sigma

Sigma Input Tables

Building data applications with Write-back and Input Tables

AUTHOR

Phil Ballai

Enterprise Architect

The Importance of Data Applications

Let's face it—traditional BI has run its course. It fails to deliver what users truly need for maximum efficiency and effectiveness in their roles. While BI vendors often tout "actionable analytics," the reality is their tools fall short of enabling direct action.

In today's data-driven landscape, organizations spanning diverse industries increasingly rely on multiple data applications to drive decision-making and operational efficiency. These applications empower users to engage with data meaningfully, shifting from static reports to dynamic, real-time data interactions.

Industries such as finance, healthcare, retail, and manufacturing harness data applications to uncover insights, streamline processes, and elevate service delivery. Financial institutions use interactive dashboards to monitor market trends and manage portfolios; healthcare providers leverage data applications to track patient outcomes and refine treatment plans; and retailers employ these tools to analyze consumer behavior and optimize inventory management.

The Importance of Data Applications

The demand for interactive data applications stems from the need for:

Real-time insights: Today's fast-paced organizations thrive on current, actionable information. Our interactive data applications deliver real-time access and analysis, empowering swift, informed decision-making.

User engagement: By integrating interactive features like input fields, sliders, and intuitive menus, users effortlessly explore data, conduct hypothetical scenarios, and extract meaningful insights.

Customization and flexibility: Every business is unique, requiring adaptable solutions that seamlessly fit their workflows. Our interactive data applications offer customizable experiences, perfectly aligned with specific organizational needs and goals.

Enhanced collaboration: Foster teamwork and synergy with our data applications, which unify insights and discussions on a single platform. This collaborative approach ensures cohesive decision-making and shared understanding across teams.

Imagine accessing actionable insights and taking immediate action—all within a single application.

Welcome to Sigma and Write-back. With Sigma, you gain powerful tools to create workflows that turn the vision of actionable data applications into reality.

By bridging traditional BI with the data-driven needs of diverse business users, Sigma empowers organizations to unlock substantial benefits.

Workato, an enterprise automation and integration platform, has streamlined complex workflows and achieved significant productivity gains since adopting Sigma. Learn more about Workato's success story here.

Nick Bunick, Principal at NewView Capital, shared his firsthand experience using Sigma during their due diligence process: “We had the privilege of leveraging Sigma during our due diligence. What impressed us most was how quickly we mastered the platform. Within minutes, we refined projections, visualized historical trends, and collaborated effectively with the Sigma team.” Discover more about NewView Capital's journey with Sigma here.

Addressing an unmet need

Workato, NewView Capital, and numerous other innovators have harnessed Sigma to pioneer new data applications centered around input tables.

Input tables stand as a cornerstone feature in crafting dynamic, interactive data applications—an exclusive hallmark of Sigma.

They serve as the crucial link between static data and user-driven interactivity, empowering users to input, adjust, and manipulate data seamlessly within the application. This capability revolutionizes conventional data analysis, fostering a more immersive and responsive user experience.

Critical attributes of input tables that underscore their indispensability in data applications include:

User input capabilities: At Sigma, we empower users to directly input data into our platform, driving real-time calculations, updating visualizations, and triggering workflows. This hands-on approach fosters interactive data analysis and enhances user engagement.

Data validation: Data accuracy is paramount. With built-in validation features, Sigma's input tables enforce custom rules and checks to uphold data quality, ensuring reliable insights.

Real-time updates: Updates made in Sigma's input tables propagate instantly across the platform, ensuring all stakeholders access the latest data. This dynamic capability boosts responsiveness and maximizes utility.

Integration with other features: Our input tables seamlessly integrate with Sigma's controls, visualizations, and external databases. This unified approach empowers users to build robust data applications tailored to diverse analytical needs.

Customization and flexibility: Customizable to meet specific requirements, Sigma's input tables offer flexibility in data entry, formatting, and interaction. This adaptability makes them ideal for a wide range of data-driven scenarios.

Minimize time to production: Sigma streamlines development cycles by enabling rapid creation and deployment of interactive data forms. This process minimizes reliance on extensive coding or IT support, while upholding stringent security and governance standards.

Across industries, the increasing demand for interactive data applications highlights the critical role of features such as input tables. Sigma's input tables empower user interactivity, ensure real-time updates, and seamlessly integrate, revolutionizing how organizations engage with and extract value from their data.

Getting started with Sigma Input Tables

How is data stored?

Before diving in, it's crucial to note: Sigma never stores customer data outside of your cloud data warehouse (CDW). Our Input Tables and Write-back functionality are essential features that empower Sigma data applications to securely send data back to your CDW for storage.

Here are the key points to understand:

1. Sigma never stores customer data outside the CDW.
2. When using Write-back, data is stored in the CDW only where you specify, as configured in the Sigma administrative interface, shown below.
3. Input table data is stored adjacent to other data, allowing it to be joined on key-value pairs if needed.
4. Sigma never overwrites data. Instead, it captures every save made by any user of an input table, maintaining a history of changes for tracking purposes.

Having grasped those points, let's dive into configuring Sigma and Snowflake to begin using input tables and discover their versatile applications.

For the most up-to-date details on supported warehouses, see here.

Configuring write back

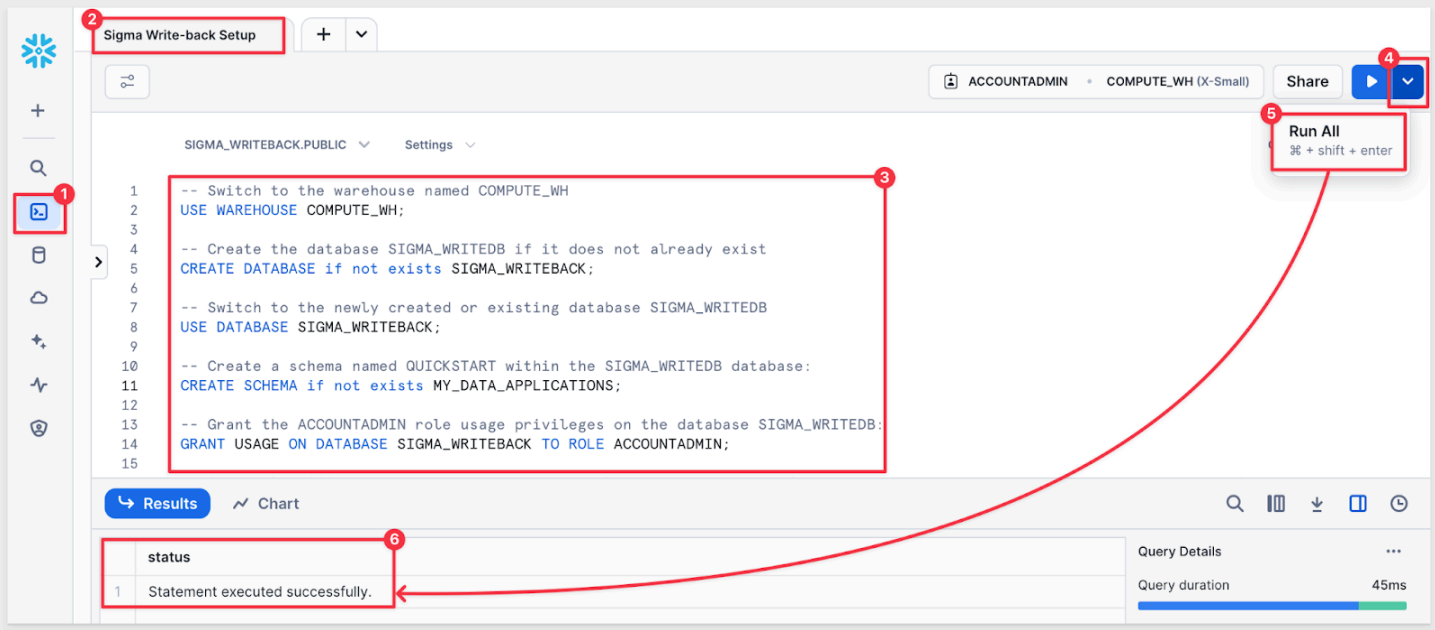
To showcase this, we'll utilize Sigma and Snowflake trial instances.

Our first step involves configuring resources within Snowflake to store data written back to the warehouse from Sigma.

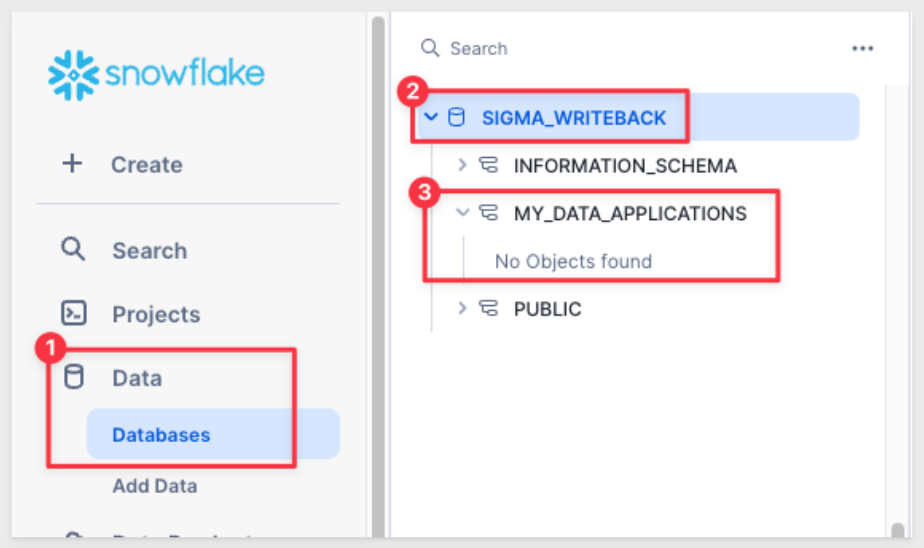
In Snowflake's interface, logged in as an "**ACCOUNTADMIN"**, create a new worksheet and run the following script, which creates a new database and schema and grants the required usage to the role "**ACCOUNTADMIN**":

```
-- RUN STATEMENTS AS "ACCOUNTADMIN" AND USE THE "COMPUTE_WH"

-- Create the database SIGMA_WRITEDB if it does not already exist
CREATE DATABASE if not exists SIGMA_WRITEBACK;

-- Switch to the newly created or existing database SIGMA_WRITEDB
USE DATABASE SIGMA_WRITEBACK;

-- Create a schema named QUICKSTART within the SIGMA_WRITEDB database:
CREATE SCHEMA if not exists MY_DATA_APPLICATIONS;

-- Grant the ACCOUNTADMIN role usage privileges on the database SIGMA_WRITEDB:
GRANT USAGE ON DATABASE SIGMA_WRITEBACK TO ROLE ACCOUNTADMIN;

-- Grant the ACCOUNTADMIN role usage, create table, create view, and create stage privileges on
the MY_DATA_APPLICATIONS schema:

GRANT USAGE, CREATE TABLE, CREATE VIEW, CREATE STAGE ON SCHEMA MY_DATA_APPLICATIONS TO ROLE
ACCOUNTADMIN;
```

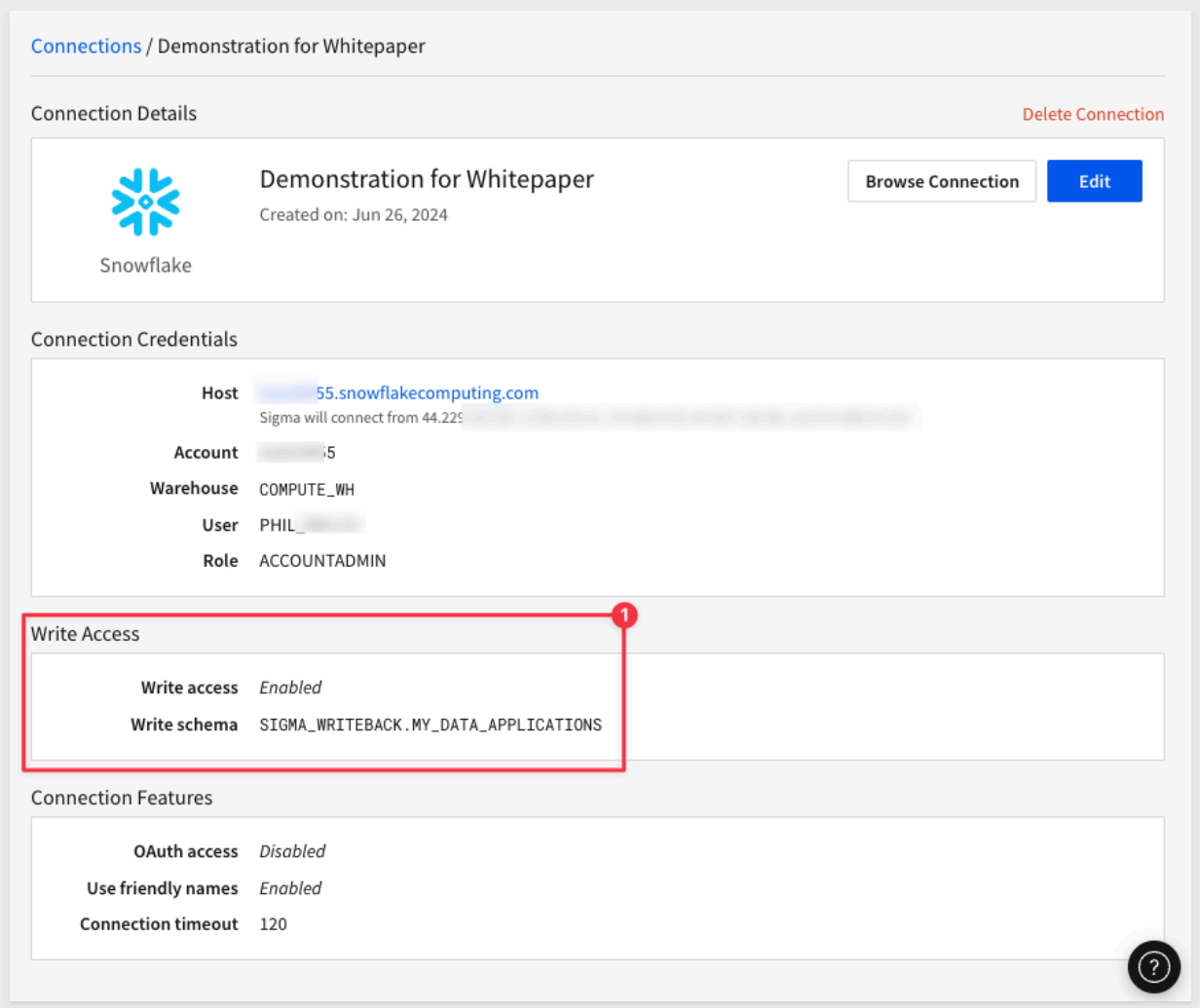
After running the script, you should see the result: "**Statement executed successfully**".



This script results are a new database and schema to hold input table data in the warehouse:



We can now jump over to Sigma, log in as an "**Administrator**," and configure Write-back using the database and schema we created above:



Once the connection is created, Sigma checks to ensure it is valid. That's it—we are ready to create our data application.

Note: Sigma also supports using [OAuth for write back enabled connections to Snowflake]

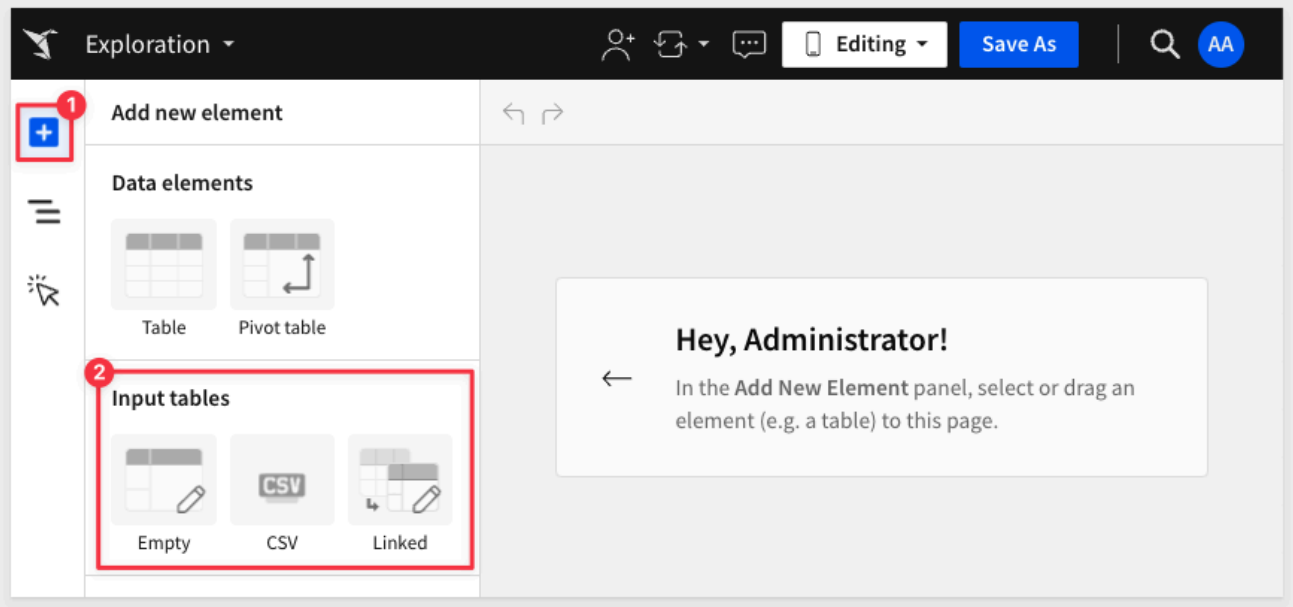
My first Sigma data application

Occasions arise when gathering even a modest amount of data from a Sigma user proves incredibly beneficial for the business. However, this does come with its own set of challenges:

- Will you have to build a new application to capture the data?
- When is the right time to capture the data?
- Will the users accept another tool?
- Once we have the data, how will it relate to existing warehouse data?
- Many other questions may arise!

Whether you're already leveraging Sigma or still relying on spreadsheets (*when you should be using Sigma*), input tables offer a powerful solution. Imagine creating a Sigma workbook enriched with the capability for users to input small data sets in real-time. Sigma takes care of the complex UI and data operations, seamlessly storing your data in the warehouse.

Within Sigma workbooks, users can choose from three types of input tables. The first two types function much like traditional spreadsheets, making them intuitive to us.



1. **Empty input tables** are blank tables that support data entry in standalone tables independent of existing data. You can edit data at the cell level and add editable rows and columns to construct your table as you see fit. You can also copy/paste values in up to 12,500 cells at once (500 rows and 25 columns).

2. **CSV input tables** support data entry in standalone tables; however, they allow you to pre-populate the table with uploaded CSV data (max 200 MB). You can then edit the uploaded data at the cell level and add editable rows and columns to construct your table as you see fit.

3. **Linked input tables** support data entry alongside existing data from other elements in the same workbook. Typically, these existing elements are other data from the warehouse that are related in some way. As a child element, a linked input table includes one or more linked columns that reference data in the parent element.

In all three types, data is written to the CDW as previously described.

Allowing users to add or supplement warehouse data opens a world of possibilities. We will demonstrate how this can be done using a very simple example.

Example use case

PROBLEM STATEMENT

Consider this scenario: Our accounting team requires real-time shipment status updates and needs to collect comments or notes from the shipping department. However, they're reluctant to grant shipping clerks access to our accounting system. Given our current budget constraints, we're searching for a solution that doesn't involve investing in new software or upgrading existing systems.

SIGMA INPUT TABLES ARE THE SOLUTION

Since Sigma is already in play, we can harness its power to swiftly and effortlessly tackle this issue, all without concerns about extra expenses, developers, or even security and governance.

Users know how to use Sigma, have permissions based on their role, and the data will be stored in the existing warehouse, so all that is handled already. Awesome!

We will use sample data provided to all Snowflake customers and join table data from the warehouse as our source of invoice data.

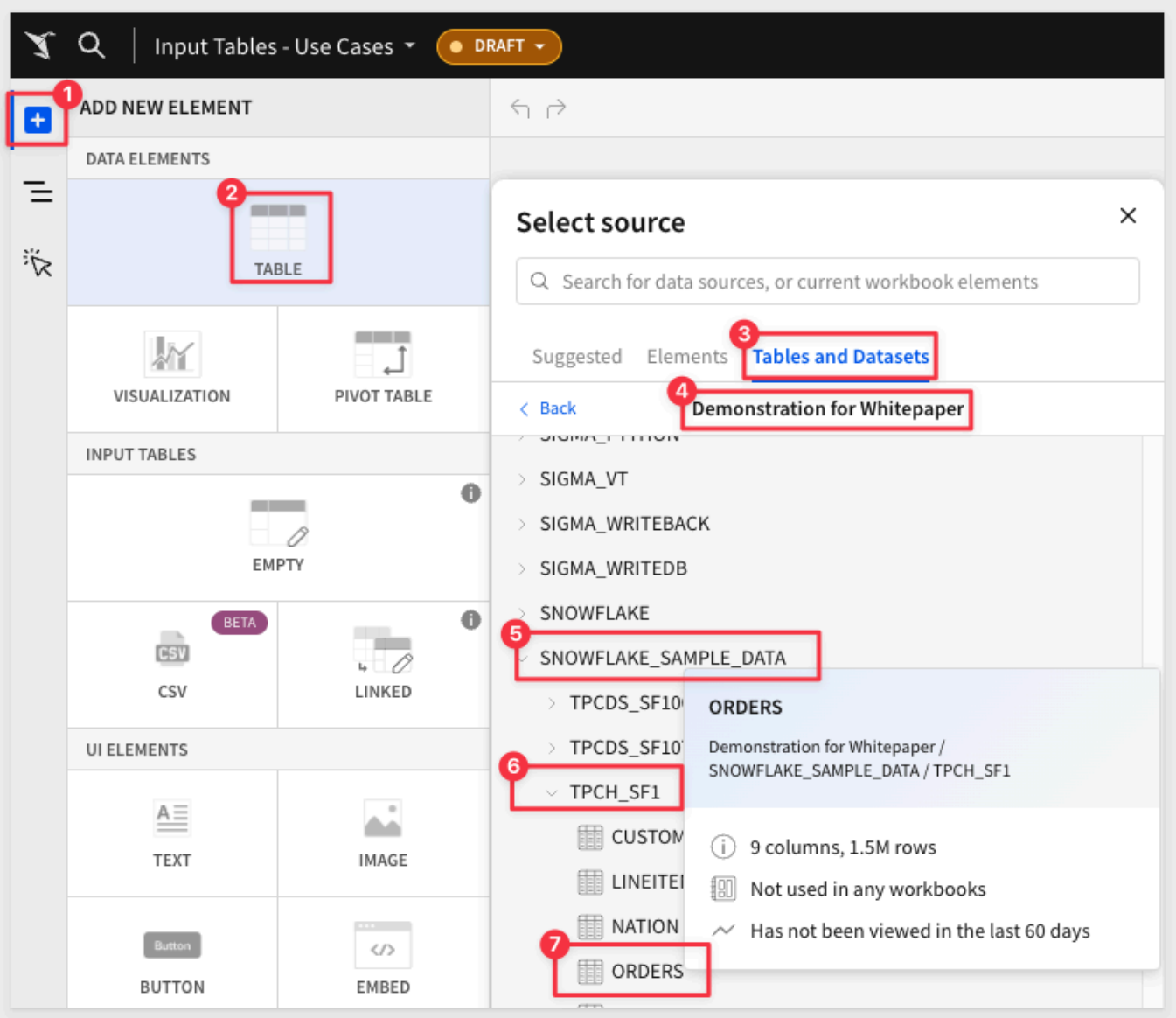
Note: Sigma provides all customers a sample dataset as well, but we will use the Snowflake sample so that we can look "under-the-hood" at the Write-back data later. Since the Sigma sample data is hosted by Sigma, we would not be able to do that.

Along the way, we will also expose you to just a few of the many amazing features that Sigma provides.

There are many ways to solve problems with Sigma, and while we don't have space to show them all, we want you to see just how fast and flexible Sigma can be while solving this use case.

1: In Sigma, create a new workbook named "**Input Tables - Use Cases**"

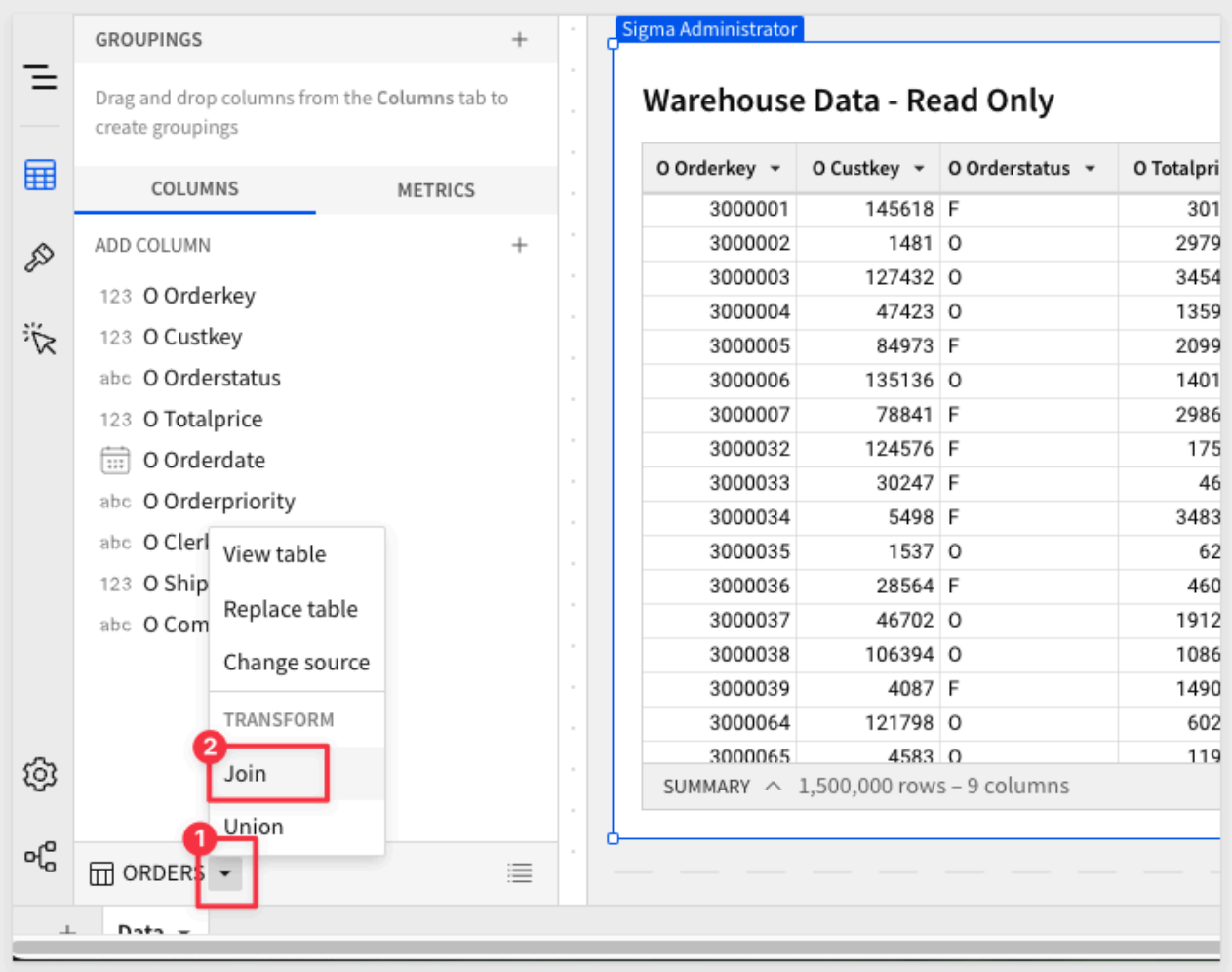
2: Add a new table element to the page and select (or search for and select) the table in "**SNOWFLAKE_SAMPLE_DATA**" > "**TPCH_SF1**" > "**ORDERS**":



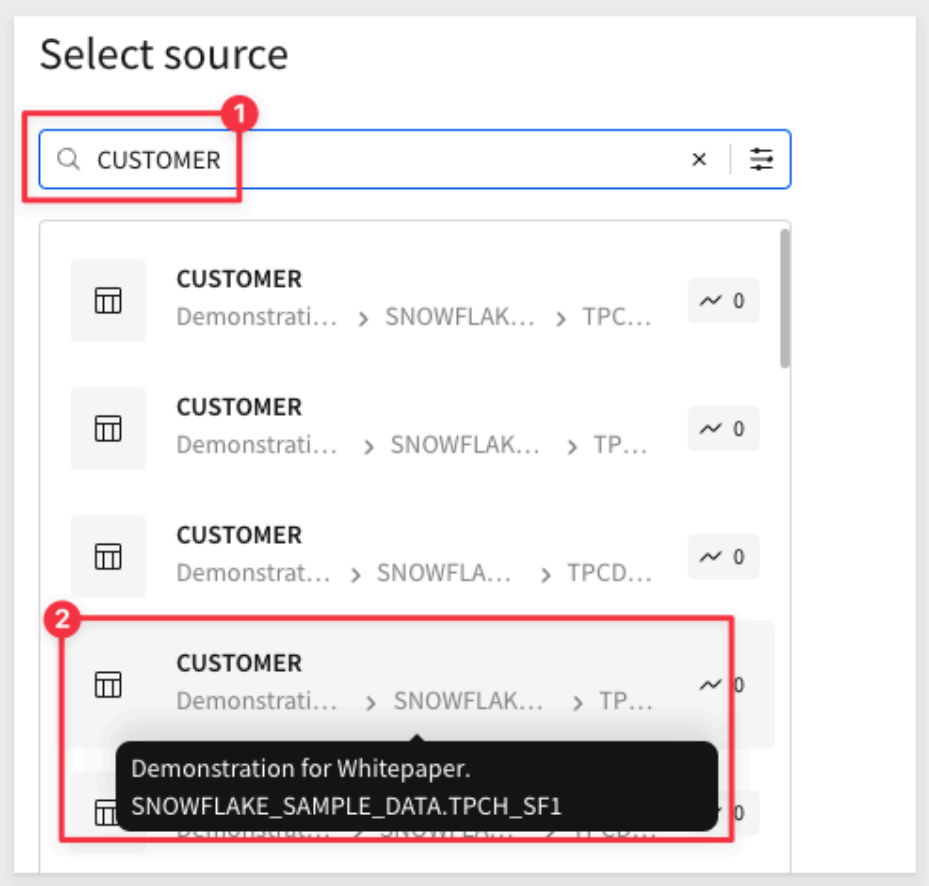
Note: We could have also just asked Sigma to show all the tables named "Orders" across all connections to find it fast!

3: Rename the new table to "**Warehouse Data - Read Only**" and the page name to "**Data**."

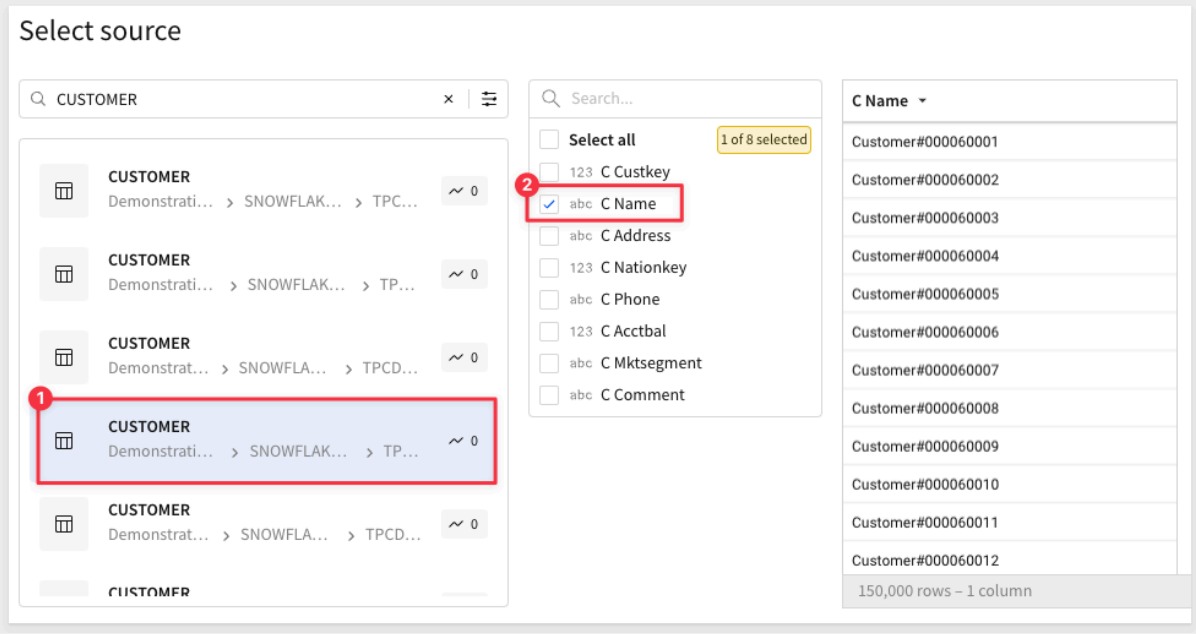
We want to enrich this data by joining it with the customer information for each order. Click the table to select it and join the additional table as shown below.



Sigma's source selector lets us search for tables named "**customer**" and select the one we want.

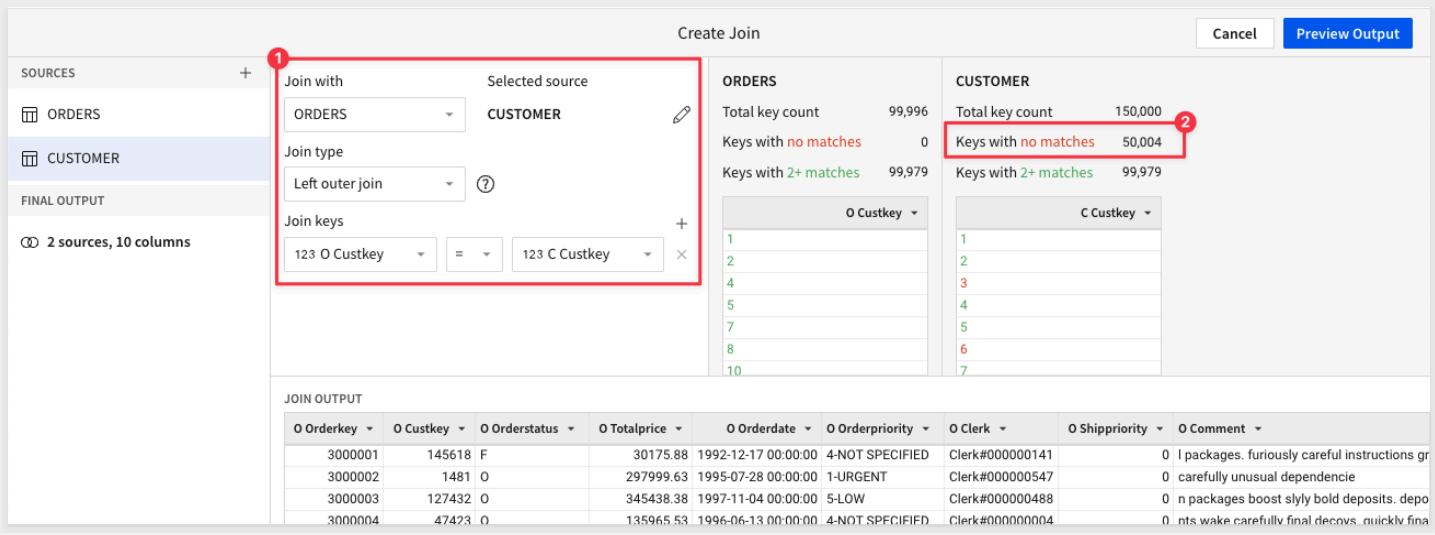


After clicking on the correct "**CUSTOMERS**" table, we can also prune the list of columns we want to include. We will keep this bare minimum and select just the "**C_NAME**" column.

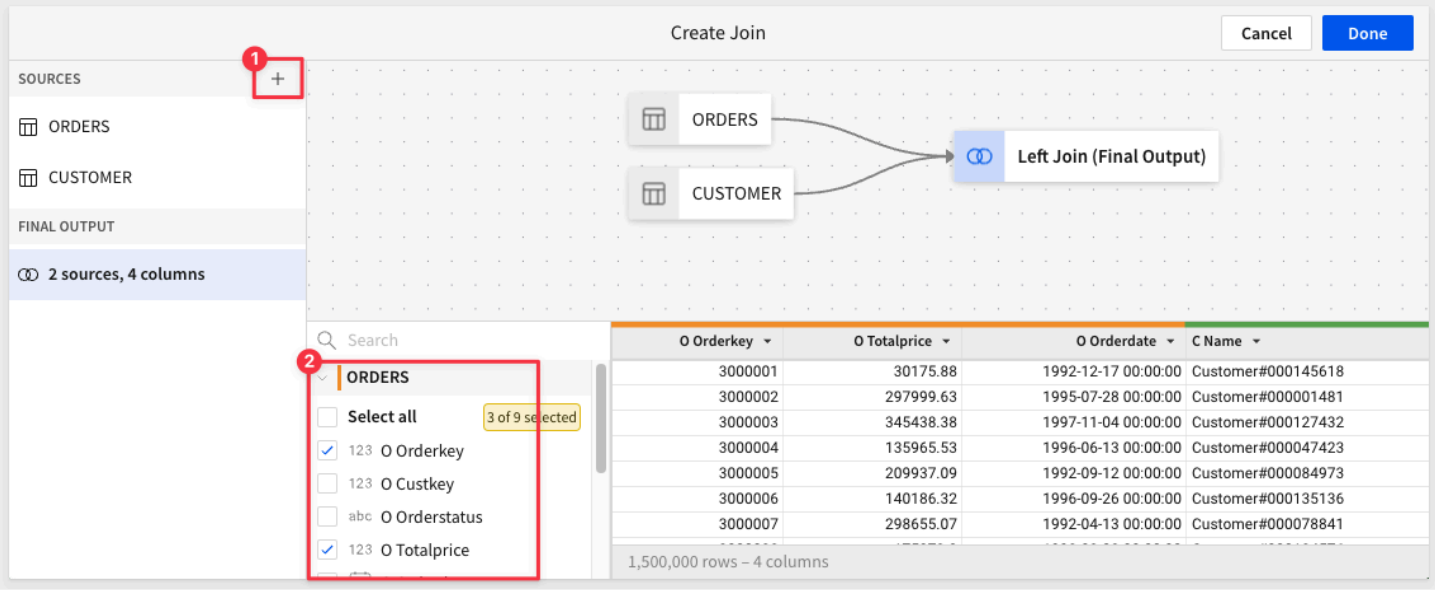


Next, we have the option to adjust the join type, join keys, and examine the matches. Make sure that the Join kets are matching on "**Customer Id**".

In this case, there are 50,004 customers who have no orders. This is normal and expected; click "**Preview Output.**"



Sigma shows us the lineage to make sure it looks correct (these can become quite complex) and prune the column list further. We don't need many columns for this use case, so we will limit to what is shown below.

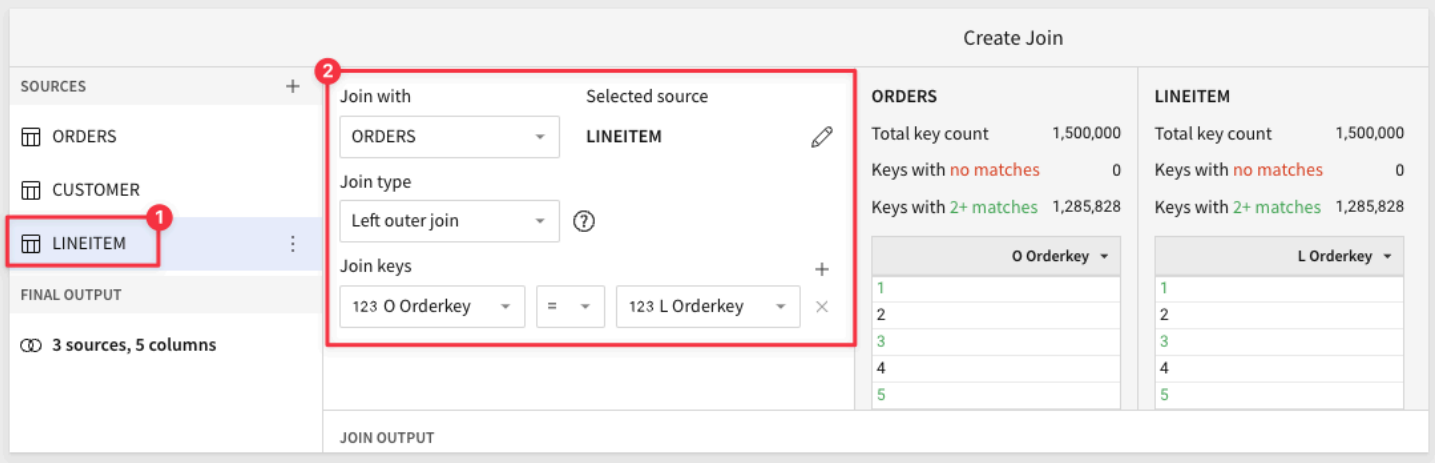


Shipping has requested that we also show the part name so that the clerk can visually verify that while looking at the shipment being final packed.

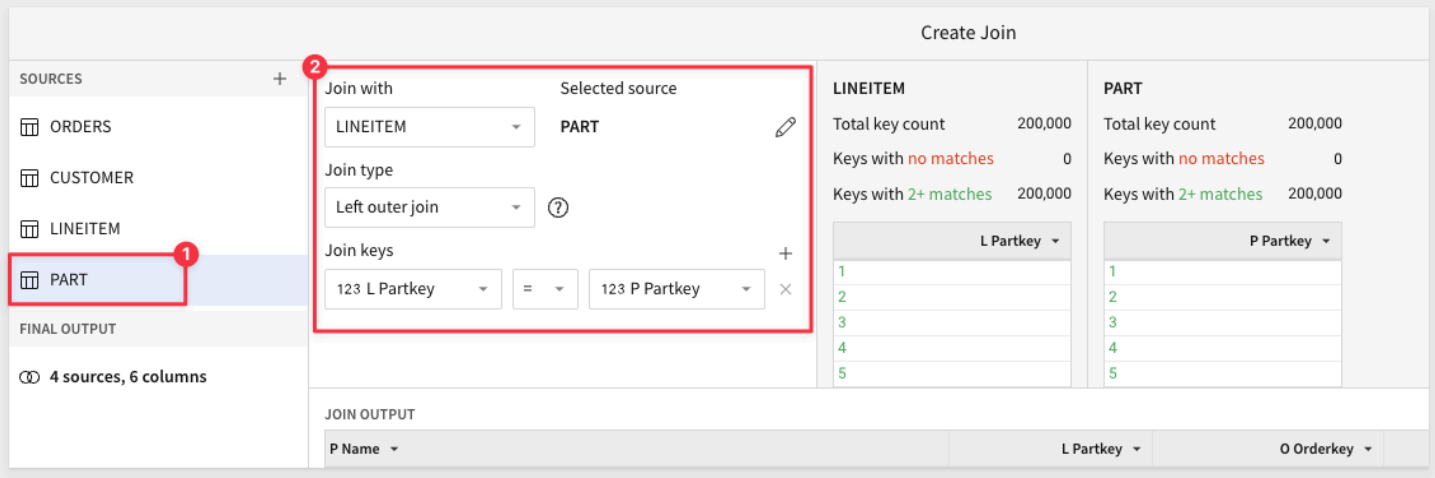
That is really simple and is just repeating the workflow we just did, adding the two required tables.

We do this by clicking on the "**+**" icon in the image above.

We need to add the "**LINEITEM**" table and the "**ORDERS**" table, joining them on "**Order Key**".

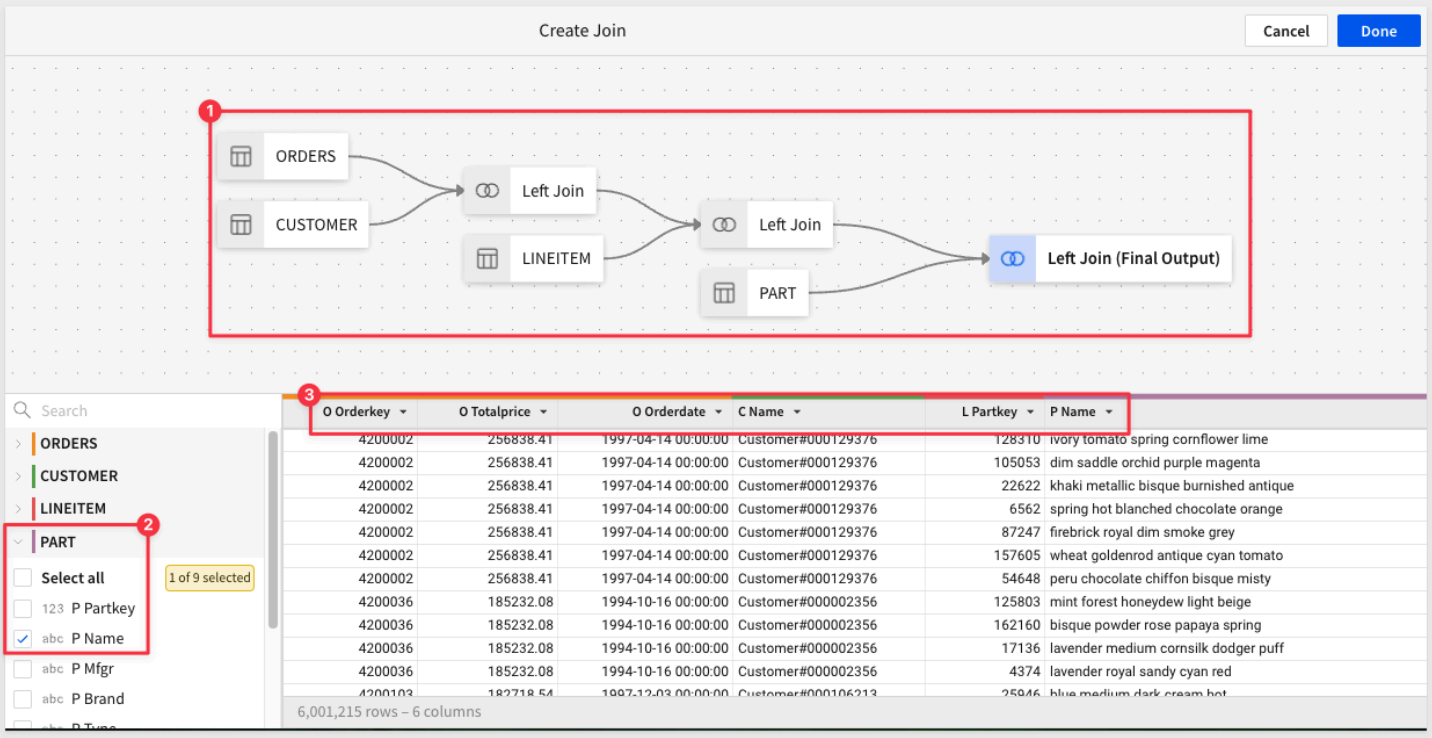


To get the actual part name, we need to add the "**PART**" table and the "**PART**" table, joining them on "**PartKey**".

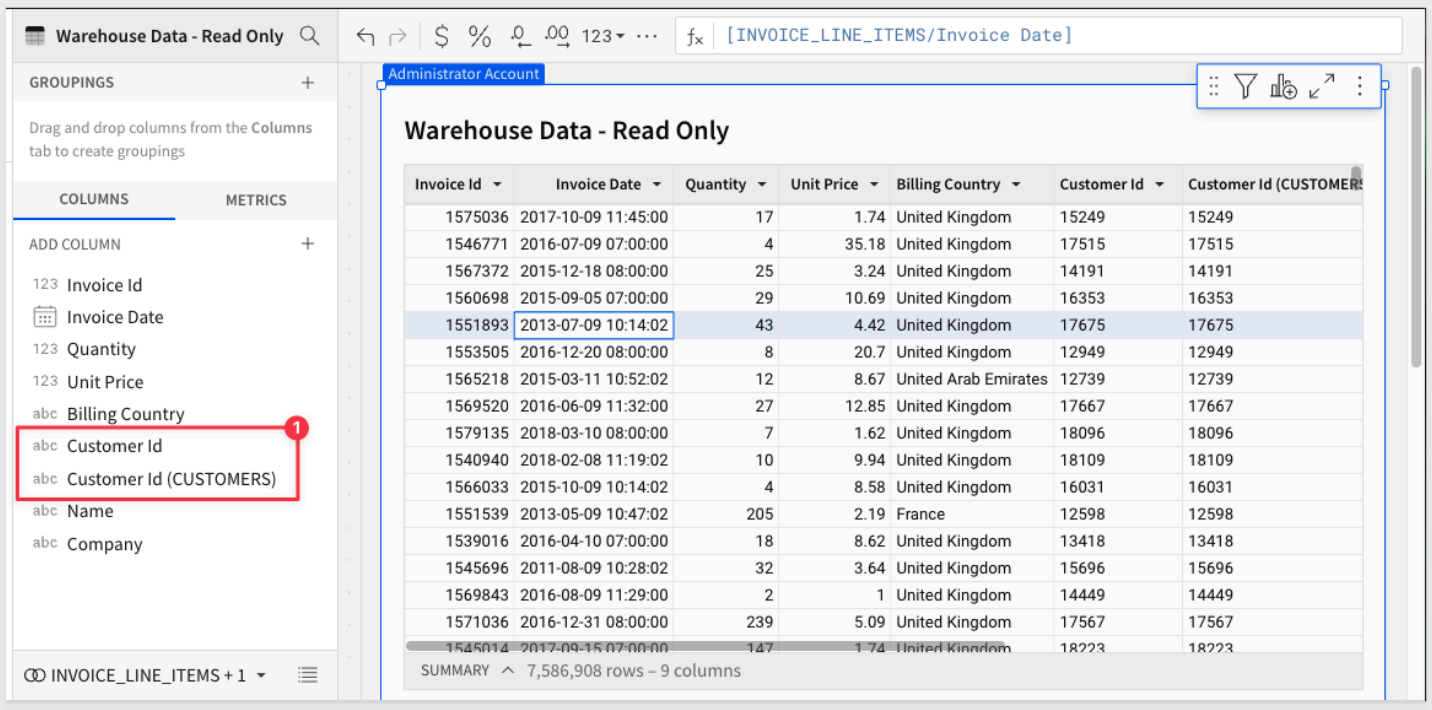


Click "**Preview Output**".

We have a revised lineage and the six source columns we need for our data application.



This source data from accounting will operate as the base data for our input table to come. We did this in just a few minutes; the Sigma interface is always working to make your job easier. Click "**Done**".



The next step is to add our input table. However, when we add a new linked input table, we will need to select a column to act as a unique identifier.

This will present a problem; most invoices contain more than one line item, so we need to have an identifier that will be unique per row.

Are we out of luck? Do we need to engage a developer? **Nope.**

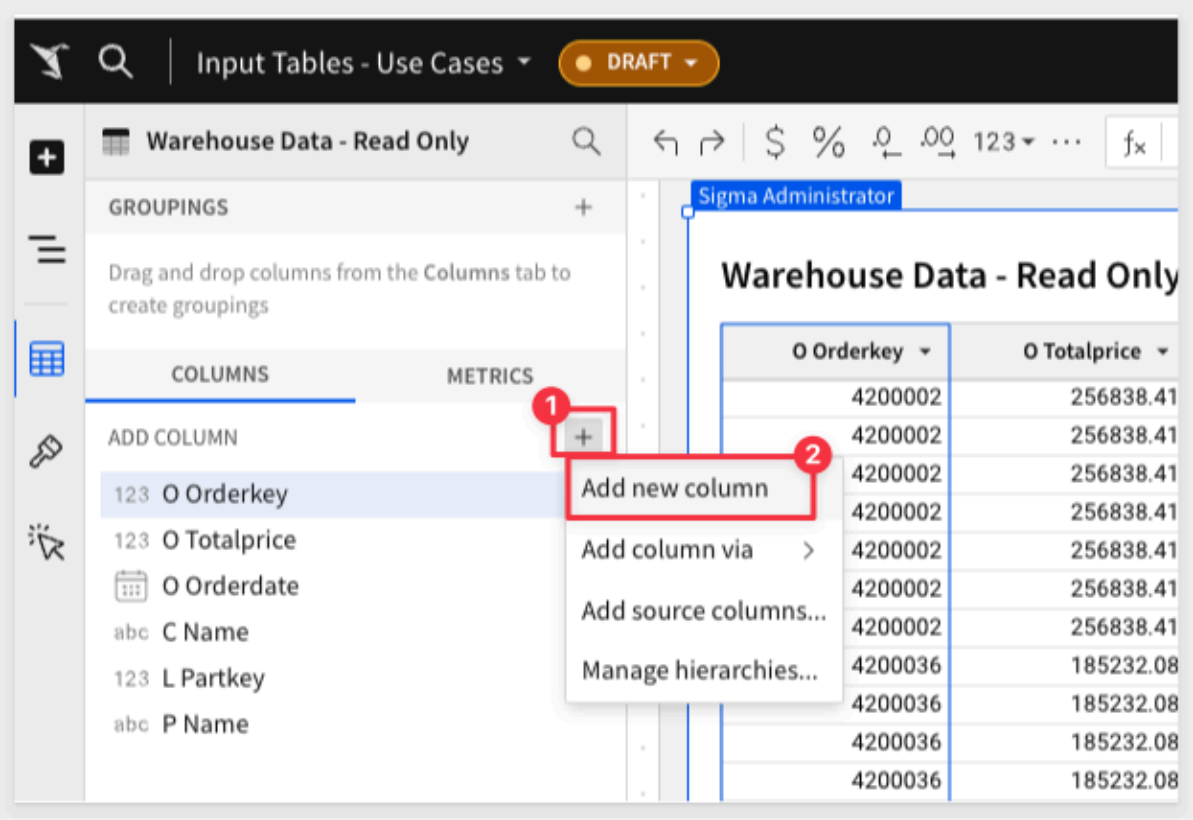
A cool feature of Sigma is its ability to enrich warehouse data on the fly, in a user's browser. This allows us to add a column to our source data that will provide the necessary uniqueness to rows, but not alter the warehouse data in any way.

To learn more about how Sigma does this and more to allow us to operate at massive scale without sacrificing performance and keeping costs in check, read this QuickStart.

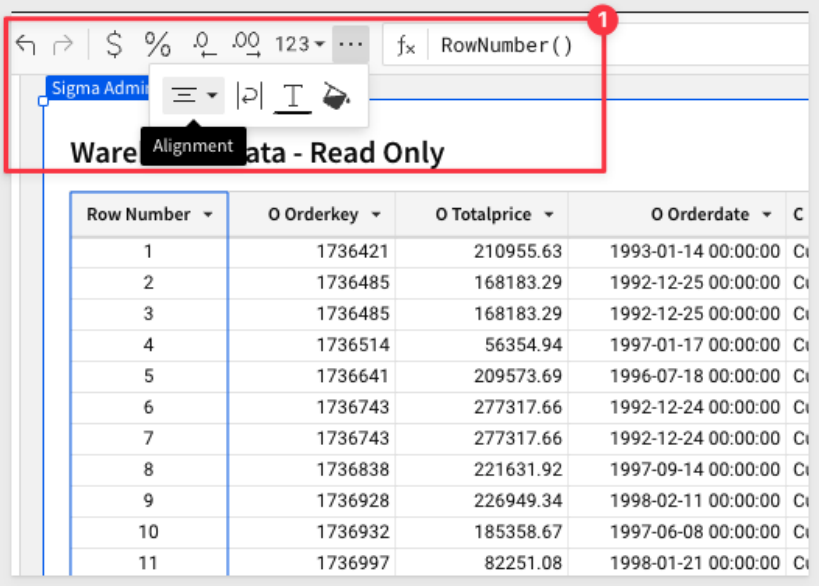
Let's not add the input table just yet.

4: On the Data page, click to add a new column, name the column "**Row Number**", and set its formula to:

```
=RowNumber()
```

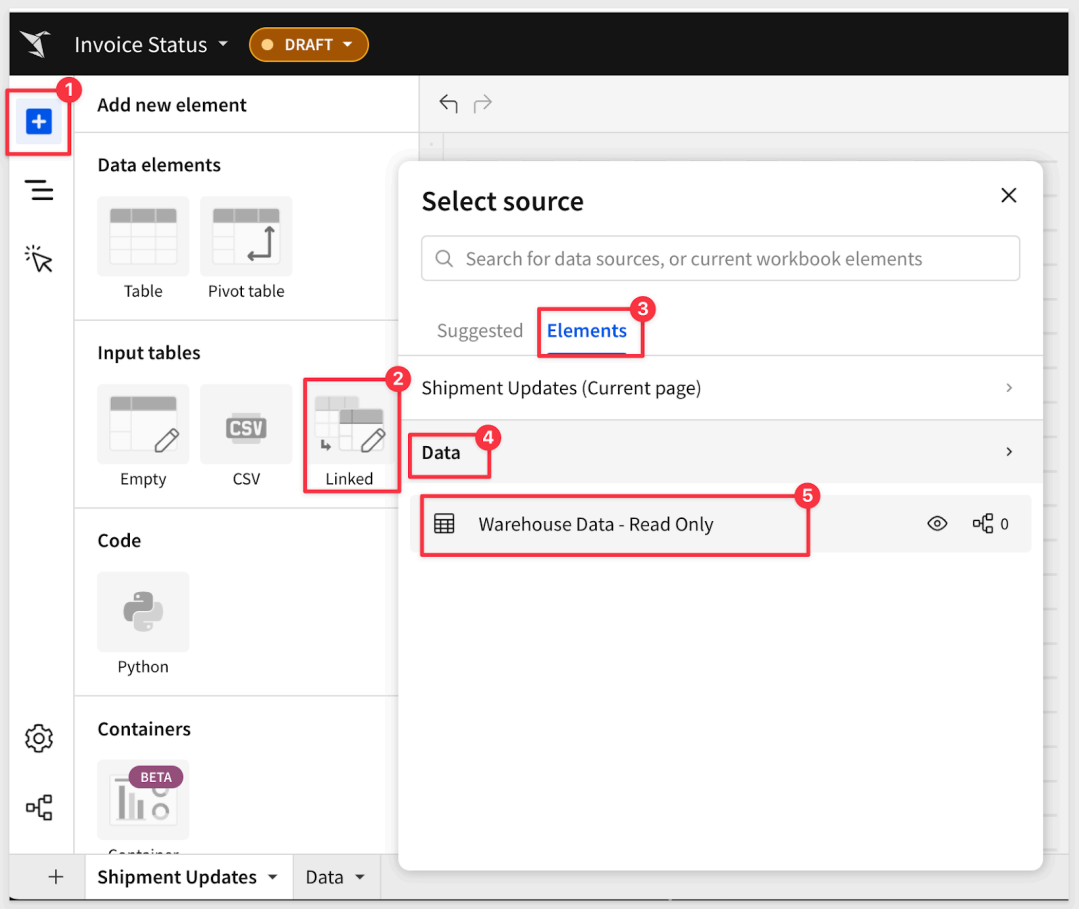


Drag the column to the first position. You may want to sort the "**Row Number**" column, but it is not required. We can apply styling and formatting to columns as shown below.

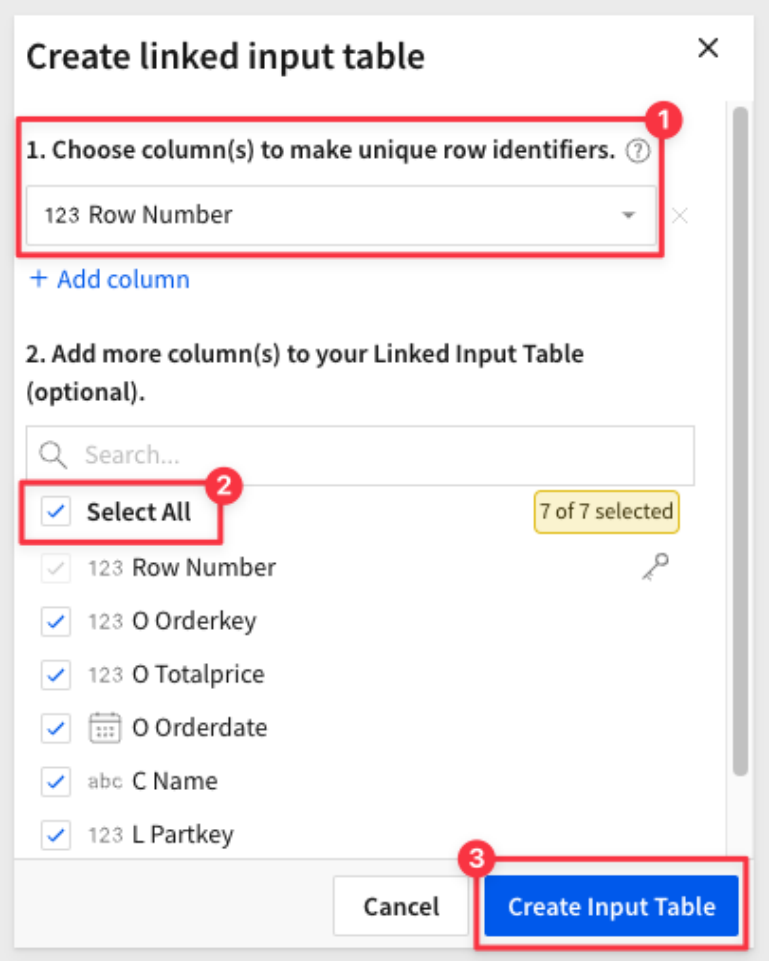


Now we are ready for our input table.

5: Add a new page to the workbook called "**Shipment Updates**" Add a "**Linked Input Table**" to the page and select the table on our "**Data**" page as shown below.



Select the "**Row Number**" column for the unique identifier and **select all other columns**. Click "**Create Input Table**"

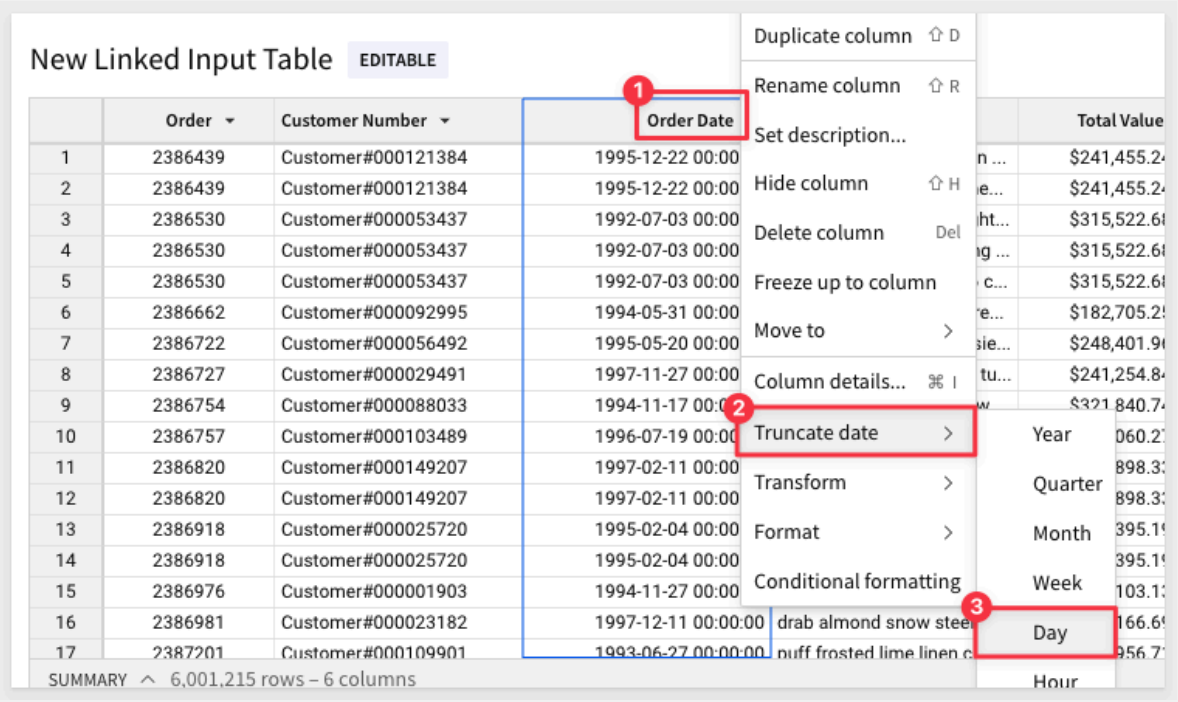


We now have our first input table ready for the next step.

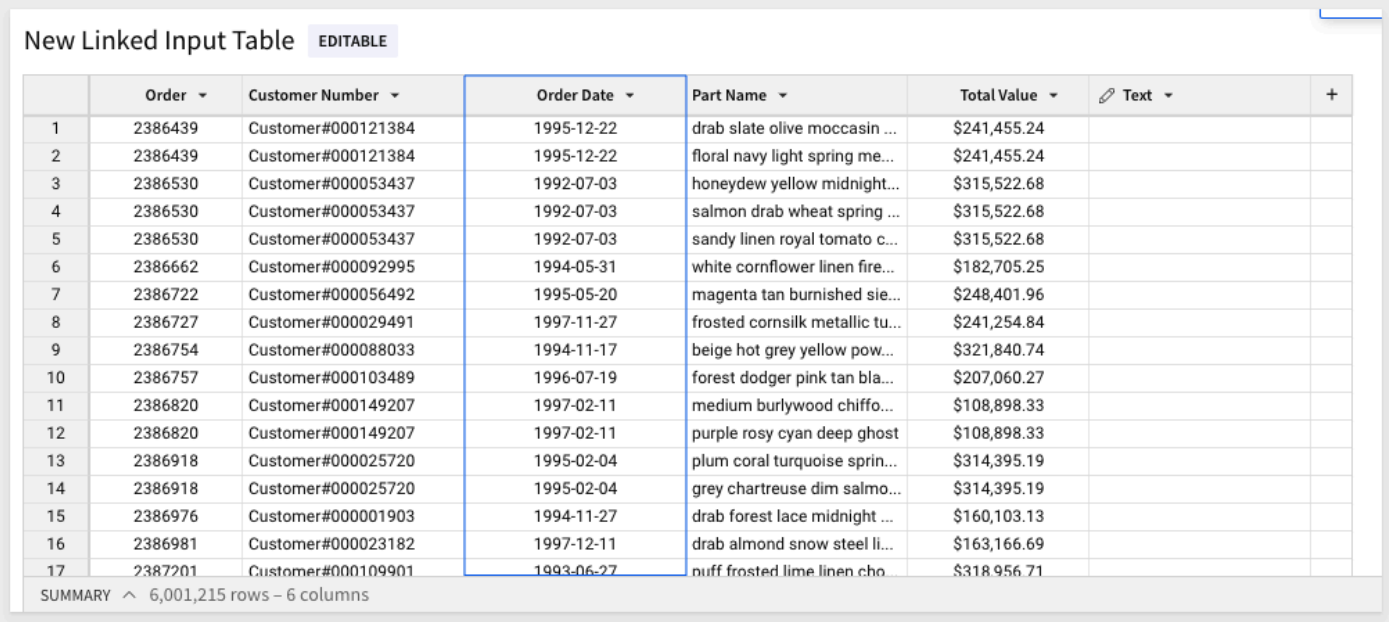
6: Let's refine the input table a little.

We can rename each column by double-clicking on the column header, to make the name more user friendly. Remember, these changes do not affect the source warehouse data in any way. We can also hide the "**Row Number**" and "**L Partkey**" columns by opening the column menu for each and selecting "**Hide**".

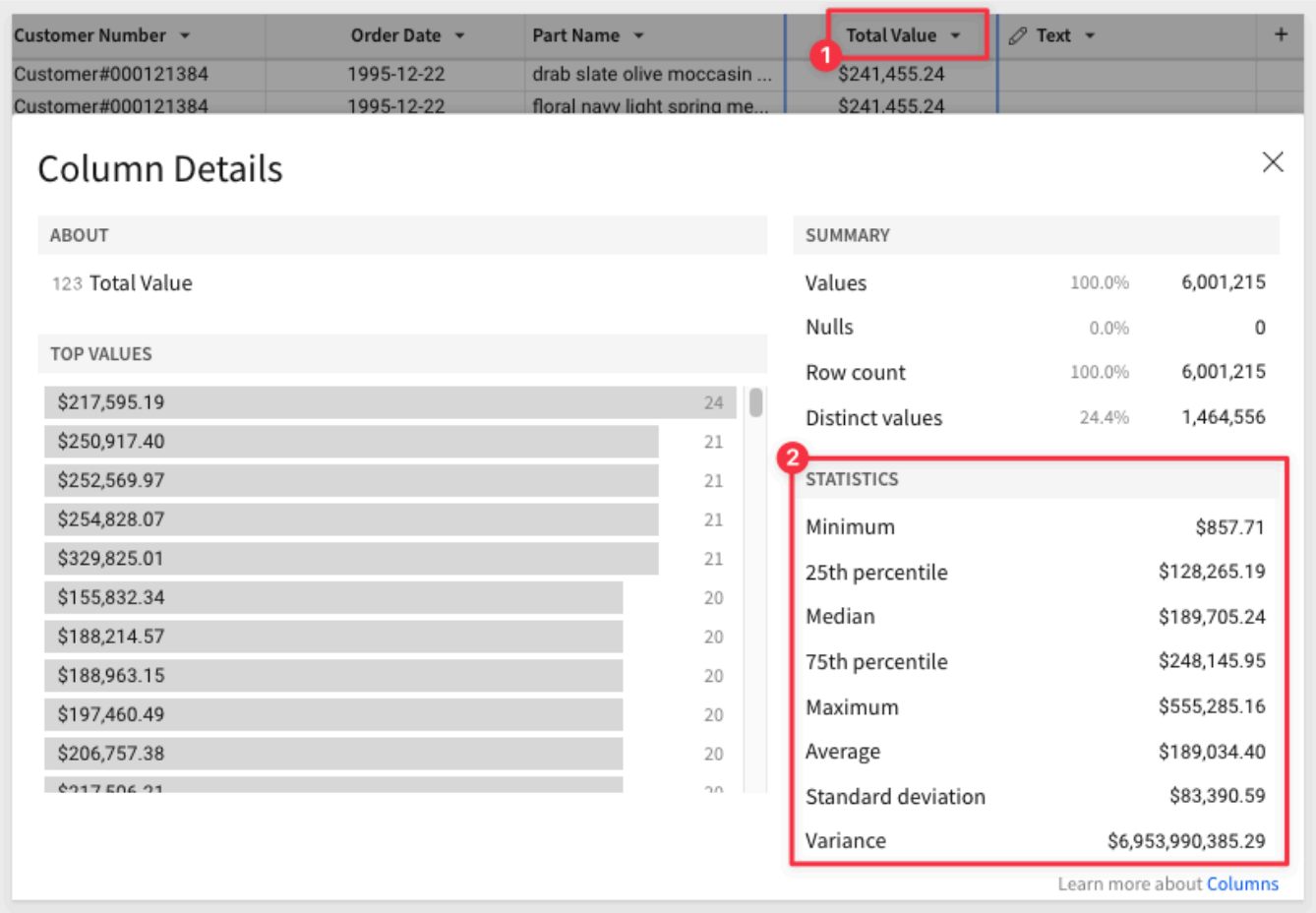
Columns can be centered, formatted as currency and more. For example, the "**Order Date**" column shows the order time and we really don't need that. We can easily truncate that column to just the day.



Our input table now looks like this.



Shipping asked for an easy way to identify the higher value shipments. Since we don't have guidance on what constitutes a "**high value shipment**", we can use column details on the "**Total Value**" column to see what the statistics show us.



With this data, we can consider values that are significantly above the median and closer to the upper percentiles. Here are a few approaches we can consider.:

Use 75th Percentile as a Threshold:

The 75th percentile value is $248,145.95. This indicates that 25% of the orders are above this amount. Using this value as a threshold can be a good indicator of "**high value**" orders.

Use Maximum Value:

The maximum order value is $555,285.16. Orders closer to this maximum value can be considered exceptionally high value.

Use Standard Deviation Above the Mean:

The average (mean) order value is $189,034.40, with a standard deviation of $83,390.59. Orders that are one standard deviation above the mean ($189,034.40 + $83,390.59 = $272,425) can be considered high value. This would mean orders above approximately $272,425 are high value.

We could do this easily by adding a calculated column, and using the Sigma function for standard deviation.

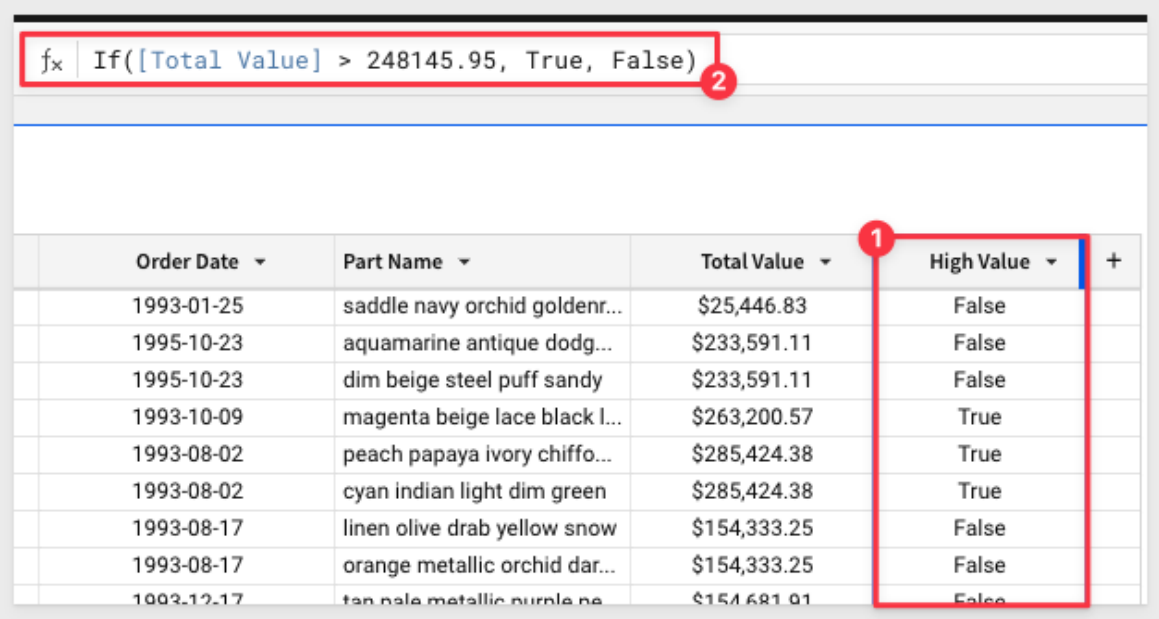
Let's keep it simple for now and use option one. The 75th percentile value is a commonly used threshold for high value, as it naturally divides the top 25% of orders from the rest.

We want to make the interface really simple for the users, so we will add a toggle control that shows only the high value orders, with a single click.

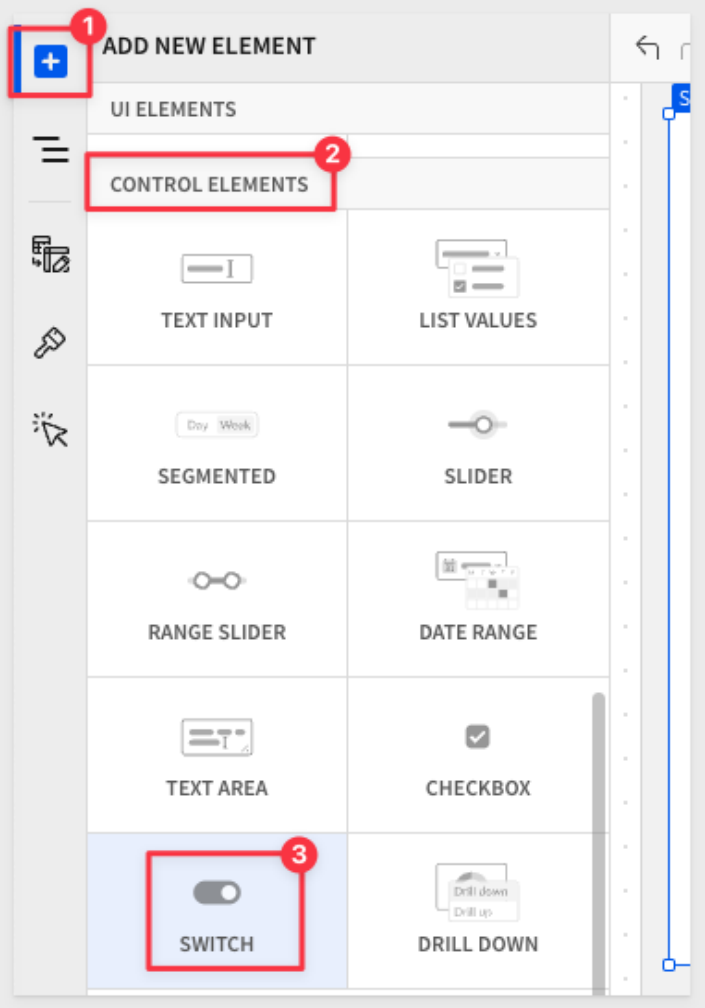
Add a new calculation column and set it's formula to:

```
If([Total Value] > 248145.95, True, False)
```

This formula is pretty self-explanatory but if the value of "**Total Value**" in the adjacent cell is greater than the 75th percentile value (as determined by the column details earlier), the cell is set to "**True**". Otherwise it is "**False**".

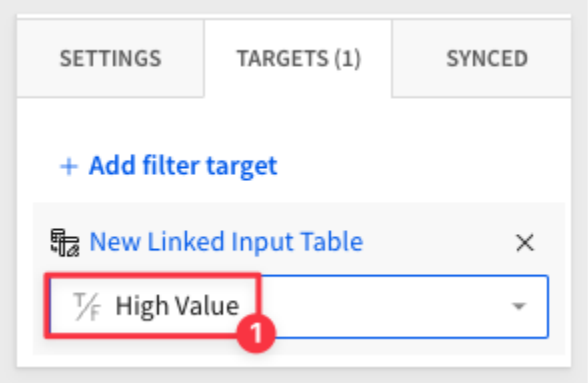


Next we add a "**Switch control**" to toggle the input table, based on "**High Value**".



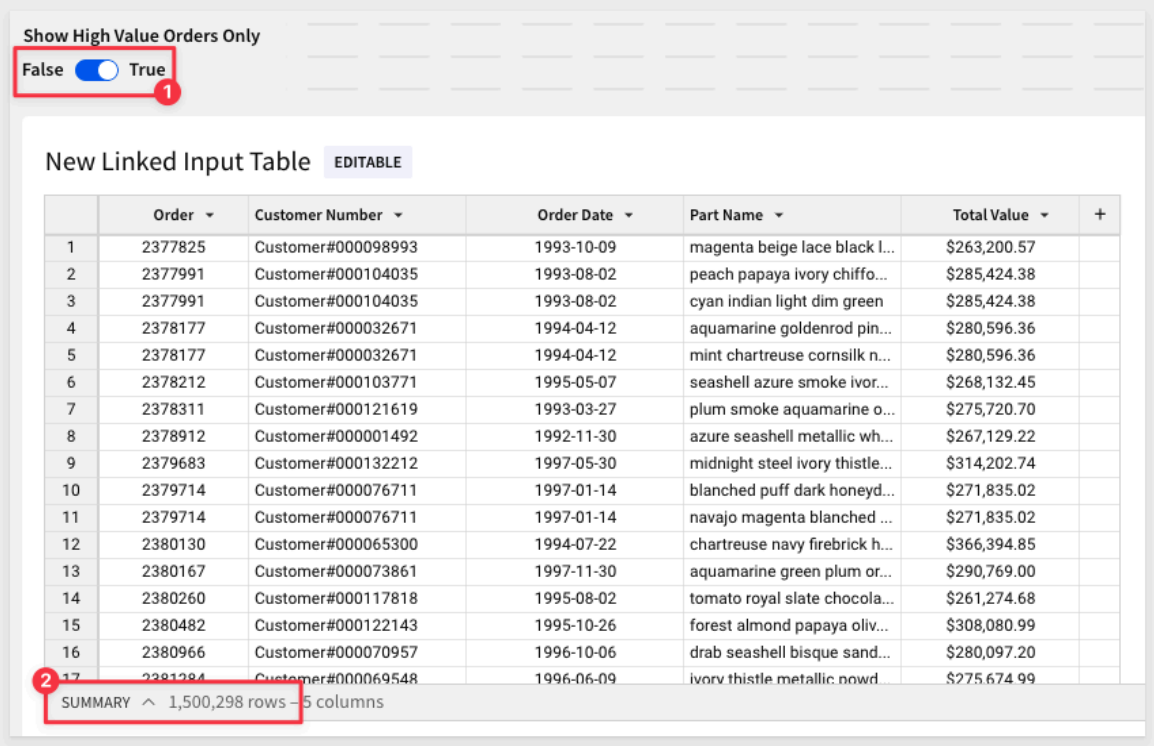
We set the controls "**Target**" to the input table.

And specify the "**High Value**" column.



With the control set to "**True**" we only show the orders that are above the 75th percentile. Notice that the row count is reduced to 1.5M rows from the previous 6M.

We also can hide the "**High Value**" column since users don't need to see it to have it work with the switch control.

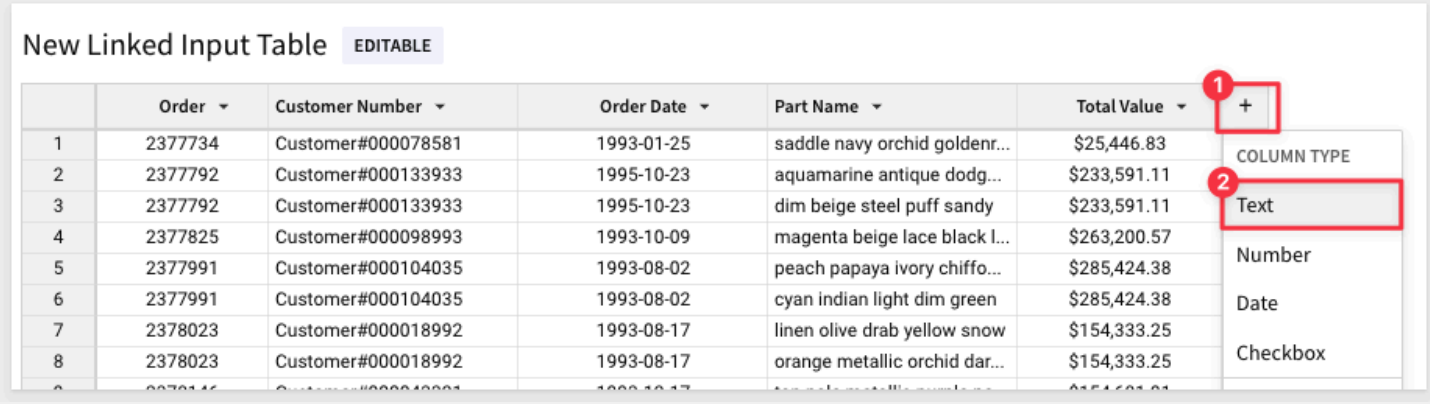


By the way, did you notice that we have been working with around 6 million rows of data like it was nothing. That is the power of Sigma.

We are done with the basic cleanup operation.

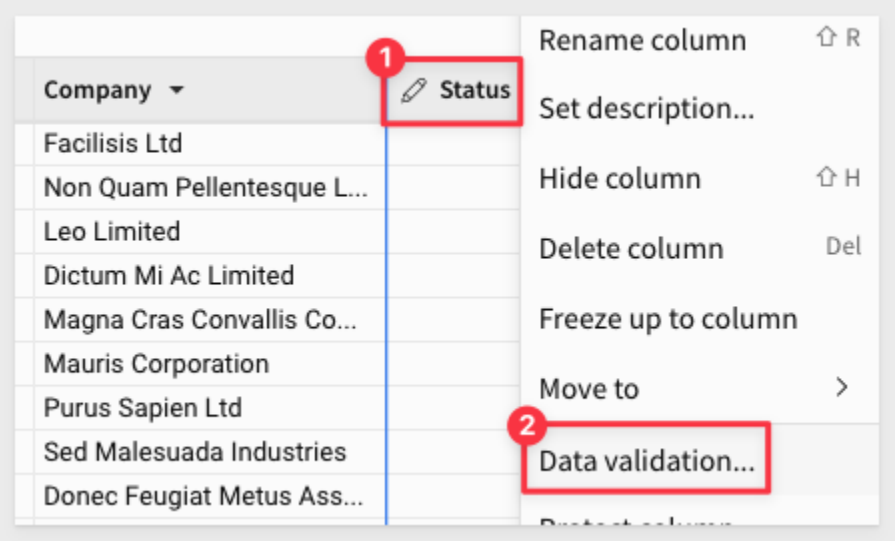
7: Now we can add the columns we want shipping to use for capturing the line item status.

Click the "**+**" after the last column in the input table and add a new "**Text**" column.

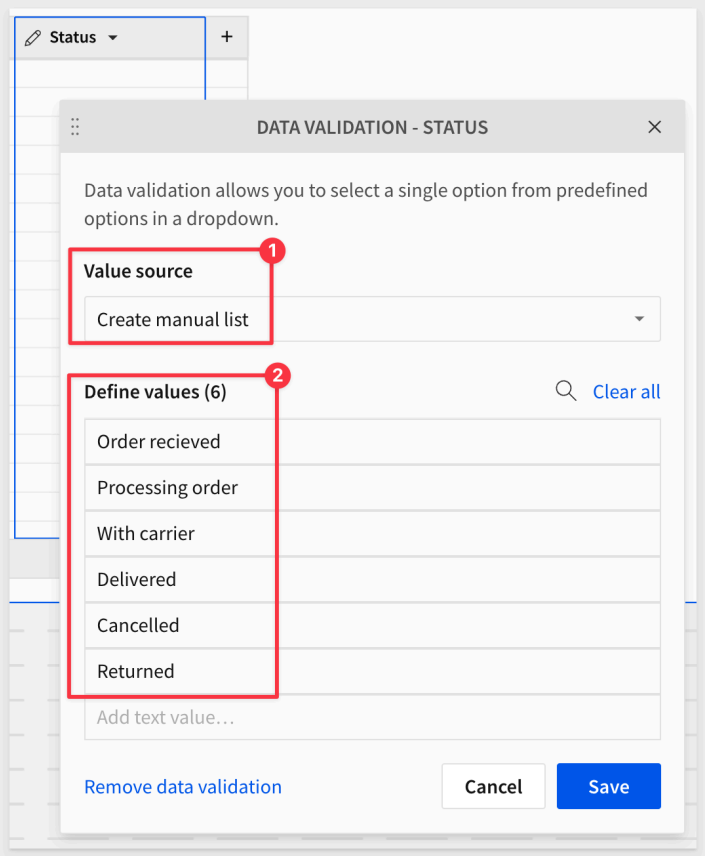


Rename the "**Text**" column to "**Status**."

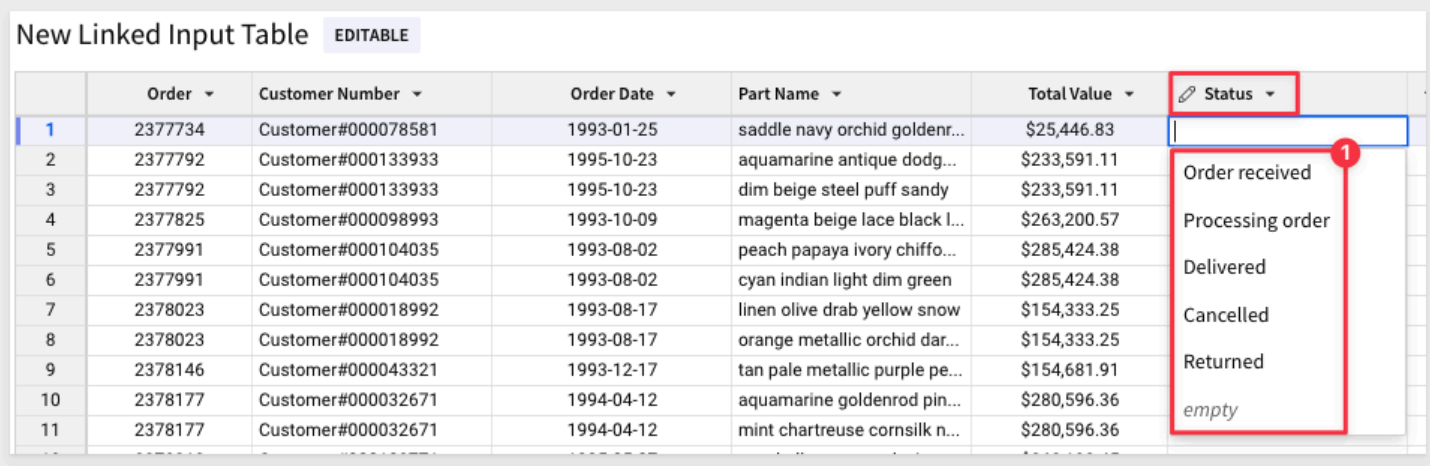
We don't want users free-form typing into the status cells, so we can add data validation. This will allow them to select from a predefined list.



We have the option to drive the validation list from another data source or create a manual list, which we did:

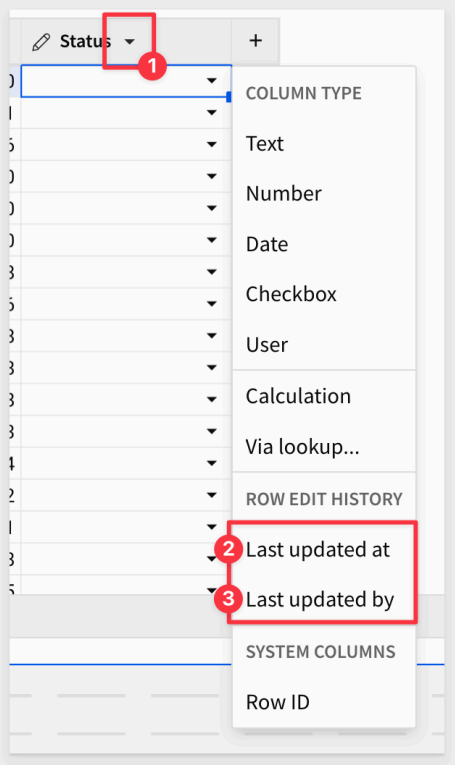


Now users are required to select from the list of valid choices or none.

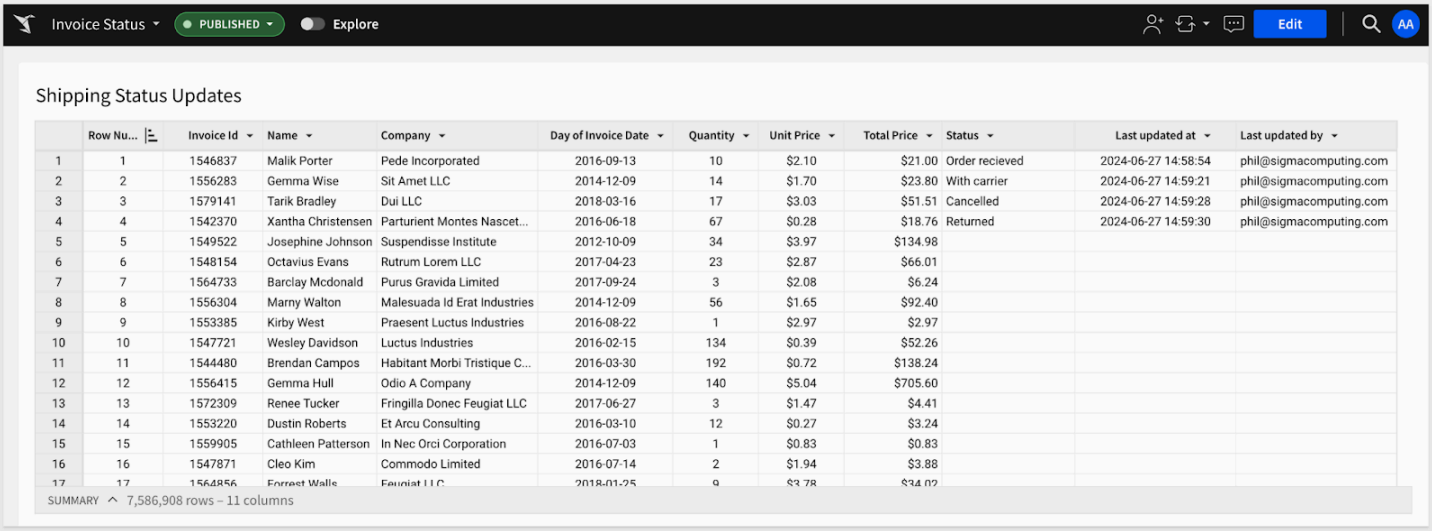


We also want to track who made the last change, and when. That is simple too.

By opening the menu for the "**Status**" column (since it is last in line, or just use the + icon as shown earlier), we can add the columns for "**Last updated at**" and "**Last updated by**".



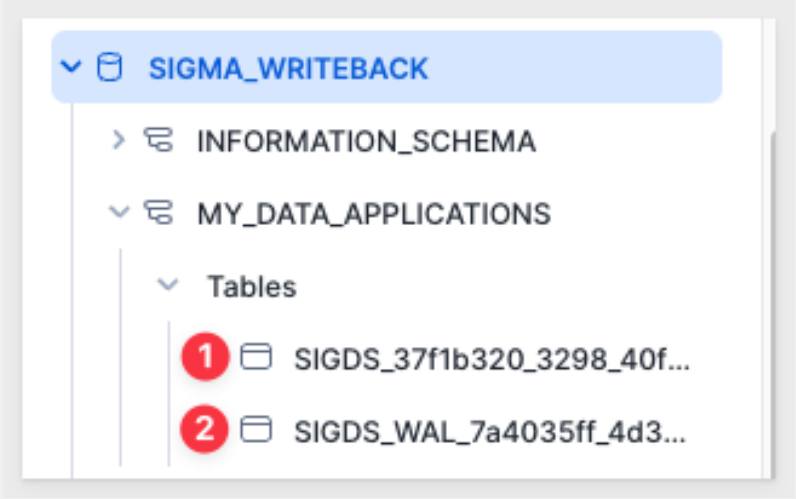
After renaming the input table to "**Shipping Status Updates,**" we have our solution ready for the shipping department to start using.



Behind the scenes

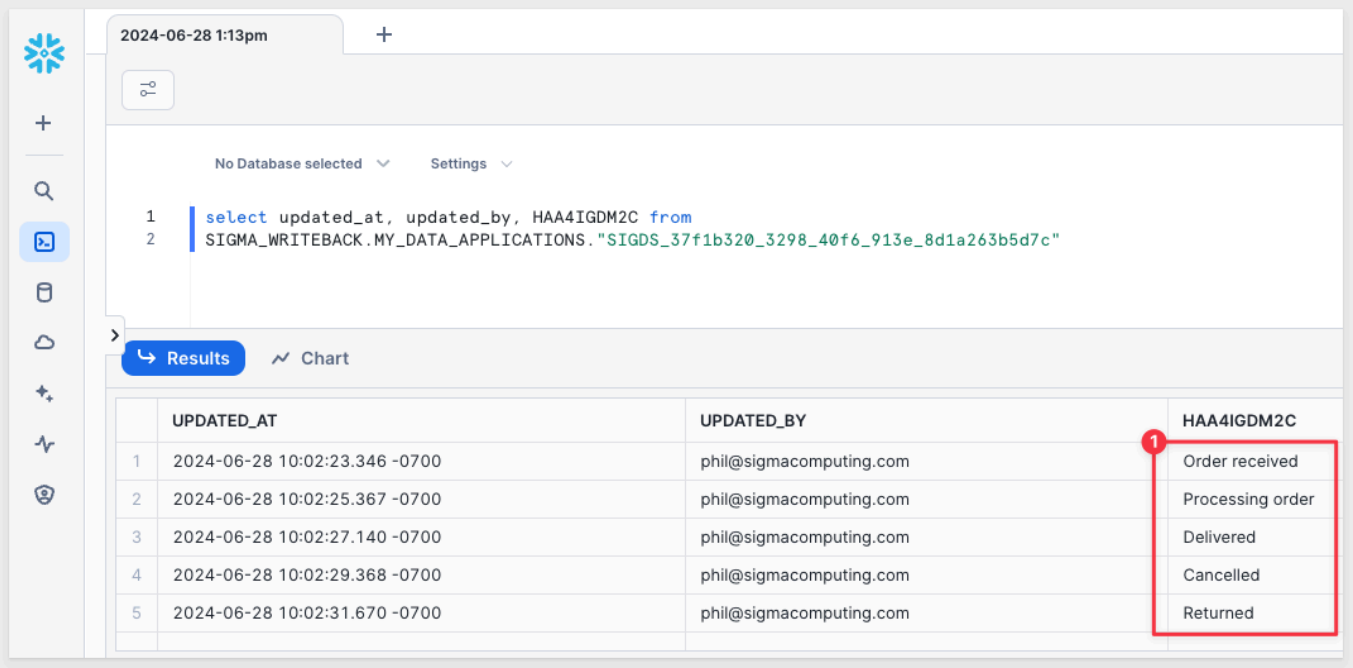
We want to verify that Sigma Write-back is working.

In the Snowflake interface, after refreshing the "**Databases**" data, we can see that Sigma has indeed created two tables in the location configured in the Sigma connection we set up earlier.



The first table (#1 in the above image) hold the input table columns that are not part of the original source data (ie: the Status column) and the tracking columns, along with other reference columns.

We can use Snowflake to query table #1 and see the five status changes we made in Sigma earlier.



The second table contains metadata about every change made to the input table, structurally.

Since this data is stored in the customer's warehouse, it can be joined with other tables as needed, bringing back user-provided data for use in other ways too.

Inevitable scope creep

Like most software projects, the scope of requirements often changes during development or just after user acceptance testing is underway. Fortunately, with Sigma, that is not a problem at all.

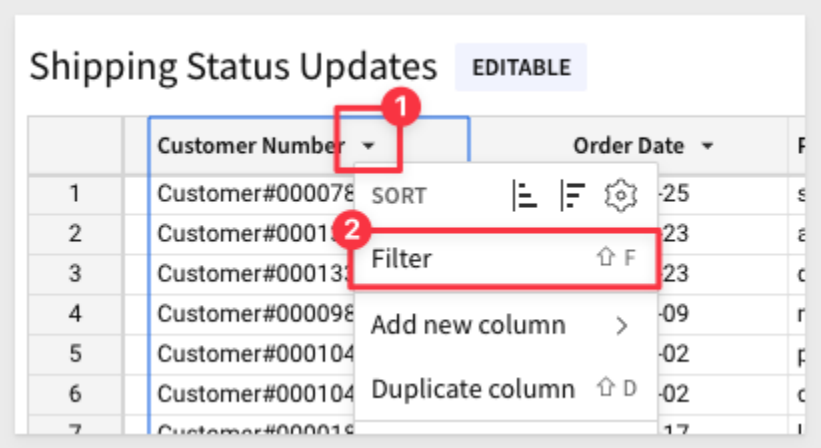
The shipping department has asked for two things, and if they can get them, they will fully accept the solution.

1. They need a way to easily find an order by customer name.

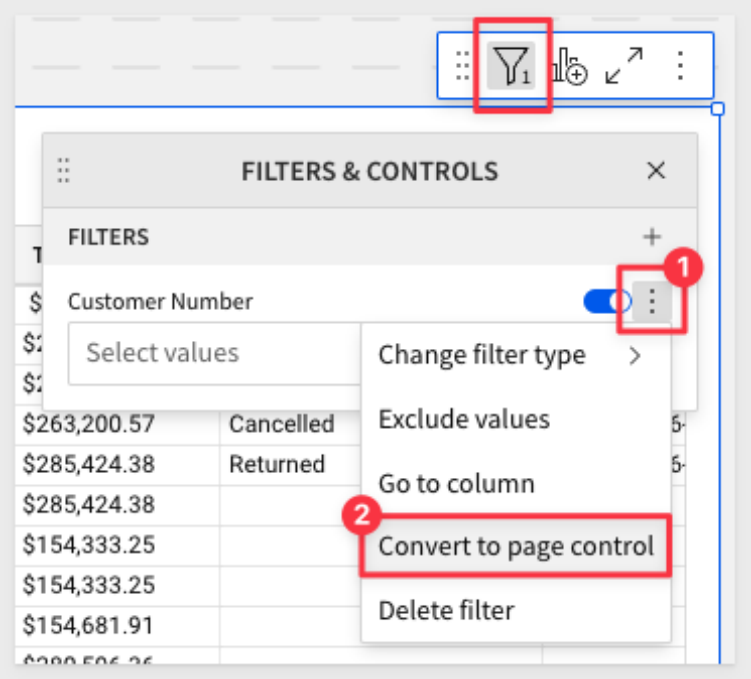
2. The department head wants to see how they are performing. They want to know the turnaround time from invoice date to when the line item is marked "**With carrier.**"

Filters and controls, Simplified

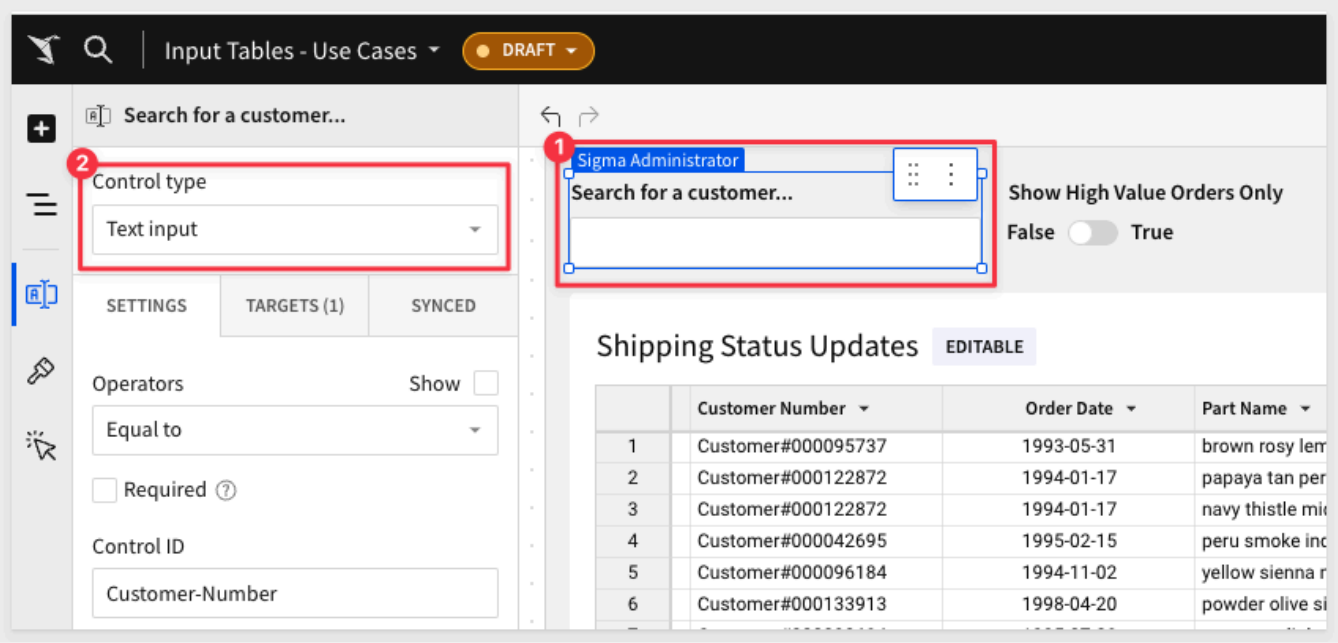
Let's tackle the first request, adding a filter control. From the "**Customer Number**" column, we select "**Filter**."



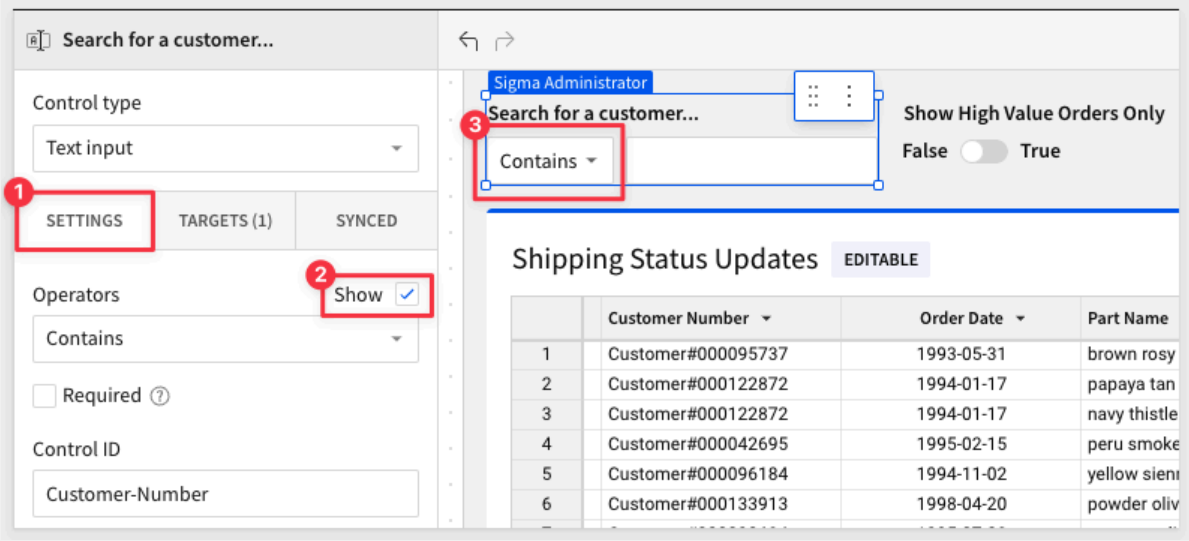
This opens the table's "**FILTERS & CONTROLS**" dialogue. Some users find that the filters icon on the table is not obvious enough and want something more noticeable, so we convert the new filter to a page control.



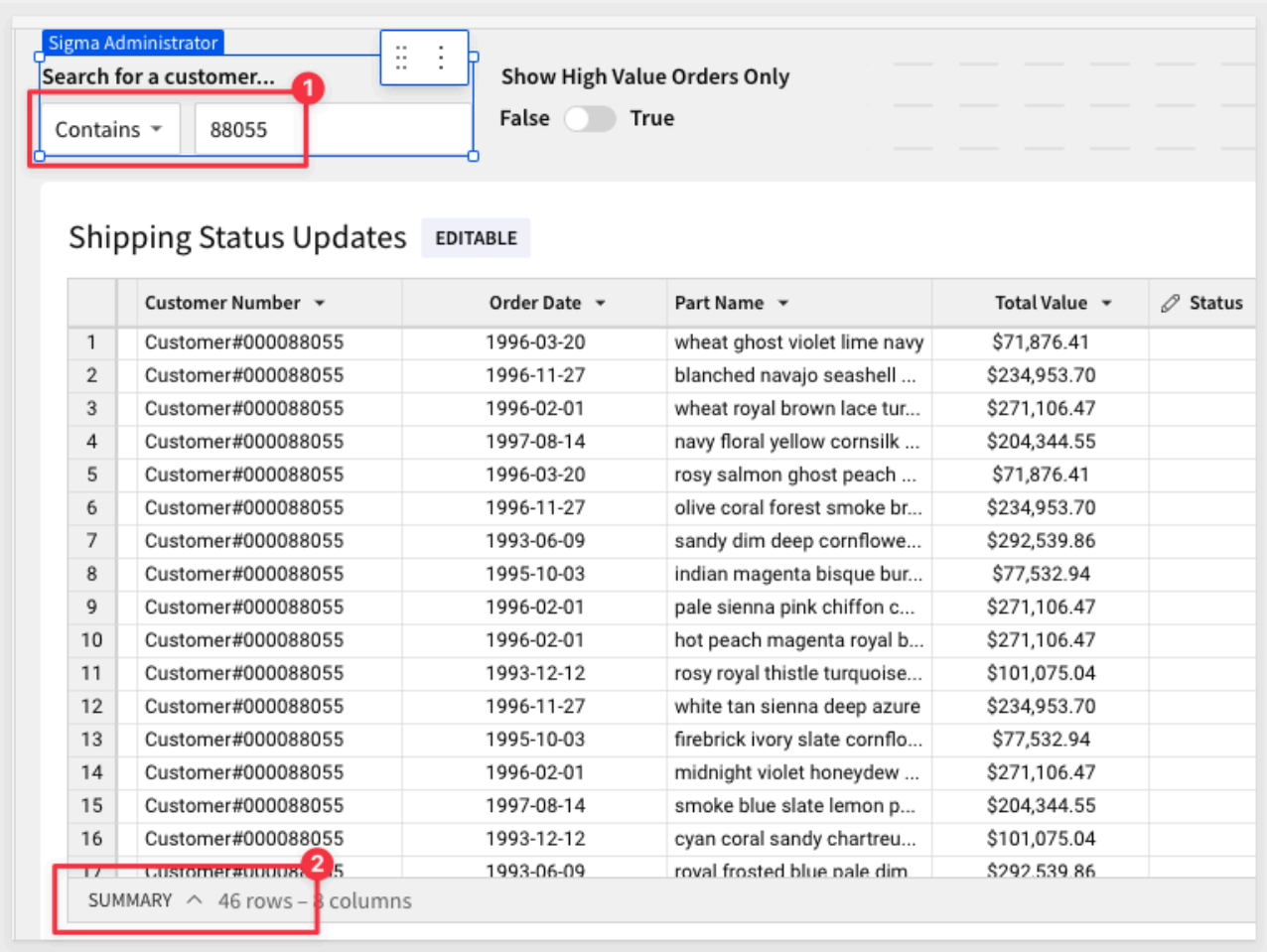
Now we have a page control, but quickly realize that a filter against 6M records is somewhat cumbersome to use, so we can change the control to a "**Text Input**" instead.



Let's give shipping some search power by enabling the user to select from all the available search operators.



Now, if we search for "**88055**," we get only 46 rows. We could refine this further with more filters, controls, and sortation, but you get just how easily Sigma handled this requirement change.



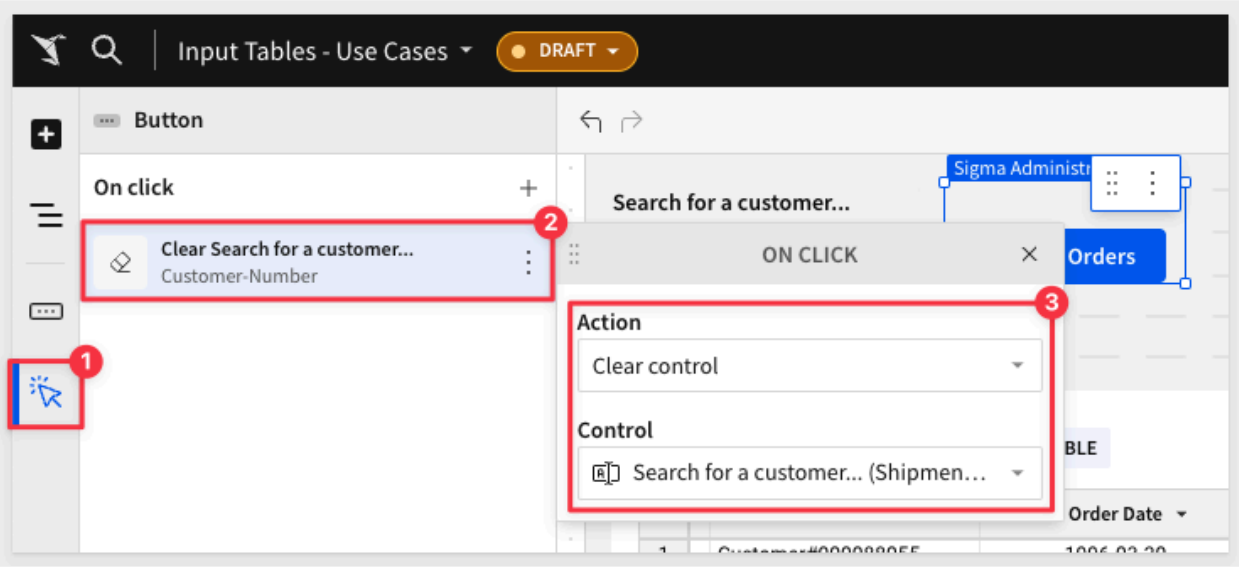
However, shipping has reported that users had difficulty clearing the search control after a search.

Actions

Sigma supports interactivity through a feature we call actions. In this case, we can simply add a button to the page and set its action to clear the filter control's value. Again, a simple example to make the point about how easy it is.

Add a "**UI element**" > "**Button**".

Configure its action.



Now, when users search, they can reset the input table with a simple and obvious button click.

Complex calculations, simplified

Restating the second request from shipping: “We want to know the turnaround time from order date to when the line item is marked with carrier”.

Note: It is important to understand that new columns are not written back on top of the original source data, but rather they are calculated on the fly, in the user's browser. Sigma alpha query handles this operation automatically.

However, input table columns that are not part of the source data are written back to the warehouse upon user save, via Sigma Write-back.

With that in mind, it opens up some interesting possibilities that are enabled by Write-back. Input table data that is written back to the warehouse could potentially be used by other applications too, since they also contain columns that can be joined on (e.g., Order ID, Customer Number, etc.) opening up a world of possibilities.

Now, you may have noticed that the "**Order Date**" column's dates are pretty old; **that is okay**. We will use that to make our point about complex calculations.

Add a new calculation column and name it "**Turnaround Time**".

Set its formula to:

```
If([Status] = "Delivered", If(IsNull([Last updated at]), "",
Concat(Text(DateDiff("year", [Order Date], [Last updated at])), " years, ",
Text(DateDiff("month", [Order Date], [Last updated at]) % 12), " months, ",
Text(DateDiff("day", [Order Date], [Last updated at]) % 30), " days")), "")
```

For those not used to these types of functions, which are similar to what spreadsheet users use all the time, here is an explanation.

Here's the step-by-step breakdown:

1: Check if the "Status" column is "Delivered":

- IF([Status] = "Delivered", … , "")
- If true, it proceeds to the next condition.
- if false, it returns an empty string.

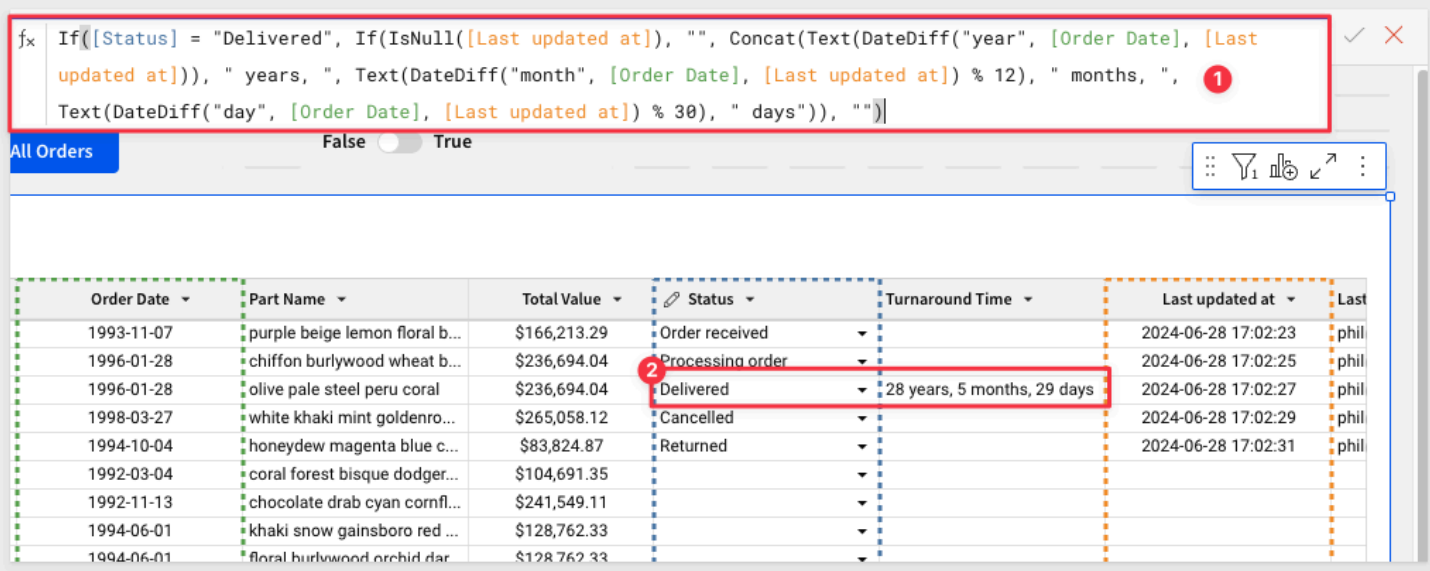
2: Check if "Last updated at" is null:

- IF(IsNull([Last updated at]), "", …)
- If true, it returns an empty string.
- If false, it proceeds to calculate the date difference and concatenate the result.

3: Calculate the date differences and concatenate the results:

- Concat(Text(DateDiff("year", [Order Date], [Last updated at])), " years, ", Text(DateDiff("month", [Order Date], [Last updated at]) % 12), " months, ", Text(DateDiff("day", [Order Date], [Last updated at]) % 30), " days")

The results are now easily readable by users.



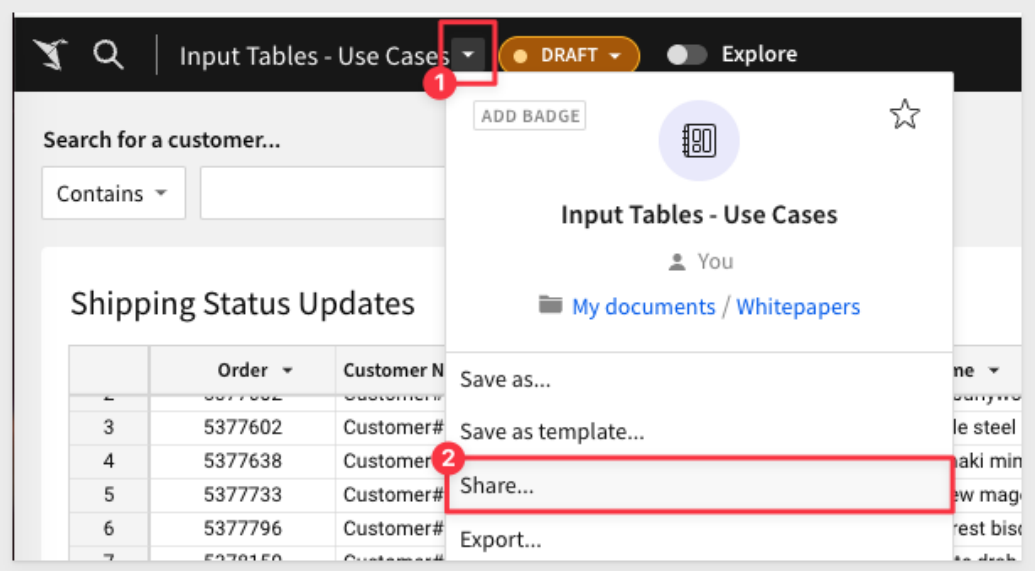
Sigma supports a wide range of functions. There is a list of the popular ones along with details on all the available functions as well.

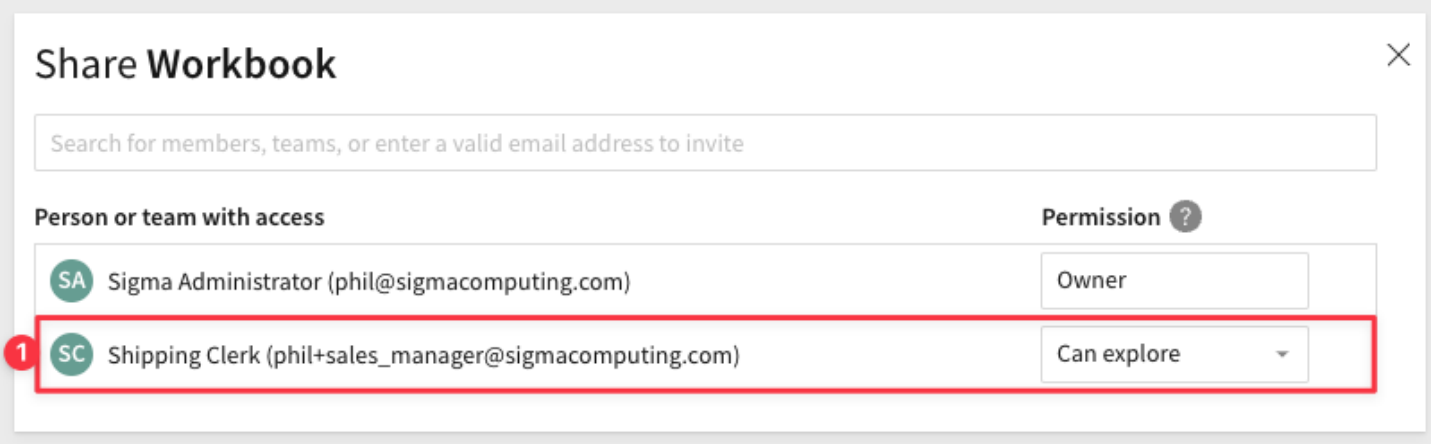
With respect to corporate governance, Sigma also supports centralizing calculations into reusable objects that can be easily added to workbooks. This ensures that calculations are standardized and consistently applied. Sigma calls these metrics; more reading here.

Clean up and sharing

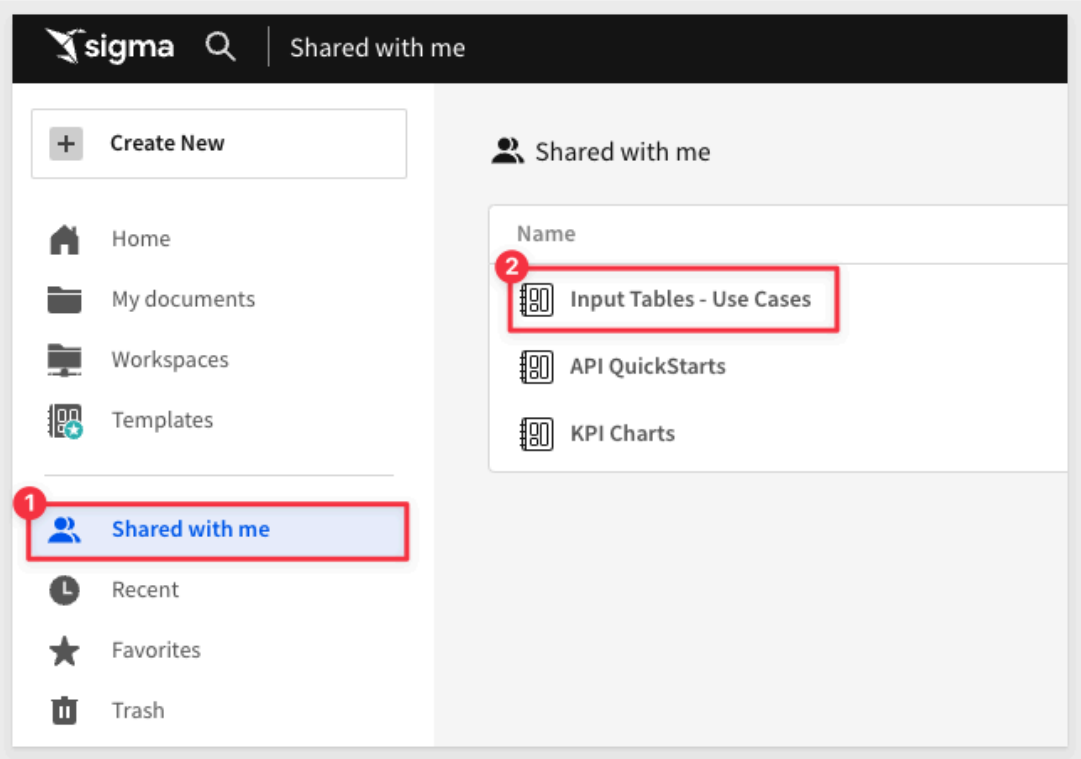
To wrap up the new data application, we added a few more page controls, a title, dynamic text to show pending shipments and an image.

The last step is to share this with the shipping team.

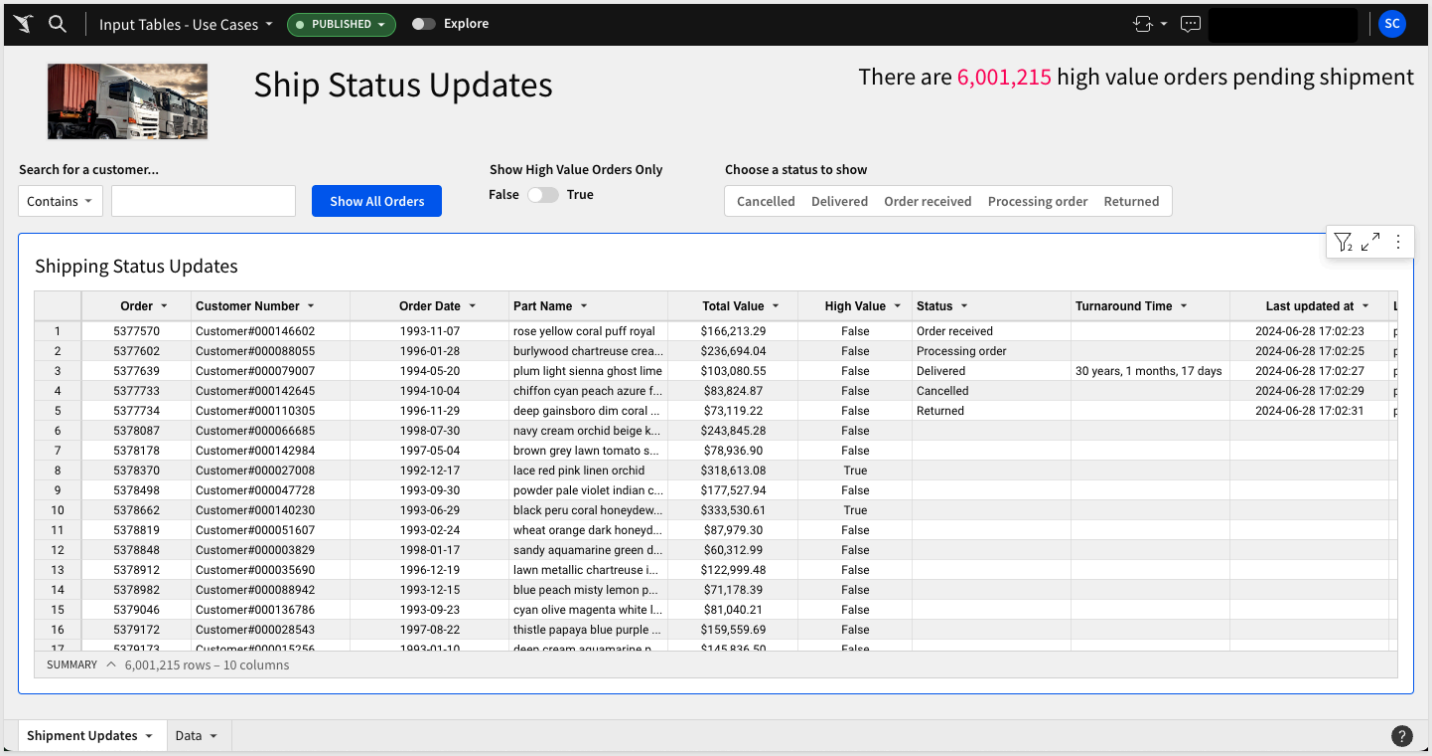




After logging into Sigma as the shipping clerk, they are able to see the "**Input Tables - Use Cases**" workbook in "**Shared with Me**" and start working straight away.



Now they can see and update the final version.



Conclusion

Unlock the potential of your data with Sigma. Traditional BI models offer static pages, but Sigma liberates cloud warehouse data. With Input Tables, capture insights previously locked in users' minds—no need for additional applications.

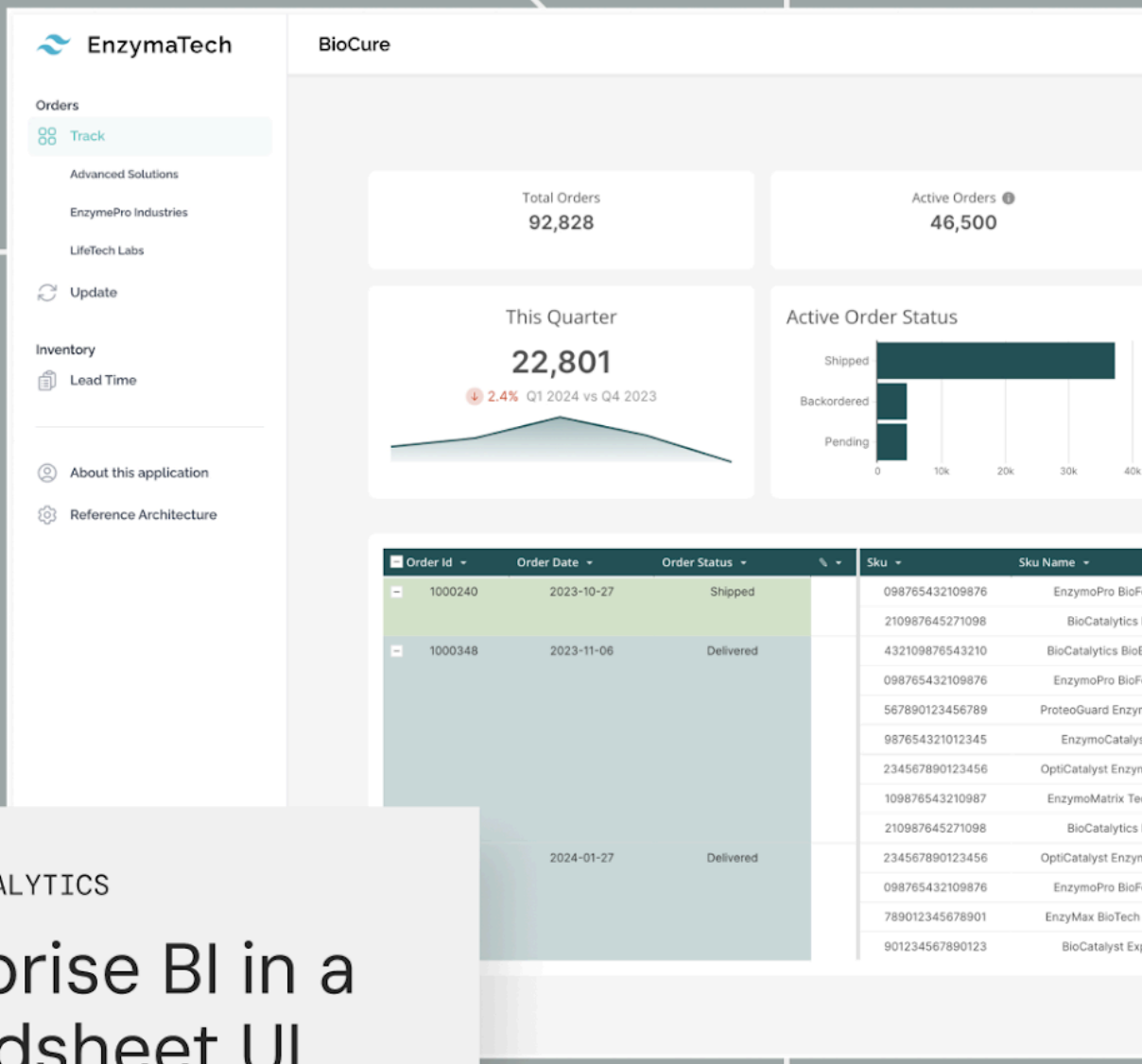
Imagine the possibilities!

Explore more Input Table use cases in our QuickStart:

- Forecast Adjustment
- Rapid Data Prototyping
- Sales Territory Planning
- Embedding Input Tables
- Data Collection

TO LEARN MORE ABOUT SIGMA INPUT TABLES, SEE HERE.

sigma



NEXT GEN ANALYTICS

Enterprise BI in a Spreadsheet UI

Run granular business intelligence on billions of records without compromising speed or security–all from an intuitive interface anyone can use.

LEARN MORE FROM SIGMACOMPUTING.COM