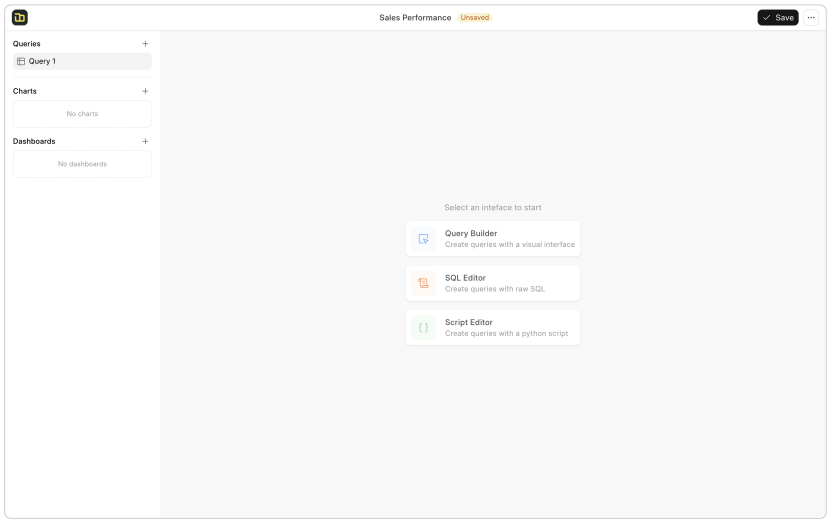
Create Your First Dashboard

Edit

In this guide, we'll create a comprehensive sales performance dashboard using the Demo Data that comes with Frappe Insights. We'll walk through creating a query, building multiple charts, and combining them into an interactive dashboard.

1. Create a Workbook

1. On the Workbook list page, click **+ New Workbook**
2. Name it "Sales Performance"
3. Click on "Query Builder"



2. Create the Base Query

Let's create a query that combines data from multiple tables. When the query builder opens, select `orders` as your source table. Add the following operations in sequence:

Join Tables

1. Click **Add Operation** → **Join Table**
   - Select `orderitems`
   - Join Type: Left Join
   - Join On: order_id = order_id
   - Select columns: price, freight_value, product_id
2. Click **Add Operation** → **Join Table**
   - Select `products`
   - Join Type: Left Join
   - Join On: product_id = product_id
   - Select columns: product_category_name
3. Click **Add Operation** → **Join Table**
   - Select `customers`
   - Join Type: Left Join
   - Join On: customer_id = customer_id
   - Select columns: customer_state

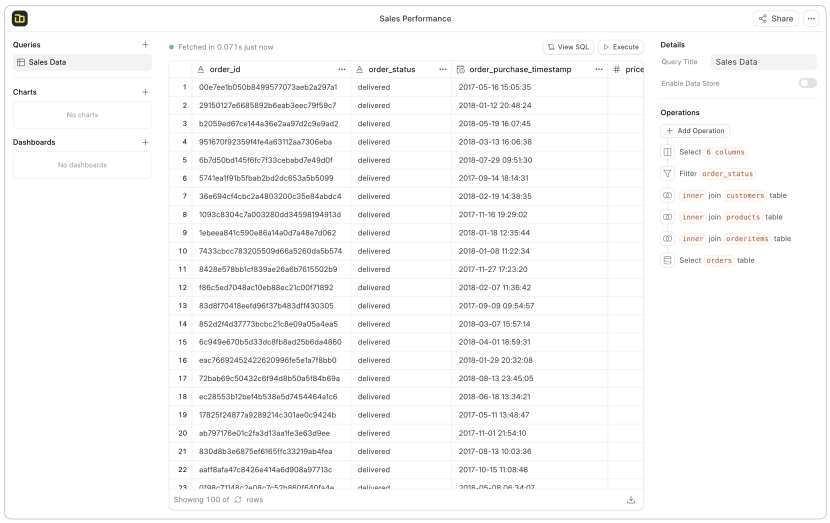
Filter and Select

4. Click **Add Operation** → **Filter Rows**
   - Column: order_status
   - Condition: Equals
   - Value: "delivered"
5. Click **Add Operation** → **Choose Columns**
   - order_id
   - order_item_id

On this page

- order_status
- order_purchase_timestamp
- price
- freight_value
- product_category_name
- customer_state

6. Name your query "Sales Data"



3. Create Charts

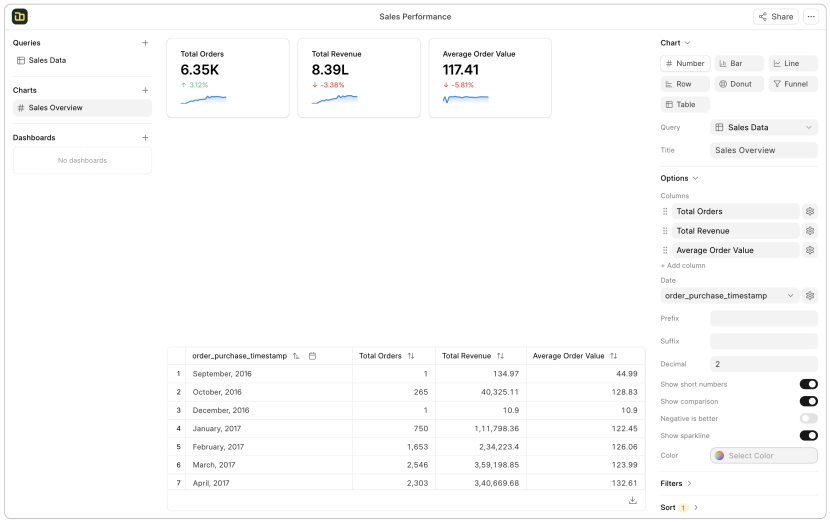
Now let's create various charts to visualize our sales data:

Sales Overview

- Click **+ New Chart**
- Chart Type: Number
- Title: "Sales Overview"
- In the Columns section:

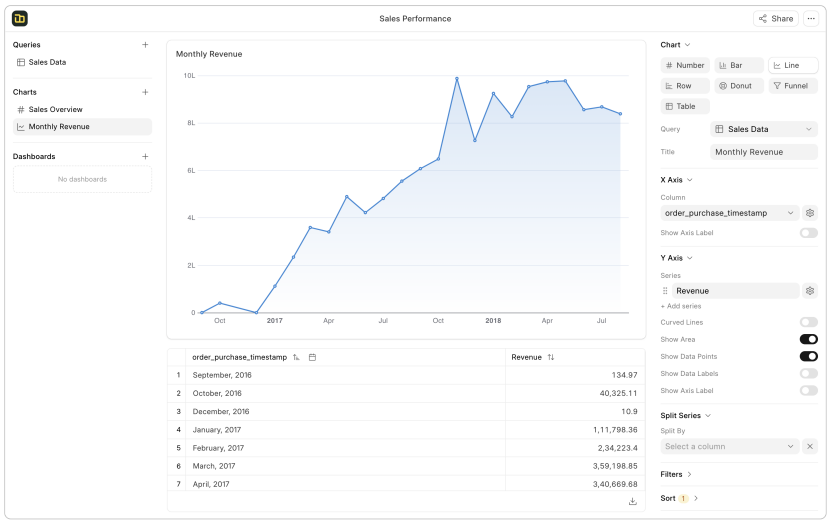
1. Click **+ Add Column**
   - Function: Count Distinct
   - Column: order_id
   - Click the gear icon to set label as "Total Orders"
2. Click **+ Add Column**
   - Function: Sum
   - Column: price
   - Click the gear icon to set label as "Total Revenue"
3. Click **+ Add Column**
   - Function: Average
   - Column: price
   - Click the gear icon to set label as "Average Order Value"

- Select "order_purchase_timestamp" as the Date Column
- Sort: order_purchase_timestamp (Ascending)
- Enable "Show Comparison"
- Enable "Show Sparkline"



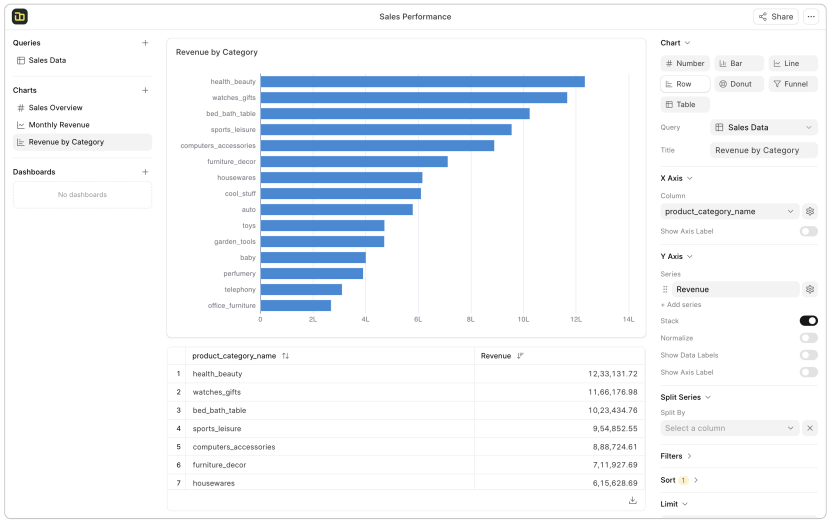
Revenue by Month

- Click **+ New Chart**
- Chart Type: Line
- Title: "Monthly Revenue"
- X-axis: `order_purchase_timestamp`
- Y-axis:
- Function: Sum of
- Column: price
- Click the gear icon to set label as "Revenue"



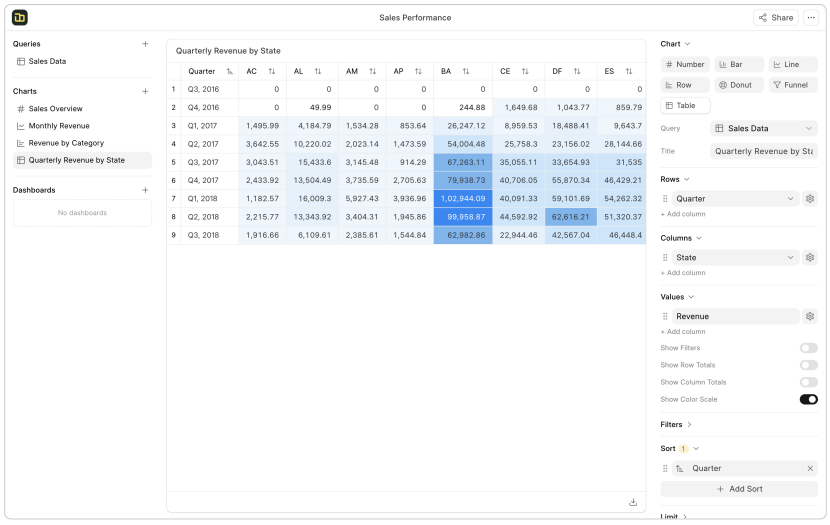
Revenue by Product Category

- Click **+ New Chart**
- Chart Type: Row
- Title: "Revenue by Category"
- X-axis: product_category_name
- Y-axis:
- Function: Sum of
- Column: price
- Click the gear icon to set label as "Revenue"
- Sort: Revenue (Descending)



Quaterly Revenue by State

- Click **+ New Chart**
- Chart Type: Table
- Title: "Quarterly Revenue by State"
- Rows:
- Column: order_purchase_timestamp
- Click the gear icon to set label as "Quarter" & granularity as "Quarter"
- Columns:
- Column: customer_state
- Click the gear icon to set label as "State"
- Values:
- Function: Sum of
- Column: price
- Click the gear icon to set label as "Revenue"
- Sort: Quarter (Ascending)
- Enable "Show Color Scale"

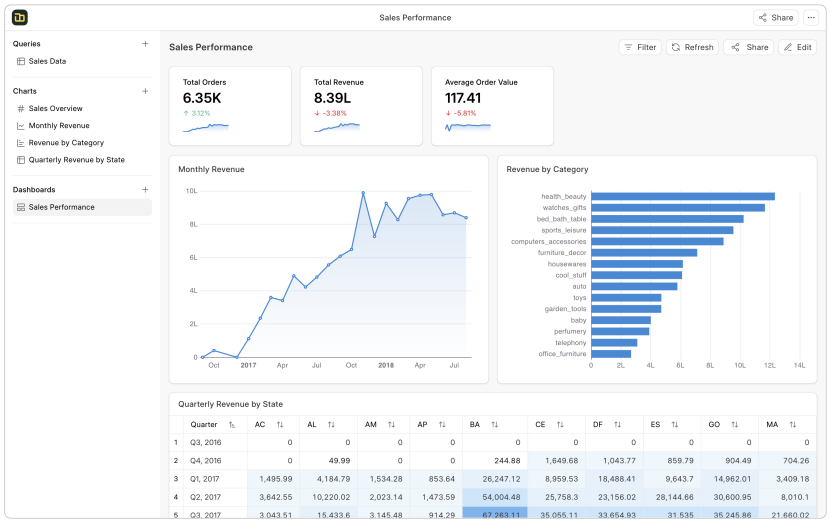


4. Build the Dashboard

Let's combine all charts into an interactive dashboard:

1. Click **+ New Dashboard**
2. Set title to "Sales Performance Dashboard"
3. Drag and drop the charts from the left panel to the dashboard:
    - First row: Add "Sales Overview" chart
    - Second row: Add "Monthly Revenue" & "Revenue by Category" charts
    - Third row: Add "Quarterly Revenue by State" chart

4. Apply dashboard filters:
    - Click on the **Filter** button on the top right
    - Click on **Add Filter** in the filter dialog
    - Select `product_category_name` as the filter column
    - Select `health_beauty` as the filter value
    - Click **Apply Filter**



This sample dashboard gives a complete view of the sales performance, from high-level metrics to detailed breakdowns by category and location. The filters allow you to drill down into specific segments and analyze the data further.

PREVIOUS PAGE
Complete Setup

NEXT PAGE
Overview

Last updated 3 months ago

Was this helpful?

Overview

Edit

What are Data Sources?

Data sources are your database connections. They act as a bridge between Insights and your database, allowing you to analyze your data seamlessly.

Once connected, you can:

- Browse your tables
- Create queries to analyze data
- Build charts and dashboards
- Share insights with your team

Supported Databases and Data Sources:

- Site DB (Auto-configured)
- CSV / Excel Upload
- MariaDB / MySQL
- PostgreSQL
- DuckDB
- ClickHouse

Not sure which database you use? Ask your IT team or database administrator.

PREVIOUS PAGE
Create Your First Dashboard

NEXT PAGE
Add Data Source

Last updated 3 months ago

Was this helpful?

Add Data Source

Edit

Prerequisites

Before connecting a data source, make sure you have the following information (ask your database admin if needed): Field Description Host Server address (e.g. db.mycompany.com or localhost) Port Connection port (e.g. 3306 for MariaDB, 5432 for PostgreSQL) Database Name The database to connect to Username Login username Password Login password SSL (optional) Whether to use a secure connection

> *"Note: Make sure you have* `Insights Admin` *role in order to add a database as datasource"*

Quick Start: Adding a Data Source

Step 1: Access Data Sources

Click Data Sources in the sidebar.

Click + Add Data Source.

Choose your database type.

Step 2: Enter Connection Details

For example, connecting to MariaDB: Title: A friendly name (e.g. "Production Database")

Host: Server address (localhost, db.mycompany.com, etc.)

Port: Usually 3306 for MariaDB (prefilled)

Database Name: Exact database name

Username / Password: Database credentials

Use SSL: Check if required

Step 3: Add and Test

Click Add Data Source once details are entered.

Insights will save your connection, load tables, and make them available for querying.

CSV / Excel Upload

Ideal for quick data imports, smaller datasets, or testing Insights without a database. How to Upload: Click + Add Data Source → Upload CSV or Excel

Choose or drag-and-drop the file

Preview data (check columns and types)

Edit table name if needed

Click Import Table

Supported Formats:

- CSV (.csv)
- Excel (.xlsx)

File size limit: 50MB (check with admin)

Tips: First row should be headers Keep column names simple Use consistent date formats Remove empty rows at the end

Site Database (Auto-Configured)

What is this?

If you're running Insights on the same site as other Frappe backend apps, it automatically connects to your site's database. This gives you instant access to all your ERPNext, Frappe or custom app data!

Features:

Auto-configured, always active No credentials needed All DocTypes available as tables Relationship links preconfigured

Common Issues:

Connection refused:

Check host and port

Access denied:

Verify username and password

Unknown database:

Check spelling

PREVIOUS PAGE
Overview

NEXT PAGE
Manage Data Sources

Last updated 3 months ago

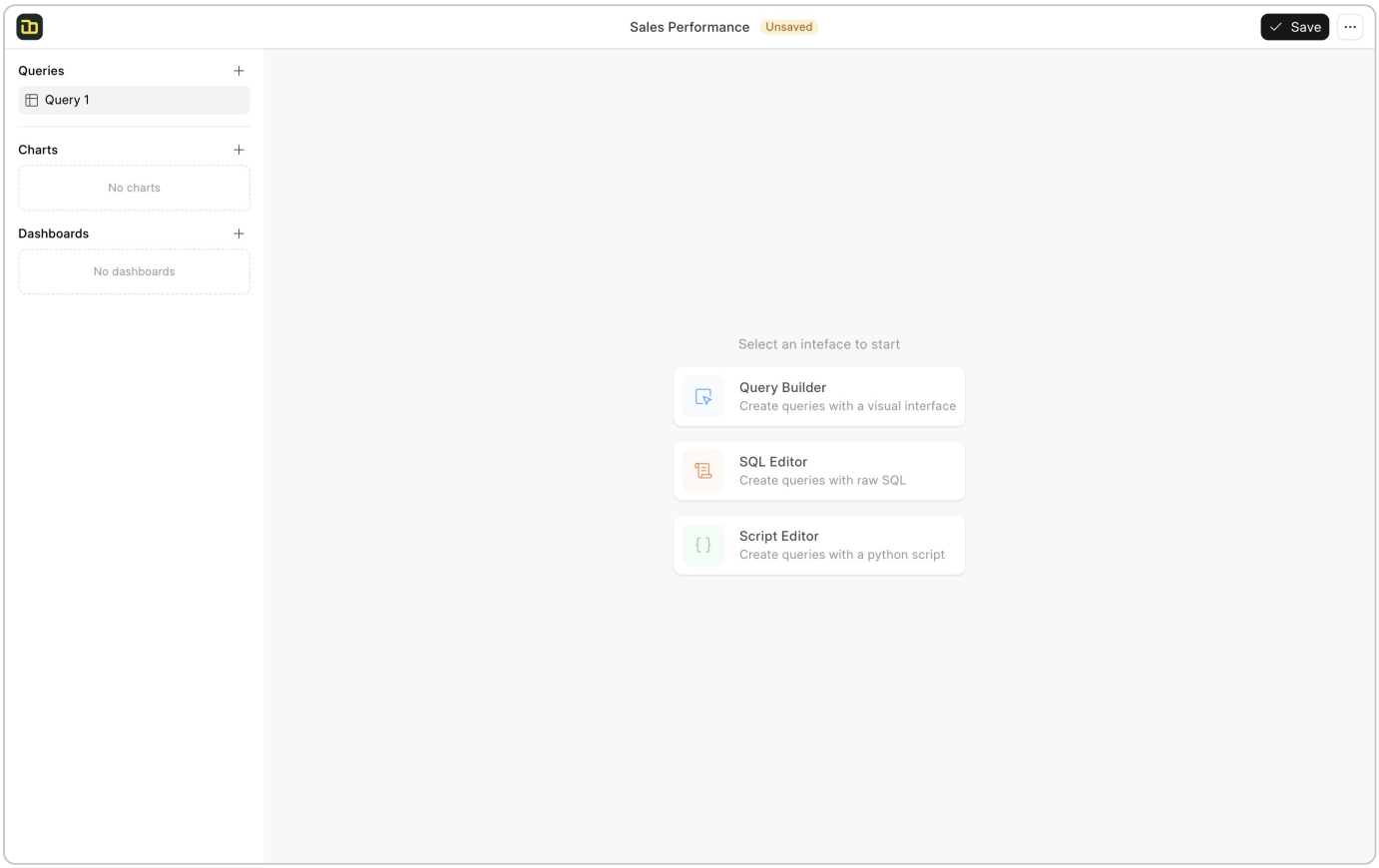
Create Your First Dashboard

Edit

In this guide, we'll create a comprehensive sales performance dashboard using the Demo Data that comes with Frappe Insights. We'll walk through creating a query, building multiple charts, and combining them into an interactive dashboard.

1. Create a Workbook

1. On the Workbook list page, click **+ New Workbook**
2. Name it "Sales Performance"
3. Click on "Query Builder"



2. Create the Base Query

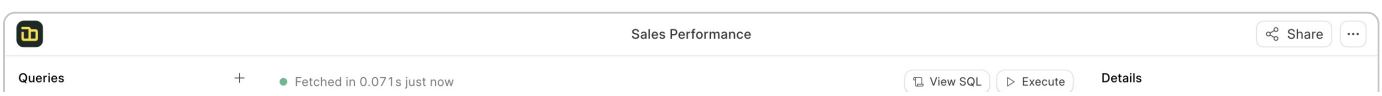
Let's create a query that combines data from multiple tables. When the query builder opens, select `orders` as your source table. Add the following operations in sequence:

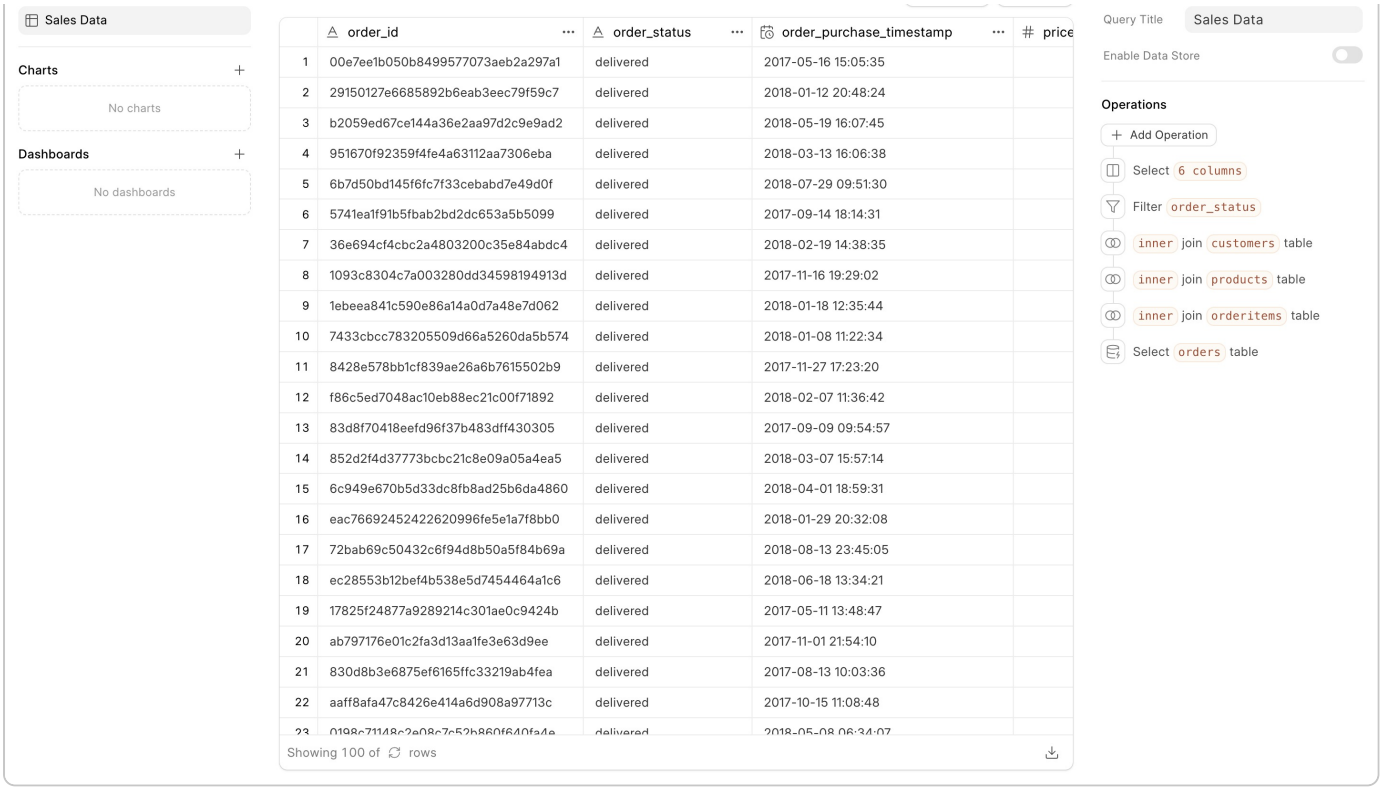
Join Tables

1. Click **Add Operation** → **Join Table**
   - Select `orderitems`
   - Join Type: Left Join
   - Join On: order_id = order_id
   - Select columns: price, freight_value, product_id
2. Click **Add Operation** → **Join Table**
   - Select `products`
   - Join Type: Left Join
   - Join On: product_id = product_id
   - Select columns: product_category_name
3. Click **Add Operation** → **Join Table**
   - Select `customers`
   - Join Type: Left Join
   - Join On: customer_id = customer_id
   - Select columns: customer_state

Filter and Select

4. Click **Add Operation** → **Filter Rows**
   - Column: order_status
   - Condition: Equals
   - Value: "delivered"
5. Click **Add Operation** → **Choose Columns**
   - order_id
   - order_item_id
   - order_status
   - order_purchase_timestamp
   - price
   - freight_value
   - product_category_name
   - customer_state
6. Name your query "Sales Data"





3. Create Charts

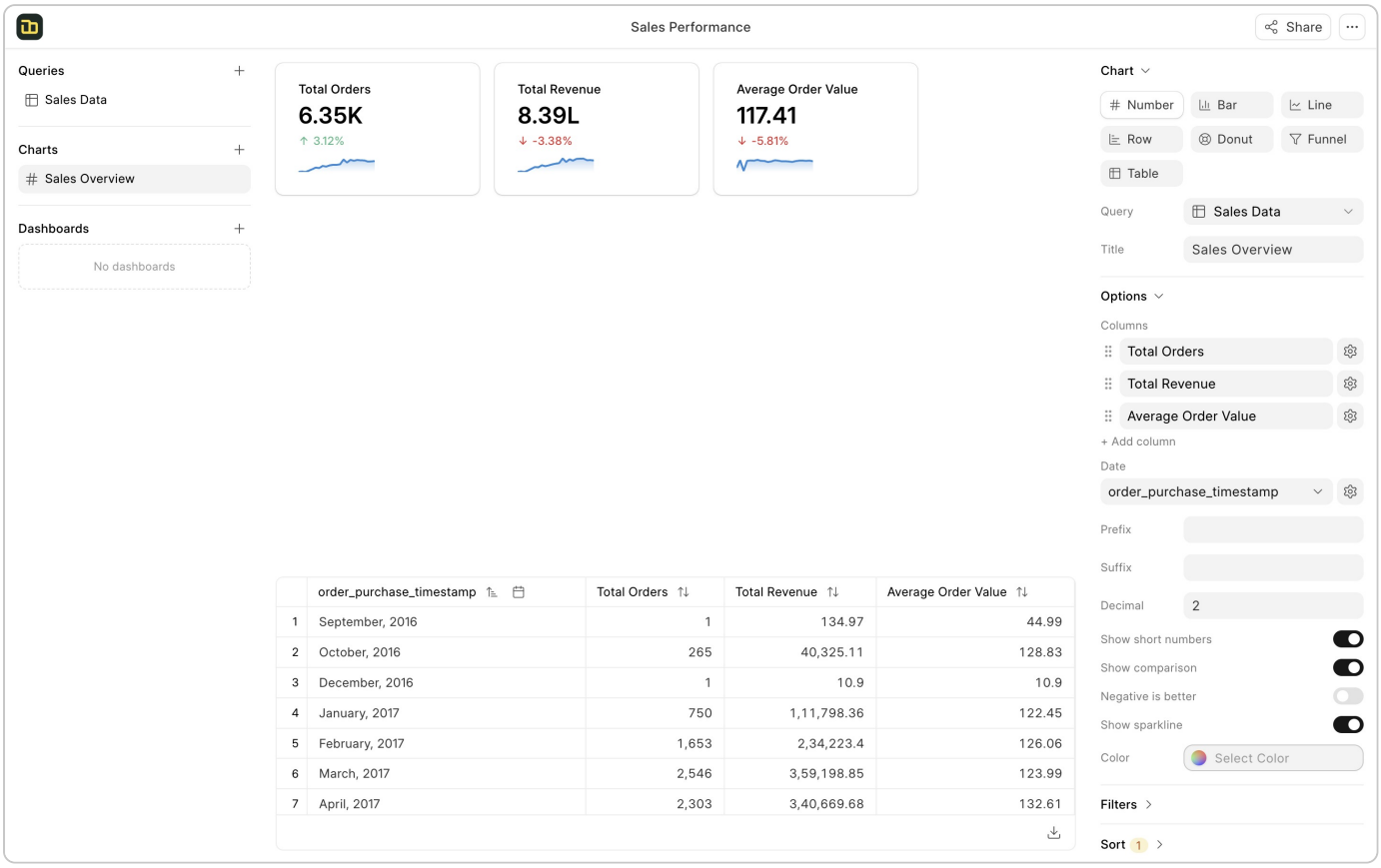
Now let's create various charts to visualize our sales data:

Sales Overview

- Click **+ New Chart**
- Chart Type: Number
- Title: "Sales Overview"
- In the Columns section:

1. Click **+ Add Column**
    - Function: Count Distinct
    - Column: order_id
    - Click the gear icon to set label as "Total Orders"
2. Click **+ Add Column**
    - Function: Sum
    - Column: price
    - Click the gear icon to set label as "Total Revenue"
3. Click **+ Add Column**
    - Function: Average
    - Column: price

    - Click the gear icon to set label as "Average Order Value"
- Select "order_purchase_timestamp" as the Date Column
- Sort: order_purchase_timestamp (Ascending)
- Enable "Show Comparison"
- Enable "Show Sparkline"

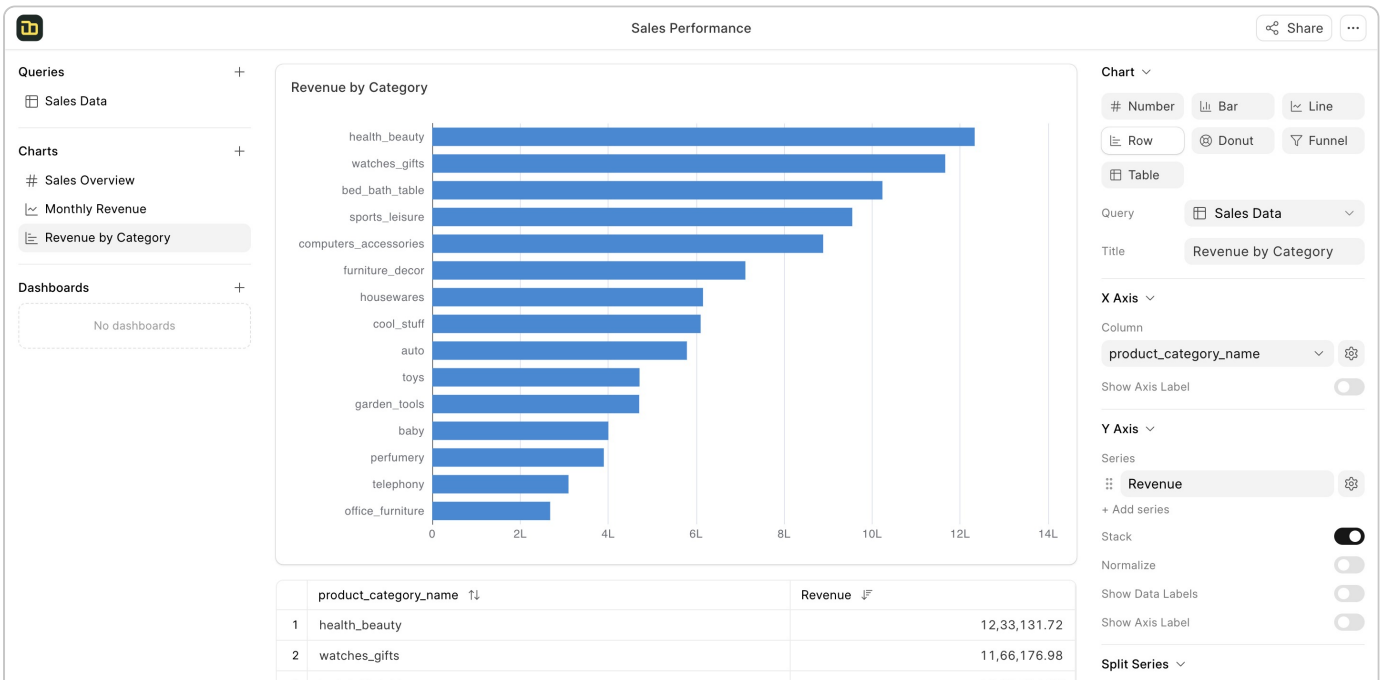


Revenue by Month

- Click **+ New Chart**
- Chart Type: Line
- Title: "Monthly Revenue"
- X-axis: `order_purchase_timestamp`
- Y-axis:
- Function: Sum of
- Column: price
- Click the gear icon to set label as "Revenue"

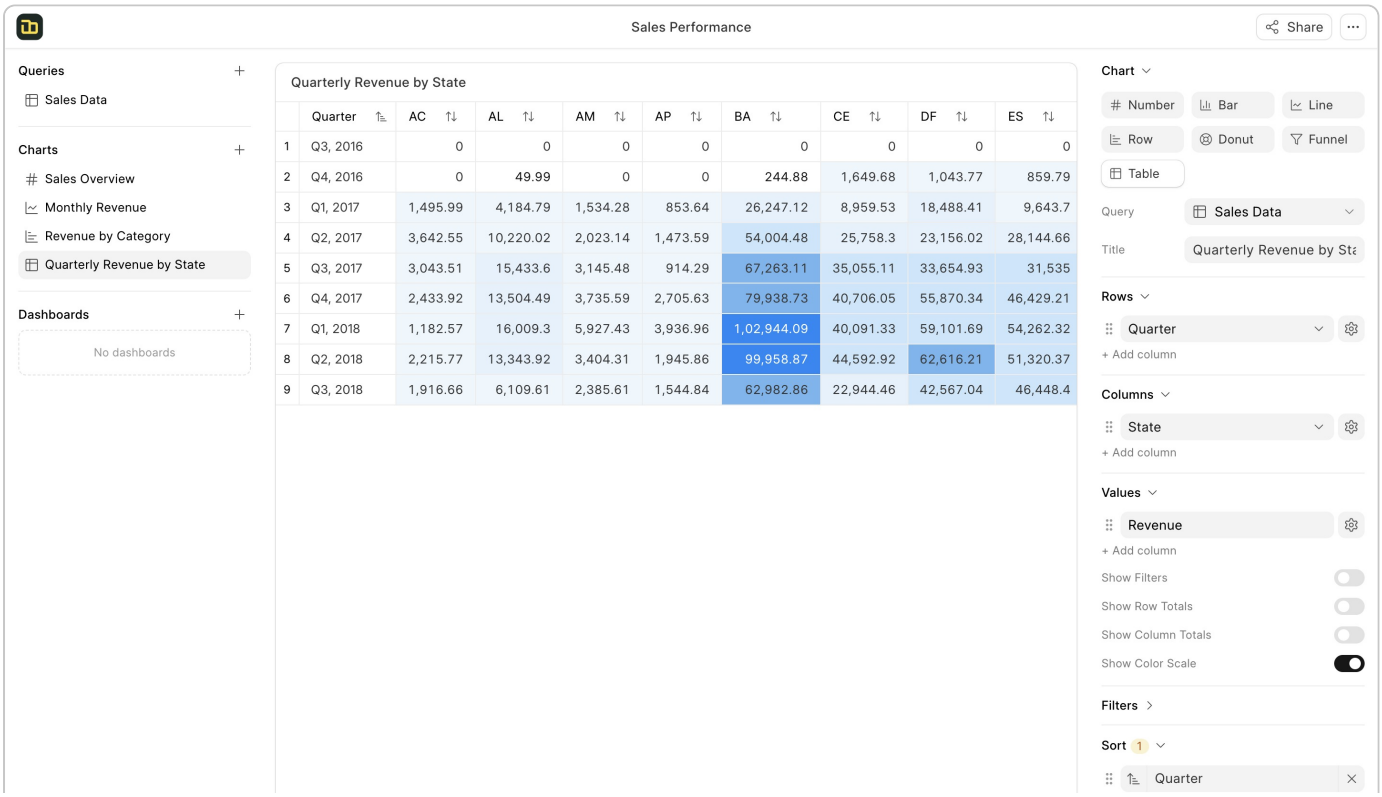
Revenue by Product Category

- Click **+ New Chart**
- Chart Type: Row
- Title: "Revenue by Category"
- X-axis: product_category_name
- Y-axis:
- Function: Sum of
- Column: price
- Click the gear icon to set label as "Revenue"
- Sort: Revenue (Descending)



Quaterly Revenue by State

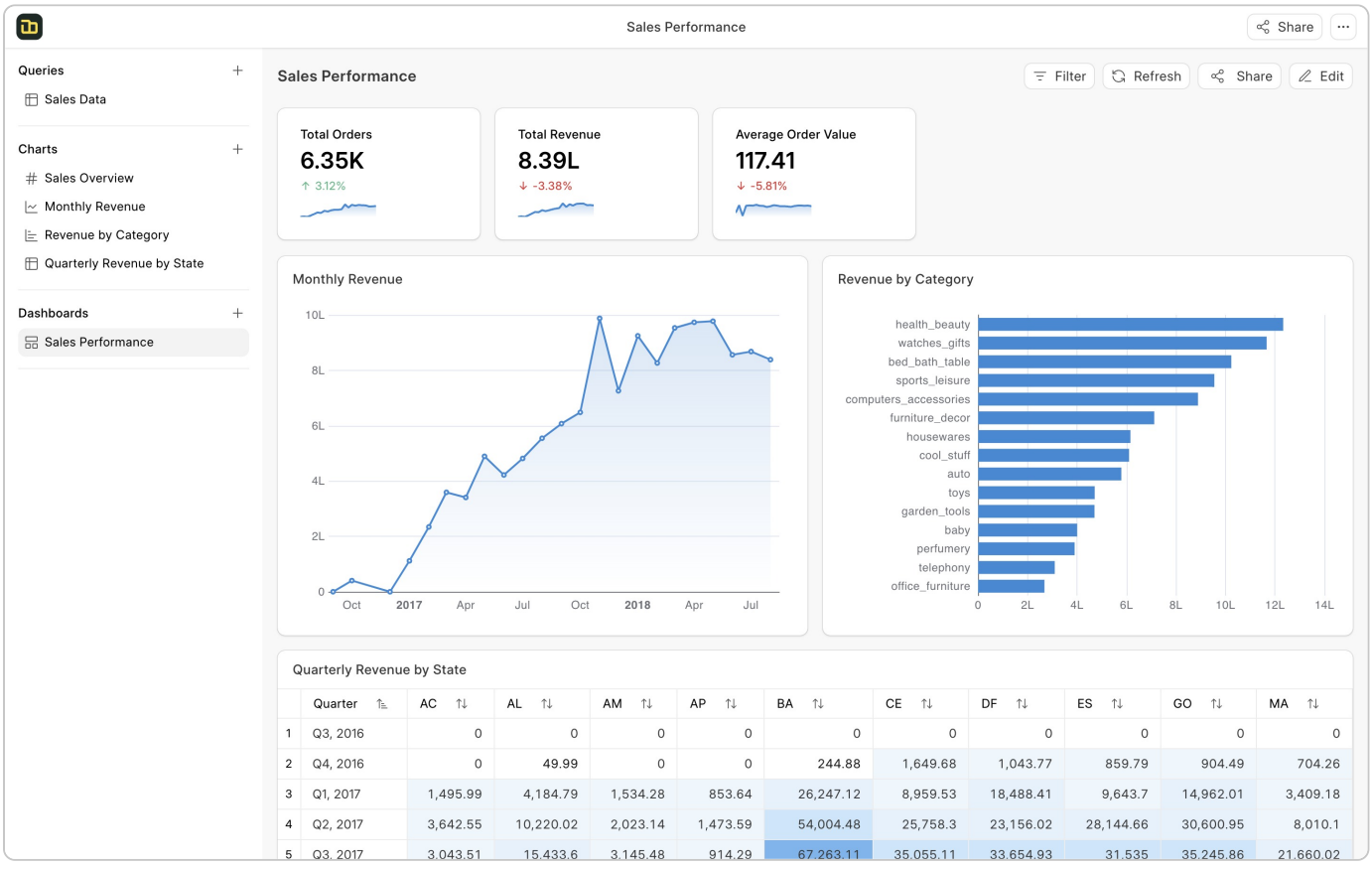
- Click **+ New Chart**
- Chart Type: Table
- Title: "Quarterly Revenue by State"
- Rows:
- Column: order_purchase_timestamp
- Click the gear icon to set label as "Quarter" & granularity as "Quarter"
- Columns:
- Column: customer_state
- Click the gear icon to set label as "State"
- Values:
- Function: Sum of
- Column: price
- Click the gear icon to set label as "Revenue"
- Sort: Quarter (Ascending)
- Enable "Show Color Scale"



4. Build the Dashboard

Let's combine all charts into an interactive dashboard:

1. Click **+ New Dashboard**
2. Set title to "Sales Performance Dashboard"
3. Drag and drop the charts from the left panel to the dashboard:
   - First row: Add "Sales Overview" chart
   - Second row: Add "Monthly Revenue" & "Revenue by Category" charts
   - Third row: Add "Quarterly Revenue by State" chart
4. Apply dashboard filters:
   - Click on the **Filter** button on the top right
   - Click on **Add Filter** in the filter dialog
   - Select `product_category_name` as the filter column
   - Select `health_beauty` as the filter value
   - Click **Apply Filter**



This sample dashboard gives a complete view of the sales performance, from high-level metrics to

detailed breakdowns by category and location. The filters allow you to drill down into specific segments and analyze the data further.

PREVIOUS PAGE
Complete Setup

NEXT PAGE
Overview

Last updated 3 months ago

Was this helpful?

Overview

Edit

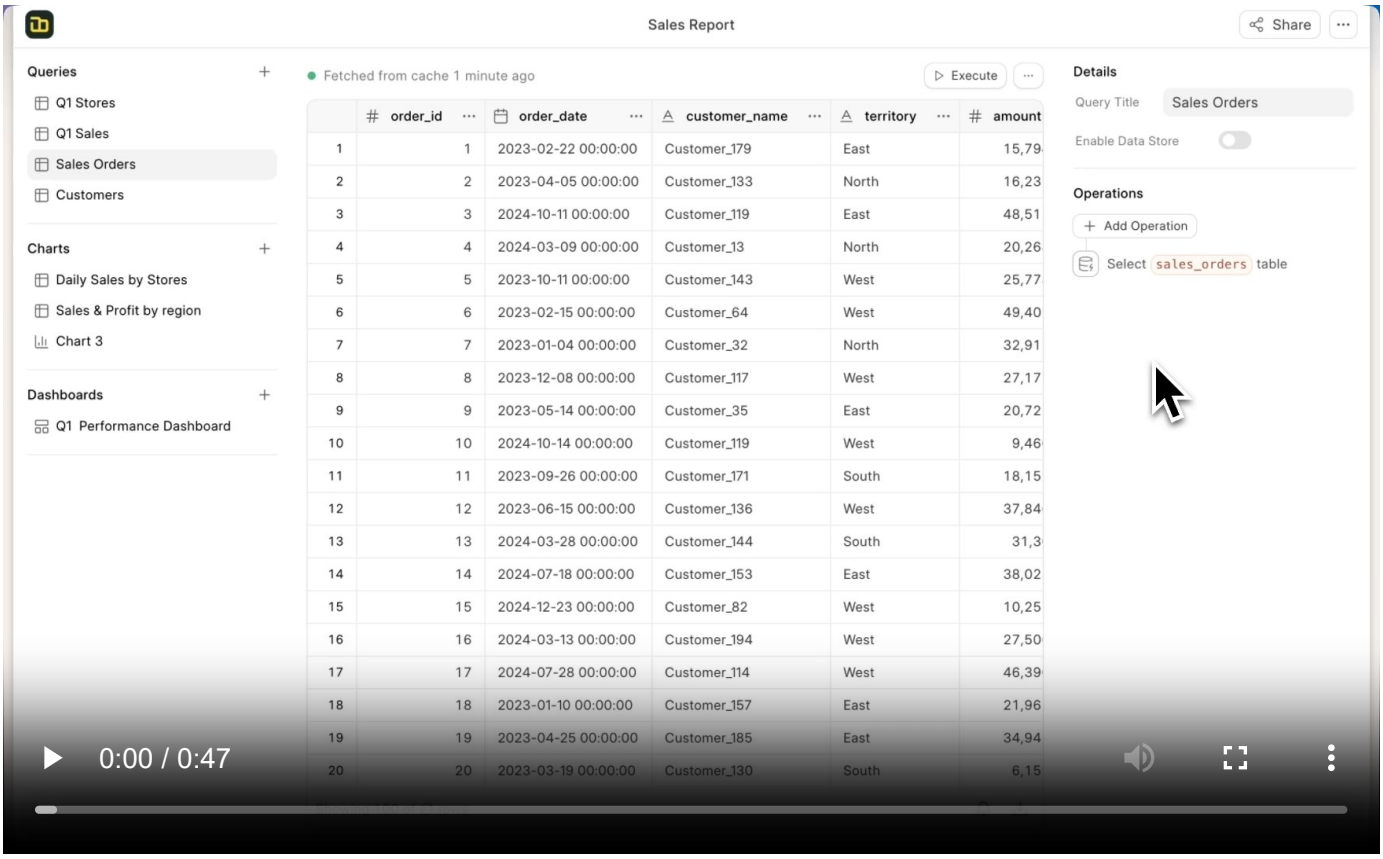
What are Queries?

Queries let you fetch and transform data from your databases. Think of a query as a recipe for getting the exact data you need. You start with a table then add steps to filter, calculate, and organize the data.

Insights offers three ways to build queries:

1. Query Builder Visual, no-code interface (recommended for most users)
2. SQL Editor Write SQL directly (for SQL experts)
3. Script Editor Python scripts for complex transformations

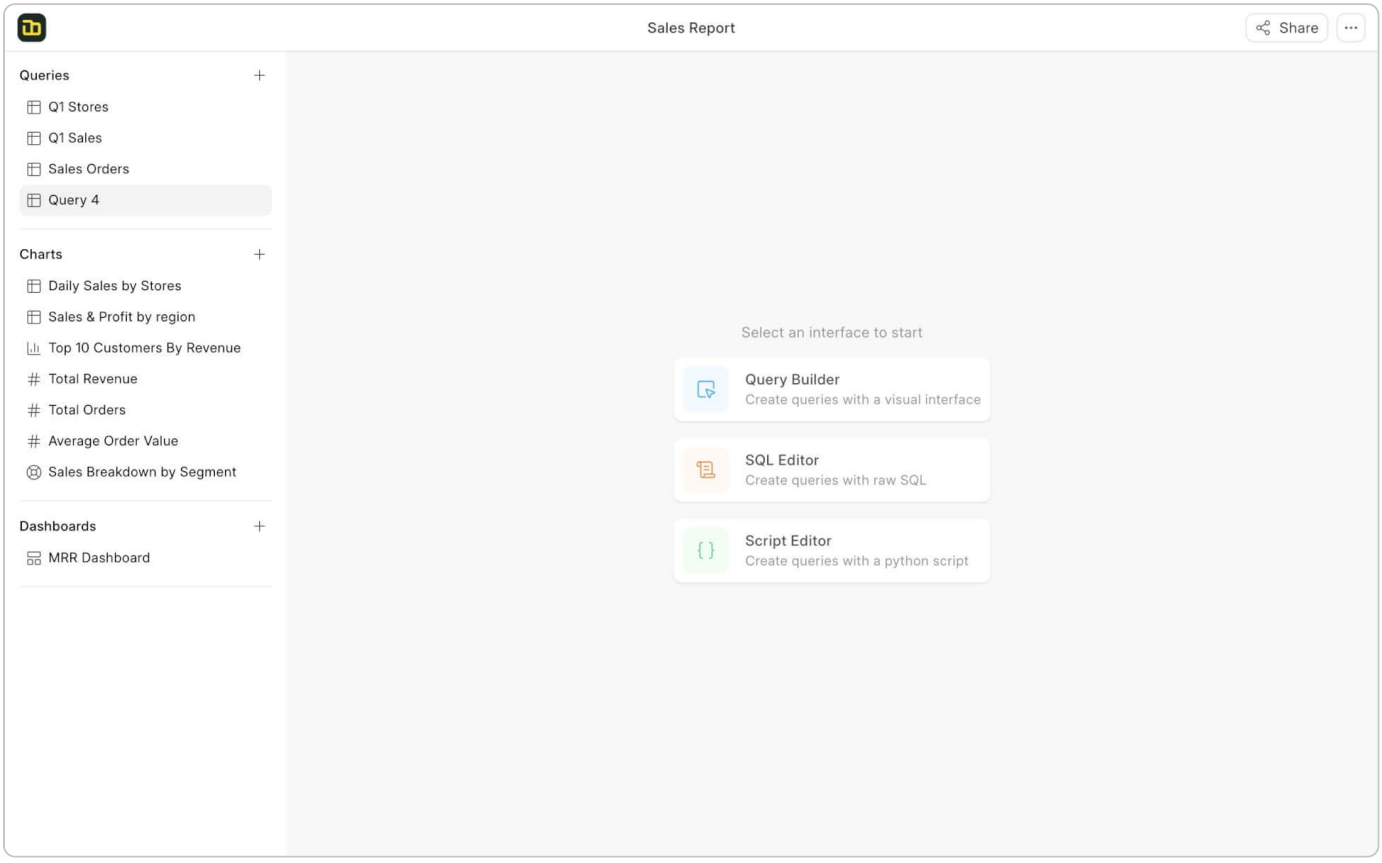
Creating Your First Query



Start a New Query

1. Open your workbook

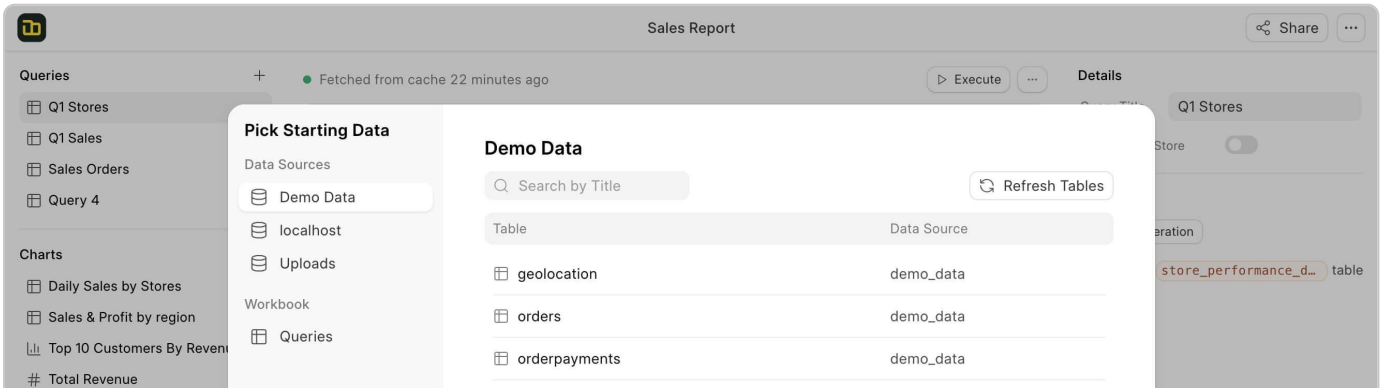
1. Click the Queries tab
2. Click New Query
3. Choose your query type:
    - Visual Query Builder -> Build visually
    - SQL Editor -> Write SQL
    - Script Editor -> Write Python

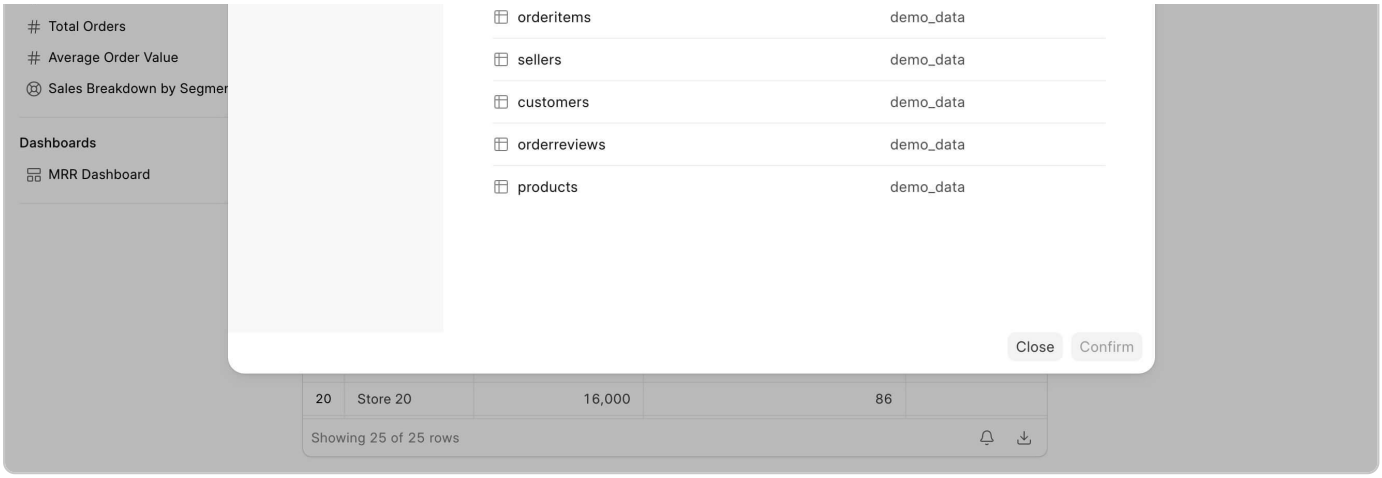


We'll focus on the Query Builder since it's the easiest to get started with.

2. Select Your Data Source

1. Click Select a table
2. Choose your database connection
3. Pick a table to start with





You can select a table from:

- Demo Data
- Site DB
- Uploads
- Other Queries in the WorkBook

Now you're ready to perform operations on the table

PREVIOUS PAGE
Manage Data Sources

NEXT PAGE
Operations

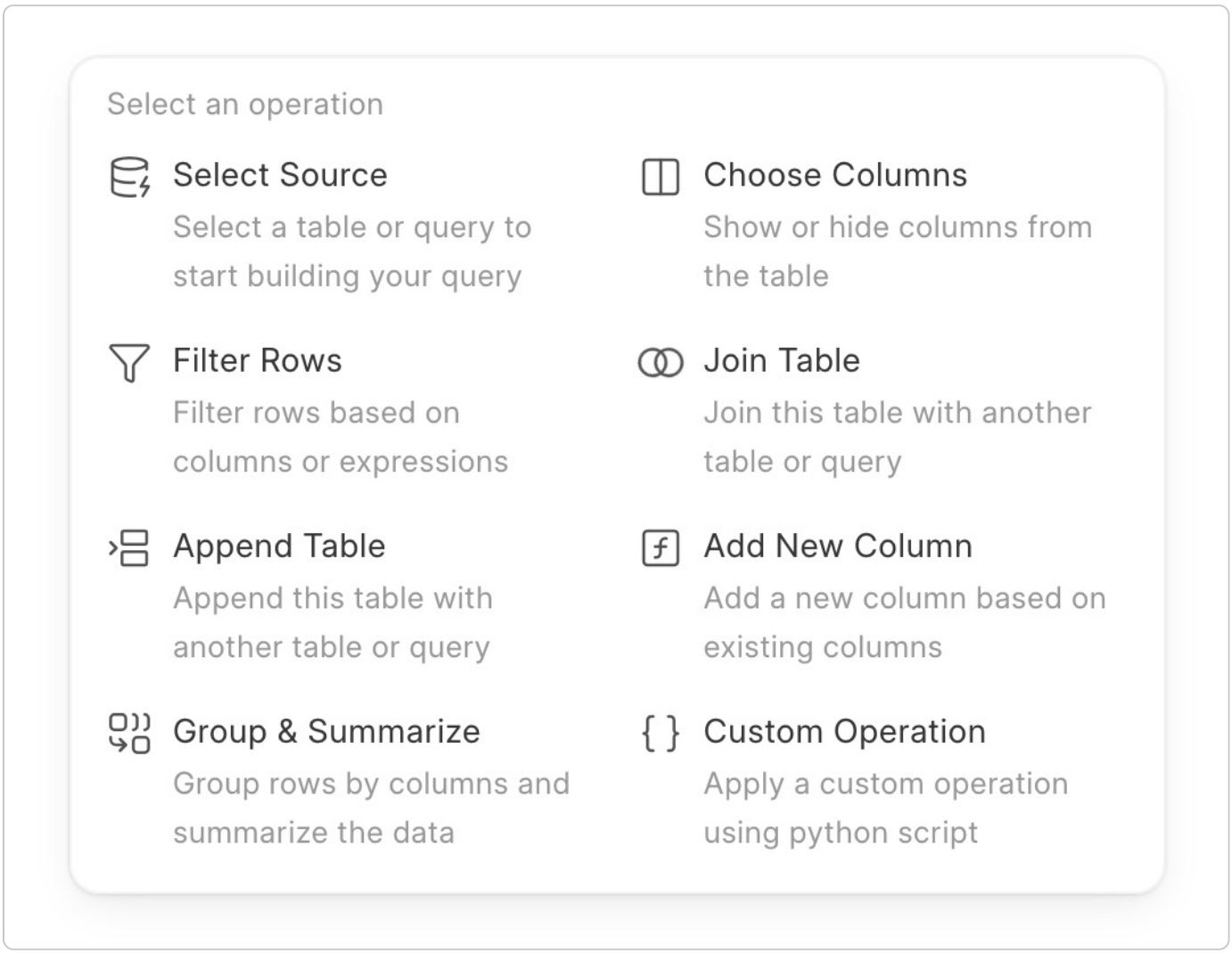
Last updated 3 months ago

Was this helpful?

Operations

Edit

Common Operations



Choose Columns

What it does: Select which columns you want to include in your results. This helps focus your analysis on the data that matters.

When to use it

- You only need specific columns from a table
- You want to hide unnecessary columns to simplify your results
- You need to improve query performance by reducing data volume

How it works

From your available columns, select only those you want to see in your results. All other columns will be hidden.

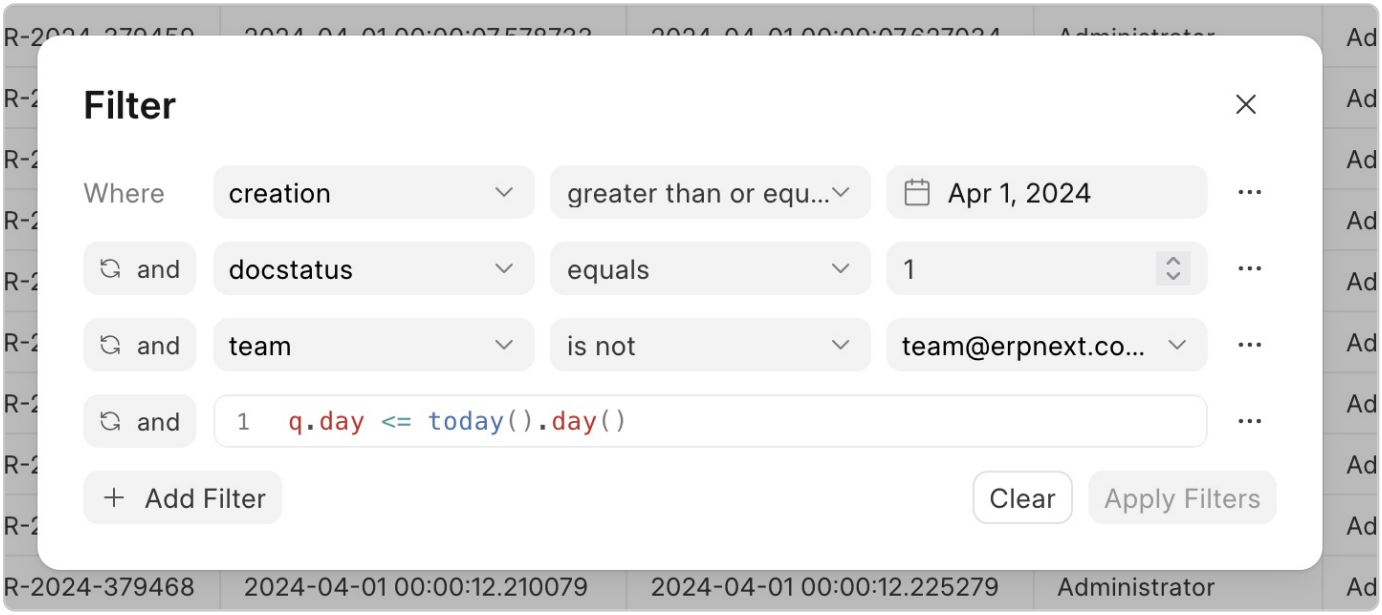
Example

From a Customer table with 20 columns, select only "Customer Name", "Email", "Region", and "Total Orders" for a customer contact list.

Tips

- Use this operation early in your query to improve performance
- Fewer columns means faster queries
- You can add columns back later if needed

Filter Rows



What it does: Show only the rows that match specific conditions. This helps you focus on relevant data.

When to use it:

- You want to analyze a specific time period
- You need data for a particular region or category
- You want to exclude certain values (like cancelled orders)
- You need to remove incomplete or invalid data

Available filter types

Comparison Filters

- Equals: Exact match

- Does not equal: Exclude specific values
- Greater than / Less than: For numbers and dates
- Greater than or equal to / Less than or equal to: Include boundary values

List Filters

- In: Match any value in a list (e.g., "New York", "Los Angeles", "Chicago")
- Not in: Exclude values in a list

Text Filters

- Contains: Find text anywhere in the column
- Does not contain: Exclude text
- Starts with: Match beginning of text
- Ends with: Match end of text

Special Filters

- Is set: Has a value (not empty)
- Is not set: Is empty or null
- Between: Value falls within a range
- Within: For relative date ranges (last 7 days, last month, etc.)

How it works You can combine multiple filters using:

- And: All conditions must be true
- Or: Any condition can be true

Examples

- Show orders from the last 30 days
- Display sales from "North" and "South" regions only
- Exclude cancelled orders
- Show customers with email addresses

Tips

- Apply filters early in your query for better performance
- Use "within" for dynamic date filtering (always shows last 7 days, etc.)
- Combine multiple filters to create precise conditions

Join Table

What it does: Combine data from two tables based on matching values in specific columns. This

lets you bring related information together.

When to use it

- You need information from multiple tables
- You want to enrich your data with additional details
- You need to combine related datasets

Join types explained

Inner Join

- Shows only rows that have matches in both tables
- Use when you only want complete data
- Example: Show orders with customer details (only for orders that have a customer)

Left Join

- Shows all rows from your current table, plus matching rows from the joined table
- Use when you want to keep all your original data
- Example: Show all customers, including those who haven't placed orders yet

Right Join

- Shows all rows from the joined table, plus matching rows from your current table
- Less commonly used than Left Join
- Example: Show all products, including those that haven't been ordered

Full Join

- Shows all rows from both tables, whether they match or not
- Use when you need comprehensive data from both sides
- Example: Show all customers and all orders, even if some don't match

How it works

1. Select the table you want to join with
2. Choose the join type
3. Specify which columns should match between the tables
4. Select which columns from the joined table you want to include

Example

Join your "Orders" table with a "Customers" table using "Customer ID" to show customer names and contact information alongside order details.

Tips

- Most common join type is Left Join
- Join columns should contain matching values (like IDs)
- Select only the columns you need from the joined table
- If column names conflict, they will be automatically renamed

Append Table

What it does: Stack rows from another table below your current data. This combines datasets vertically.

When to use it

- You want to combine data from multiple time periods
- You need to merge data from different regions or locations
- You have similar data in separate tables that should be analyzed together

How it works

1. Select the table to append
2. Choose whether to remove duplicate rows
3. Only columns that exist in both tables will be included

Options

- Include duplicates: Keep all rows, even if they're identical
- Remove duplicates: Show each unique row only once

Example: Combine "Q1 Sales" and "Q2 Sales" tables to analyze sales for the first half of the year.

Tips

- Both tables must have at least one column with the same name
- Only matching columns appear in the results
- Column types are automatically aligned
- Use "Remove duplicates" if you think there might be overlapping data

Add New Column

What it does: Create a new calculated column based on your existing columns using formulas or expressions.

When to use it

- You need to perform calculations on your data
- You want to create derived metrics

- You need to combine information from multiple columns

Common calculations

- Math operations: Add, subtract, multiply, divide
- Percentages: (part / total) * 100
- Profit calculations: revenue - cost
- Text combinations: Combine first and last names
- Conditional logic: Different values based on conditions

How it works

1. Give your new column a name
2. Choose the data type (Number, Text, Date, etc.)
3. Enter your formula using existing column names

Examples

- Calculate profit: revenue - cost
- Find profit margin percentage: (profit / revenue) * 100
- Calculate order total: quantity * unit_price
- Combine text: first_name + " " + last_name

Tips

- Use clear, descriptive names for your calculated columns
- Make sure columns are the correct data type before calculating
- Test your formulas with a small sample first
- You can reference columns by their names in your formulas

Group & Summarize

What it does: Group your data by categories and calculate summary statistics like totals, averages, and counts.

When to use it

- You want to see totals by category (sales by region, orders by product)
- You need to calculate averages or other statistics
- You want to analyze trends over time
- You need to create aggregated reports

Summary functions

Sum: Add up all values

- Example: Total sales, total quantity

Count: Count the number of rows

- Example: Number of orders, number of customers

Count Distinct: Count unique values only

- Example: Number of unique customers, number of different products

Average: Calculate the mean value

- Example: Average order value, average age

Minimum: Find the smallest value

- Example: Lowest price, earliest date

Maximum: Find the largest value

- Example: Highest price, most recent date

How it works

1. Choose dimensions (what to group by): Categories like Region, Product, Date
2. Choose measures (what to calculate): Metrics like Total Sales, Average Price, Order Count
3. Results show one row per unique combination of dimensions

Time-based grouping: When grouping by dates, you can choose the time period:

- Second, Minute, Hour (for detailed timestamps)
- Day (daily analysis)
- Week (weekly summaries)
- Month (monthly trends)
- Quarter (quarterly reports)
- Year (yearly comparisons)

Examples

- Total sales by region: Group by "Region", Sum of "Sales Amount"
- Average order value by product: Group by "Product Name", Average of "Order Value"
- Monthly revenue: Group by "Order Date" (by Month), Sum of "Revenue"
- Customer count by city: Group by "City", Count of "Customer ID"

Tips

- You can group by multiple dimensions (e.g., Region and Product)

- You can calculate multiple measures (e.g., Total Sales and Average Price)
- Use descriptive names for your measures
- For dates, choose the right time granularity for your analysis
- This operation enables drill-down features in charts

Custom Operation

What it does: Apply advanced transformations using custom Python expressions. This is for advanced users who need operations beyond the standard options.

When to use it

- You need transformations not available in other operations
- You have complex business logic to implement
- You need advanced data manipulation
- You're comfortable with Python programming

What you can do

- Access all your columns as variables
- Use Python expressions to transform data
- Apply complex conditional logic
- Perform advanced calculations

Example

Create ranking or apply windowing functions that aren't available through the standard operations.

Important notes:

- Requires knowledge of Python and data manipulation
- Use only when standard operations don't meet your needs
- Test carefully before applying to large datasets
- The expression must return a valid data table

PREVIOUS PAGE
Overview

NEXT PAGE
Expressions

Last updated 3 months ago

Expressions

Edit

Expressions

The expression/formula language in Insights is based on python syntax. The underlying library used for query building is `ibis`, you can check the documentation here to know more.

Following are the list of helper functions that make it easier to write expressions in Insights:

Aggregate Functions

1. Count
   - Syntax: `count(column)`
   - Description: Returns the count of non-null values in the column.
   - Example: `count(sales)`
   - Returns: 100
2. Count Distinct
   - Syntax: `distinct_count(column)`
   - Description: Returns the count of distinct values in the column.
   - Example: `distinct_count(customer_id)`
   - Returns: 50
3. Sum
   - Syntax: `sum(column)`
   - Description: Returns the sum of values in the column.
   - Example: `sum(sales)`
   - Returns: 10000
4. Average
   - Syntax: `avg(column)`
   - Description: Returns the average of values in the column.
   - Example: `avg(sales)`
   - Returns: 100
5. Minimum
   - Syntax: `min(column)`

- Description: Returns the minimum value in the column.
- Example: `min(sales)`
- Returns: 50

6. Maximum
   - Syntax: `max(column)`
   - Description: Returns the maximum value in the column.
   - Example: `max(sales)`
   - Returns: 200
7. Group Concat
   - Syntax: `group_concat(column)`
   - Description: Concatenates the values in the column.
   - Example: `group_concat(product_name)`
   - Returns: "Product A, Product B, Product C"
8. Sum if
   - Syntax: `sum_if(condition, column)`
   - Description: Returns the sum of values in the column that satisfy the condition.
   - Example: `sum_if(sales > 1000, sales)`
   - Returns: 5000
9. Count if
   - Syntax: `count_if(condition, column)`
   - Description: Returns the count of values in the column that satisfy the condition.
   - Example: `count_if(sales > 1000, sales)`
   - Returns: 5
10. Distinct Count if
    - Syntax: `distinct_count_if(condition, column)`
    - Description: Returns the count of distinct values in the column that satisfy the condition.
    - Example: `distinct_count_if(sales > 1000, customer_id)`
    - Returns: 3

Conditional Functions

1. If Else
   - Syntax: `if_else(condition, true_value, false_value)`
   - Description: Returns the true_value if the condition is true, else returns the false_value.
   - Example: `if_else(sales > 1000, "High", "Low")`

- Returns: "High"

2. Case
   - Syntax: `case(condition1, value1, condition2, value2, ..., default_value)`
   - Description: Returns the value corresponding to the first true condition. If no condition is true, returns the default_value.
   - Example: `case(sales > 1000, "High", sales > 500, "Medium", "Low")`
   - Returns: "High"
3. Coalesce
   - Syntax: `coalesce(column1, column2, ..., default_value)`
   - Description: Returns the first non-null value from the columns. If all columns are null, returns the default_value.
   - Example: `coalesce(product_name, product_code, "Unknown")`
   - Returns: "Product A"

String Functions

1. Lower
   - Syntax: `lower(column)`
   - Description: Converts the column values to lowercase.
   - Example: `lower(product_name)`
   - Returns: "product a"
2. Upper
   - Syntax: `upper(column)`
   - Description: Converts the column values to uppercase.
   - Example: `upper(product_name)`
   - Returns: "PRODUCT A"
3. Concat
   - Syntax: `concat(column1, column2, ..., separator)`
   - Description: Concatenates the columns with the separator.
   - Example: `concat(first_name, last_name, " ")`
   - Returns: "John Doe"
4. Replace
   - Syntax: `replace(column, old_value, new_value)`
   - Description: Replaces the old_value with the new_value in the column.
   - Example: `replace(product_name, "A", "B")`
   - Returns: "Product B"

5. Substring
    - Syntax: `substring(column, start, length)`
    - Description: Returns a substring of the column starting from the start index with the specified length.
    - Example: `substring(product_name, 0, 7)`
    - Returns: "Product"
6. Contains
    - Syntax: `contains(column, substring)`
    - Description: Returns true if the column contains the substring, else false.
    - Example: `contains(product_name, "Product")`
    - Returns: True
7. Not Contains
    - Syntax: `not_contains(column, substring)`
    - Description: Returns true if the column does not contain the substring, else false.
    - Example: `not_contains(product_name, "A")`
    - Returns: False
8. Starts With
    - Syntax: `starts_with(column, prefix)`
    - Description: Returns true if the column starts with the prefix, else false.
    - Example: `starts_with(product_name, "Product")`
    - Returns: True
9. Ends With
    - Syntax: `ends_with(column, suffix)`
    - Description: Returns true if the column ends with the suffix, else false.
    - Example: `ends_with(product_name, "A")`
    - Returns: True
10. Length
    - Syntax: `length(column)`
    - Description: Returns the length of the column.
    - Example: `length(product_name)`
    - Returns: 10

Numeric Functions

1. Abs

- Syntax: `abs(column)`
- Description: Returns the absolute value of the column.
- Example: `abs(sales)`
- Returns: 100

2. Round
    - Syntax: `round(column, decimals)`
    - Description: Rounds the column to the specified number of decimals.
    - Example: `round(sales, 2)`
    - Returns: 100.50
3. Floor
    - Syntax: `floor(column)`
    - Description: Returns the largest integer less than or equal to the column.
    - Example: `floor(sales)`
    - Returns: 100
4. Ceil
    - Syntax: `ceil(column)`
    - Description: Returns the smallest integer greater than or equal to the column.
    - Example: `ceil(sales)`
    - Returns: 101

Date Functions

1. Year
    - Syntax: `year(column)`
    - Description: Returns the year part of the date column.
    - Example: `year(order_date)`
    - Returns: 2022
2. Quarter
    - Syntax: `quarter(column)`
    - Description: Returns the quarter part of the date column.
    - Example: `quarter(order_date)`
    - Returns: 3
3. Month
    - Syntax: `month(column)`
    - Description: Returns the month part of the date column.

- Example: `month(order_date)`
- Returns: 10

4. Week of Year
   - Syntax: `week_of_year(column)`
   - Description: Returns the week of the year of the date column.
   - Example: `week_of_year(order_date)`
   - Returns: 41
5. Day of Year
   - Syntax: `day_of_year(column)`
   - Description: Returns the day of the year of the date column.
   - Example: `day_of_year(order_date)`
   - Returns: 288
6. Day of Week
   - Syntax: `day_of_week(column)`
   - Description: Returns the day of the week (0-6) of the date column.
   - Example: `day_of_week(order_date)`
   - Returns: 4
7. Day
   - Syntax: `day(column)`
   - Description: Returns the day part of the date column.
   - Example: `day(order_date)`
   - Returns: 15
8. Hour
   - Syntax: `hour(column)`
   - Description: Returns the hour part of the date column.
   - Example: `hour(order_date)`
   - Returns: 10
9. Minute
   - Syntax: `minute(column)`
   - Description: Returns the minute part of the date column.
   - Example: `minute(order_date)`
   - Returns: 30
10. Second

- Syntax: `second(column)`
- Description: Returns the second part of the date column.
- Example: `second(order_date)`
- Returns: 45

11. Microsecond
    - Syntax: `microsecond(column)`
    - Description: Returns the microsecond part of the date column.
    - Example: `microsecond(order_date)`
    - Returns: 500000
12. Date Diff
    - Syntax: `date_diff(column1, column2, unit)`
    - Description: Returns the difference between two date columns in the specified unit. Units can be "year", "quarter", "month", "week", "day", "hour", "minute", "second", "millisecond", "microsecond", "nanosecond".
    - Example: `date_diff(order_date, delivery_date, "day")`
    - Returns: 5
13. Format Date
    - Syntax: `format_date(column, format)`
    - Description: Formats the date column according to the specified format. The format should be a valid ANSI `strftime` format.
    - Example: `format_date(order_date, "%Y-%m-%d")`
    - Returns: "2022-10-15"
14. Now
    - Syntax: `now()`
    - Description: Returns the current date and time.
    - Example: `now()`
    - Returns: "2022-10-15 10:30:00"
15. Today
    - Syntax: `today()`
    - Description: Returns the current date.
    - Example: `today()`
    - Returns: "2022-10-15"

Boolean Functions

1. Is in

- Syntax: `is_in(column, value1, value2, ...)`
- Description: Returns true if the column value is in the list of values, else false.
- Example: `is_in(product_name, "Product A", "Product B")`
- Returns: True

2. Is Not in
   - Syntax: `is_not_in(column, value1, value2, ...)`
   - Description: Returns true if the column value is not in the list of values, else false.
   - Example: `is_not_in(product_name, "Product A", "Product B")`
   - Returns: False
3. Is Set
   - Syntax: `is_set(column)`
   - Description: Returns true if the column value is not null, else false.
   - Example: `is_set(product_name)`
   - Returns: True
4. Is Not Set
   - Syntax: `is_not_set(column)`
   - Description: Returns true if the column value is null, else false.
   - Example: `is_not_set(product_name)`
   - Returns: False
5. Is Between
   - Syntax: `is_between(column, start, end)`
   - Description: Returns true if the column value is between the start and end values, else false.
   - Example: `is_between(sales, 1000, 5000)`
   - Returns: True
6. Is Not Between
   - Syntax: `is_not_between(column, start, end)`
   - Description: Returns true if the column value is not between the start and end values, else false.
   - Example: `is_not_between(sales, 1000, 5000)`
   - Returns: False

Utility Functions

1. USD to INR

- Syntax: `to_inr(currency_column, amount_column, rate=83)`
- Description: Converts the values in the amount column from USD to INR based on the rate. If the currency is not USD, the amount is returned as is. Default rate is 83.
- Example: `to_inr(currency, amount, 83)`
- Returns: 8300

2. INR to USD
   - Syntax: `to_usd(currency_column, amount_column, rate=83)`
   - Description: Converts the values in the amount column from INR to USD based on the rate. If the currency is not INR, the amount is returned as is. Default rate is 83.
   - Example: `to_usd(currency, amount, 83)`
   - Returns: 100
3. Row Number
   - Syntax: `row_number()`
   - Description: Returns the row number of the current row.
   - Example: `row_number()`
   - Returns: 1
4. Previous Period Value
   - Syntax: `previous_period_value(column, date_column, offset=1)`
   - Description: Returns the value of the column for the previous period based on the date column. Offset specifies the number of periods to go back.
   - Example: `previous_period_value(sales, order_date, 1)`
   - Returns: 1000
5. Next Period Value
   - Syntax: `next_period_value(column, date_column, offset=1)`
   - Description: Returns the value of the column for the next period based on the date column. Offset specifies the number of periods to go forward.
   - Example: `next_period_value(sales, order_date, 1)`
   - Returns: 2000
6. Percentage Change
   - Syntax: `percentage_change(column, date_column, offset=1)`
   - Description: Returns the percentage change in the value of the column compared to the previous period based on the date column. Offset specifies the number of periods to go back.
   - Example: `percentage_change(sales, order_date, 1)`
   - Returns: 10

7. Is First Row
    - Syntax: `is_first_row(group_by, order_by, sort_order="asc")`
    - Description: Returns 1 if the current row is the first row in the group, else 0. Sort order can be "asc" or "desc".
    - Example: `is_first_row(customer_id, order_date, "asc")`
    - Returns: 1
8. Create Buckets
    - Syntax: `create_buckets(column, num_buckets)`
    - Description: Creates buckets based on the values in the column. The number of buckets specifies the number of buckets to create.
    - Example: `create_buckets(sales, 5)`
    - Returns: "1000-2000"

PREVIOUS PAGE
Operations

NEXT PAGE
Overview

Last updated 3 months ago

Overview

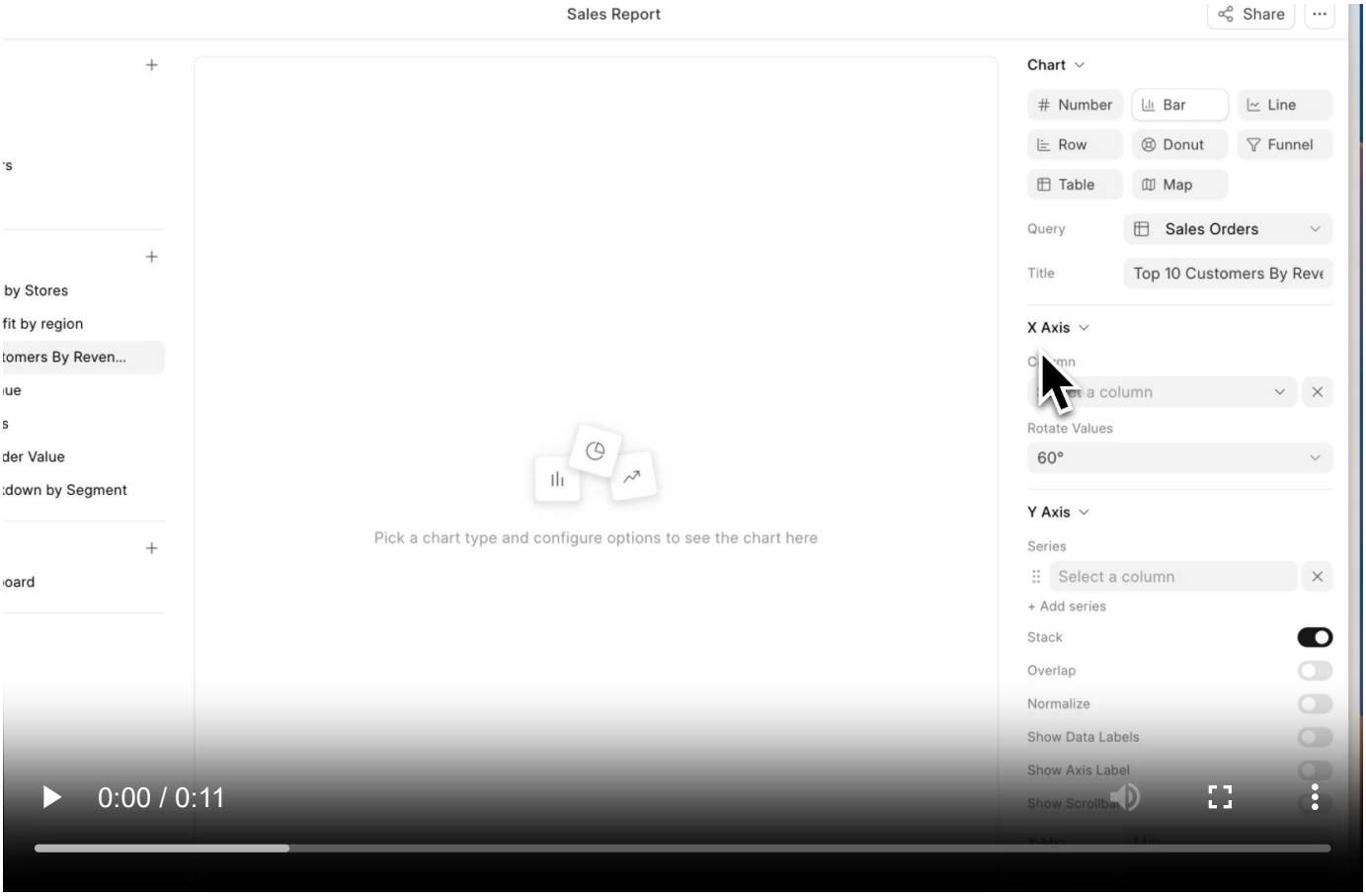
Edit

What are Charts?

Charts turn your data into pictures making it easy to spot trends, compare values, and share insights. Instead of staring at rows of numbers, you can see your data as bars, lines, pie slices, and more.

Insights has 8 chart types to choose from, each perfect for different situations. List of charts...

Creating Your First Chart



1. Open your workbook
2. Click the Charts tab
3. Click New Chart
4. Choose a Query (the data source)
5. Pick a Chart Type

6. Configure your chart (we'll explain each type below)
7. Give it a name
8. Click outside the title to save

That's it! Your chart will appear and update automatically.

PREVIOUS PAGE
Expressions

NEXT PAGE
Choosing the right Chart

Last updated 3 months ago

Was this helpful?

Choosing the right Chart

Edit

Chart Types Explained

Number Chart

Best for: Displaying one or more key metrics prominently

Perfect when

- You want to show a KPI (Key Performance Indicator) at a glance
- You're building a dashboard header with important numbers
- You want to compare this month vs last month

Example uses

- Total Revenue: $1.2M
- Active Users: 5,432
- Conversion Rate: 3.4%

How to Configure

1. Columns: Choose which numbers to display Can show multiple numbers side by side Each gets its own big display
2. Date Column (optional): For comparisons and sparklines Enables "Show comparison" feature Shows percent change from previous period
3. Formatting: Prefix: Add $ or other symbols before the number Suffix: Add % or other symbols after Decimal places: How many decimals to show Shorten numbers: Turn 1,200,000 into 1.2M
4. Advanced: Show comparison: Display % change from previous period (needs date column) Negative is better: Green when numbers go down (for costs, errors, etc.) Show sparkline: Mini line chart showing the trend Sparkline color: Choose the line color

Pro Tips

Use Number charts at the top of dashboards for immediate impact Keep it to 34 numbers max for readability Enable comparisons to show if metrics are improving Use sparklines to show trend direction at a glance

Bar Chart

Best for: Comparing values across categories

Perfect when

Comparing sales across regions Showing revenue by product Ranking top performers

Example: Monthly sales comparison, Product category performance

How to Configure

1. XAxis: Choose your categories (Region, Product, Month, etc.) This is what appears along the bottom Can rotate labels if names are long
2. YAxis: Choose what to measure (Sales, Count, Average, etc.) Add multiple measures to show several bars per category Set min/max values to zoom in on a range Show data labels to display exact values on bars
3. Chart Style: Stack: Stack multiple series on top of each other (see total height) Overlap: Show bars side by side overlapping Normalize: Show as 100% stacked (proportions)
4. Split By (optional): Break down each category further Example: Sales by Month, split by Region Each month shows multiple bars (one per region) Limit max splits to avoid clutter (default: 10)

Pro Tips

Use horizontal labels for short names, rotate for long names Stack bars when you want to show total AND composition Side-by-side bars are easier to compare exact values Limit categories to 1015 for readability

When to Use Bar vs Row: Bar (vertical): Better for time periods (months, quarters) Row (horizontal): Better for many categories or long labels

Row Chart

Best for: Comparing categories horizontally (same as Bar chart, rotated)

Perfect when

You have many categories to compare Category names are long You want to rank items top to bottom

Example: Top 20 products by sales, Employee performance rankings

Row charts are configured exactly like Bar charts, just rotated 90 degrees. All the same options apply!

When to use Row instead of Bar

Long category names (easier to read horizontally) Ranking lists (top to bottom feels natural) Many categories (vertical scrolling is easier)

Line Chart

Best for: Showing trends over time

Perfect when

Tracking revenue over months Monitoring daily active users Showing seasonal patterns

Example: Sales trend over 12 months, Website traffic by day

How to Configure

1. XAxis: Usually a date/time column Set granularity: day, week, month, year Rotate labels if needed
2. YAxis: What to plot (Sales, Users, Orders, etc.) Add multiple lines to compare trends Each line can be a different color
3. Line Style: Curved lines: Smooth curves vs straight lines Show area: Fill the area under the line Show data points: Display dots at each data point
4. Split By (optional): Create multiple lines Example: Revenue over time, split by Region Creates one line per region Each line autocolored differently

Pro Tips

Line charts are THE best choice for time series data Use curved lines for a polished look Show area fill to emphasize magnitude Avoid more than 57 lines (gets messy) Use different colors for each line

Common Patterns

Month-over-month growth: Xaxis = Month, Yaxis = Total Sales Daily actives: Xaxis = Day, Yaxis = Count of Users Comparison: Use split by to show multiple products on same chart

Donut Chart

Best for: Showing parts of a whole (proportions)

Perfect when

Market share breakdown Budget allocation Traffic sources distribution

Example: Sales by Product Category (shows % of total), Revenue by Region

How to Configure

1. Label Column: The categories (Product, Region, Source, etc.) Each unique value becomes a slice
2. Value Column: What to measure (Sales, Count, etc.) Determines slice size Automatically shows as percentages
3. Display Options: Legend Position: Top, Bottom, Left, or Right Max Slices: Limit slices (groups small values into "Others") Inline Labels: Show labels inside slices instead of in legend

Pro Tips

Best with 37 categories (too many slices = hard to read) Set max slices to group small values into "Others" Use inline labels when you have few, large slices Colors autoassigned, but you can't customize them per slice

Donut vs Pie: Donuts look more modern and are easier to label. They're the same thing functionally!

Funnel Chart

Best for: Showing stages in a process where values decrease

Perfect when

Sales funnel (Leads → Qualified → Demo → Closed) Conversion funnel (Visitors → Signup →

Activated → Paid) Application process (Applied → Screened → Interviewed → Hired)

Example: Ecommerce funnel showing dropoff at each step

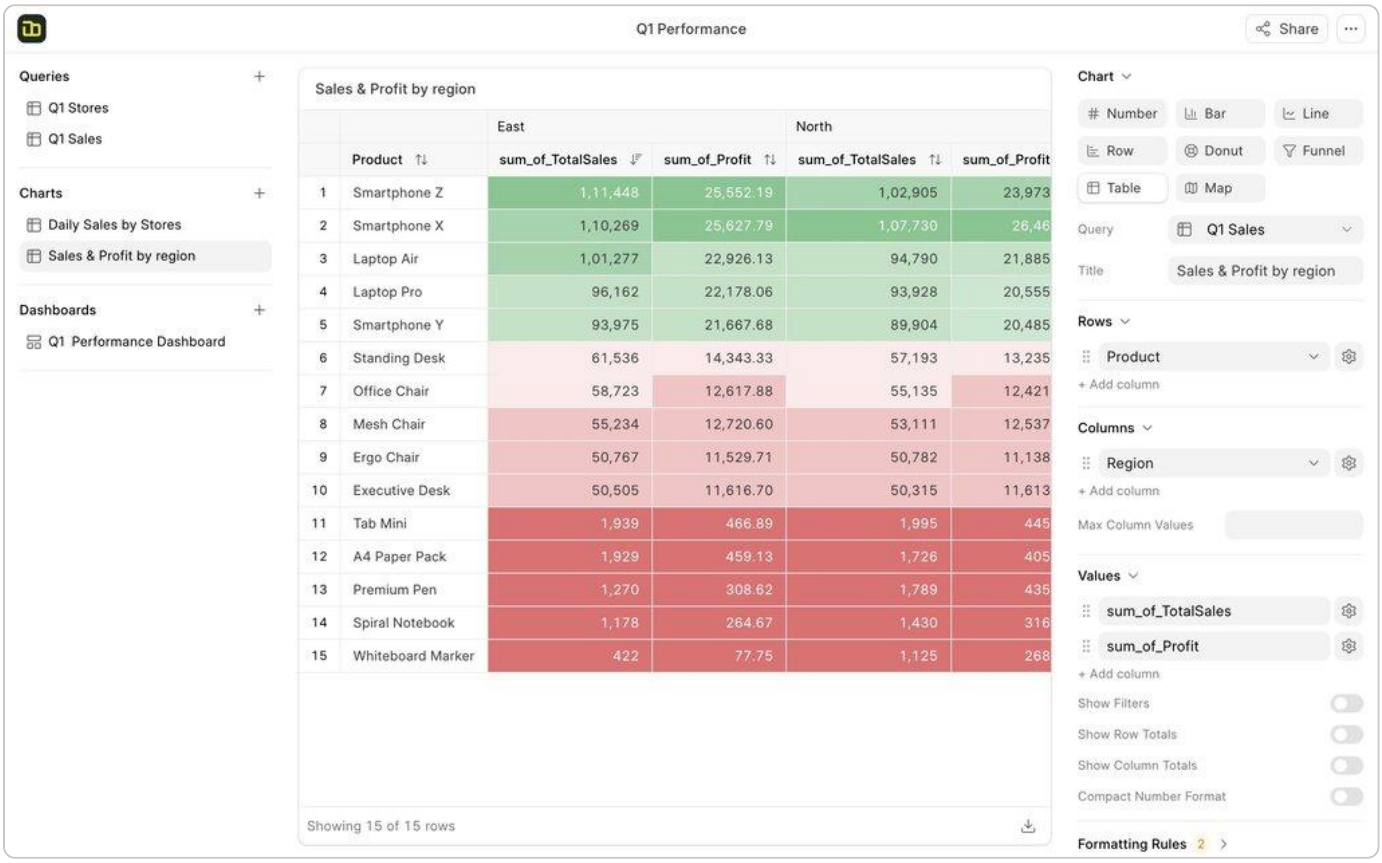
How to Configure

1. Label Column: The stage names (Visit, Cart, Checkout, Purchase) Order matters! Sort your query to put stages in order
2. Value Column: The count or amount at each stage Funnel automatically shows dropoff between stages
3. Label Position: Left: Labels on left side Right: Labels on right side Alternate: Alternate left and right

Pro Tips

Always sort your data in funnel order before charting Funnel automatically calculates conversion rates between stages Great for presentations instantly shows where you lose people

Table Chart



Best for: Displaying detailed data in rows and columns

Perfect when

You need to show exact values Data needs to be searchable/filterable Creating reports people will export Building pivot tables

Example: Customer list with details, Sales by Product and Month

How to Configure

1. Rows: What becomes row headers (Customer, Product, Date, etc.) Can add multiple row dimensions Creates groupings/hierarchies
2. Columns (optional): Create pivot table Values become column headers Example: Months become columns
3. Values: What to display in cells (Sales, Count, Average, etc.) Can show multiple values
4. Display Options: Show filter row: Add search boxes to each column Show totals: Add row and column totals Compact numbers: Show 1.2K instead of 1,200 Color scale: Heat map coloring (high values = darker) Sticky columns: Pin columns when scrolling
5. Conditional Formatting: Color cells based on rules Highlight values above/below threshold Color scales (red to green) Custom rules

Pro Tips

Tables are perfect when exact numbers matter Use pivot mode (add columns) for crosstabulation Enable filter row for large datasets Sticky columns keep important columns visible while scrolling Color scale mode makes patterns jump out

Map Chart

Best for: Visualizing data geographically

Perfect when

Sales by country Users by region Store locations and performance

Example: Revenue by Country, Support tickets by State

How to Configure

1. Map Type: Choose your geography World Map: All countries India: Indian states
2. Location Column: Column with location names Must match map region names Examples: "India", "Delhi"
3. Value Column: What to show (Sales, Count, etc.) Determines color intensity Higher values = darker colors

Pro Tips

Location names must match exactly (check spelling) Colors autoscale from light (low) to dark (high) Great for spotting geographic patterns Currently limited to World and India maps

Location Name Tips

Countries: Use full names ("United Kingdom" not "UK") US States: Full names or abbreviations work ("California" or "CA") India: Use official state names

Last updated 3 months ago

Was this helpful?

Overview

Edit

Dashboards in Insights allow you to bring together multiple charts, filters, and text blocks into a single, interactive view. Think of a dashboard as a canvas where you can arrange your visualizations to tell a complete data story.

What you can do with dashboards

- Combine multiple charts in one place
- Add interactive filters that control multiple charts
- Include text for context and explanations
- Share dashboards with your team or make them public
- Arrange everything in a flexible grid layout

Perfect for

- Executive summaries with key metrics
- Team performance monitoring
- Customer analytics overviews
- Sales and revenue tracking
- Operational dashboards

PREVIOUS PAGE
Choosing the right Chart

NEXT PAGE
Sharing and Access Control

Last updated 3 months ago

Was this helpful?

Sharing and Access Control

Edit

Sharing Options

1. Private (Default)
   - Only you can see the dashboard
   - Not shared with anyone
2. Share with Specific People
   - Select individual users to grant access
   - They see it in their dashboard list
   - You control who has access
3. Share with Organization
   - Everyone in your organization can access
   - Appears in their dashboard list
   - Great for team dashboards
4. Public Link
   - Anyone with the link can view
   - Perfect for external sharing
   - Link remains active until you revoke it

How to Share

1. Click the "Share" button in dashboard header
2. Choose sharing option:
   - General Access: Private, Organization, or Public
   - Add People: Search and select specific users
3. Copy the share link if public
4. Click "Save" to apply changes

Share Link Features

Copy Link

- Click "Copy Link" to get the shareable URL

- Send via email, chat, or embed in documents

Embed Code

- Click "Get Embed Code" for iframe code
- Paste in websites or internal portals
- Dashboard displays in the embedded frame

Managing Access

- See everyone who has access
- Remove users by clicking the X next to their name
- Change between private/organization/public anytime

Access Levels

View Only

- Users can see charts and apply filters
- Cannot edit the dashboard
- Cannot change layout or add items

Edit Access

- Available to dashboard creator and workbook editors
- Can modify layout, add/remove items
- Can change sharing settings

Pro Tips

- Start with private, share when ready
- Use organization sharing for internal dashboards
- Use public links for client reports or external stakeholders
- Review who has access periodically
- Remove access when team members change roles

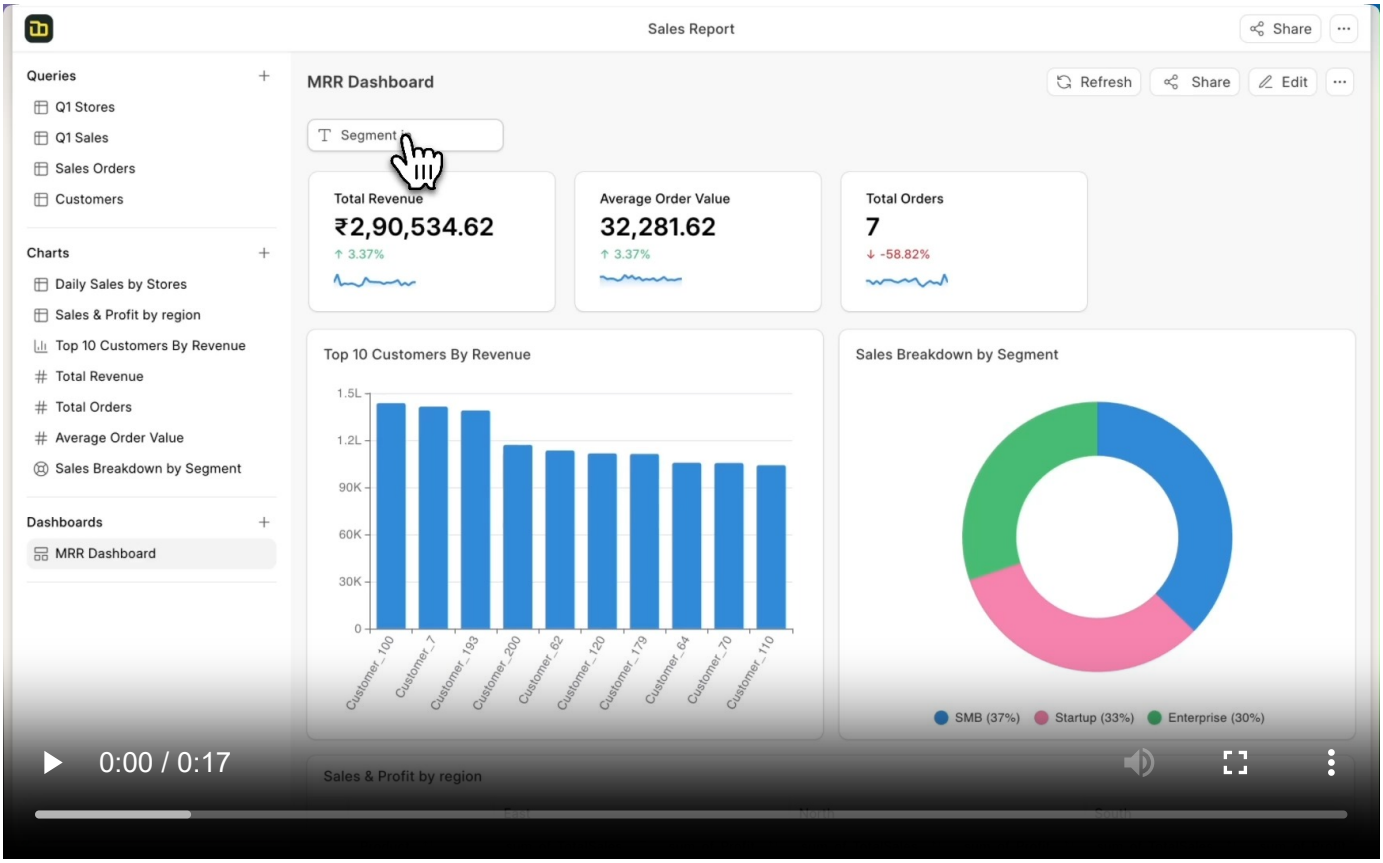
PREVIOUS PAGE
Overview

NEXT PAGE
Creating your first Dashboard

Last updated 3 months ago

Creating your first Dashboard

Edit



1. Navigate to Dashboards
    - Click on "Dashboards" in workbook page
    - Click "+" Icon
2. Give it a Name

```
- Click on the title to rename it
- Use a clear and descriptive name like "Sales Overview" or "Monthly Performance"
```

3. Add Content

```
- Click "Add Chart" to add visualizations
- Click "Add Filter" to add interactive controls
- Click "Add Text" to add descriptions or headers
```

4. Arrange Your Layout

```
- Drag items to reposition them
- Resize items by dragging the corners
- Items snap to a grid for clean alignment
```

5. Save and Share

```
- Changes save automatically
- Click "Done" when finished editing
- Use the "Share" button to give others access
```

Dashboard Components

Charts

Charts are the heart of your dashboard - they display your data visually.

Adding Charts

- Click "Add Chart" button
- Select one or more charts from your workbook
- Charts appear on the dashboard in a grid

Working with Charts

- Move: Click and drag to reposition
- Resize: Drag the bottom-right corner to change size
- View Details: Click the expand icon to see the full chart
- Remove: Click the trash icon to delete from dashboard
- Edit: Click the pencil icon to modify the chart

Pro Tips

- Group related charts together
- Put the most important metrics at the top
- Use consistent sizing for visual harmony
- Consider the reading flow (left to right, top to bottom)

Filters

Filters let viewers interact with your dashboard by controlling what data is displayed across

multiple charts.

How Filters Work

- One filter can control multiple charts
- Viewers select values to filter all linked charts simultaneously
- Great for date ranges, regions, products, or any category

Adding a Filter

1. Click "Add Filter" button
2. Give it a clear label (e.g., "Select Region", "Date Range")
3. Choose the filter type:

```
- Text: For categories, names, statuses
- Number: For quantities, amounts, IDs
- Date: For time-based filtering
```

4. Link it to charts:

```
- Toggle on the charts you want to control
- Select which column in each chart should be filtered
```

5. Click "Apply"

Setting Up Filter Links

- You can link one filter to different columns in different charts
- Example: A "Region" filter might control:
  - "Region" column in Sales chart
  - "Store Region" column in Inventory chart
  - "Customer Region" column in Orders chart

Filter Types Explained:

Text Filters

- Use for: Categories, names, regions, statuses
- Operators: equals, contains, starts with, in list
- Example: Filter by Product Category = "Electronics"

Number Filters

- Use for: Amounts, quantities, IDs, scores
- Operators: equals, greater than, less than, between
- Example: Filter by Order Amount > 1000

Date Filters

- Use for: Time periods, deadlines, dates
- Options: Specific date, date range, or relative (last 7 days, this month)
- Example: Filter by Order Date = Last 30 Days

Pro Tips

- Place filters at the top of the dashboard for easy access
- Use clear labels that explain what the filter does
- Limit to 3-5 filters to avoid overwhelming users
- Group related filters together
- Use relative date filters for dynamic dashboards

Text Blocks

Text blocks add context, headers, and explanations to your dashboard.

Adding Text

1. Click "Add Text" button
2. Type or paste your content
3. Use the editor to format text
4. Click outside or press Done to save

What to Include

- Headers: Section titles to organize the dashboard
- Descriptions: Explain what charts show
- Instructions: Guide viewers on how to use filters
- Notes: Important context or caveats about the data
- Dates: Show when data was last updated

Formatting Options

- Bold, italic, underline
- Bullet points and numbered lists
- Headers (different sizes)
- Links to other resources

Pro Tips

- Keep text concise and scannable
- Use text to divide dashboard into sections
- Add a summary text at the top explaining the dashboard's purpose
- Include "Last Updated" information for time-sensitive data

PREVIOUS PAGE
Sharing and Access Control

NEXT PAGE
Enable Permissions

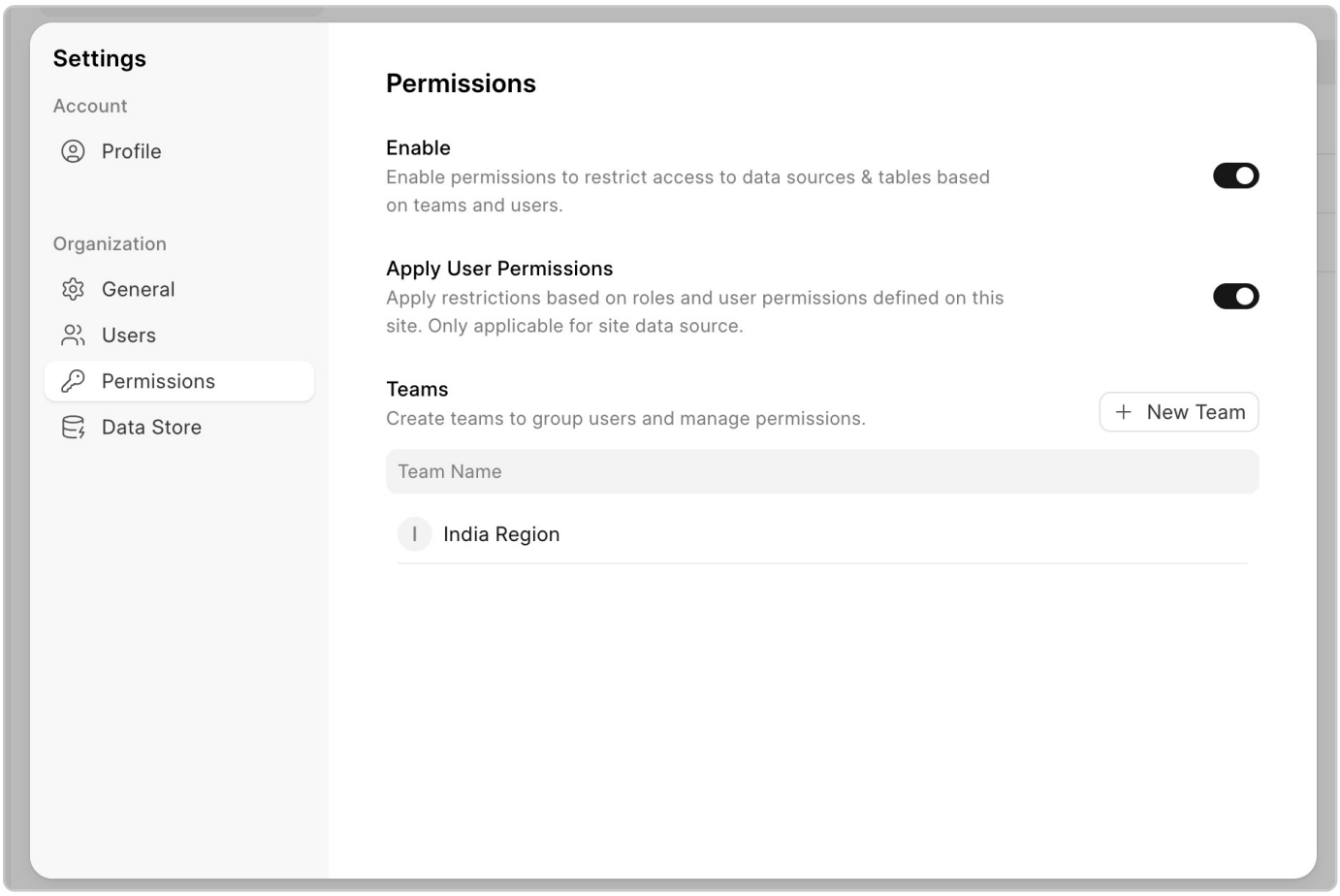
Last updated 3 months ago

Was this helpful?

Enable Permissions

Edit

Step 1: Enable Team Permissions



1. Go to **Settings → Permissions**
2. Toggle **"Enable permissions to restrict access..."**
3. This activates team-based resource access control

Step 2: Enable User Permissions

For Frappe site data sources only:

1. Toggle **"Apply User Permissions"**
2. This applies Frappe's role-based permissions on top of team permissions
3. Only works for site data sources (not external databases)

PREVIOUS PAGE
Creating your first Dashboard

NEXT PAGE
Teams

Last updated 3 months ago

Was this helpful?

Teams

Edit

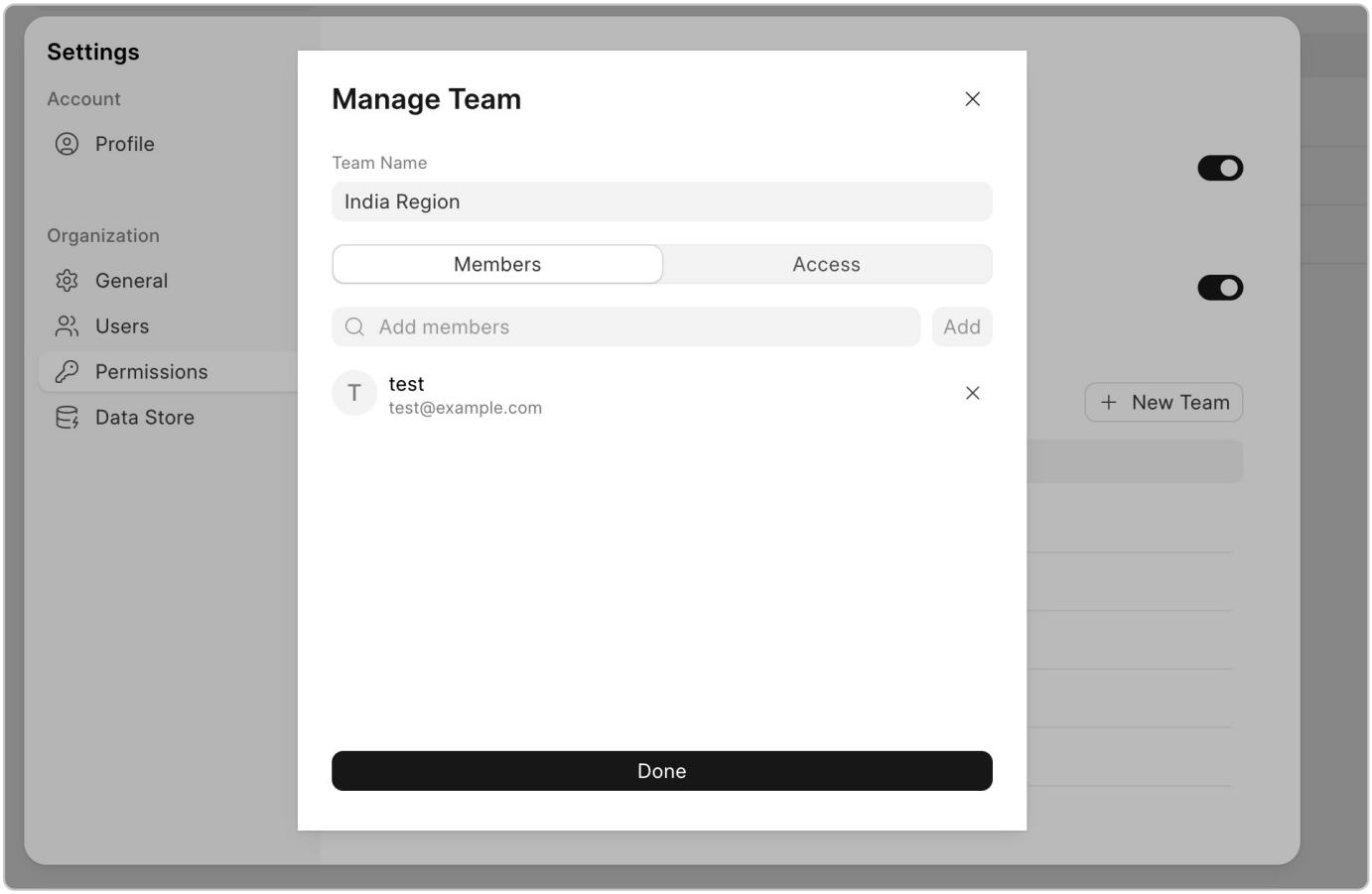
What are Teams?

Teams are groups of users who share access to the same resources. Every Insights installation has an **Admin** team with full access.

Creating Teams

1. Go to **Settings** → **Permissions**
2. Click **"Create Team"**
3. Enter team name (e.g., "Sales Team West", "Finance Team")
4. Click **Create**

Managing Team Members



1. Click on a team from the list

2. Go to the **Members** tab
3. Add/remove users using email addresses

Example:

```
Team: Sales Team West
Members:
  - john@company.com
  - jane@company.com
  - robert@company.com
```

PREVIOUS PAGE
Enable Permissions

NEXT PAGE
Overview

Last updated 3 months ago

Overview

Edit

This guide explains how to control data access in Insights using teams, permissions and row-level security

> *"Note: Make sure you have* `Insights Admin` *role in order to configure user permissions"*

Insights provides three levels of access control:

1. **Application-level**: Role-based access (Insights Admin, Insights User)
2. **Resource-level**: Team-based access to data sources, tables, dashboards, charts
3. **Row-level**: Filter-based restrictions on table data (e.g., show only assigned customers)

Permission Hierarchy

```
User → Team(s) → Resources (Data Sources/Tables) → Row Filters
```

- Users are assigned to one or more teams
- Teams are granted access to specific resources
- Resources can have row-level filters applied

PREVIOUS PAGE
Teams

NEXT PAGE
Resource Permissions

Last updated 3 months ago

Was this helpful?

Resource Permissions

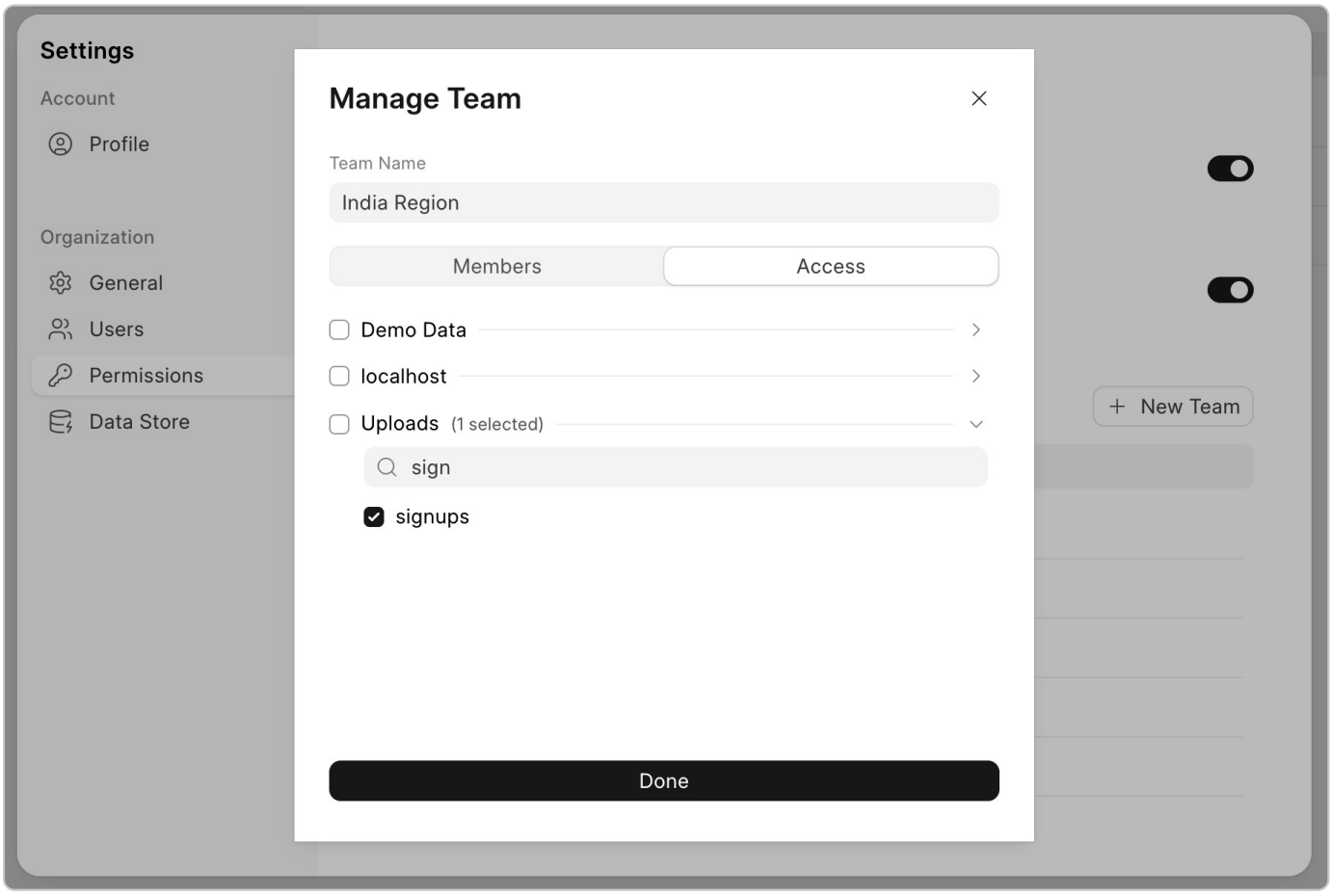
Edit

What are Resources?

Resources that can be shared with teams:

- **Data Sources**: Database connections
- **Tables**: Individual tables within data sources
- **Dashboards**: Dashboard access
- **Charts**: Chart access
- **Workbooks**: Workbook access (through sharing)

Granting Access to Data



1. Click on a team
2. Go to the **Access** tab

3. Select data sources and/or specific tables
4. Click **Done** to save

Access Patterns

Pattern 1: Full Data Source Access

Grant access to entire data source:

- User can query all tables in that data source
- Best for admins or broad access needs

Pattern 2: Specific Table Access

Grant access to individual tables:

- User can only query selected tables
- More granular control
- Better for restricted access

Example:

```
Sales Team West has access to:
  ✓ Data Source: Production Database
    ✓ Table: customers
    ✓ Table: orders
    ✓ Table: invoices
    ✗ Table: employee_salaries (not accessible)
```

PREVIOUS PAGE
Overview

NEXT PAGE
Table Restrictions

Last updated 3 months ago

Table Restrictions

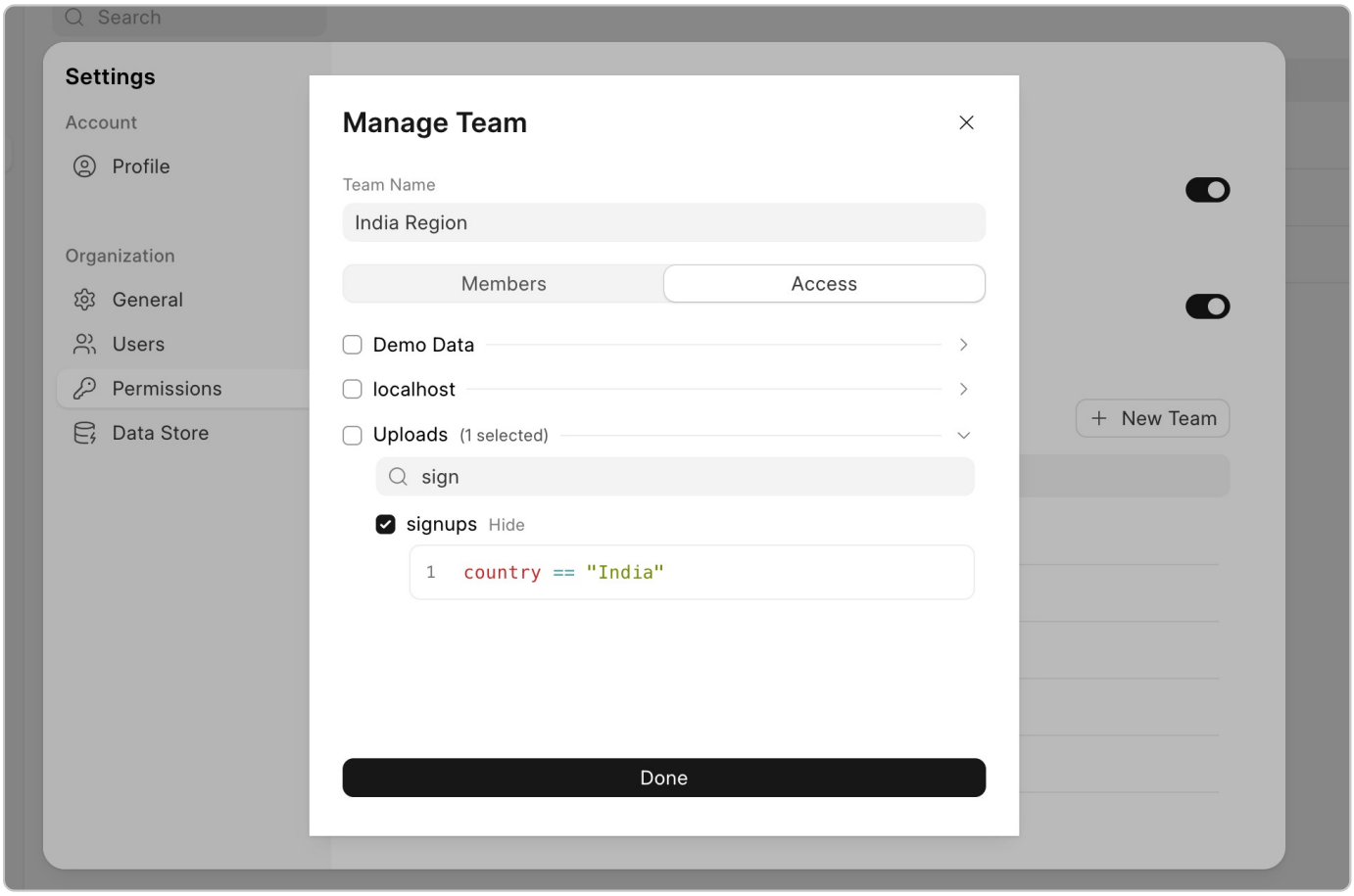
Edit

Row-Level Security (Table Restrictions)

What are Table Restrictions?

Table restrictions are **filter expressions** that automatically limit which rows users can see in a table. This is the key feature for implementing "users can only see their own data" scenarios.

How to Set Table Restrictions



1. Click on a team → **Access** tab
2. Select a data source to expand it
3. Check the tables you want to grant access to
4. Click **"Set Filters"** next to a table name
5. Enter a filter expression

6. Click **Done** to save

Filter Expression Syntax

Expressions use Python-like syntax with table column names:

```python
# Basic equality
column_name == 'value'

# Multiple conditions
region == 'West' AND active == 1

# Comparisons
revenue > 100000

# String operations
customer_name.startswith('A')

# NULL checks
manager IS NOT NULL
```

PREVIOUS PAGE
Resource Permissions

NEXT PAGE
Examples

Last updated 3 months ago

Examples

Edit

Use Cases & Examples

Use Case 1: Sales Team - Territory-Based Access

Scenario: Sales reps should only see customers in their assigned territory.

Setup:

1. Create team: "Sales Team North"
2. Add team members: northern sales reps
3. Grant access to `customers` table
4. Set table restriction:

```python
territory == 'North'
```

Result: Team members only see customers where `territory = 'North'`

Use Case 2: Sales Rep - Individual Assignment

Scenario: Each sales rep should only see their assigned customers and their sales data.

Setup:

1. Create team: "Sales Representatives"
2. Add all sales reps as members
3. Grant access to tables:
   - `customers`
   - `sales_orders`
4. Set table restrictions:

   For `customers` table:

```python
sales_person == frappe.session.user
```

For `sales_orders` table:

```python
sales_person == frappe.session.user
```

Result:

- John (john@company.com) only sees customers/orders where `sales_person = 'john@company.com'`
- Jane (jane@company.com) only sees customers/orders where `sales_person = 'jane@company.com'`

Use Case 3: Manager - Team View

Scenario: Sales managers should see all data for their team members.

Setup:

1. Create team: "Sales Managers West"
2. Add managers as members
3. Grant access to same tables as reps
4. Set table restrictions:

   For `customers` table:

   ```python
   sales_person_region == 'West'
   ```

   Or if you have a manager column:

   ```python
   manager == frappe.session.user
   ```

Result: Managers see all their team's data, reps see only their own.

Use Case 4: Finance Team - AR Visibility

Scenario: AR team should see all invoices, but only for active customers.

Setup:

1. Create team: "Accounts Receivable"
2. Add AR team members
3. Grant access to:
   - `customers`
   - `sales_invoices`
   - `payments`
4. Set table restrictions:

   For `customers` table:

   ```python
   status == 'Active' AND outstanding_amount > 0
   ```

   For `sales_invoices` table:

   ```python
   status != 'Paid' AND due_date IS NOT NULL
   ```

Result: AR team sees all unpaid invoices for active customers.

Use Case 5: Multi-Company Access

Scenario: Users should only see data for their assigned company/branch.

Setup:

1. Create teams per company: "Company A Team", "Company B Team"
2. Grant access to shared tables
3. Set table restrictions:

   ```python
   company == 'Company A'
   ```

Result: Users in "Company A Team" only see Company A data across all tables.

Use Case 6: Time-Based Access

Scenario: Analysts should only see recent data.

Setup:

1. Create team: "Junior Analysts"
2. Grant access to transaction tables
3. Set table restrictions:

```python
transaction_date >= '2024-01-01'
```

Result: Analysts can't see historical data older than specified date.

PREVIOUS PAGE
Table Restrictions

NEXT PAGE
Troubleshooting

Troubleshooting

Edit

Users can't see any data

Check:

1. Is permissions enabled? (Settings → Permissions)
2. Is user in a team?
3. Does team have access to the data source/table?
4. Are table restrictions too restrictive?
5. Test with admin account first

Users see wrong data

Check:

1. Review table restriction expressions
2. Verify column names match database schema exactly
3. Check for typos in filter expressions
4. Test with SQL query directly

Security Considerations

What Permissions DON'T Protect

- **Direct database access**: Permissions only apply within Insights
- **API access**: If users can access backend APIs directly
- **Exported data**: Once data is exported (CSV, Excel), filters don't apply
- **Shared workbooks**: Sharing bypasses team permissions

What Permissions DO Protect

- All queries created within Insights
- Dashboard and chart views
- Data exploration and query builder

- API calls through Insights frontend

PREVIOUS PAGE
Examples

NEXT PAGE
Creating Date Table

Last updated 3 months ago