SQL Sentry Portal & SentryOne Monitor Top SQL

Last Modified on 02 November 2021

Applies to the following products and features: The SentryOne Monitor product and the on-premises SQL Sentry Portal feature for SQL Sentry.

Introduction

The Top SQL view displays a unified picture of collected SQL statements. It's designed to help you quickly identify queries, applications, logins, and more that are causing the most waits, using the most resources, taking the most time, and putting the most load on your SQL Server.

Note: Top SQL data is retained for 15 days in **SentryOne Monitor**. If you're using the **SQL Sentry Portal** feature for SQL Sentry, this default value may be changed and is controlled by the Monitoring Service Settings.

Available Charts

The full viewing options for this card's charts are:

- Waits
- Resources
- Queries ›
- By App ›
- By DB ›
- By Host ›
- By Login ›

Select the ⋮ button in the upper right for additional **options** such as reset, show/hide axis labels, and show/hide axes.

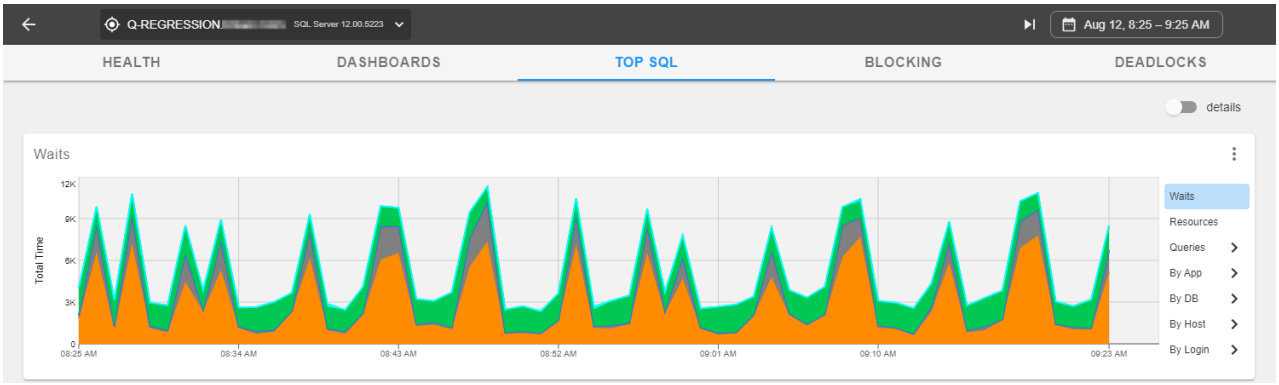
Show/hide axes has the following options:

- Avg Duration (ms)
- CPU Time (ms)
- Exec Count
- Reads Logical
- Writes Logical
- Reads Physical

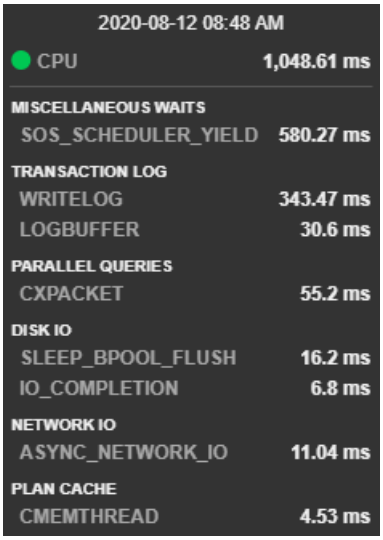
On the options with the ❯ (chevron-right), additional choices similar to the above are available by selecting the ❯ symbol. For example, *By App - Duration (ms)* and *Queries - Reads (P)* are available chart options.

Waits

The first card in **Top SQL** defaults to a **Waits** view. Waits displayed here are from the SQL Server instance level. For a better understanding of waits, see the SQL Server Waits Stats section of the Dashboards article and this blog post What to do (or not do) about top wait stats.

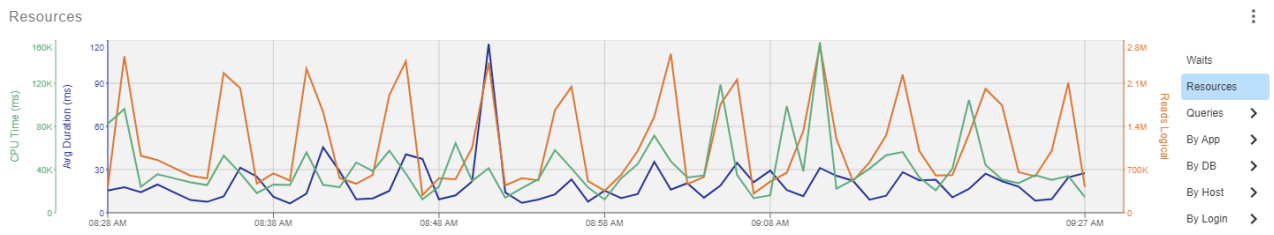


Hover over an area on the chart to view additional details about the waits:

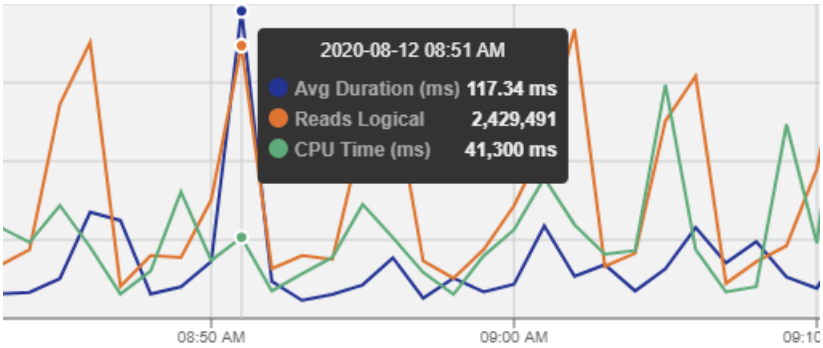


Resources

View resource usage from Top SQL, based on query and procedure stats:



Hover over a point on the chart to view additional details about the resources being used:

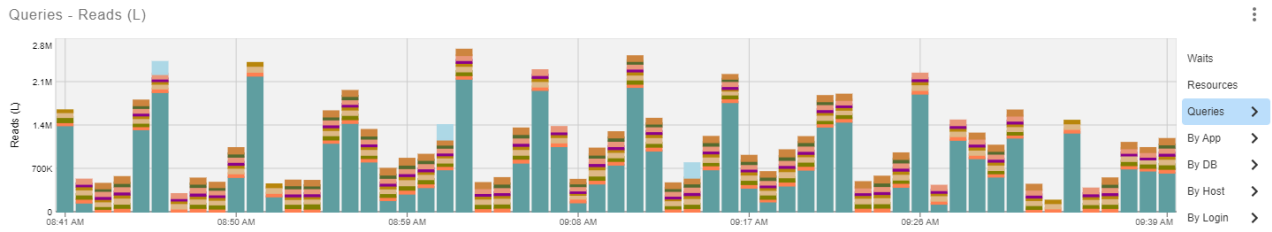


Queries

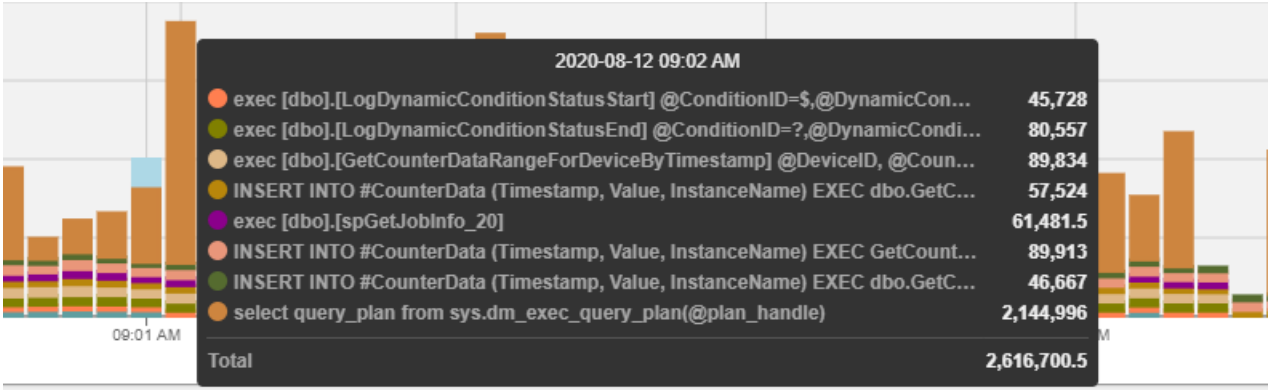
Select the ❯ (chevron-right) to view charts for:

- CPU
- Duration
- Exec Count
- Reads (L) - The default selection
- Reads (P)
- Writes (L)

The information displayed here is from query stats, procedure stats, and trace data.



Hover over an area in the chart to view more details about the queries:



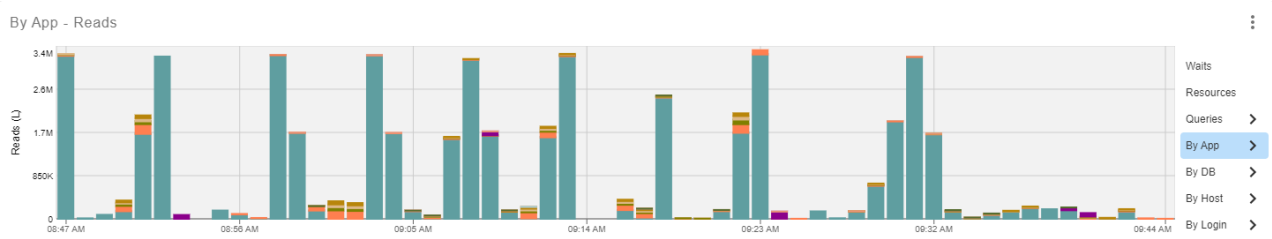
By App

Select the ❯ (chevron-right) to view charts for:

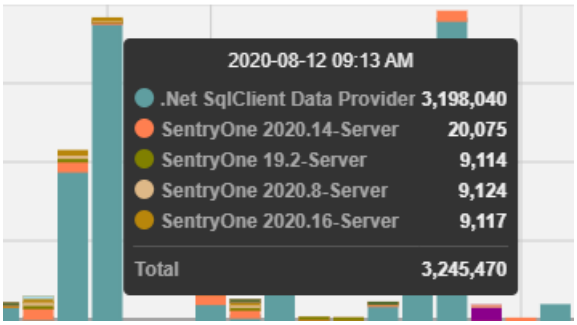
- CPU

- Duration
- Exec Count
- Reads (L) - The default selection
- Writes (L)

The information displayed here is from trace data.



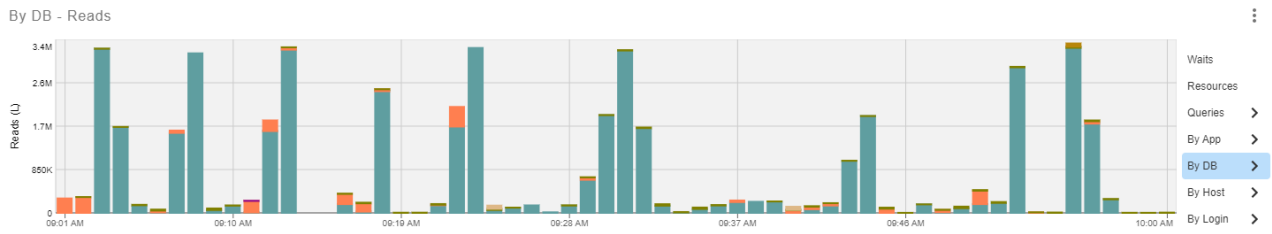
Hover over an area on the chart to view more details about the applications:



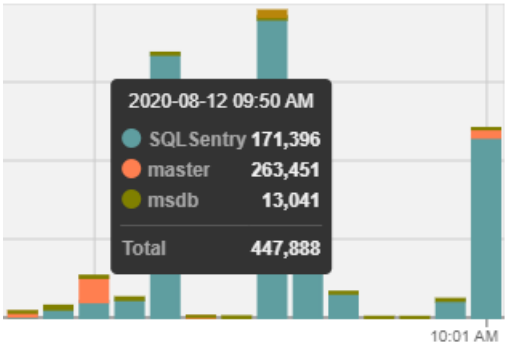
By DB

Select the ❯ (chevron-right) to view charts for:

- CPU
- Duration
- Exec Count
- Reads (L) - The default selection
- Reads (P)



Hover over an area on the chart to view more details about the databases:

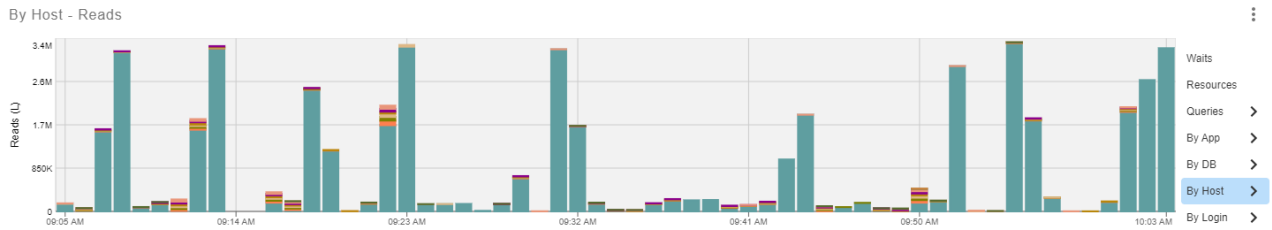


By Host

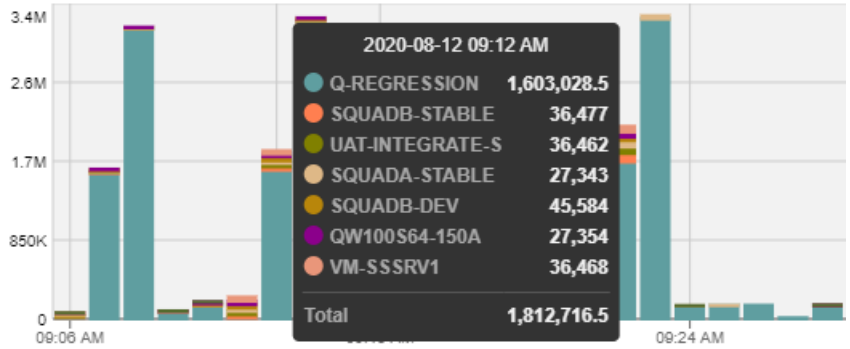
Select the ❯ (chevron-right) to view charts for:

- CPU
- Duration
- Exec Count
- Reads (L) - The default selection
- Writes (P)

The information displayed here is from trace data.



Hover over an area on the chart to view more details about hosts:



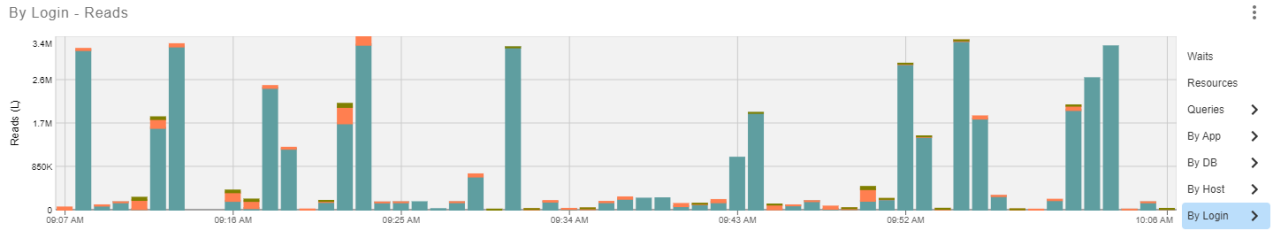
By Login

Select the ❯ (chevron-right) to view charts for:

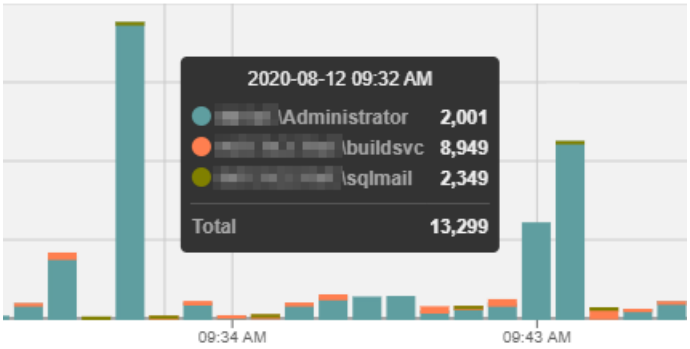
- CPU
- Duration

- Exec Count
- Reads (L) - The default selection
- Writes (P)

The information displayed here is from trace data.

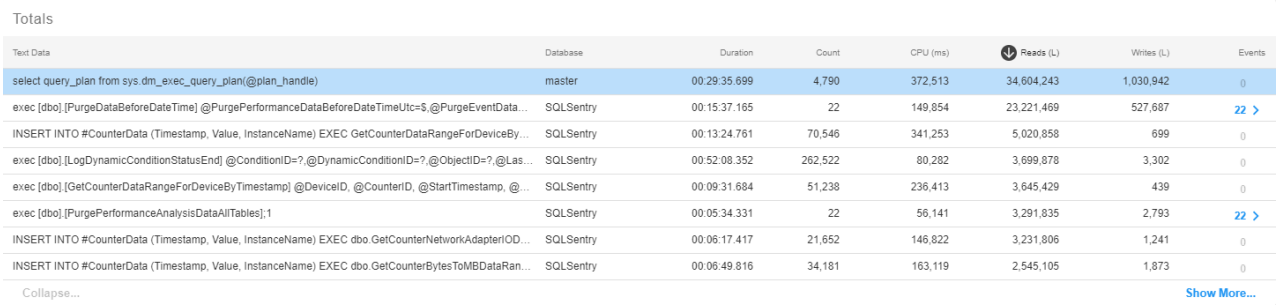


Hover over an area on the chart to view more details about logins:



Totals

The **Totals** table displays the text data and associated information such as database, duration, count, and CPU for the collected SQL statements (including procedure stats, query stats, and completed queries).

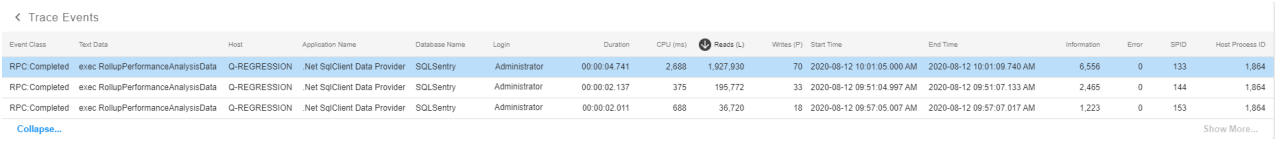


> **Note:** By default, the **Totals** grid displays the top 8 queries by logical reads (descending). For all grids, the arrow with the circle around it highlights by which column the data is sorted; ascending or descending.

Trace Events

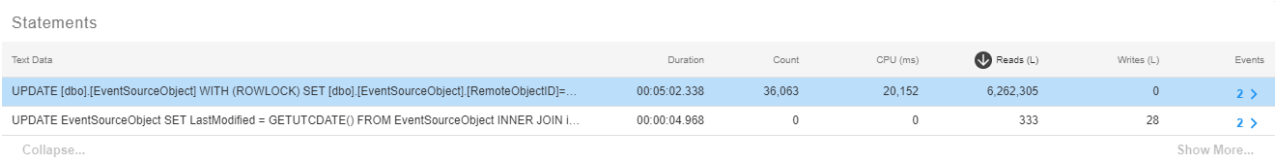
Select the (chevron-right) under the **Events** column to display details for any collected **Trace Events** such as

RPC:Completed.



Note: In the upper right of the **Top SQL** page, there is a **details** switch. The switch is off by default. Select the switch to turn on details (details), which flips all totals and statements tables to the detailed **Trace Events** and **Trace Events Statements** view on the page.

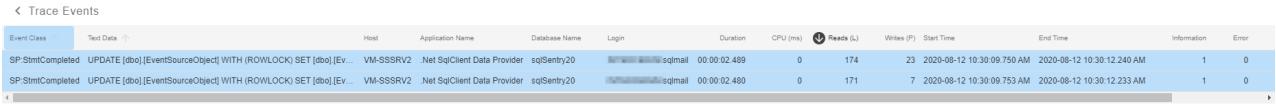
Statements



The **Statements** card displays additional information about **Totals** where applicable, including plan diagrams, text data, parameters, and plan XML.

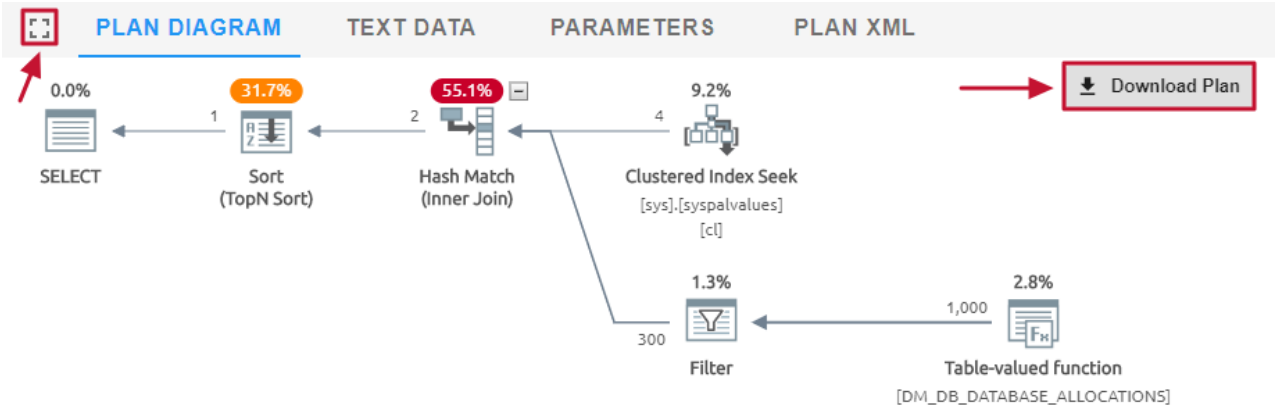
Trace Events Statements

Select the ❯ (chevron-right) under the **Events** column to display statement details to any collected Trace Events such as *SP:StmtCompleted*.



Plan Diagram

Use the **full screen** button in the upper left to expand a larger plan, or use the **Download Plan** button to download the entire **.sqlplan** file.

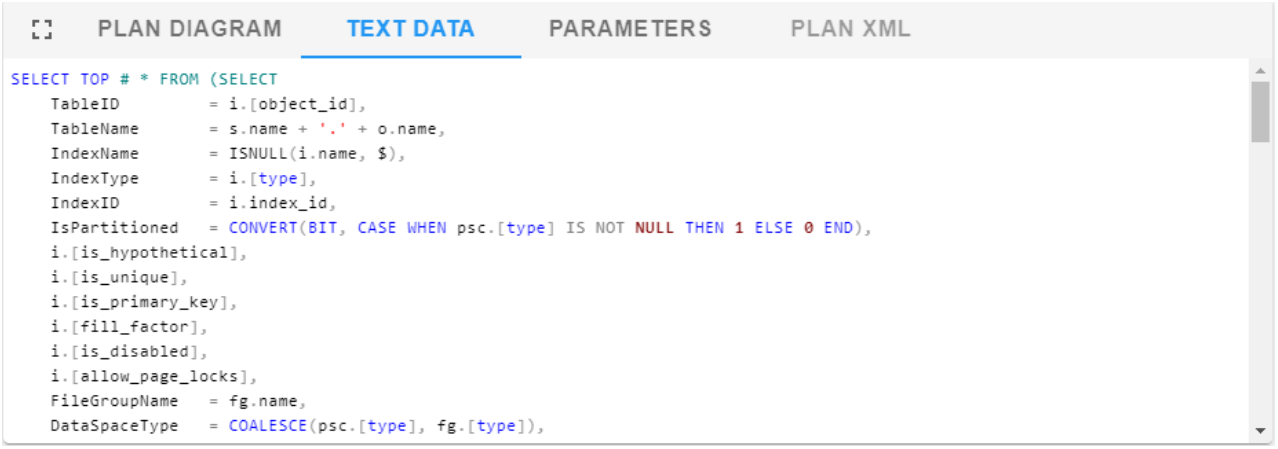


Note:

- Sometimes the **plan XML** may be populated, but there's not a statement that can be matched for the **plan diagram**. This can be caused by things like nested procedures or individual statements falling outside of the collection thresholds on their own. In this case, the **plan diagram** will display a message such as "*The selected statement was not found in the plan XML. Download the full plan to view in SentryOne Plan Explorer*" and provide a **Download Plan XML** button.
- If the **Query History** chart shows a disabled point (i.e. a gray triangle), then the plan diagram will display a message to indicate that there are no plans available (e.g. "*There is no data to display*").

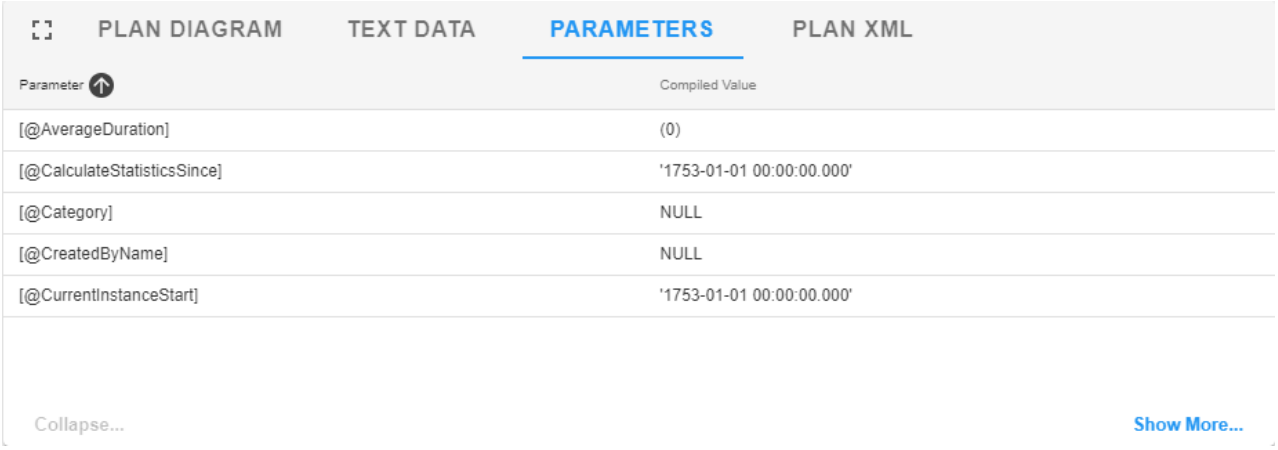
Text Data

Use the **Text Data** tab to view a formatted and syntax color-coded copy of the statement.



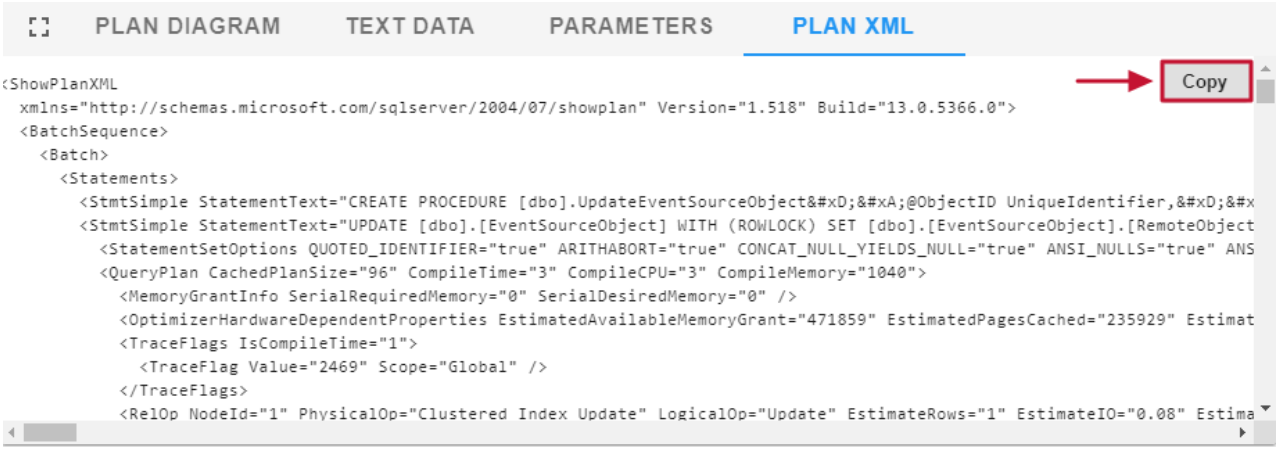
Parameters

Use the **Parameters** tab to view compiled values for statement parameters.



Plan XML

Use the **Plan XML** tab to view or copy the ShowPlanXML output.



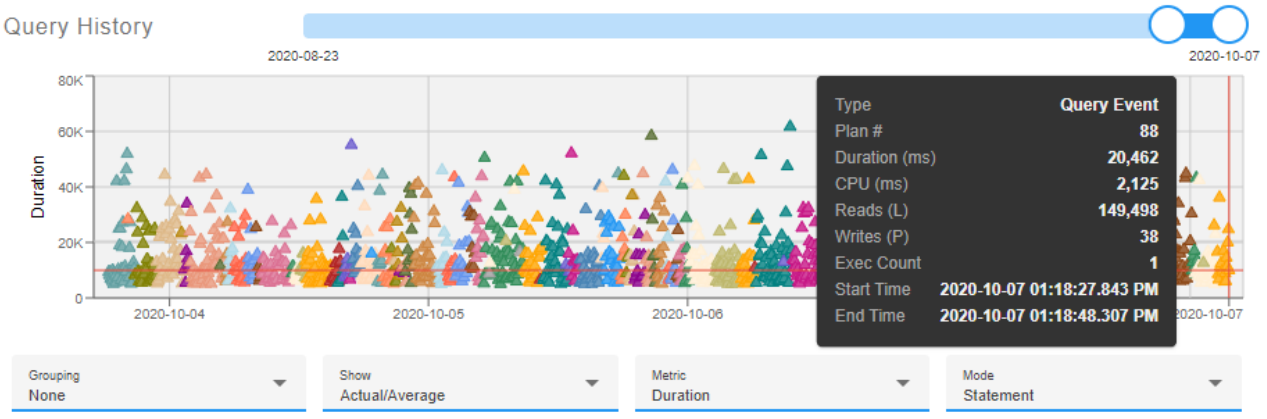
Note: The **Copy** button is only available when you are using **HTTPS** (requires an SSL certificate for your SQL Sentry Portal installation).

Query History

Select a query or statement, then view the **Query History** for it. **Query History** displays a graphical representation of the selected query over a specified range of time. **Query History** provides information about the query execution plans, if and when they were changed, and how they impacted different resources.

Query Event

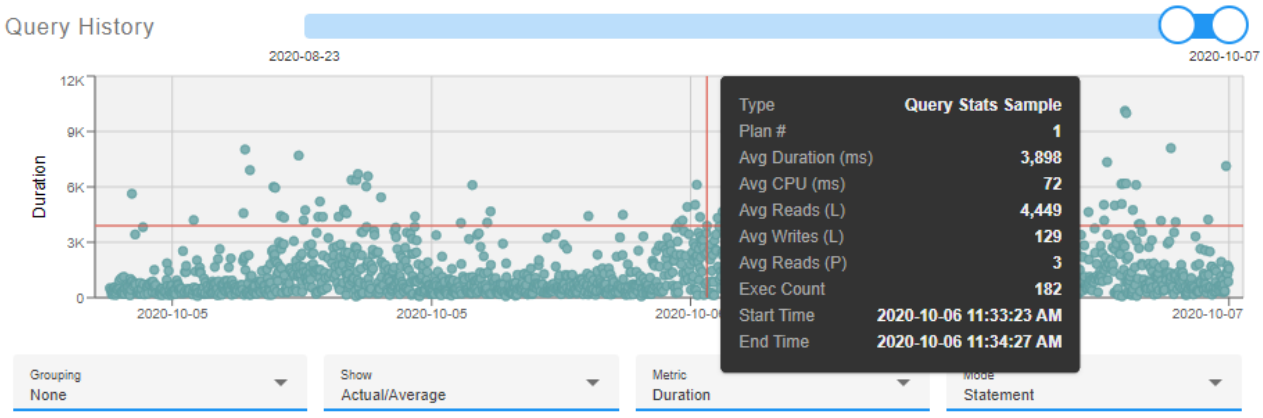
Each triangle represents a **Query Event**.



Note: The triangle colors represent execution plans. Triangles of the same color are using the same plan. If there is excessive plan drift (beyond 25 plans), then the 25 colors will start to be reused in the same order. Select a specific event to the **Plan #** represented.

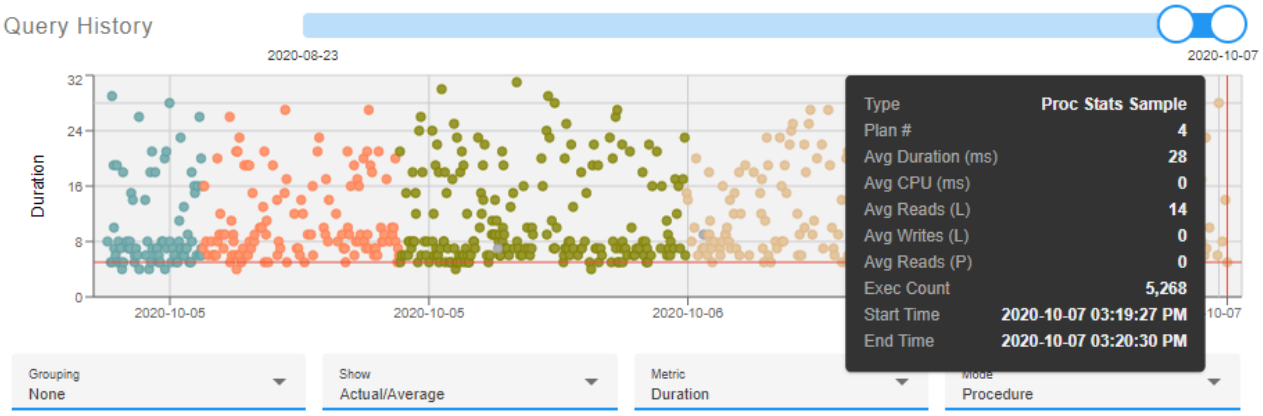
Query Stats Sample

Each dot represents a **Query Stats Sample** or a Proc Stats Sample.



Proc Stats Sample

Each dot represents a **Proc Stats Sample** or a Query Stats Sample.



> **Note:** The **Mode** for the **Proc Stats Sample** is set to *Procedure*.

Additional Options

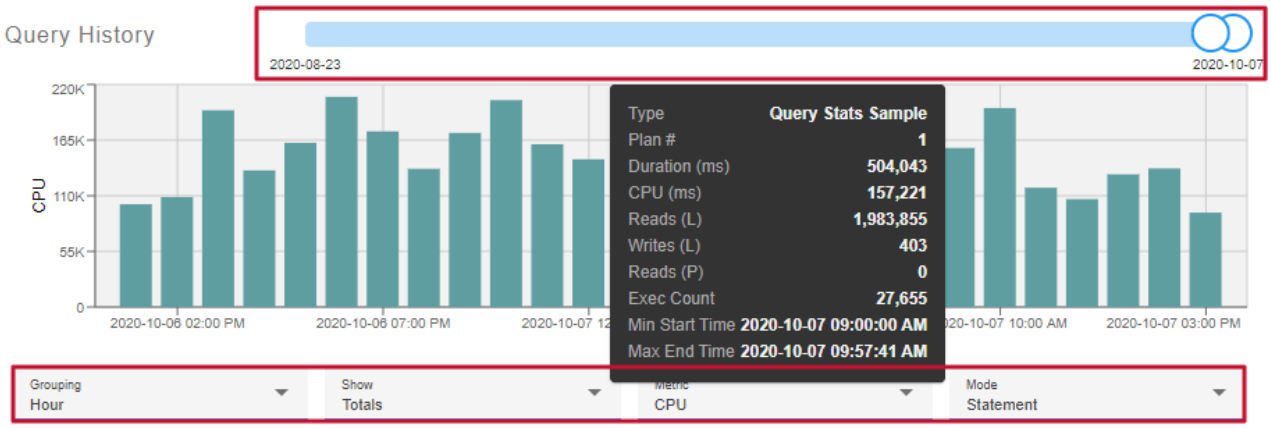
Use the options below the chart to adjust the **Grouping**, **Show**, **Metric**, **Mode**, or **Dates** slider window.

- **Grouping**
  - *None*
  - *Hour*
  - *Day*
  - *Week*
- **Show**
  - *Actual/Average*

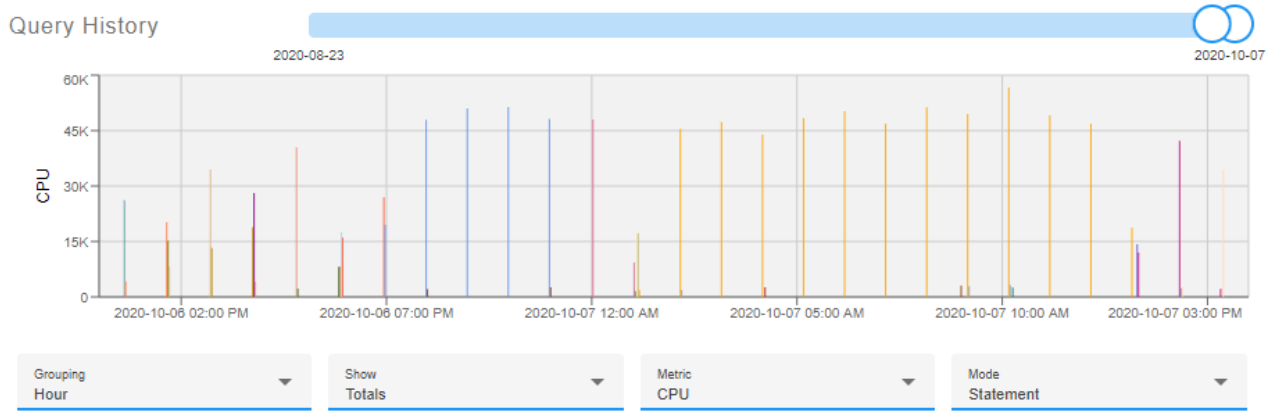
    - *Totals*
- **Metric**
    - *Duration*
    - *CPU*
    - *IO*
- **Mode**
    - *Procedure*
    - *Statement*

> **Note:** In *Procedure* mode, the chart reflects changes in the procedure stats (plan_handle), whereas *Statement* mode displays the changes in query stats (query_plan_hash).

Example with **Grouping** by *Hour*, **Show** *Totals*, **Metric** *CPU*, and **Mode** *Statement*:



An additional example with multiple plans represented:



Additional Information:

- Multiple Plans for an "Identical" Query blog post by Aaron Bertrand on SQLPerformance

- Different Plans for "Identical" Servers blog post by Aaron Bertrand on SQL Performance
- Analyzing "death by a thousand cuts" workloads blog post by Erin Stellato on SQLPerformance
- How useful are query_hash and query_plan_hash for troubleshooting? blog post by Jonathan Kehayias on SQLskills