



WINDOW-LEVEL TOOL

Query

A full SQL editor — write, run, cancel, and tune any SQL statement or batch, review every result set, and see how SQL Server executes it.

Pro tier. The Query window, including Query History, is a Pro feature; Base users see an upgrade prompt instead.

Who

WHO IT IS FOR

Users comfortable with SQL who want a fast editor for ad-hoc statements, scripts, stored-procedure work, and query tuning.

What

WHAT IT DOES

The Query window has a syntax-highlighted SQL editor with line numbers on top and a tabbed results pane below, separated by a draggable splitter. Click Execute or press F5 to run the whole script, or only the selected text; Cancel stops a running query on the server. Scripts split on GO run as separate batches, each result set gets its own “Results #” tab, and the Messages tab numbers each batch’s output. Use a variable you never declared, such as @Address, and Execute prompts for its value instead of failing; after a successful prompted run you can choose Add to write the matching DECLARE lines into the script. Plan shows the estimated plan without running anything, and Actual runs the SQL inside a transaction that is always rolled back to measure each operator’s real time and rows (it asks first if the SQL changes data). Either produces a plan grid with the two costliest operators coloured red and orange, a Suggestions tab listing up to five ways to make the SQL faster with SQL to try, a Missing indexes tab with CREATE INDEX statements to review, and an SSMS-style plan diagram you can save as a .sqlplan file; clicking a plan row, suggestion, or diagram box bookmarks the editor lines it came from. 🕒 History lists every query and search run against the database so you can load one back. The Custom SQL dropdown is your personal query library — save, load, delete, export, and import named queries. Each result tab can be charted, exported to CSV, Excel, JSON, TSV, Markdown, or HTML (or copied to the clipboard), filtered or sorted by clicking a column header, and formatted by right-clicking a Date, DateTime, or numeric header.

When

WHEN TO USE IT

Use it for ad-hoc exploration, debugging stored procedures, testing schema changes, running one-off administrative scripts, or finding out why a query is slow.

Where

WHERE TO FIND IT

Open it from the Query ... button at the top right of the database window. The toolbar runs left to right: Execute, Plan, Actual, Cancel, the Results toggle (show or hide the results pane), the Custom SQL dropdown with its add and delete icons, , History, then Find, Replace, Word Wrap, Save SQL (to a .sql file), Compare (against the last SQL you ran), export and import of your Custom SQL library, the connection icon, the camera icon, and the lightbulb Tip of the Day. Result, plan, and Messages tabs sit along the top of the results pane, with the export and chart icons at its right when a result tab with rows is active; a status bar with a spinner and row counts runs along the bottom. Paste with Ctrl+V — right-click paste is not available.

Why

WHY IT MATTERS

Sometimes the fastest path is to type the SQL. The Query window gives you a proper editor, results you can chart and export, and a plan analyser that points at the slow part of a query, without leaving McSaver Search.

SCREENSHOTS

McSaver Search — Query

Query - AdventureWorks2019 [your-server]:9

```
-- Select Custom SQL --
```

Execute Plan Actual Results

```

1 SELECT TOP 100000
2     [Sales.SalesOrderHeader].[OrderDate] AS [Order Date],
3     [Sales.SalesOrderHeader].[TotalDue] AS [Total Due],
4     [Sales.SalesTerritory: ].[Name] AS [Sales Territory + Name]
5 FROM [AdventureWorks2019].[Sales].[SalesOrderHeader] AS [Sales.SalesOrderHeader] WITH (NOLOCK)
6 LEFT JOIN [Sales].[SalesTerritory] AS [Sales.SalesTerritory: ] WITH (NOLOCK) ON [Sales.SalesTerritory: ].[TerritoryID] = [Sales.SalesOrderHeader].[TerritoryID]
7 WHERE
8     ([Sales.SalesOrderHeader].[OrderDate] BETWEEN '2010-01-01' AND '2020-01-01' )
9 ORDER BY [Order Date]
```

Messages Results #1: 31 k

Order Date	Total Due	Sales Territory -- Name
2011-05-31 00:00:00	23153.2239	Southeast
2011-05-31 00:00:00	1457.3288	Southeast
2011-05-31 00:00:00	36865.8012	Canada
2011-05-31 00:00:00	32474.9324	Canada
2011-05-31 00:00:00	472.3108	Southwest
2011-05-31 00:00:00	27510.4109	Northwest
2011-05-31 00:00:00	16158.6961	Northwest
2011-05-31 00:00:00	5694.8564	Southwest
2011-05-31 00:00:00	6876.3649	Central
2011-05-31 00:00:00	40487.7233	Canada
2011-05-31 00:00:00	807.2585	Northwest
2011-05-31 00:00:00	6893.2549	Central
2011-05-31 00:00:00	9153.6054	Northwest
2011-05-31 00:00:00	6895.4100	Canada
2011-05-31 00:00:00	4216.0258	Northeast
2011-05-31 00:00:00	2955.0542	Canada
2011-05-31 00:00:00	6434.0848	Central
2011-05-31 00:00:00	15992.7446	Southeast
2011-05-31 00:00:00	8773.6810	Canada
2011-05-31 00:00:00	11036.3964	Southwest
2011-05-31 00:00:00	1481.1742	Canada
2011-05-31 00:00:00	12832.9009	Southwest
2011-05-31 00:00:00	15524.0686	Southeast
2011-05-31 00:00:00	4363.7107	Northeast
2011-05-31 00:00:00	48284.0662	Northwest
2011-05-31 00:00:00	6301.6258	Southeast

Completed in 335 ms | Last execute: 10/5/2026 6:23:50 PM

Execute: the Searcher's query run in the Query window, with its result set on the Results #1 tab.

McSaver Search — Query

Show estimated plan: compile the SQL (or the selection) without running it, and list the costliest operators, missing indexes and plan warnings.

Execute Plan Actual Results

-- select custom SQL --

```

1 |SELECT TOP 100000
2 |    [Sales.SalesOrderHeader].[OrderDate] AS [Order Date],
3 |    [Sales.SalesOrderHeader].[TotalDue] AS [Total Due],
4 |    [Sales.SalesTerritory: ].[Name] AS [Sales Territory → Name]
5 |FROM [AdventureWorks2019].[Sales].[SalesOrderHeader] AS [Sales.SalesOrderHeader] WITH (NOLOCK)
6 |LEFT JOIN [Sales].[SalesTerritory] AS [Sales.SalesTerritory: ] WITH (NOLOCK) ON [Sales.SalesTerritory: ].[TerritoryID] = [Sales.SalesOrderHeader].[TerritoryID]
7 |WHERE
8 |    ( [Sales.SalesOrderHeader].[OrderDate] BETWEEN '2010-01-01' AND '2020-01-01' )
9 |ORDER BY [Order Date]

```

Messages
Estimated plan: 1 statement(s)
Suggestions: 2
Missing indexes: 1
Estimated plan diagram

Stmt	Lines	Op %	Operator	Object	Est. Rows	Op Cost	Subtree Cost
1	2, 8	72.1	sort (TopN Sort)		31465	2.1693	3.0072
1	5-6	8.6	↳ Hash Match (Right Outer Join)		31465	0.2583	0.8379
1	6	0.1	↳ Index Scan	[Sales].[SalesTerritory].[AK_SalesTerritory_Name]	10	0.0033	0.0033
1	8	0.9	↳ Filter		31465	0.0277	0.5763
1	3	0.1	↳ compute Scalar		31465	0.0031	0.5486
1	3	0.1	↳ compute Scalar		31465	0.0031	0.5455
1	5	18	↳ Clustered Index Scan	[Sales].[SalesOrderHeader].[PK_SalesOrderHeader_SalesOrderID]	31465	0.5423	0.5423

Estimated plan in 156 ms (nothing was executed) | 10/5/2026 6:24:15 PM

Plan: the estimated plan, compiled without running anything; the two costliest operators are coloured in Op %.

McSaver Search — Query

Actual plan: RUNS the SQL (or the selection) to measure it, inside a transaction that is then rolled back, and shows each operator's real time (ms) and rows. Asks first if the SQL changes data.

Execute Plan Actual Results Select Custom SQL

History

```

1 SELECT TOP 100000
2     [Sales.SalesOrderHeader].[OrderDate] AS [Order Date],
3     [Sales.SalesOrderHeader].[TotalDue] AS [Total Due],
4     [Sales.SalesTerritory: ].[Name] AS [Sales Territory → Name]
5 FROM [AdventureWorks2019].[Sales].[SalesOrderHeader] AS [Sales.SalesOrderHeader] WITH (NOLOCK)
6 LEFT JOIN [Sales].[SalesTerritory] AS [Sales.SalesTerritory: ] WITH (NOLOCK) ON [Sales.SalesTerritory: ].[TerritoryID] = [Sales.SalesOrderHeader].[TerritoryID]
7 WHERE
8     ([Sales.SalesOrderHeader].[OrderDate] BETWEEN '2010-01-01' AND '2020-01-01' )
9 ORDER BY [Order Date]

```

Messages
Actual plan: 1 statement(s)
Suggestions: 2
Missing indexes: 1
Actual plan diagram

Stmt	Lines	Own ms	Elapsed ms	CPU ms	Actual Rows	Est. Rows	Op %	Operator	Object
1	2, 8	19	31	31	31465	31465	72.1	Sort (TopN Sort)	
1	5-6	6	12	12	31465	31465	8.6	↳ Hash Match (Right Outer Join)	
1	6	0	0	0	10	10	0.1	↳ Index Scan	[Sales].[SalesTerritory].[AK_SalesTerritory_Name]
1	8	2	6	6	31465	31465	0.9	↳ Filter	
1	3	NULL	NULL	NULL	NULL	31465	0.1	↳ compute scalar	
1	3	NULL	NULL	NULL	NULL	31465	0.1	↳ compute scalar	
1	5	4	4	4	31465	31465	18	↳ clustered Index Scan	[Sales].[SalesOrderHeader].[PK_SalesOrderHeader_SalesOrderID]

Actual plan in 213 ms (ran, then rolled back) | 10/5/2026 6:24:42 PM

Actual: the query ran inside a transaction that was rolled back, with Own ms, Elapsed ms, CPU ms and Actual Rows per operator.

McSaver Search — Query

Query - AdventureWorks2019 [your-server]: 9

Execute

Plan

Actual

Results

-- Select Custom SQL --

History

```

1 SELECT TOP 100000
2   [Sales.SalesOrderHeader].[OrderDate] AS [Order Date],
3   [Sales.SalesOrderHeader].[TotalDue] AS [Total Due],
4   [Sales.SalesTerritory: ].[Name] AS [Sales Territory → Name]
5 FROM [AdventureWorks2019].[Sales].[SalesOrderHeader] AS [Sales.SalesOrderHeader] WITH (NOLOCK)
6 LEFT JOIN [Sales].[SalesTerritory] AS [Sales.SalesTerritory: ] WITH (NOLOCK) ON [Sales.SalesTerritory: ].[TerritoryID] = [Sales.SalesOrderHeader].[TerritoryID]
7 WHERE
8   ([Sales.SalesOrderHeader].[OrderDate] BETWEEN '2010-01-01' AND '2020-01-01' )
9 ORDER BY [Order Date]

```

Messages

Actual plan: 1 statement(s)

Suggestions: 2

Missing indexes: 1

Actual plan diagram

Stmt	Lines	Own ms	Elapsed ms	CPU ms	Actual Rows	Est. Rows	Op %	Operator	Object
1	2, 8	19	31	31	31465	31465	72.1	Sort (TopN Sort)	
1	5-6	6	12	12	31465	31465		Left Outer Join	
1	6	0	0	0	10	10	0.1	Index Scan	[Sales].[SalesTerritory].[AK_SalesTerritory_Name]
1	8	2	6	6	31465	31465	0.9	Filter	
1	3	NULL	NULL	NULL	31465	31465	0.1	compute Scalar	
1	3	NULL	NULL	NULL	31465	31465	0.1	compute Scalar	
1	5	4	4	4	31465	31465	18	clustered Index Scan	[Sales].[SalesOrderHeader].[PK_SalesOrderHeader_SalesOrderID]

Actual plan in 213 ms (ran, then rolled back) | 10/5/2026 6:24:42 PM

Click a plan row to bookmark the editor lines it came from (line 8 here).

McSaver Search — Query

Query - AdventureWorks2019 [your-server]: 9

Execute

Plan

Actual

Results

-- Select Custom SQL --

History

```

1 SELECT TOP 100000
2     [Sales.SalesOrderHeader].[OrderDate] AS [Order Date],
3     [Sales.SalesOrderHeader].[TotalDue] AS [Total Due],
4     [Sales.SalesTerritory: ].[Name] AS [Sales Territory → Name]
5 FROM [AdventureWorks2019].[Sales].[SalesOrderHeader] AS [Sales.SalesOrderHeader] WITH (NOLOCK)
6 LEFT JOIN [Sales].[SalesTerritory] AS [Sales.SalesTerritory: ] WITH (NOLOCK) ON [Sales.SalesTerritory: ].[TerritoryID] = [Sales.SalesOrderHeader].[TerritoryID]
7 WHERE
8     ([Sales.SalesOrderHeader].[OrderDate] BETWEEN '2010-01-01' AND '2020-01-01' )
9 ORDER BY [Order Date]

```

Messages

Actual plan: 1 statement(s)

Suggestions: 2

Missing indexes: 1

Actual plan diagram

Stmt

Lines

Impact %

Table

Suggested index (review before creating)

1

1-9

18

[Sales].[SalesOrderHeader]

CREATE NONCLUSTERED INDEX [IX_SalesOrderHeader_OrderDate] ON [Sales].[SalesOrderHeader] ([OrderDate]) INCLUDE ([TerritoryID], [TotalDue]);

Actual plan in 213 ms (ran, then rolled back) | 10/5/2026 6:24:42 PM

Missing indexes: the index SQL Server asked for, with a CREATE INDEX statement to review.

● ● ●
McSaver Search — Query

Query - AdventureWorks2019 [your-server]: 9

Execute
Plan
Actual
Results

-- Select Custom SQL --

History

```

1 SELECT TOP 100000
2     [Sales.SalesOrderHeader].[OrderDate] AS [Order Date],
3     [Sales.SalesOrderHeader].[TotalDue] AS [Total Due],
4     [Sales.SalesTerritory: ].[Name] AS [Sales Territory → Name]
5 FROM [AdventureWorks2019].[Sales].[SalesOrderHeader] AS [Sales.SalesOrderHeader] WITH (NOLOCK)
6 LEFT JOIN [Sales].[SalesTerritory] AS [Sales.SalesTerritory: ] WITH (NOLOCK) ON [Sales.SalesTerritory: ].[TerritoryID] = [Sales.SalesOrderHeader].[TerritoryID]
7 WHERE
8     ( [Sales.SalesOrderHeader].[OrderDate] BETWEEN '2010-01-01' AND '2020-01-01' )
9 ORDER BY [Order Date]

```

Messages
Actual plan: 1 statement(s)
Suggestions: 2
Missing indexes: 1
Actual plan diagram

1. Avoid the sort of 31,465 rows
~35% of the run time

Statement 1, lines 2, 8
A Sort reads all of its input before returning a row; this one is 35% of the run time.

CREATE NONCLUSTERED INDEX [IX_salesorderheader_OrderDate] ON [Sales].[SalesOrderHeader] ([OrderDate]); -- rows come out already in order
Copy

2. Create the index SQL Server asks for on [Sales].[SalesOrderHeader]
~18% of the run time

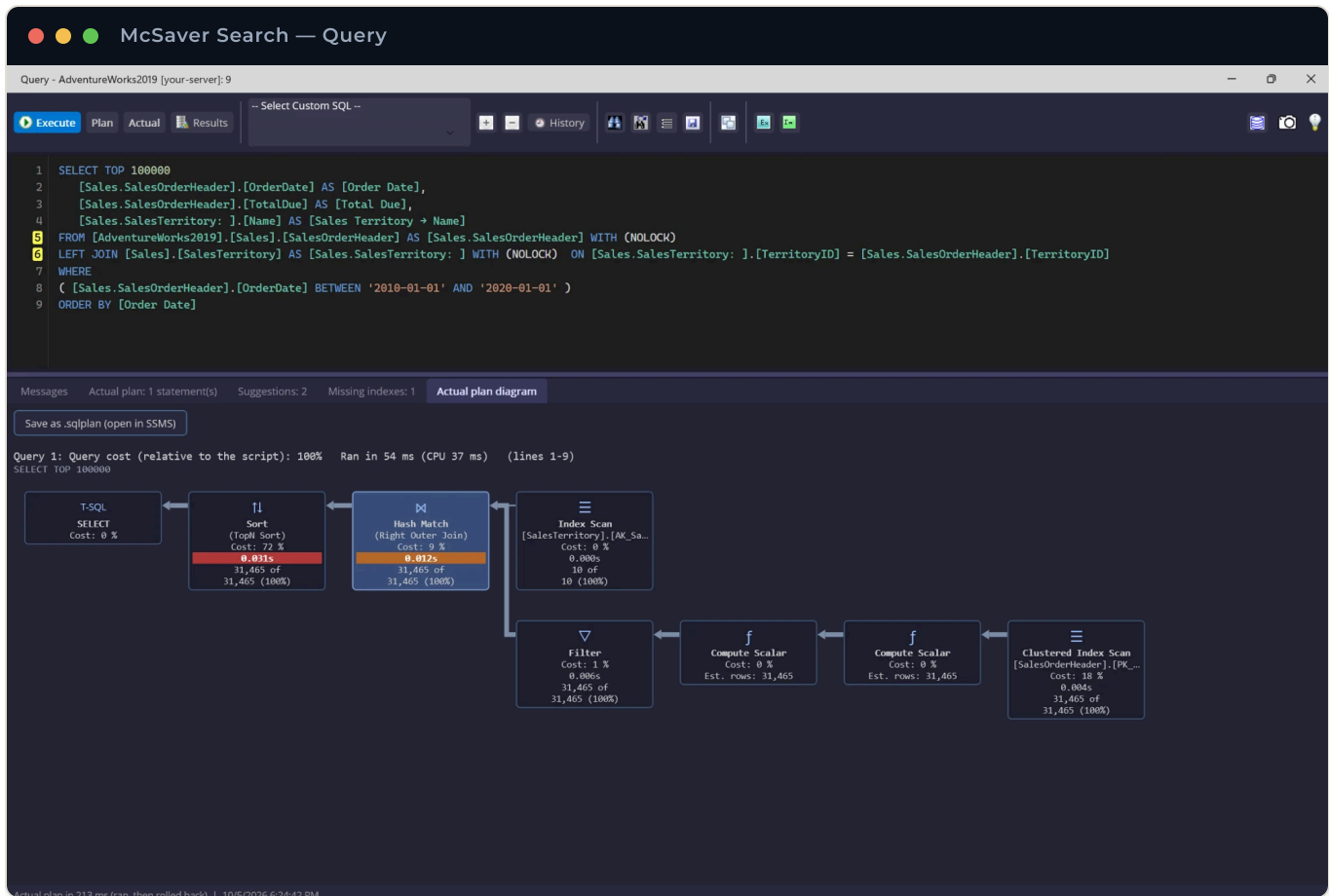
Statement 1, lines 5, 8
The optimizer estimates it would cut statement 1's cost by 18%. Check it against the table's existing indexes first: widening one may do instead.

CREATE NONCLUSTERED INDEX [IX_SalesOrderHeader_OrderDate] ON [Sales].[SalesOrderHeader] ([OrderDate]) INCLUDE ([TerritoryID], [TotalDue]);
Copy

Read from the plan, not tested: try one change at a time and run Actual before and after to see what it gained. The percentage is how much of the run the problem accounts for, so the most there is to gain.

Actual plan in 213 ms (ran, then rolled back) | 10/5/2026 6:24:42 PM

Suggestions: ways to make the query faster, ranked by their share of the run time, each with SQL to try and a Copy button.



The actual plan diagram, drawn the way SSMS draws it; the selected Hash Match bookmarks lines 5 and 6.

Tip of the Day 49 tips

Every Tip of the Day for this tab — open them in the app from the lightbulb icon.

VALUE Write and execute any SQL statement with syntax highlighting, multi-result-set support, and a built-in query library — all without leaving the app.

OVERVIEW The Query window pairs a full SQL editor on top with a tabbed results pane below, connected by a draggable splitter so you can balance editing and output space.

ACTION Click Execute (or press F5) to run the SQL in the editor; if text is selected, only the selection runs — otherwise the entire editor content executes.

ACTION Toggle the Results button to show or hide the results pane below the editor.

NAVIGATION Use the Custom SQL dropdown to load a previously saved query into the editor.

ACTION Click the Add Custom SQL icon to save the current editor contents under a name you choose, adding it to your personal query library.

ACTION Click the Delete Custom SQL icon to remove the currently selected saved query from your library.

ACTION Click the Find icon to search for text within the editor; click Replace to find and replace.

ACTION	Toggle Word Wrap to control whether long lines wrap or scroll horizontally in the editor.
EXPORT	Click the Save SQL icon to write the current editor contents to a .sql file on disk.
ACTION	Click the Compare icon to diff the current editor text against the last SQL you executed, so you can see what changed.
ACTION	Click Export Custom SQLs to save your entire query library to a file for backup or sharing with a colleague.
ACTION	Click Import Custom SQLs to load a previously exported query library file back into the editor.
ACTION	Click the Connection icon to switch which database the Query window runs against without closing the window.
EXPORT	Click the camera icon to capture a screenshot of the Query window; it saves to your export folder and opens that folder automatically.
VISUAL	The status bar at the bottom shows execution progress and row counts; an activity spinner appears while a query is running.
FEATURE	Results appear in tabs — each result set from a batch gets its own tab, and a Messages tab shows print output and error messages.
FEATURE	Every query you execute is recorded to a shared SQL Change Log — the same log the Table Editor and Finder feed — opened from the Show SQL Log icon in the database window header.
TECHNIQUE	Write multiple statements separated by GO to execute them as separate batches, each producing its own result tab.
FEATURE	The editor uses a monospace font (Consolas) with line numbers and a dark theme for comfortable reading.
FEATURE	The Query window is a Pro feature — without a Pro license you will see an upgrade prompt instead.
FEATURE	Paste SQL into the editor with the keyboard (Ctrl+V, or Cmd+V on macOS) — right-click paste is not available, so the keyboard shortcut is the paste path.
ACTION	Click the chart icon on the results tab bar to open the focused result set in the interactive chart window, where you pick the axes and plotting operation; it appears only when a result tab with rows is active.
EXPORT	Click the export icon on the results tab bar to save the focused result set — the options popup offers CSV, Excel, JSON, TSV, Markdown, and HTML, and exports the full result set rather than only the on-screen rows.
EXPORT	Check "Export to Clipboard" in the export popup to copy the result set instead of saving a file; Excel is copied as tab-separated text, so it pastes straight into Excel or Sheets as cells.
TECHNIQUE	Right-click a Date, DateTime, or Numeric column header in the results grid to pick a display format for that column; it re-renders the column and is display-only, leaving the query and data untouched.

FILTER	Click a column header in the results grid to open the Filter/Sort popup and filter or sort that result set by the column; it only changes what is shown, and the header's  marker tells you a filter or sort is active.
FEATURE	When a script split on GO runs, the Messages tab numbers each batch's output — "1.", "2.", and so on — so you can tell which batch printed what; a batch that returns no result sets reads "Commands completed successfully."
PERFORMANCE	Each result tab builds its grid the first time you open it and keeps it, so switching back and forth between result tabs is instant.
FEATURE	Use a variable you never declare — for example WHERE x = @Address — and Execute asks for its value before running, instead of failing with "Must declare the scalar variable"; SQL Server works out which variables are undeclared without running anything.
TECHNIQUE	At a variable prompt, a whole number is sent as bigint and a decimal as decimal; wrap the value in 'quotes' to send it as text (useful for zip codes like '00123'), or type NULL for a null. Anything else is sent as text.
VISUAL	After a run that prompted for variables, the values used are listed at the top of the Messages tab, and the next run pre-fills each prompt with the value you entered last time.
WORKFLOW	After a prompted run succeeds, choose Add to write matching DECLARE lines at the top of the script so it runs without prompting next time — use Compare to see exactly what was added.
FEATURE	If adding a foreign key fails because existing rows break it, click "Show offending rows" in the error to open a Query window pre-filled with, and already running, a SELECT of exactly the blocking child rows.
ACTION	Click Cancel beside Execute to stop a running query on the server; closing the window also cancels it.
FEATURE	Click  History on the toolbar to see every query and search run against this database — newest first, with when, from where, the outcome, rows and time; Load SQL puts it in the editor, and Copy SQL, Delete and Clear all tidy the list. Viewing history is Pro; it is recorded on every tier.
PERFORMANCE	Click Plan to see the estimated plan without running anything — SQL Server only compiles the SQL (or your selection) and the Estimated plan tab lists every operator, root first, with its cost and the editor lines it came from.
PERFORMANCE	Click Actual to measure the SQL for real: it runs inside a transaction that is always rolled back and shows each operator's Own ms, Elapsed ms, CPU ms and Actual Rows next to the estimates; it asks first if the SQL changes data or schema.
SAFETY	If a script commits its own transaction, Actual stops at that batch and says so in Messages — the rollback can't undo work the script already committed.
VISUAL	In a plan grid the two costliest operators of each statement are coloured — red, then orange — in the Op % column (estimated) or Own ms column (actual); hover the cell to see its rank.
NAVIGATION	Click a row in a plan or Missing indexes tab to bookmark the editor lines it came from, so you can see exactly which join, filter or ORDER BY the operator belongs to.

FEATURE	The Suggestions tab lists up to five ways to make the SQL faster — missing indexes, filters that can't seek (LIKE '%...!', functions on a column), key lookups, sorts, tempdb spills, implicit conversions, bad row estimates and more — ranked by the share of time or cost each accounts for.
ACTION	Each Suggestions card explains why the plan points to the fix and gives SQL to try with a Copy button; click the card to bookmark the editor lines it is about.
FEATURE	The Missing indexes tab shows the indexes SQL Server wished it had, each with a CREATE INDEX statement to review — check it against your existing indexes before you run it.
VISUAL	The plan diagram tab draws each statement the way SSMS does: boxes show the operator, object and cost (plus time and actual-of-estimated rows for an actual plan), arrows thicken with the rows they carry, and the two costliest boxes are coloured. Click a box to bookmark its lines; hover for details.
EXPORT	Click Save as .sqlplan on the plan diagram to save every statement's plan in one file and open it in SSMS as a graphical plan.
TECHNIQUE	Plan a CREATE or ALTER PROCEDURE and you get the plan of its body — parameters are declared for you — and a CREATE or ALTER VIEW is planned as its SELECT; Messages notes when this happens.
CLARIFICATION	Plan and Actual need the SHOWPLAN permission; if your login lacks it, the error tells you the GRANT SHOWPLAN to ask your DBA for.
ACTION	Click the empty space below a short script to put the caret after the last character; Shift+click there extends the selection to the end.