FOR SQL SERVER MANAGEMENT STUDIO 21 / 22

SSMS SQL Formatter

User Guide & Reference — rule-based and AI-assisted T-SQL formatting, comment handling, SELECT * expansion, and Excel/CSV export for SQL Server Management Studio.

Version **2.12.0** • Updated 2026-08-24 • Made by DesignAlign

Download the latest release

Contents

1. Quick reference

The extension only defines 4 global keyboard shortcuts — everything else lives in the **Tools** menu (or right-click in the editor for Format T-SQL Script). A dedicated **SQL Formatter** toolbar with the same four most-used commands (Format, Preview, Expand SELECT *, Export to Excel) also docks automatically the first time the extension loads.

Shortcut	Action
`Ctrl+Shift+Alt+F`	Format T-SQL Script
`Ctrl+Shift+Alt+P`	Preview Format
`Ctrl+Shift+Alt+X`	Export Results to Excel
`Ctrl+Shift+Alt+A`	Add Results as Sheet

Tools menu (no shortcut)	Action
Format T-SQL Script Help	Usage summary + link to docs, shown inside SSMS
Open Results in Excel	Write copied results to a temp workbook and open it
Expand SELECT *	Resolve SELECT * to explicit columns, with preview
Format Files…	Format one or more .sql files on disk
Format All Open Files	Format every open .sql document, one undo per window
Export / Import Formatter Settings	Save or load the whole configuration as JSON

With text selected, Format T-SQL Script and Preview Format apply to the selection only; otherwise the whole document. Ctrl+Z always undoes a format as one step.

2. Getting started

SSMS doesn't expose the Visual Studio Marketplace in its own extension manager, so installation is manual: open the SSMS SQL Formatter Marketplace listing, select **Download**, then double-click `SsmsSqlFormatter.vsix` (the installer should list "SQL Server Management Studio 21/22" as a target — tick it and install) and restart SSMS. If the installer doesn't offer SSMS as a target, run `VSIXInstaller.exe` directly from the SSMS installation folder against the downloaded VSIX.

Once installed, open a query window and press `Ctrl+Shift+Alt+F`, right-click → **Format T-SQL Script**, or use the Tools menu. With text selected, only the selection is formatted; otherwise the whole document. The change lands as a single undo step, so `Ctrl+Z` reverts the entire format in one press.

> A script that fails to parse is always left completely untouched — you'll see the parse error with line/column instead of a partially-mangled file. This applies everywhere: the main command, format-on-save, format-on-paste, batch formatting, and the CLI.

All settings live under **Tools → Options → Format T-SQL Script** (General and AI Engine pages). A **Format T-SQL Script Help** entry in the Tools menu shows a quick usage summary without leaving SSMS.

Microsoft does not officially support extensions in SSMS 21/22 — they're tolerated and widely used, but if SSMS misbehaves with an extension installed, Microsoft won't help. Keep the VSIX handy so you can uninstall via VSIXInstaller if needed.

3. Formatting engines

Set under **Tools** → **Options** → **General** → **Formatting engine**. Everything except the interactive Format T-SQL Script command (and Expand SELECT *) always uses Rule-based, regardless of this setting.

Engine	How it works	Strengths	Trade-offs
Rule-based	Microsoft's ScriptDom parser + script generator, fully offline	Instant, deterministic, understands all T-SQL	Regenerating from the parse tree loses comments on its own — see Comment handling
AI	Anthropic Claude Messages API, or GitHub Copilot / any OpenAI-compatible chat endpoint, using your own key or token	Preserves comments exactly, follows free-form style instructions ("leading commas", "align equals signs")	Needs network + a key/token; script text is sent to the provider

Two built-in style presets apply to the Rule-based engine (set under **Style preset**): **Classic** (old-format compact — trailing commas, inline JOINs, fewer line breaks) and **Modern** (each column/JOIN/predicate on its own line, aligned bodies). Choose **Custom** to control every individual option yourself — see the next section.

4. Style presets & basics

These live in **Tools** → **Options** → **General**, category **02. Basics**. They only take effect when Style preset is **Custom** — Classic and Modern use their own built-in combinations.

Option	What it does
Keyword casing	Uppercase, Lowercase, or PascalCase for T-SQL keywords.
Built-in function casing	Casing for built-in function names (GETDATE, COUNT, ISNULL…). Only names immediately followed by '(' are affected, so a column sharing a function's name is left alone.
Data type casing	Casing for data type keywords (INT, VARCHAR, DATETIME…).
Indent size (spaces)	Default 4.
Indent with tabs	Converts each indent level into a tab character.
Re-indent subqueries	Guarantees every nested subquery/derived table is indented at least one level per nesting depth. Only ever adds indentation.
Comma placement	Trailing (end of line) or Leading (start of next line — handy for easy column commenting). Applies to every preset.
Add semicolons	Terminate statements with semicolons.
Max line length (0 = off)	Wraps a top-level comma-separated list (SELECT/GROUP BY/ORDER BY) one item per line once it exceeds this many characters. Doesn't apply inside parentheses (IN-lists, function arguments).
AS keyword on its own line	Places AS on a new line in views/procedures (Custom preset only).

5. Line breaks, lists & alignment

03. Line breaks (Custom preset only)

Each of these independently controls whether that clause starts on a new line: **New line before FROM / WHERE / JOIN / GROUP BY / ORDER BY / HAVING / OUTPUT / OFFSET**, plus **New line before (/)** for the opening and closing parenthesis of a multiline list. All default to on.

04. Lists

Option	What it does
Multiline SELECT list	Each selected column on its own line.
Multiline WHERE predicates	Each AND/OR predicate on its own line.
Multiline INSERT lists	Same, for INSERT column/value lists.
Align clause bodies	Aligns the body of clauses (SELECT list, SET clauses, etc.).
Align column definitions	Aligns column definition fields in CREATE TABLE.
Multiline view columns	Each column in a view's column list on its own line.
Indent view body / Indent SET clause	Indents the body of a view, or an UPDATE statement's SET clause.
Align '=' in assignments *(off by default)*	In runs of 2+ consecutive "name = expression" lines (SET clauses, old-style SELECT aliasing), pads the shorter left-hand sides so every '=' lines up. A line with more than one top-level '=' (a comparison, not an assignment) breaks the run and is left alone.
Align ON in JOIN clauses *(off by default)*	Condenses each JOIN onto one line ("JOIN table ON condition" — ScriptDom's generator otherwise always splits this across three lines) and, in runs of 2+ consecutive joins, pads the shorter ones so every ON lines up.
Align THEN in CASE expressions *(off by default)*	Expands a CASE with 2+ WHEN branches onto multiple lines (ScriptDom never does this on its own) and pads the shorter conditions so every THEN lines up. A single-branch CASE is left untouched.

6. Blank lines & GO

Category **05. Blank lines and GO**. Every count below accepts **-1** to mean "leave as-is."

Option	What it does
Blank lines before / after GO	Exact blank-line count around each GO batch separator. Set both to -1 to leave GO spacing untouched entirely.
Blank lines between statements	Exact blank-line count between consecutive statements, at every nesting level — top-level batch statements and statements inside BEGIN…END, IF/WHILE, TRY/CATCH, and procedure/function/trigger bodies. A comment above a statement stays attached to it (blank lines go above the comment, not between it and the statement).
Max consecutive blank lines	Collapses runs of blank lines anywhere down to this many (-1 = unlimited). Never touches blank lines inside /* */ comments; the GO settings above take precedence around GO.
Trim trailing whitespace	Removes spaces/tabs at line ends.

7. Comment handling

The rule-based engine regenerates the script from its parse tree, which doesn't retain comments on its own — **Comment handling** (category **06. Safety**) controls what happens to them instead. Comments are *never* silently deleted by default.

Mode	Behavior
Inline *(default)*	Re-attaches each comment to the same code it was next to — trailing comments stay at line ends, standalone comments keep their own line. If a comment can't be confidently repositioned, it's appended at the end under a banner instead of being dropped.
MoveToEnd	Strips every comment out of its original position and collects them all at the end of the script, in their original order.
Discard	Drops comments entirely. **Warn when script contains comments** (on by default) shows a confirmation before this happens.

The AI engine always preserves comments exactly, regardless of this setting. A separate **Enable formatting cache** option (off by default, also under 06. Safety) caches identical script+settings combinations in memory to speed up repeated formatting.

8. Expand SELECT *

Replaces `SELECT *` (and `alias.*`) with an explicit, ordered column list resolved from the real table/view structure — using whatever database the active query window is already connected to, so there's nothing extra to configure. Anywhere it can't confidently resolve the source, it leaves that `SELECT *` untouched, decided independently for each occurrence in the script — it never guesses.

Two ways to use it

- Turn on **Expand SELECT * when formatting** under **Tools → Options → General** (category **11. SELECT * expansion**, off by default) to have it run automatically as part of Format T-SQL Script.
- Or run **Tools → Expand SELECT *** directly at any time, regardless of that setting — it applies directly, with no preview step, by default. Turn on **Show preview before applying** (same category) to see a preview first instead, the same as Preview Format.

> Requires the query window to actually be connected to a database. If no connection can be determined, the command tells you so and nothing changes.

What it can and can't resolve

Resolves plain base tables and views reachable through the connection — including old-style system compatibility views like `sysobjects` and modern catalog views/DMVs like `sys.all_columns` or `sys.dm_exec_sessions` — and four-part linked-server references (`LinkedServer.Database.Schema.Table`), by querying the linked server's own catalog directly. That only works when the linked server is itself SQL Server; anything else (Oracle, ODBC, flat file, ...) fails the same lookup and is left untouched, same as any other unresolvable table. It does **not** resolve: CTEs, derived tables/subqueries, table-valued functions, temp tables, table variables, or synonyms — any of these are simply left as `SELECT *`. Windows/Integrated Authentication connections are the reliable case; SQL Server Authentication connections may not always be resolvable.

Interactive only — this never runs on format-on-save, format-on-paste, batch formatting, or the CLI, since it requires a live database connection.

9. Format on save & format on paste

Format on save

Automatically formats a .sql document immediately before it's written to disk (Ctrl+S, Save All, closing a dirty document, etc.), using your General settings. Always uses the Rule-based engine — never AI — so saving is never delayed by a network call or a confirmation prompt. A script that fails to parse is saved untouched.

Format on paste

Automatically reformats a .sql document right after pasting multi-line text into it. Detected as a single edit that inserts multi-line text, so ordinary typing is never affected. Same Rule-based-only, parse-failures-left-untouched behavior as format on save.

Both are off by default — turn them on under **Tools** → **Options** → **General** (category **09. Format on save**).

10. Batch formatting & the CLI

Format Files…

Tools menu → pick one or more .sql files and format them on disk in place (Rule-based engine, original file encoding preserved). A file that fails to parse is left untouched and reported at the end.

Format All Open Files

Formats every currently-open .sql document in place. Each is a normal editor edit — Ctrl+Z in that window undoes it — and nothing is written to disk unless you save afterward.

Command line (CI)

`tools/SsmsSqlFormatter.Cli` builds a standalone `ssmssqlfmt.exe` — Rule-based engine only, no AI, no SSMS or Visual Studio install required — for gating a build or PR on "SQL is formatted":

```
ssmssqlfmt check <file-or-directory...> [--config settings.json]   # reports what would change, exit
ssmssqlfmt format <file-or-directory...> [--config settings.json]  # formats in place, same as Forma
```

A directory argument is searched recursively for `*.sql` files. `--config` takes the same JSON produced by **Export Formatter Settings**, so a team can share one style file between the IDE and CI. Example GitHub Actions step:

```
- name: Check SQL formatting
  run: tools\SsmsSqlFormatter.Cli\bin\Release\net48\ssmssqlfmt.exe check sql\ --config .sqlformatter
```

11. Shared team config

Drop a `.sqlformatter.json` file (same format produced by **Export Formatter Settings**) into a repo folder, and every format of a file under that folder — or a descendant folder — picks it up automatically, both in SSMS and from the CLI. It overrides just that one operation; your own Tools > Options settings are never modified or persisted. If the file is missing or fails to parse, formatting silently falls back to your own settings rather than blocking.

Turn this off with **Use folder-level .sqlformatter.json** under **Tools → Options → General** (category **10. Shared config**, on by default) if you don't want a folder's contents to ever affect formatting.

12. Exporting results to Excel & CSV

Click in the results grid, select cells (`Ctrl+A` for all), press `Ctrl+Shift+Alt+X` , then Ctrl+V in Excel — you get a real table with bold headers and, by default, every cell as Text so leading zeros and long IDs survive. **Add Results as Sheet** (`Ctrl+Shift+Alt+A`) queues additional result sets into the same workbook before exporting.

> SSMS's results grid builds its own private context menu, so right-clicking it won't show the extension's commands. Use the keyboard shortcut, the Tools menu, or add a toolbar button (**Tools** → **Customize** → **Commands** → **Toolbar** → **Standard** → **Add Command** → **Tools** → **Copy Results as Excel Table**) — a toolbar button is always visible and survives upgrades.

07. Export results to Excel

Option	What it does
Export format	Xlsx (styled workbook) or Csv (styling options below don't apply to Csv).
Output folder	Leave empty to use the Windows temp folder. Created if it doesn't exist; falls back to temp if it can't be used.
Ask where to save	Shows a Save As dialog per export instead of writing straight to the output folder.
Include query on a separate sheet	Adds a "Query" worksheet with the SQL that produced the data (Xlsx only).
Try to copy the grid automatically	Sends the grid's Copy-with-Headers keystroke before converting — only works when the grid still has focus, i.e. via the Ctrl+Shift+Alt+X shortcut. A toolbar/menu click moves focus away, so whatever you copied yourself is converted instead.
First row contains headers	Formats the first copied row as a bold header. Turn off if you copied without headers.
Paste all cells as text	Keeps leading zeros and prevents long numbers turning into scientific notation or ID-like values into dates. Disable to let Excel auto-detect types.
Paste NULL as empty cells	Converts the grid's literal "NULL" text into empty cells.

08. Excel appearance

Option	What it does
Font name / size	Applied to the exported workbook.
Bold header row	
Header background / text colour	Colour picker via the dropdown.
Show cell borders / Border colour	
Freeze header row	Keeps headers visible while scrolling long result sets.
Add AutoFilter to headers	Adds Excel filter dropdowns to the header row.
Banded rows / Band colour	Shades every second data row for readability (off by default).

13. AI Engine settings

Configured on the separate **Tools → Options → AI Engine** page.

1. Connection

Option	What it does
Provider	Anthropic (Messages API) or Copilot (Copilot-style chat endpoint).
API key / token	Your Anthropic API key or Copilot bearer token. Stored in Windows Credential Manager for the current user — never in plain text.
Model	e.g. `claude-sonnet-4-5` (Anthropic) or a chat model your Copilot-compatible endpoint supports.
API endpoint	Defaults to `https://api.anthropic.com/v1/messages`; point this at your Copilot-compatible chat completions endpoint (or any OpenAI-compatible endpoint returning a `choices[].message.content` shape) to switch providers.
Max output tokens / Timeout	Raise both for very large scripts.

2. Behaviour

Option	What it does
Custom style instructions	Free-form text appended to the prompt, e.g. "leading commas", "align equals signs in SET clauses", "keep short CASE expressions on one line".
Send General options as style guide	Translates your rule-based settings (casing, indent, line breaks) into the AI prompt so both engines produce a consistent style.
Fall back to rule-based on error	If the AI call fails (no network, bad key, timeout), silently formats with the rule-based engine instead.

3. Privacy

Confirm before sending script (on by default) asks for confirmation before sending SQL to the provider — your script text, including any embedded literals or data, leaves the machine when using the AI engine. Get an Anthropic key at console.anthropic.com (billed separately from any Claude.ai subscription).

14. Export / Import Formatter Settings

Export Formatter Settings (Tools menu) saves your current configuration to a JSON file — the same format read by **Import Formatter Settings** and by `.sqlformatter.json` / the CLI's `--config`. Use it to share one style across a team, or to keep a personal backup before experimenting with settings.

15. Tips & troubleshooting

Question	Answer
Format T-SQL Script doesn't appear when I right-click	Some SSMS builds don't add it to the query editor's private context menu automatically. The keyboard shortcut and Tools menu always work; to add the context-menu entry yourself (one-time, survives upgrades): **Tools → Customize → Commands tab → Context menu → "Editor Context Menus | Code Window" → Add Command → category Tools → Format T-SQL Script → OK → Close.**
Right-clicking the results grid shows no export option	The grid builds its own private context menu. Use `Ctrl+Shift+Alt+X`, the Tools menu, or add a toolbar button — see Exporting results to Excel & CSV, above.
My comments moved or disappeared	Check **Comment handling** under Tools > Options > General — Discard mode drops them (with a warning first). Inline or MoveToEnd never drop a comment; switch to the AI engine if you need them to stay in their exact original spot.
Expand SELECT * isn't finding my table	It only resolves plain base tables/views reachable through the active connection (including linked-server tables, when the linked server is itself SQL Server) — not CTEs, derived tables, table-valued functions, temp tables, synonyms, or non-SQL-Server linked servers. Confirm the query window shows a live connection, and that the table/linked-server name and schema are spelled correctly.
A script wasn't touched at all	By design — a script that fails to parse is always left completely untouched. You'll see the parse error with line/column instead.
Tools > Options shows an error, or won't load, right after upgrading	An in-place VSIX upgrade can occasionally leave a stale extension cache behind. Close SSMS and run `tools/ssms-diagnose-and-reset.ps1` (included in the project source) — it prints the relevant SSMS activity log entries, uninstalls the extension, and clears `ComponentModelCache` (SSMS rebuilds it automatically). Reinstall the latest VSIX afterwards, or pass `-VsixPath` to have the script do that in the same run.
How do I uninstall?	Run `tools/ssms-diagnose-and-reset.ps1` (see above), or manually: `VSIXInstaller.exe /uninstall:SsmsSqlFormatter.7f3d2a1e-5b8c-4e9f-a6d0-1c4b7e2f9a35`

SSMS SQL Formatter User Guide — generated for version 2.12.0. This document is updated alongside new features; if something you see in the extension isn't covered here, check **Tools → Format T-SQL Script Help** and get the current release from the **Visual Studio Marketplace**. Free to use, licensed under the MIT License. Made by **DesignAlign**.