MCP SERVER | NO CODE | CLOUD HOSTED

SQL Syntax Validator MCP

Stop running flawed queries against your production DB.

SQL Syntax Validator immediately audits any SQL query for structural errors before it hits your database. It uses Abstract Syntax Tree parsing to find missing commas, misplaced parentheses, or invalid join structures. This prevents runtime crashes and deadlocks caused by flawed code generated by AI agents.

A+ Quality Score 100/100

sql-validation | syntax-checking | ast-parsing | query-optimization | error-prevention

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/sql-syntax-validator

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

SQL Syntax Validator MCP

1 tools available
Cloud-hosted on Vinkius

AI agents are great at drafting complex queries, but they're terrible reviewers. They often leave behind subtle syntax mistakes—a forgotten comma in a massive JOIN statement, an incorrectly closed parenthesis, or using duplicate column aliases. Running these flawed queries directly against production data is a recipe for bottlenecks and deadlocks.

This MCP solves that problem by validating the query structure locally before execution. It doesn't need to talk to your database; it just parses the code itself. By supporting major dialects like PostgreSQL, MySQL, MariaDB, and BigQuery, you can trust that your generated SQL is syntactically sound across different environments.

When you connect this MCP through Vinkius, you gain an immediate safety check for any AI client or agent. You just pass the raw query string, and it tells you exactly where the syntax breaks down, pinpointing the error location before you even think about running it.

Core Capabilities

01 — Audit Query Structure

The MCP checks a raw SQL string against established grammar rules to confirm its structural integrity.

02 — Pinpoint Syntax Errors

It reports the exact location (line and column) of any syntax mistake in the query, saving debugging time.

03 — Support Multiple Dialects

The validator accepts different SQL dialects, including MySQL, PostgreSQL, MariaDB, and BigQuery, making it universal for your stack.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/sql-syntax-validator** — connect your AI agent in three steps.

01 You provide the MCP with the raw SQL query string and specify the target database dialect (e.g., 'postgresql').

02 The underlying engine processes the code using Abstract Syntax Tree parsing, checking it against grammar rules without connecting to a live database.

03 The system returns a clear status: either confirmation that the query is valid or a detailed report pointing out specific syntax errors.

The bottom line is you get immediate, non-destructive feedback on your SQL code's structure.

Built For

Data Engineers and Backend Developers who rely on AI to draft complex database queries. If you frequently spend time debugging runtime errors caused by subtle syntax flaws in generated SQL, this MCP is for you.

Data Engineer

They use the validator to pre-check massive JOIN statements and ETL query drafts before deploying them into production pipelines.

Backend Developer

They check service layer database interactions, ensuring that any SQL generated by an agent will compile correctly for the specific dialect (e.g., MySQL vs. Postgres).

Data Analyst

They validate complex analytical queries drafted by AI agents to ensure they don't contain subtle structural mistakes that would return empty or partial results.

What Changes When You Connect

01 Prevents deadlocks and bottlenecks. By using the validator, you confirm syntax integrity before execution, stopping database crashes caused by malformed AI-generated code.

02 Supports major dialects out of the box. Whether you're working with PostgreSQL, MySQL, or BigQuery, this MCP validates against the correct grammar rules for your environment.

03 Pinpoints errors instantly. Instead of getting a vague 'query failed,' you get precise feedback on the line and column where the syntax broke down.

04 Saves time during development. You don't have to run every draft query through a local sandbox just to check commas or parentheses; this MCP handles it upfront.

05 Reduces agent risk. When using AI agents, this validator acts as a mandatory safety layer, ensuring that complex queries are structurally sound before they reach the final execution stage.

Real-World Applications

Refining an Agent-Generated SELECT Query

A data analyst asks their agent to pull user metrics across three tables. The agent drafts a huge query, but misses a comma in the WHERE clause. Instead of running it and getting a vague 'syntax error,' they feed the draft into this MCP's `validate_sql` tool. It immediately flags the missing comma, letting them fix the query before any damage is done.

Switching Database Backends

A backend developer wrote a complex reporting query for their PostgreSQL staging environment. Now they need to port it to MariaDB. They run the same SQL through this MCP, specifying 'mariadb' as the dialect. The validator catches syntax differences unique to MariaDB that would have failed otherwise.

Auditing Multi-Dialect Joins

A team is building a universal service layer using AI agents. They need one query structure that works across both MySQL and BigQuery. They use the MCP to test the same core logic against both dialects, ensuring compatibility for all client environments.

Handling Complex UNION Statements

A data engineer needs to combine results from multiple sources using a complex UNION query structure. Before committing the code, they use this MCP's `validate_sql` function to audit the entire block, confirming that all column types and structures are consistent across every SELECT clause.

Patterns to Avoid

Assuming AI-Generated SQL is Perfect

✗ AVOID

Running an agent's complex query directly against production data because the prompt said it 'should work.' This often leads to silent failures or resource deadlocks.

✓ INSTEAD

Always run any generated SQL through this MCP's `validate_sql` tool first. Specify the target dialect (e.g., postgresql) and verify the syntax is clean before execution.

Copy-Pasting Code Without Checking

✗ AVOID

Manually copying a query from an old documentation page into a new service layer, only to find out later that the dialect (e.g., using MySQL specific functions in a Postgres environment) causes a failure.

✓ INSTEAD

Use this MCP to validate the code snippet, making sure you specify the correct database dialect for the target system.

Ignoring Parenthesis Balance

✗ AVOID

Leaving an extra parenthesis or missing a closing bracket in a deeply nested subquery. This mistake is difficult to spot manually but causes immediate runtime failures.

✓ INSTEAD

The MCP's AST parsing detects these structural imbalances automatically, saving you the headache of manual debugging.

The Right Fit

Use this MCP if your primary concern is *structural correctness* and preventing execution failure. If you are generating or reviewing SQL queries using any AI agent, run them through this validator first. It checks grammar (syntax) and dialect compatibility—the foundational layer of safety.

Don't use it if you need the tool to perform data transformations itself (e.g., calculating averages). This MCP only validates *what* you write; it doesn't execute or modify data. If your problem is identifying a missing piece of logic, you should use a general-purpose AI agent for brainstorming, not this validator. This is purely a pre-flight syntax check.

The Dreaded Debugging Loop

You write a complex query in your IDE, relying on an AI assistant to handle the tricky JOINs and aggregations. You run it against staging, but instead of clean results, you get an opaque error message: 'Query failed.' Then comes the cycle: copy that failure log into Google, read ten forum threads, guess if it's a comma, a bracket, or a dialect issue, and spend the next two hours fixing one character.

With this MCP, you stop guessing. You pass that flawed query directly to the validator. It doesn't need access to your database; it just parses the text itself. Within seconds, it tells you exactly where the syntax breaks down, pointing out whether it was a missing comma or an invalid keyword for BigQuery.

SQL Syntax Validator: Guaranteed Pre-Flight Checks

The most tedious manual step that vanishes is the initial sanity check. You no longer have to manually test every single query draft against multiple environments or worry about whether your agent used a dialect-specific function incorrectly.

Now, you trust the system's ability to catch structural flaws—from basic missing commas to advanced parenthesis mismatches—before they ever reach a live connection. The code is clean, verified, and ready.

SQL Syntax Validator: 1 Tool

Use the available tools to audit any raw SQL query string, checking it for structural integrity against multiple major database dialects.

#	TOOL	DESCRIPTION
01	`validate_sql`	Pass a raw SQL query string and an optional dialect to check for syntax errors offline, preventing runtime database crashes.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U Validate this PostgreSQL query before execution: `SELECT id, name FROM users WHERE email = 'test@example.com' ORDER BY created_at DESC;`

✅ **Valid SQL Query:** {"valid": true, "database": "postgresql", "type": "select"}

U Check if this MySQL query is syntactically sound: `SELECT * FROM orders WHERE amount > 100 AND GROUP BY user_id`

❌ **Syntax Error:** Expected end of input, found 'GROUP' at line 1, column 47.

U Audit this complex BigQuery join.

✅ **Valid SQL Query:** Checked AST correctly.

Frequently Asked Questions

01 **Does SQL Syntax Validator handle all major database dialects?**

Yes, it supports MySQL, PostgreSQL, MariaDB, BigQuery, and others. You simply specify the dialect you are targeting when running the validation check.

02 **Can I use validate_sql on incomplete queries?**

Absolutely. The tool doesn't require a runnable query; it just needs enough syntax to determine if an error exists, making it perfect for auditing drafts.

03 **Is this MCP faster than running the query live?**

Yes, because it only performs local Abstract Syntax Tree parsing. It never connects to your actual database, so validation is extremely fast and resource-light.

04 **What kind of errors can SQL Syntax Validator prevent?**

It prevents structural issues like unmatched parentheses, missing commas in JOIN clauses, or using keywords that are invalid for the specified dialect.

05 **Does validate_sql check my data integrity?**

No. This MCP is strictly a syntax validator. It checks if the *grammar* of the query is correct; it doesn't verify if the columns or tables actually exist in your database.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"sql-syntax-validator": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

SQL Syntax Validator is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by SQL Syntax Validator. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	SQL Syntax Validator MCP
Server ID	`019e38f2-383d-7117-ac33-a5dedef87cde`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/sql-syntax-validator.