


















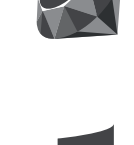




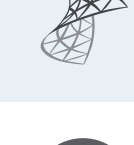









-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  **T-SQL**
-  VB.NET
-  VB6
-  XML



T-SQL static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your T-SQL code

All rules 80

 Vulnerability 1












 Bug 16

 Security Hotspot 4

 Code Smell 59





Tags 

Search by name... 

Empty statements should be removed		Code Smell
Track uses of "TODO" tags		Code Smell
A primary key should be specified during table creation		Code Smell
Track lack of copyright and license headers		Code Smell
SHA-1 and Message-Digest hash algorithms should not be used in secure contexts		Vulnerability
"NOCOUNT" should be activated on "PROCEDURE" and "TRIGGER" definitions		Code Smell
Control flow statements "IF", "WHILE" and "TRY" should not be nested too deeply		Code Smell
"CASE" expressions should end with "ELSE" clauses		Code Smell
"IF ... ELSEIF" constructs should end with "ELSE" clauses		Code Smell
Control structures should use BEGIN...END blocks		Code Smell
String literals should not be duplicated		Code Smell

Empty statements should be removed

Analyze your code

 Code Smell  Minor   unused


Empty statements, i.e. `;`, are usually introduced by mistake, for example because:


- It was meant to be replaced by an actual statement, but this was forgotten.
- There was a typo which lead the semicolon to be doubled, i.e. `;;`.


See

- [CERT, MSC12-C](#). - Detect and remove code that has no effect or is never executed
- [CERT, MSC51-J](#). - Do not place a semicolon immediately following an if, for, or while condition
- [CERT, EXP15-C](#). - Do not place a semicolon on the same line as an if, for, or while statement

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition