


















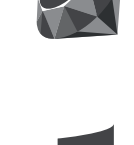




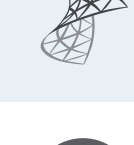









-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  **T-SQL**
-  VB.NET
-  VB6
-  XML



# 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... 

Jump statements should not be redundant	 Code Smell
"CATCH" clauses should do more than rethrow	 Code Smell
Boolean checks should not be inverted	 Code Smell
Multiple variables should not be declared on the same line	 Code Smell
Unused local variables should be removed	 Code Smell
Local variable and parameter names should comply with a naming convention	 Code Smell
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

## Jump statements should not be redundant

[Analyze your code](#)

 Code Smell

 Minor 

 redundant clumsy

Jump statements, such as `RETURN` and `CONTINUE` let you change the default flow of program execution, but jump statements that direct the control flow to the original direction are just a waste of keystrokes.

### Noncompliant Code Example

```
CREATE PROCEDURE MyProc
AS
    DECLARE @return_status int = 0;
    WHILE @return_status = 0
    BEGIN
        EXEC @return_status = something;
        CONTINUE; -- Noncompliant
    END;
    RETURN; -- Noncompliant
GO
```

### Compliant Solution

```
CREATE PROCEDURE MyProc
AS
    DECLARE @return_status int = 0;
    WHILE @return_status = 0
    BEGIN
        EXEC @return_status = something;
    END;
GO
```

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition