















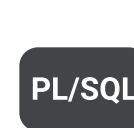


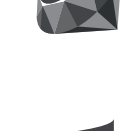




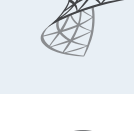









-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  **T-SQL**
-  VB.NET
-  VB6
-  XML














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... 

Redundant pairs of parentheses should be removed
 Code Smell
Functions and procedures should not have too many parameters
 Code Smell
Collapsible "if" statements should be merged
 Code Smell
Unused labels should be removed
 Code Smell
Using hardcoded IP addresses is security-sensitive
 Security Hotspot
Column references should not have more than two-parts
 Code Smell
Triggers should not "PRINT", "SELECT", or "FETCH"
 Code Smell
"LIKE" clauses should not be used without wildcards
 Code Smell
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

Redundant pairs of parentheses should be removed

[Analyze your code](#)

 Code Smell  Major   confusing

The use of parentheses, even those not required to enforce a desired order of operations, can clarify the intent behind a piece of code. But redundant pairs of parentheses could be misleading, and should be removed.

Noncompliant Code Example

```
DECLARE @x INT = (@y / 2 + 1); -- Compliant even if the parentheses are ignored
IF (@x > 0) AND ((@x+@y > 0)) -- Noncompliant
BEGIN
    -- ...
END
```

Compliant Solution

```
DECLARE @x INT = (@y / 2 + 1);
IF (@x > 0) AND (@x+@y > 0)
BEGIN
    -- ...
END
```

Available In:

sonarlint  | **sonarcloud**  | **sonarqube**  Developer Edition