


































-  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... 

"COALESCE", "IIF", and "CASE" input expressions should not contain subqueries

 Code Smell

"CHECK" or "NOCHECK" should be specified explicitly when constraints are activated

 Code Smell

Deprecated features should not be used

 Code Smell

Multiline blocks should be enclosed in BEGIN...END blocks

 Code Smell

Two branches in a conditional structure should not have exactly the same implementation

 Code Smell

"LIKE" clauses should not start with wildcard characters

 Code Smell

Column names should be used in an "ORDER BY" clause

 Code Smell

Queries should not join too many tables

 Code Smell

Function and procedure names should comply with a naming convention

 Code Smell

Columns to be read with a "SELECT" statement should be clearly defined

 Code Smell

"CASE" expressions should not have too many "WHEN" clauses

 Code Smell

Sections of code should not be commented out

"COALESCE", "IIF", and "CASE" input expressions should not contain subqueries

Analyze your code

 Code Smell  Major  unpredictable

COALESCE and IIF (which evaluate to CASE expressions under the covers), as well as CASE input expressions should not be used with subqueries because the subquery will be evaluated once for each option in the expression, and each evaluation could return different results depending on the isolation level. To ensure consistent results, use the `SNAPSHOT ISOLATION` isolation level. To ensure consistent results *and* better performance, move the subquery out of the expression.

Note it is also an option to replace COALESCE with ISNULL.

Noncompliant Code Example

```
...
COALESCE((SELECT a FROM b WHERE c) , 1) -- Noncompliant
...
```

```
...
CASE
WHEN (SELECT COUNT(*) FROM A) > 0 THEN (SELECT COUNT(*) FROM A) + 42
...
ELSE otherExpression
END
...
```

Compliant Solution

```
SET @a = SELECT a FROM b WHERE c
...
COALESCE(@a, 1)
...
```

or

```
SET TRANSACTION ISOLATION LEVEL SNAPSHOT
BEGIN TRANSACTION
...
COALESCE((SELECT a FROM b WHERE c) , 1)
...
```

```
...

SET @a = SELECT COUNT(*) FROM A

CASE
WHEN @a > 0 THEN @a + 42
...
ELSE otherExpression
END
...
```

Available In:

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