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...

The number of variables in a FETCH statement should match the number of columns in the cursor
Bug

"CASE" input expressions should be invariant
Bug

Nullable subqueries should not be used in "NOT IN" conditions
Bug

Using weak hashing algorithms is security-sensitive
Security Hotspot

Dynamically executing code is security-sensitive
Security Hotspot

"SELECT" statements used as argument of "EXISTS" statements should be selective
Code Smell

Size should be specified for "varchar" variables and parameters
Code Smell

Conditionally executed code should be denoted by either indentation or BEGIN...END block
Code Smell

Conditionals should start on new lines
Code Smell

"INSERT" statements should explicitly list the columns to be set
Code Smell

Output parameters should be assigned
Bug

Queries that use "TOP" should have an "ORDER BY"

The number of variables in a FETCH statement should match the number of columns in the cursor

Analyze your code

Bug Critical

A `FETCH` statement fails when the number of variables does not match the number of columns selected in the CURSOR definition.

Noncompliant Code Example

```
DECLARE c1 cursor FOR SELECT FirstName, LastName FROM customer;
OPEN c1;
FETCH NEXT FROM c1 INTO @Name; -- Noncompliant
```

Compliant Solution

```
DECLARE c1 cursor FOR SELECT FirstName, LastName FROM customer;
OPEN c1;
FETCH NEXT FROM c1 INTO @FirstName, @LastName;
```

Available In:
sonarlint sonarcloud sonarqube Developer Edition