- 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

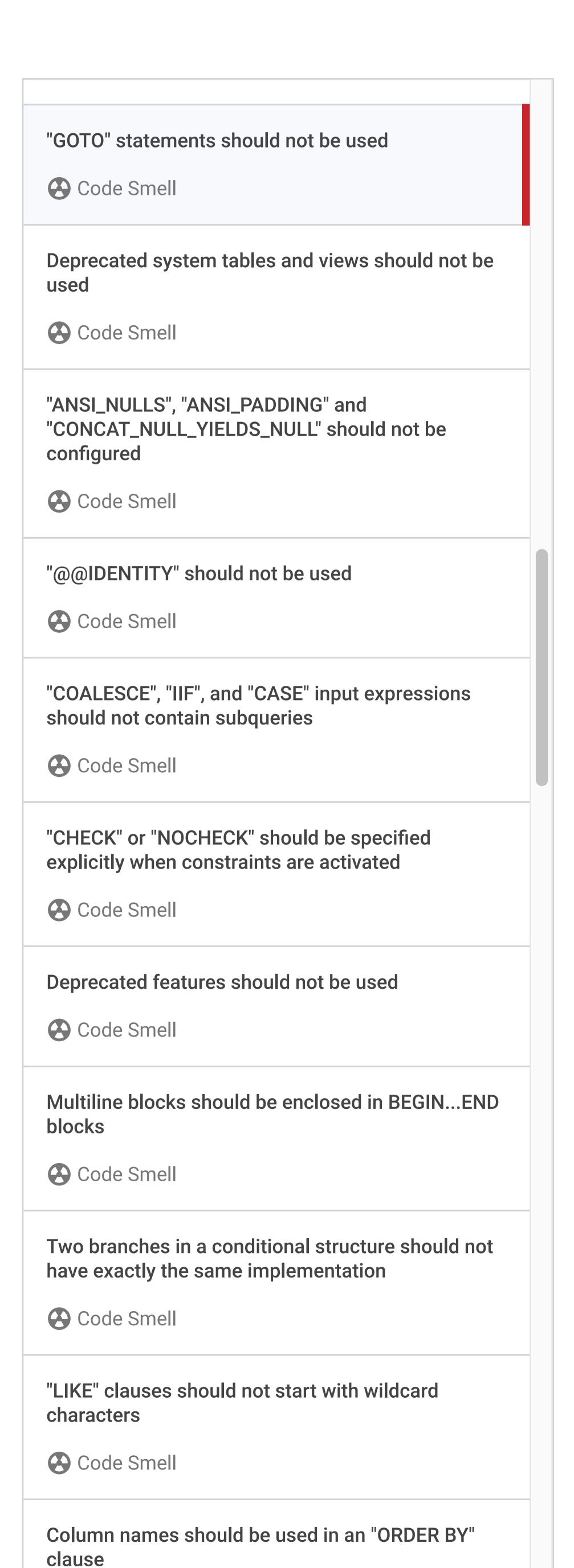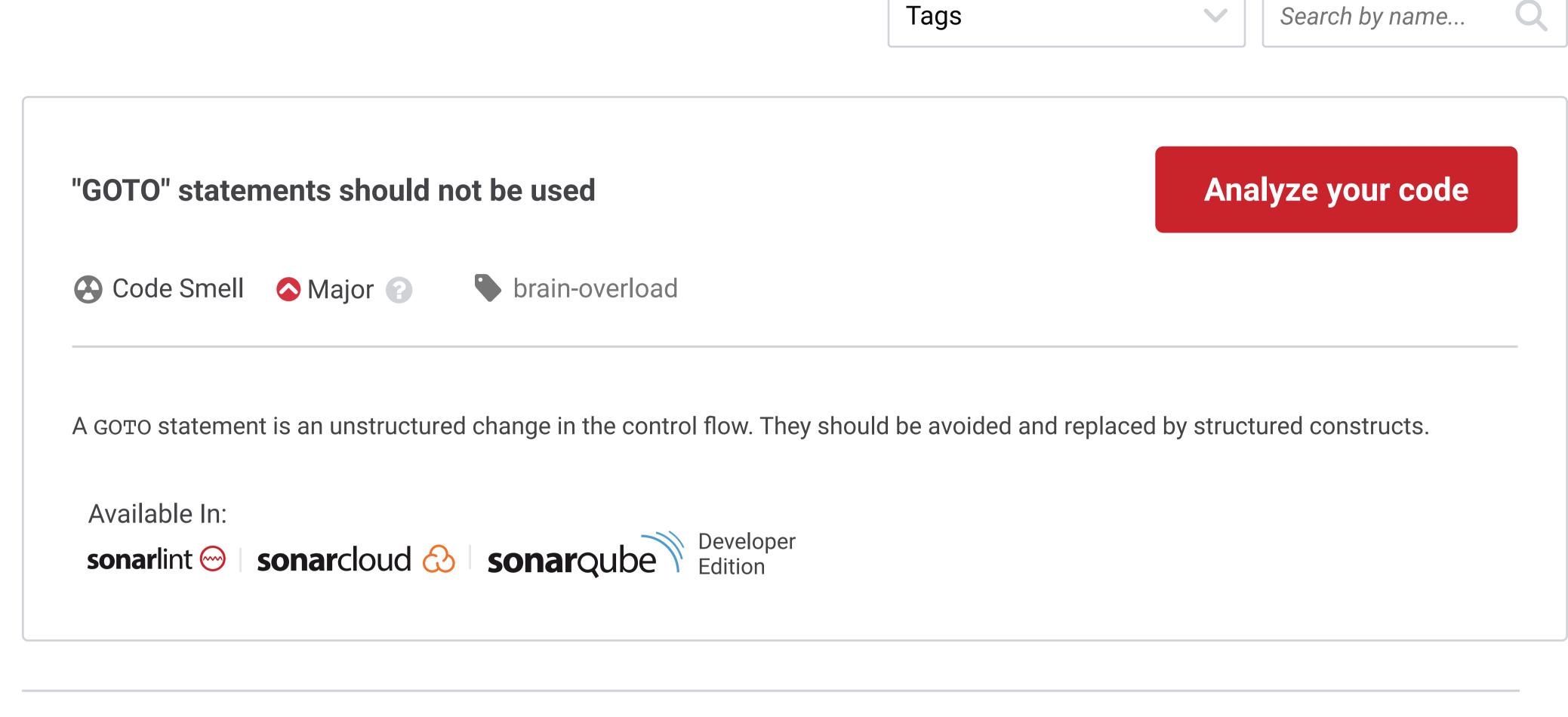# T-SQL static code analysis

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... 🔍

"GOTO" statements should not be used
⊗ Code Smell

Deprecated system tables and views should not be used
⊗ Code Smell

"ANSI_NULLS", "ANSI_PADDING" and "CONCAT_NULL_YIELDS_NULL" should not be configured
⊗ Code Smell

"@@IDENTITY" should not be used
⊗ Code Smell

"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

"GOTO" statements should not be used

Analyze your code

⊗ Code Smell   🔺 Major ⓘ   🏷 brain-overload

A GOTO statement is an unstructured change in the control flow. They should be avoided and replaced by structured constructs.

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition