

































-  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

PL/SQL static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PL/SQL code

All rules

188

 Vulnerability

4

 Bug

45

 Security Hotspot

2

 Code Smell

137

Tags

Search by name...



"LIKE" clauses should not be used without wildcards

 Code Smell

Weak "REF CURSOR" types should not be used

 Code Smell

Whitespace and control characters in string literals should be explicit

 Code Smell

Blocks containing "EXECUTE IMMEDIATE" should trap all exceptions

 Code Smell

"EXCEPTION_INIT -20,NNN" calls should be centralized

 Code Smell

"CREATE OR REPLACE" should be used instead of "CREATE"

 Code Smell

Block labels should appear on the same lines as "END"

 Code Smell

"LOOP ... END LOOP;" constructs should be avoided

 Code Smell

"IF" statements should not be nested too deeply

 Code Smell

"CASE" expressions should end with "ELSE" clauses

 Code Smell

String literals should not be duplicated

 Code Smell

Constant names should comply with a naming convention

 Code Smell

Output parameters should be assigned

"LIKE" clauses should not be used without wildcards

Analyze your code

 Code Smell  Critical   sql

The use of `LIKE` in a SQL query without one or more wildcards in the sought value is suspicious. A maintainer can suppose that either `=` was meant instead, or that the wildcard was unintentionally omitted.

Note that in some cases using `LIKE` without a wildcard may return different results than the use of `=`. Thus, the use of `LIKE` without a wildcard may be intentional. However, it is highly likely to confuse maintainers who either are unaware of this fact, or don't understand that such circumstances apply to the query in question.

Noncompliant Code Example

```
SELECT name
FROM product
WHERE name LIKE 'choc'
```

Compliant Solution

```
SELECT name
FROM product
WHERE name LIKE 'choc%'
```

or

```
SELECT name
FROM product
WHERE name = 'choc'
```

Available In:

sonarlint



sonarcloud



sonarqube



Developer Edition