


















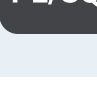














-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

PL/SQL

## 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...

🔍

 Bug

Identical expressions should not be used on both sides of a binary operator

 Bug

All code should be reachable

 Bug

Loops with at most one iteration should be refactored

 Bug

Variables and columns should not be self-assigned

 Bug

"IF" statement conditions should not evaluate unconditionally to "TRUE" or to "FALSE"

 Bug

The "result\_cache" hint should be avoided

 Bug

"GOTO" statements should not be used

 Code Smell

"TO\_NUMBER" should be used with a format model

 Code Smell

Labels should not be reused in inner scopes

 Code Smell

"FUNCTIONS" should not have "OUT" parameters

 Code Smell

Variables should be nullable

 Code Smell

"VARCHAR2" should be used

 Code Smell

Native SQL joins should be used

Identical expressions should not be used on both sides of a binary operator

Analyze your code

 Bug  Major 

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified.

This rule ignores operators +, \* and ||, and expressions: 1=1, 1<>1, 1!=1, 1~=1 and 1^=1.

Noncompliant Code Example

```
SELECT code
  FROM Person
 WHERE first_name IS NULL OR first_name IS NULL; -- Noncompliant

SELECT * FROM Users
  INNER JOIN Clients ON Clients.id = Clients.id; -- Noncompliant
```

Compliant Solution

```
SELECT code
  FROM Person
 WHERE first_name IS NULL OR last_name IS NULL;

SELECT * FROM Users
  INNER JOIN Clients ON Clients.id = Users.id;
```

Exceptions

This rule ignores \*, +, and =.

See

- [CERT, MSC12-C](#) - Detect and remove code that has no effect or is never executed
- {rule:plsql:S1656} - Implements a check on =.

Available In:

sonarlint

|

sonarcloud

|

sonarqube

Developer Edition