

































-  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...



Related "IF/ELSIF" statements and "WHEN" clauses in a "CASE" should not have the same condition

 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

Related "IF/ELSIF" statements and "WHEN" clauses in a "CASE" should not have the same condition

Analyze your code

 Bug  Major  unused pitfall

A CASE and a chain of IF/ELSIF statements is evaluated from top to bottom. At most, only one branch will be executed: the first one with a condition that evaluates to `true`.

Therefore, duplicating a condition automatically leads to dead code. Usually, this is due to a copy/paste error. At best, it's simply dead code and at worst, it's a bug that is likely to induce further bugs as the code is maintained, and obviously it could lead to unexpected behavior.

Noncompliant Code Example

```
IF param == 1 THEN
  x := 'A';
ELSIF param == 2 THEN
  x := 'B';
ELSIF param == 1 THEN -- Noncompliant, for sure this is a bug
  x := 'C';
END IF;

result := CASE param
  WHEN 1 THEN 'A'
  WHEN 2 THEN 'B'
  WHEN 1 THEN 'C'  -- Noncompliant
  ELSE 'D'
END;
```

Compliant Solution

```
IF param == 1 THEN
  result := 'A';
ELSIF param == 2 THEN
  result := 'B';
ELSIF param == 3 THEN
  result := 'C';
END IF;

result := CASE param
  WHEN 1 THEN 'A'
  WHEN 2 THEN 'B'
  WHEN 3 THEN 'C'
  ELSE 'D'
END;
```

See

- [CERT, MSC12-C](#). - Detect and remove code that has no effect or is never executed

Available In:

sonarlint 

sonarcloud 

sonarqube 

Developer Edition