


















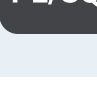














-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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...

🔍

"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

 Code Smell

"FORALL" should be used

 Code Smell

"FETCH ... BULK COLLECT INTO" should be used

 Code Smell

Column aliases should be defined using "AS"

 Code Smell

Procedures and functions should be encapsulated in packages

 Code Smell

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

Analyze your code

 Bug  Major   cwe

IF statements with conditions that are always false have the effect of making blocks of code non-functional. This can be useful during debugging, but should not be checked in. IF statements with conditions that are always true are completely redundant, and make the code less readable. In either case, unconditional IF statements should be removed.

Noncompliant Code Example

```
IF TRUE THEN
    do_something;
END IF;

IF FALSE THEN
    do_something_else;
END IF;
```

Compliant Solution

```
do_something;
```

See

- [MITRE, CWE-489](#) - Leftover Debug Code
- [MITRE, CWE-570](#) - Expression is Always False
- [MITRE, CWE-571](#) - Expression is Always True

Available In:

sonarlint  | **sonarcloud**  | **sonarqube**  Developer Edition