


















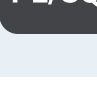














-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

# 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...

🔍

"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

 Bug

"ROWNUM" should not be used at the same query level as "ORDER BY"

 Bug

All branches in a conditional structure should not have exactly the same implementation

 Bug

Strings should only be moved to variables or columns which are large enough to hold them

 Bug

"WHERE" clause conditions should not be contradictory

 Bug

Unary prefix operators should not be repeated

 Bug

DML events clauses should not include multiple "OF" clauses

 Bug

"PACKAGE BODY" initialization sections should not

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

 Code Smell  Critical   pitfall

Simple loops, of the form `LOOP . . . END LOOP`, behave by default as infinite ones, since they do not have a loop condition. They can often be replaced by other, safer, loop constructs.

## Noncompliant Code Example

```
SET SERVEROUTPUT ON

DECLARE
  i PLS_INTEGER;
BEGIN
  i := 1;
  LOOP -- Noncompliant, an infinite loop by default and therefore dangerous
    DBMS_OUTPUT.PUT_LINE('First loop i: ' || i);

    i := i + 1;
    EXIT WHEN i > 10;
  END LOOP;

END;
/
```

## Compliant Solution

```
SET SERVEROUTPUT ON

DECLARE
  i PLS_INTEGER;
BEGIN
  FOR i IN 1..10 LOOP -- Compliant, much safer equivalent alternative
    DBMS_OUTPUT.PUT_LINE('Second loop i: ' || i);
  END LOOP;
END;
/
```

Available In:

 |  |  Developer Edition