

**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... 

<b>labeled</b>	 Code Smell
<b>In labeled loops "EXIT" should exit the label</b>	 Code Smell
<b>"EXIT WHEN" should be used rather than "IF ... THEN EXIT; END IF;"</b>	 Code Smell
<b>"FOR" loop end conditions should not be hard-coded</b>	 Code Smell
<b>Large item lists should not be used with "IN" clauses</b>	 Code Smell
<b>"GOTO" should not be used within loops</b>	 Code Smell
<b>"FULL OUTER JOINS" should be used with caution</b>	 Code Smell
<b>"CASE" should be used rather than "DECODE"</b>	 Code Smell
<b>"CROSS JOIN" queries should not be used</b>	 Code Smell
<b>"END" statements of labeled blocks should be labeled</b>	 Code Smell
<b>Two branches in a conditional structure should not have exactly the same implementation</b>	 Code Smell
<b>Unused assignments should be removed</b>	 Code Smell

## "END" statements of labeled loops should be labeled

## Analyze your code

☢ Code Smell    ⬆ Major    ?    🏷 convention pitfall

Labeled loops are useful, especially when the code is badly indented, to match the begin and end of each loop. This rule raises an issue when the end of a labeled loop is unlabeled.

## Noncompliant Code Example

```
BEGIN
  <<myLoopLabel1>>
  LOOP
    EXIT;
  END LOOP; -- Noncompliant; this labeled loop has no ending label

  LOOP
    EXIT;
  END LOOP; -- Compliant; not a labeled loop
END;
/
```

## Compliant Solution

```
BEGIN
  <<myLoopLabel1>>
  LOOP
    EXIT;
  END LOOP myLoopLabel1;

  LOOP
    EXIT;
  END LOOP;
END;
/
```

Available In:

sonarlint  | sonarcloud  | sonarqube  Developer Edition