
















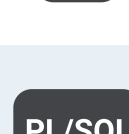
















-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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... 🔍

 Code Smell
Nested blocks should be labeled
 Code Smell
"RESULT_CACHE" should not be used
 Code Smell
Columns should be aliased
 Code Smell
Track parsing failures
 Code Smell
Files should not be too complex
 Code Smell
Function and procedure parameters should comply with a naming convention
 Code Smell
Magic literals should not be used
 Code Smell
"UNION" should be used with caution
 Code Smell
"GROUP BY" should not be used in SQL "SELECT" statements
 Code Smell
Track breaches of an XPath rule
 Code Smell
Track uses of "NOSONAR" comments
 Code Smell
Track comments matching a regular expression
 Code Smell

## Nested blocks should be labeled

Analyze your code

-  Code Smell
-  Major 
-  convention

Labeled blocks are useful, especially when the code is badly indented, to help maintainers match the beginning and ending of each block. When blocks are nested, labeling them can improve the code's readability. This rule detects nested block which do not have a start label.

### Noncompliant Code Example

```
BEGIN -- Compliant, this is not a nested block
  NULL;
END;
/

BEGIN
  BEGIN -- Noncompliant; this nested block has no label
    NULL;
  END;
END;
/

BEGIN
  BEGIN -- Noncompliant; this nested block has only an end label
    NULL;
  END myBlockLabel1;

  <<myBlockLabel2>> -- Compliant
  BEGIN
    NULL;
  END;
END;
/
```

### Compliant Solution

```
BEGIN
  NULL;
END;
/

BEGIN
  BEGIN myBlockLabel0
    NULL;
  END myBlockLabel0;
END;
/

BEGIN
  BEGIN myBlockLabel1
    NULL;
  END myBlockLabel1;

  <<myBlockLabel2>>
  BEGIN
    NULL;
  END;
END;
/
```

Available In:

 |  |  Developer Edition