
















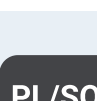
















-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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...

🔍

 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 contain "RETURN" statements

 Bug

"NULL" should not be compared directly

 Bug

"MLSLABEL" should not be used

 Bug

"VARCHAR2" and "NVARCHAR2" should be used

 Bug

String literals should not be duplicated

Analyze your code

 Code Smell

 Critical



 design

Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.

On the other hand, constants can be referenced from many places, but only need to be updated in a single place.

Noncompliant Code Example

With the default threshold of 3:

```
BEGIN
  prepare('action1');
  execute('action1');
  release('action1');
END;
/
```

Compliant Solution

```
DECLARE
  action CONSTANT VARCHAR2(7) := 'action1';
BEGIN
  prepare(action);
  execute(action);
  release(action);
END;
/
```

Exceptions

To prevent generating some false-positives, literals having less than 5 characters are excluded.

Available In:

sonarlint



| **sonarcloud**



| **sonarqube**



Developer Edition