





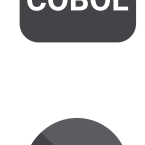



























-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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
"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
Statements should be on separate lines
 Code Smell

"RESULT_CACHE" should not be used

Analyze your code

 Code Smell

 Major 

 performance

Because `RESULT_CACHE`-enabled functions increase memory consumption, one should double-check that the gain in performances is significant, and avoid over-using this feature in general.

Noncompliant Code Example

```
CREATE FUNCTION myFastFunction RETURN PLS_INTEGER RESULT_CACHE AS -- Noncompliant
BEGIN
    RETURN 42;
END;
/

DROP FUNCTION myFastFunction;
```

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition