

















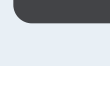















-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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... 

Functions and procedures should not have too many parameters

 Code Smell

Collapsible "if" statements should be merged

 Code Smell

Unused labels should be removed

 Code Smell

Compound triggers should define at least two triggers

 Code Smell

"EXIT" should not be used in loops

 Code Smell

Jump statements should not be redundant

 Code Smell

"EXCEPTION WHEN ... THEN" clauses should do more than "RAISE"

 Code Smell

Single line comments should start with "--"

 Code Smell

An "ORDER BY" direction should be specified explicitly

 Code Smell

Oracle's join operator (+) should not be used

 Code Smell

"cursor%NOTFOUND" should be used instead of "NOT cursor%FOUND"

 Code Smell

Object attributes should comply with a naming convention

Functions and procedures should not have too many parameters

Analyze your code

 Code Smell  Major  brain-overload

Having functions and procedures which take too many parameters decreases the code's readability and usability. It is likely that such a function/procedure is not modular enough, and should be split into several smaller ones.

Noncompliant Code Example

With the default threshold of 10:

```
SET SERVEROUTPUT ON

CREATE FUNCTION sumWithTooManyParameters( -- Noncompliant, too many parameters
  a1 PLS_INTEGER,
  a2 PLS_INTEGER,
  a3 PLS_INTEGER,
  a4 PLS_INTEGER,
  a5 PLS_INTEGER,
  a6 PLS_INTEGER,
  a7 PLS_INTEGER,
  a8 PLS_INTEGER,
  a9 PLS_INTEGER,
  a10 PLS_INTEGER,
  a11 PLS_INTEGER
)
  RETURN PLS_INTEGER AS
BEGIN
  RETURN a1 + a2 + a3 + a4 + a5 + a6 + a7 + a8 + a9 + a10 + a11;
END;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE('Sum is ' || sumWithTooManyParameters(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11));
END;
/

DROP FUNCTION sumWithTooManyParameters;
```

Compliant Solution

```
SET SERVEROUTPUT ON

CREATE FUNCTION sumCorrected(n PLS_INTEGER) RETURN PLS_INTEGER AS -- Compliant
BEGIN
  RETURN (1 + n)*(n / 2);
END;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE('Sum is ' || sumCorrected(11));
END;
/

DROP FUNCTION sumCorrected;
```

Available In:

sonarlint  | **sonarcloud**  | **sonarqube**  Developer Edition