
















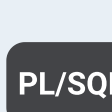
















-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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

Loop start and end labels should match

 Code Smell

Block start and end labels should match

 Code Smell

Cipher algorithms should be robust

 Vulnerability

"COMMIT" should not be used inside a loop

 Bug

Individual "WHERE" clause conditions should not be unconditionally true or false

 Bug

Nullable subqueries should not be used in "NOT IN" conditions

 Bug

"COMMIT" and "ROLLBACK" should not be called from non-autonomous transaction triggers

 Bug

Positional and named arguments should not be mixed in invocations

 Bug

Functions should end with "RETURN" statements

 Bug

Collections should not be iterated in "FOR" loops

 Bug

Using weak hashing algorithms is security-sensitive

 Security Hotspot

Dynamically executing code is security-sensitive

 Security Hotspot

SQL "JOIN" conditions should involve all joined tables

 Code Smell

Quoted identifiers should not be used

Analyze your code

 Code Smell

 Blocker



 pitfall

Quoted identifiers are confusing to many programmers, as they look similar to string literals. Moreover, for maximum portability, identifiers should be self-descriptive and should not contain accents. Quoted identifiers can contain any character, which can be confusing.

Noncompliant Code Example

```
SET SERVEROUTPUT ON

DECLARE

  "x + y" PLS_INTEGER := 0; -- Noncompliant, quoted identifiers are confusing
  x PLS_INTEGER := 40;
  y PLS_INTEGER := 2;
  "hello" VARCHAR2(42) := 'world';  -- Noncompliant

BEGIN
  DBMS_OUTPUT.PUT_LINE("x + y"); -- Noncompliant, displays 0
  DBMS_OUTPUT.PUT_LINE("hello"); -- Noncompliant, confusing, displays "world" and not "hello"
END;
/
```

Compliant Solution

```
SET SERVEROUTPUT ON

DECLARE

  my_int PLS_INTEGER := 0;
  x PLS_INTEGER := 40;
  y PLS_INTEGER := 2;
  greeting VARCHAR2(42) := 'hello';

BEGIN
  DBMS_OUTPUT.PUT_LINE(my_int);
  DBMS_OUTPUT.PUT_LINE(x + y); -- Compliant, displays 42

  DBMS_OUTPUT.PUT_LINE(greeting);
END;
/
```

Available In:

sonarlint



sonarcloud



sonarqube

Developer Edition