


















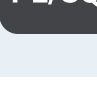














-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

# 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...

🔍

contain "RETURN" statements

 Bug

"NULL" should not be compared directly

 Bug


"MLSLABEL" should not be used

 Bug

"VARCHAR2" and "NVARCHAR2" should be used

 Bug

Related "IF/ELSIF" statements and "WHEN" clauses in a "CASE" should not have the same condition

 Bug

Identical expressions should not be used on both sides of a binary operator

 Bug

All code should be reachable

 Bug

Loops with at most one iteration should be refactored

 Bug

Variables and columns should not be self-assigned

 Bug

"IF" statement conditions should not evaluate unconditionally to "TRUE" or to "FALSE"

 Bug

The "result\_cache" hint should be avoided

 Bug

"GOTO" statements should not be used

 Code Smell

"TO\_NUMBER" should be used with a format model

 Code Smell

## "PACKAGE BODY" initialization sections should not contain "RETURN" statements

Analyze your code

-  Bug
-  Major 
-  unused

In a CREATE PACKAGE BODY, the purpose of the initialization section is to set the initial values of the package's global variables. It is therefore surprising to find a RETURN statement there, as all its following statements will be unreachable.

### Noncompliant Code Example

```
SET SERVEROUTPUT ON

CREATE OR REPLACE PACKAGE foo AS
    FUNCTION getBar RETURN PLS_INTEGER;
    bar PLS_INTEGER;
END;
/

CREATE OR REPLACE PACKAGE BODY foo AS
    FUNCTION getBar RETURN PLS_INTEGER AS
    BEGIN
        RETURN bar; -- Compliant
    END;
BEGIN
    bar := 42;
    DBMS_OUTPUT.PUT_LINE('package loaded');
    RETURN; -- Noncompliant
    DBMS_OUTPUT.PUT_LINE('this is unreachable code');
END;
/

DROP PACKAGE BODY foo;

DROP PACKAGE foo;
```

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition