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...

- "GOTO" statements should not be used
  Code Smell
- "TO_NUMBER" should be used with a format model
  Code Smell
- Labels should not be reused in inner scopes
  Code Smell
- "FUNCTIONS" should not have "OUT" parameters
  Code Smell
- Variables should be nullable
  Code Smell
- "VARCHAR2" should be used
  Code Smell
- Native SQL joins should be used
  Code Smell
- "FORALL" should be used
  Code Smell
- "FETCH ... BULK COLLECT INTO" should be used
  Code Smell
- Column aliases should be defined using "AS"
  Code Smell
- Procedures and functions should be encapsulated in packages
  Code Smell
- Procedures should have parameters
  Code Smell
- "EXECUTE IMMEDIATE" should be used instead of DBMS_SQL procedure calls
  Code Smell

## "GOTO" statements should not be used

Analyze your code

Code Smell  Major  brain-overload

A `GOTO` statement is an unstructured change in the control flow. They should be avoided and replaced by structured constructs.

**Noncompliant Code Example**

```
SET SERVEROUTPUT ON

DECLARE
  i PLS_INTEGER := 42;
BEGIN
  IF i < 0 THEN
    GOTO negative; -- Noncompliant
  END IF;

  DBMS_OUTPUT.PUT_LINE('positive');
  goto cleanup; -- Noncompliant

  <<negative>>
  DBMS_OUTPUT.PUT_LINE('negative!');

  <<cleanup>>
  NULL;
END;
/
```

**Compliant Solution**

```
SET SERVEROUTPUT ON

DECLARE
  i PLS_INTEGER := 42;
BEGIN
  IF i < 0 THEN
    DBMS_OUTPUT.PUT_LINE('negative!'); -- Compliant
  ELSE
    DBMS_OUTPUT.PUT_LINE('positive');
  END IF;
END;
/
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition