



















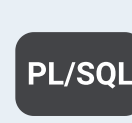

























-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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

<div>EXIT WHEN should be used rather than IF ... THEN EXIT; END IF;"</div> <div> Code Smell</div>
<div>"FOR" loop end conditions should not be hard-coded</div> <div> Code Smell</div>
<div>Large item lists should not be used with "IN" clauses</div> <div> Code Smell</div>
<div>"GOTO" should not be used within loops</div> <div> Code Smell</div>
<div>"FULL OUTER JOINS" should be used with caution</div> <div> Code Smell</div>
<div>"CASE" should be used rather than "DECODE"</div> <div> Code Smell</div>
<div>"CROSS JOIN" queries should not be used</div> <div> Code Smell</div>
<div>"END" statements of labeled blocks should be labeled</div> <div> Code Smell</div>
<div>Two branches in a conditional structure should not have exactly the same implementation</div> <div> Code Smell</div>
<div>Unused assignments should be removed</div> <div> Code Smell</div>
<div>"LIKE" clauses should not start with wildcard characters</div> <div> Code Smell</div>
<div>Column names should be used in a SQL "ORDER BY" clause</div> <div> Code Smell</div>

## "FOR" loop end conditions should not be hard-coded

Analyze your code

 Code Smell

 Major

 brain-overload

Hard-coding bounds in FOR loops is a bad practice, just as magic numbers in general are. Often, those magic bounds can be replaced by dynamic values. If that is not possible, replacing the literal number with a constant is still better.

### Noncompliant Code Example

```
SET SERVEROUTPUT ON

DECLARE
    TYPE myCollectionType IS VARRAY(3) OF VARCHAR2(42);
    myCollection myCollectionType := myCollectionType('David', 'John', 'Richard');

BEGIN

    FOR i IN 2 .. 3 -- Noncompliant; magic numbers used for the loop bounds
    LOOP
        DBMS_OUTPUT.PUT_LINE('name = ' || myCollection(i));
    END LOOP;

    FOR i IN 2 .. myCollection.LAST -- Noncompliant, better but still magic
    LOOP
        DBMS_OUTPUT.PUT_LINE('name = ' || myCollection(i));
    END LOOP;

END;
/
```

### Compliant Solution

```
SET SERVEROUTPUT ON

DECLARE
    TYPE myCollectionType IS VARRAY(3) OF VARCHAR2(42);
    myCollection myCollectionType := myCollectionType('David', 'John', 'Richard');

BEGIN
    FOR i IN myCollection.FIRST .. myCollection.LAST
    LOOP
        DBMS_OUTPUT.PUT_LINE('name = ' || myCollection(i));
    END LOOP;
END;
/
```

Available In:

sonarlint

sonarcloud

sonarqube

Developer Edition