















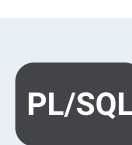

















-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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

 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
 Code Smell
Record fields should comply with a naming convention
 Code Smell
Cursors should follow a naming convention
 Code Smell
Types should follow a naming convention
 Code Smell

Tags ▾

Search by name... 🔍

"EXIT" should not be used in loops

Analyze your code

 Code Smell

 Minor 

 bad-practice

FOR and WHILE loops are structured control flow statements.

A FOR loop will iterate once for each element in the range, and the WHILE iterates for as long as a condition holds.

However, inserting an EXIT statement within the loop breaks this structure, reducing the code's readability and making it harder to debug.

Noncompliant Code Example

```
SET SERVEROUTPUT ON

DECLARE
  TYPE myCollectionType IS VARRAY(10) OF VARCHAR2(42);
  myCollection myCollectionType := myCollectionType('Foo', 'Bar', NULL, 'Baz', 'Qux');

  i PLS_INTEGER;
BEGIN
  i := 1;
  WHILE i <= myCollection.LAST LOOP
    EXIT WHEN myCollection(i) IS NULL; -- Noncompliant, breaks the structure of the WHILE

    DBMS_OUTPUT.PUT_LINE('Got: ' || myCollection(i));
    i := i + 1;
  END LOOP;
```

Compliant Solution

```
SET SERVEROUTPUT ON

DECLARE
  TYPE myCollectionType IS VARRAY(10) OF VARCHAR2(42);
  myCollection myCollectionType := myCollectionType('Foo', 'Bar', NULL, 'Baz', 'Qux');

  i PLS_INTEGER;
BEGIN
  i := 1;
  WHILE i <= myCollection.LAST AND myCollection(i) IS NOT NULL LOOP
    DBMS_OUTPUT.PUT_LINE('Got: ' || myCollection(i));
    i := i + 1;
  END LOOP;
END;
/
```

Available In:

 |  |  Developer Edition