





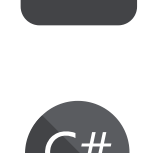








































-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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
"WHEN" clauses should not have too many lines
 Code Smell
Magic numbers should not be used
 Code Smell
Files should not have too many lines of code
 Code Smell
Lines should not be too long
 Code Smell
Explicitly opened cursors should be closed
 Bug
Identifiers should be written in lower case
 Code Smell
"PLS_INTEGER" types should be used
 Code Smell
Reserved words should be written in upper case
 Code Smell
Parameter "IN" mode should be specified explicitly
 Code Smell
Lines in a multiline comment should start with "*"
 Code Smell
CASE should be used for sequences of simple tests
 Code Smell
SQL tables should be joined with the "JOIN" keyword
 Code Smell

Tags 

Search by name... 

"WHEN" clauses should not have too many lines

Analyze your code

 Code Smell

 Major 

 brain-overload

The `CASE` statement should be used only to clearly define some new branches in the control flow. As soon as a `WHEN` clause contains too many statements this highly decreases the readability of the overall control flow statement. In such case, the content of `WHEN` clause should be extracted in a dedicated function.

Noncompliant Code Example

```
CASE my_variable
  WHEN 0 THEN -- 6 lines till next WHEN
    procedure1;
    procedure2;
    procedure3;
    procedure4;
    procedure5;
  WHEN 1 THEN
-- ...
END CASE;
```

Compliant Solution

```
DECLARE
  PROCEDURE do_something AS
  BEGIN
    procedure1;
    procedure2;
    procedure3;
    procedure4;
    procedure5;
  END;
BEGIN
  CASE my_variable
    WHEN 0 THEN
      do_something;
    WHEN 1 THEN
-- ...
    END CASE;
END;
/
```

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition