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

Tags

Search by name...

Two branches in a conditional structure should not have exactly the same implementation
Code Smell

Unused assignments should be removed
Code Smell

"LIKE" clauses should not start with wildcard characters
Code Smell

Column names should be used in a SQL "ORDER BY" clause
Code Smell

Procedures and functions should be documented
Code Smell

SQL statements should not join too many tables
Code Smell

Function and procedure names should comply with a naming convention
Code Smell

Columns to be read with a "SELECT" statement should be clearly defined
Code Smell

"CASE" structures should not have too many "WHEN" clauses
Code Smell

Deprecated LONG and LONG RAW datatypes should no longer be used
Code Smell

Sections of code should not be commented out
Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Analyze your code

Code Smell | Major | design suspicious

Having two branches in an `IF/ELSIF` chain with the same implementation is at best duplicate code, and at worst a coding error.

If the same logic is truly needed for both instances, then in an `IF` chain they should be combined.

Noncompliant Code Example

```
IF param = 1 THEN
    sort_order := 0;
    column := 'LastName';
ELSIF param = 2 THEN
    sort_order := 0;
    column := 'LastName'; -- Noncompliant
ELSE
    sort_order := 1;
    column := 'FirstName';
END IF;
```

Exceptions

Branches in an `IF/ELSIF` chain with implementation that contains a single line of code are ignored.

```
IF param = 1 THEN
    sort_order := 0;
ELSIF param = 2 THEN
    sort_order := 1;
ELSE
    sort_order := 0; -- No issue, usually this is done on purpose to increase the readability
END IF;
```

But this exception does not apply to `IF` chains without `ELSE`-s when all branches have the same single line of code. In case of `IF` chains with `ELSE`-s rule {rule:plsql:S3923} raises a bug.

```
IF param = 1 THEN  -- Noncompliant, this might have been done on purpose but probably not
    sort_order := 0;
ELSIF param = 2 THEN
    sort_order := 0;
END IF;
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition