

**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... 

 Code Smell	
<b>"NUMBER" variables should be declared with precision</b>	
 Code Smell	
<b>Nested subqueries should be avoided</b>	
 Code Smell	
<b>Nested loops should be labeled</b>	
 Code Smell	
<b>Nested blocks should be labeled</b>	
 Code Smell	
<b>"RESULT_CACHE" should not be used</b>	
 Code Smell	
<b>Columns should be aliased</b>	
 Code Smell	
<b>Track parsing failures</b>	
 Code Smell	
<b>Files should not be too complex</b>	
 Code Smell	
<b>Function and procedure parameters should comply with a naming convention</b>	
 Code Smell	
<b>Magic literals should not be used</b>	
 Code Smell	
<b>"UNION" should be used with caution</b>	
 Code Smell	
<b>"GROUP BY" should not be used in SQL "SELECT" statements</b>	

## "NUMBER" variables should be declared with precision

## Analyze your code

-  Code Smell  Major   performance

Declaring a `NUMBER` variable without any precision wastes memory because Oracle supports up to 38 decimal digits by default (or the maximum supported by your system, whichever is less). If you don't need that large a value, you should specify whatever matches your needs. This will save memory and provide extra integrity checking on input.

This rule also applies to some **NUMBER** subtypes as well: **NUMERIC**, **DEC**, and **DECIMAL**.

## Noncompliant Code Example

```
DECLARE
    var1 NUMBER; -- Noncompliant
    var2 NUMERIC; -- Noncompliant
BEGIN
    NULL;
END;
/
```

## Compliant Solution

```
DECLARE
    var1 NUMBER(9,2);
    var2 NUMERIC(4,0);
BEGIN
    NULL;
END;
/
```

Available In:

