

































-  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









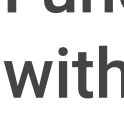




All rules 188

 Vulnerability 4

 Bug 45

 Security Hotspot 2

 Code Smell 137

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

Tags ▾

Search by name... 🔍

## Nested subqueries should be avoided

Analyze your code

 Code Smell

 Major



 performance sql

Subqueries are nested when they appear in the `WHERE` clause of the parent statement. When an Oracle database evaluates a statement with a nested subquery, it must evaluate the subquery portion multiple times and may overlook some efficient access paths or joins.

Subquery unnesting unnests and merges the body of the subquery into the body of the statement that contains it, allowing the optimizer to consider the queries together when evaluating access paths and joins. The optimizer can unnest most subqueries, with some exceptions. Those exceptions include:

- hierarchical subqueries
- subqueries that contain a `ROWNUM` pseudocolumn
- subqueries that contain one of the set operators
- subqueries that contain a nested aggregate function
- subqueries that contain a correlated reference to a query block that is not the immediate outer query block of the subquery.

Assuming no restrictions exist, the optimizer automatically unnests some (but not all) of the following nested subqueries:

- Uncorrelated `IN` subqueries
- `IN` and `EXISTS` correlated subqueries, as long as they do not contain aggregate functions or a `GROUP BY` clause

You can enable extended subquery unnesting by instructing the optimizer to unnest additional types of subqueries:

- You can unnest an uncorrelated `NOT IN` subqueries by specifying the `HASH_AJ` or `MERGE_AJ` hint in the subquery.
- You can unnest other subqueries by specifying the `UNNEST` hint in the subquery.

Because these optimizations are dependant on the version of Oracle used, it is best to avoid using nested subqueries in the first place when possible.

### Noncompliant Code Example

```
BEGIN
  SELECT col1
  BULK COLLECT INTO result
  FROM table1
  WHERE col2 IN (SELECT col3 FROM table2); -- Noncompliant
END;
/
```

### Compliant Solution

```
BEGIN
  SELECT col1
  BULK COLLECT INTO result
  FROM table1
  JOIN table2 ON col2 = col3;
END;
/
```

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition