Oracle Correlated Subquery

Introduction to the Oracle correlated subquery

See the following products table in the [sample database](https://www.oracletutorial.com/getting-started/oracle-sample-database/):



The following [query](https://www.oracletutorial.com/oracle-basics/oracle-select/) returns the cheapest products from the products table using a [subquery](https://www.oracletutorial.com/oracle-basics/oracle-subquery/) in the [WHERE](https://www.oracletutorial.com/oracle-basics/oracle-where/) clause.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT      product\_id,      product\_name,      list\_price  FROM      products  WHERE      list\_price =(          SELECT              MIN( list\_price )          FROM              products      ); |

Let’s examine this query in more detail.

First, you can execute the subquery independently.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      MIN( list\_price )  FROM      products; |

Second, Oracle evaluates the subquery only once.

Third, after the subquery returns a result set, the outer query makes use of them. In other words, the outer query depends on the subquery. However, the subquery is isolated and not dependent on the values of the outer query.

Unlike the above subquery, a correlated subquery is a [subquery](https://www.oracletutorial.com/oracle-basics/oracle-subquery/) that uses values from the outer query. In addition, a correlated subquery may be evaluated once for each row selected by the outer query. Because of this, a query that uses a correlated subquery could be slow.

A correlated subquery is also known as a **repeating subquery** or a **synchronized subquery**.

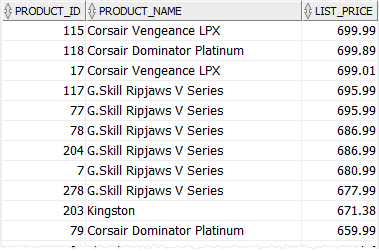
Oracle correlated subquery examples

Let’s take some examples of the correlated subqueries to better understand how they work.

A) Oracle correlated subquery in the WHERE clause example

The following query finds all products whose list price is above [average](https://www.oracletutorial.com/oracle-aggregate-functions/oracle-avg/) for their category.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | SELECT      product\_id,      product\_name,      list\_price  FROM      products p  WHERE      list\_price > (          SELECT              AVG( list\_price )          FROM              products          WHERE              category\_id = p.category\_id      ); |



In the above query, the outer query is:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      product\_id,      product\_name,      list\_price  FROM      products p  WHERE      list\_price > |

And the correlated subquery is:

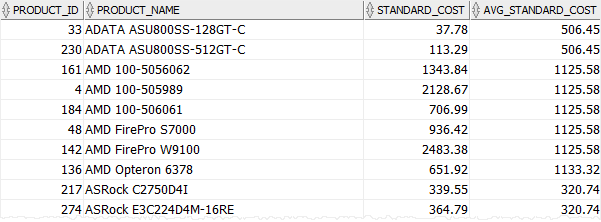
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      AVG( list\_price )  FROM      products  WHERE      category\_id = p.category\_id |

For each product from the products table, Oracle must execute the correlated subquery to calculate the average price by category.

B) Oracle correlated subquery in the SELECT clause example

The following query returns all products and the [average](https://www.oracletutorial.com/oracle-aggregate-functions/oracle-avg/) standard cost based on the product category:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | SELECT      product\_id,      product\_name,      standard\_cost,      ROUND(          (              SELECT                  AVG( standard\_cost )              FROM                  products              WHERE                  category\_id = p.category\_id          ),          2      ) avg\_standard\_cost  FROM      products p  ORDER BY      product\_name; |



For each product from the products table, Oracle executed the correlated subquery to calculate the average standard of cost for the product category.

Note that the above query used the [ROUND()](https://www.oracletutorial.com/oracle-date-functions/oracle-round/) function to round the average standard cost to two decimals.

C) Oracle correlated subquery with the EXISTS operator example

We usually use a correlated subquery with the [EXISTS](https://www.oracletutorial.com/oracle-basics/oracle-exists/) operator. For example, the following statement returns all customers who have no orders:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | SELECT      customer\_id,      name  FROM      customers  WHERE      NOT EXISTS (          SELECT              \*          FROM              orders          WHERE              orders.customer\_id = customers.customer\_id      )  ORDER BY      name; |