**What is subquery**

Sql Statement …..(Select…………);  
  
﻿**A subquery is a SELECT statement within another SQL statement.** The SQL statement can be SELECT, WHERE clause, FROM clause, JOIN, INSERT, UPDATE, DELETE, SET, DO, or another subquery.

The query that contains the subquery is normally called outer query and the subquery itself is called inner query.

**Advantages of using subquery**

* Subqueries structure a complex query broken down into a series of logical steps for easy understanding and code maintenance.
* Subqueries allow you to use the results of another query in the outer query.

**single-value subquery or scalar subquery.**

**When the subquery returns a single value,** **the subquery is only evaluated once** and then the value is returned to outer query to use. This kind of subqueries are also known as single-value subquery or scalar subquery.

An aggregate function or an expression is normally used in the subquery.

select OrderID, CustomerID  
from orders  
where ShippedDate = (select max(ShippedDate) from orders);

**column subquery**

**When the subquery returns a list of values, the subquery is only evaluated once** and then the list of values is returned to outer query to use. This kind of subqueries are also known as column subquery.

Column subquery is normally used in WHERE clause with an **IN operator** that tests each value in the returned list from the subquery.

/\*  
This query retrieves a list of customers that made   
purchases after the date 1998-05-01.\*/

Select CustomerID, CompanyName  
from customers  
where CustomerID in   
(  
 select CustomerID   
 from orders   
 where orderDate > '1998-05-01'  
);

\*

Following query returns the same result as the one  
in Practice #1 but here no subquery is used.   
Instead, we used inner join.

select a.CustomerID, a.CompanyName  
from customers as a  
inner join orders as b on a.CustomerID = b.CustomerID  
where b.orderDate > '1998-05-01'

**Row Subquery(only MySQL support)**

A subquery that returns one or more rows of values is also known as row subquery.

When the subquery returns one or more rows of values, the subquery is only evaluated once and then the row(s) of values is returned to outer query to use. **Row subquery is MySQL specific.** Other major database systems (SQL Server, Oracle, etc) don't have this type of subquery.

/\*  
This query finds out all the employees who live  
in the same city and country as customers.

select EmployeeID, FirstName, LastName, City, Country  
from employees  
where row(City, Country) in  
(select City, Country from customers);

**Using correlated subqueries**

**The name of correlated subqueries means that a subquery is correlated with the outer query. The correlation means the subquery executes once for every row in the outer query.**

A correlated subquery can usually be rewritten as a join query. Using joins enables the database engine to use the most efficient execution plan. The query optimizer is more mature for joins than for subqueries, so in many cases a statement that uses a subquery should normally be rephrased as a join to gain the extra speed in performance.

Note that **alias** **must** **be** **used** to distinguish table names in the SQL query that contains correlated subqueries.

/\*  
Because a product can be sold at a discount,   
we want to know the highest unit price ever   
sold for each product.  
   
This query uses correlated subquery to get  
the highest unit price for each product sold.   
   
Alias is used for order\_details in both the  
subquery and outer query. \*/

select distinct a.ProductID,   
 a.UnitPrice as Max\_unit\_price\_sold  
from order\_details as a  
where a.UnitPrice =   
(  
 select max(UnitPrice)  
 from order\_details as b  
 where a.ProductID = b.ProductID  
)  
order by a.ProductID;

*Using different tables in subquery and outer query.*

\*  
This query finds out a list of orders and their customers   
who ordered more than 20 items of product Grandma's   
Boysenberry Spread (ProductID 6) on a single order.  
\*/  
select a.OrderID,   
 a.CustomerID  
from orders as a  
where   
(  
 select Quantity   
 from order\_details as b   
 where a.OrderID = b.OrderID and b.ProductID = 6  
) > 20;

**Using EXISTS and NOT EXISTS in correlated subqueries**

EXISTS and NOT EXISTS are used with a subquery in WHERE clause to examine if the result the subquery returns is TRUE or FALSE. The true or false value is then used to restrict the rows from outer query select. Because EXISTS and NOT EXISTS only return TRUE or FALSE in the subquery, the SELECT list in the subquery does not need to contain actual column name(s). Normally use SELECT \* (asterisk) is sufficient but you can use SELECT column1, column2, ... or anything else. It does not make any difference.

Because EXISTS and NOT EXISTS are used with correlated subqueries, **the subquery executes once for every row in the outer query**. In other words, for each row in outer query, by using information from the outer query, the subquery checks if it returns TRUE or FALSE, and then the value is returned to outer query to use.

\*  
This query uses EXISTS keyword in WHERE clause   
to return a list of customers whose products   
were shipped to UK.  
   
Note that the outer query only returns a row  
where the subquery returns TRUE.  
\*/  
select CustomerID, CompanyName   
from customers as a  
where exists  
(  
 select \* from orders as b  
 where a.CustomerID = b.CustomerID  
 and ShipCountry = 'UK'  
);  
   
/\*

This query uses INNER JOIN and returns the same   
result set as the query above.

select distinct a.CustomerID, a.CompanyName   
from customers as a  
inner join orders as b  
on a.CustomerID = b.CustomerID  
where b.ShipCountry = 'UK';

link to refer all

<https://www.geeksengine.com/database/subquery/exists.php>