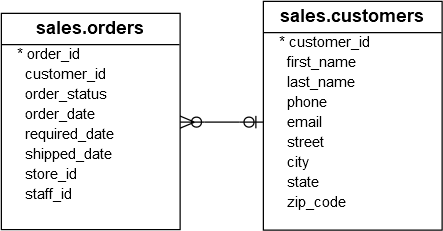
## Introduction to SQL Server subquery

## Q1(Q2(Q3))

A subquery is a query nested inside another statement such as [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/), [INSERT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/), [UPDATE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-update/), or [DELETE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-delete/).

Let’s see the following example.

Consider the orders and customers tables from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/).



The following statement shows how to use a subquery in the [WHERE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/) clause of a [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) statement to find the sales orders of the customers who locate in New York:

SELECT

order\_id,

order\_date,

customer\_id

FROM

sales.orders

WHERE

customer\_id IN (

SELECT

customer\_id

FROM

sales.customers

WHERE

city = 'New York'

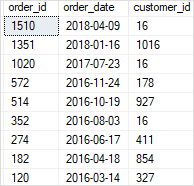
)

ORDER BY

order\_date DESC;

Code language: SQL (Structured Query Language) (sql)

Here is the result:



In this example, the following statement is a subquery:

SELECT

customer\_id

FROM

sales.customers

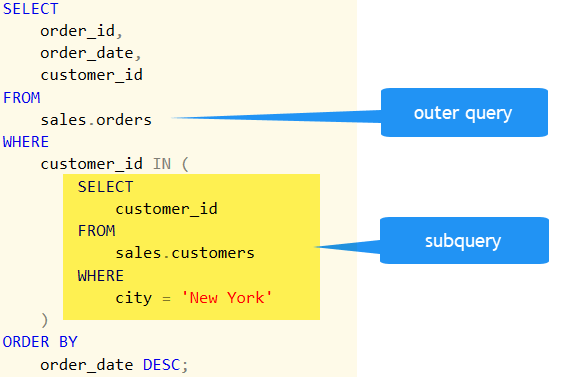
WHERE

city = 'New York'

Code language: SQL (Structured Query Language) (sql)

Note that you must always enclose the SELECT query of a subquery in parentheses ().

A subquery is also known as an inner query or inner select while the statement containing the subquery is called an outer select or outer query:



SQL Server executes the whole query example above as follows:

First, it executes the subquery to get a list of customer identification numbers of the customers who locate in New York.

SELECT

customer\_id

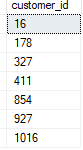
FROM

sales.customers

WHERE

city = 'New York'

Code language: SQL (Structured Query Language) (sql)



Second, SQL Server substitutes customer identification numbers returned by the subquery in the [IN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-in/) operator and executes the outer query to get the final result set.

As you can see, by using the subquery, you can combine two steps together. The subquery removes the need for selecting the customer identification numbers and plugging them into the outer query. Moreover, the query itself automatically adjusts whenever the customer data changes.

## Nesting subquery

A subquery can be nested within another subquery. SQL Server supports up to 32 levels of nesting. Consider the following example:

SELECT

product\_name,

list\_price

FROM

production.products

WHERE

list\_price > (

SELECT

AVG (list\_price)

FROM

production.products

WHERE

brand\_id IN (

SELECT

brand\_id

FROM

production.brands

WHERE

brand\_name = 'Strider'

OR brand\_name = 'Trek'

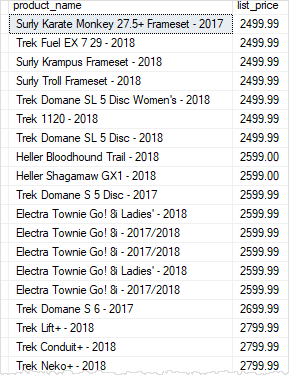
)

)

ORDER BY

list\_price;

Code language: SQL (Structured Query Language) (sql)



First, SQL Server executes the following subquery to get a list of brand identification numbers of the Strider and Trek brands:

SELECT

brand\_id

FROM

production.brands

WHERE

brand\_name = 'Strider'

OR brand\_name = 'Trek';

Code language: SQL (Structured Query Language) (sql)

SQL Server Subquery brand id list

Second, SQL Server calculates the average price list of all products that belong to those brands.

SELECT

AVG (list\_price)

FROM

production.products

WHERE

brand\_id IN (6,9)

Code language: SQL (Structured Query Language) (sql)

Third, SQL Server finds the products whose list price is greater than the average list price of all products with the Strider or Trek brand.

## SQL Server subquery types

You can use a subquery in many places:

* In place of an expression
* With [IN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-in/) or [NOT IN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-in/)
* With [ANY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-any/) or [ALL](https://www.sqlservertutorial.net/sql-server-basics/sql-server-all/)
* With [EXISTS](https://www.sqlservertutorial.net/sql-server-basics/sql-server-exists/) or NOT EXISTS
* In [UPDATE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-update/), [DELETE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-delete/), or[INSERT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) statement
* In the FROM clause

### **SQL Server subquery is used in place of an expression**

If a subquery returns a single value, it can be used anywhere an expression is used.

In the following example, a subquery is used as a column expression named max\_list\_price in a SELECT statement.

SELECT

order\_id,

order\_date,

(

SELECT

MAX (list\_price)

FROM

sales.order\_items i

WHERE

i.order\_id = o.order\_id

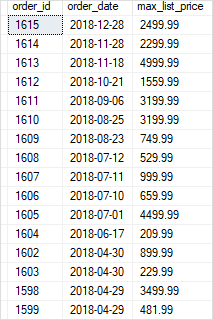
) AS max\_list\_price

FROM

sales.orders o

order by order\_date desc;

Code language: SQL (Structured Query Language) (sql)



### **SQL Server subquery is used with IN operator**

A subquery that is used with the [IN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-in/) operator returns a set of zero or more values. After the subquery returns values, the outer query makes use of them.

The following query finds the names of all mountain bikes and road bikes products that the Bike Stores sell.

SELECT

product\_id,

product\_name

FROM

production.products

WHERE

category\_id IN (

SELECT

category\_id

FROM

production.categories

WHERE

category\_name = 'Mountain Bikes'

OR category\_name = 'Road Bikes'

);

Code language: SQL (Structured Query Language) (sql)



This query is evaluated in two steps:

1. First, the inner query returns a list of category identification numbers that match the names Mountain Bikes and code Road Bikes.
2. Second, these values are substituted into the outer query that finds the product names which have the category identification number match with one of the values in the list.

### **SQL Server subquery is used with ANY operator**

The subquery is introduced with the ANY operator has the following syntax:

scalar\_expression comparison\_operator ANY (subquery)

Code language: SQL (Structured Query Language) (sql)

Assuming that the subquery returns a list of value v1, v2, … vn. The ANY operator returns TRUE if one of a comparison pair (scalar\_expression, vi) evaluates to TRUE; otherwise, it returns FALSE.

For example, the following query finds the products whose list prices are greater than or equal to the average list price of any product brand.

SELECT

product\_name,

list\_price

FROM

production.products

WHERE

list\_price >= ANY (

SELECT

AVG (list\_price)

FROM

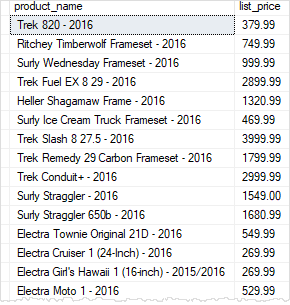
production.products

GROUP BY

brand\_id

)

Code language: SQL (Structured Query Language) (sql)



For each brand, the subquery finds the maximum list price. The outer query uses these max prices and determines which individual product’s list price is greater than or equal to any brand’s maximum list price.

### **SQL Server subquery is used with ALL operator**

The [ALL](https://www.sqlservertutorial.net/sql-server-basics/sql-server-all/) operator has the same syntax as the [ANY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-any/) operator:

scalar\_expression comparison\_operator ALL (subquery)

Code language: SQL (Structured Query Language) (sql)

The [ALL](https://www.sqlservertutorial.net/sql-server-basics/sql-server-all/) operator returns TRUE if all comparison pairs (scalar\_expression, vi) evaluate to TRUE; otherwise, it returns FALSE.

The following query finds the products whose list price is greater than or equal to the average list price returned by the subquery:

SELECT

product\_name,

list\_price

FROM

production.products

WHERE

list\_price >= ALL (

SELECT

AVG (list\_price)

FROM

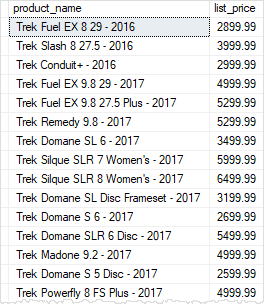
production.products

GROUP BY

brand\_id

)

Code language: SQL (Structured Query Language) (sql)



### **SQL Server subquery is used with EXISTS or NOT EXISTS**

The following illustrates the syntax of a subquery introduced with [EXISTS](https://www.sqlservertutorial.net/sql-server-basics/sql-server-exists/) operator:

WHERE [NOT] EXISTS (subquery)

Code language: SQL (Structured Query Language) (sql)

The EXISTS operator returns TRUE if the subquery return results; otherwise it returns FALSE.

On the other hand, the NOT EXISTS is opposite to the EXISTS operator.

The following query finds the customers who bought products in 2017:

SELECT

customer\_id,

first\_name,

last\_name,

city

FROM

sales.customers c

WHERE

EXISTS (

SELECT

customer\_id

FROM

sales.orders o

WHERE

o.customer\_id = c.customer\_id

AND YEAR (order\_date) = 2017

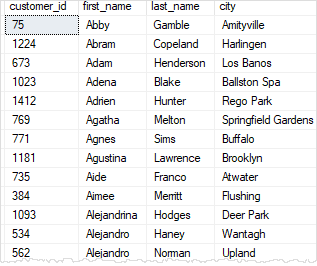
)

ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)



If you use the NOT EXISTS instead of EXISTS, you can find the customers who did not buy any products in 2017.

SELECT

customer\_id,

first\_name,

last\_name,

city

FROM

sales.customers c

WHERE

NOT EXISTS (

SELECT

customer\_id

FROM

sales.orders o

WHERE

o.customer\_id = c.customer\_id

AND YEAR (order\_date) = 2017

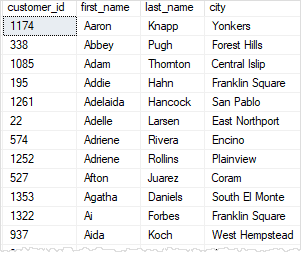
)

ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)



### **SQL Server subquery in the FROM clause**

Suppose that you want to find the average of the sum of orders of all sales staff. To do this, you can first find the number of orders by staffs:

SELECT

staff\_id,

COUNT(order\_id) order\_count

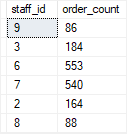
FROM

sales.orders

GROUP BY

staff\_id;

Code language: SQL (Structured Query Language) (sql)



Then, you can apply the AVG() function to this result set. Since a query returns a result set that looks like a virtual table, you can place the whole query in the FROM clause of another query like this:

SELECT

AVG(order\_count) average\_order\_count\_by\_staff

FROM

(

SELECT

staff\_id,

COUNT(order\_id) order\_count

FROM

sales.orders

GROUP BY

staff\_id

) t;

Code language: SQL (Structured Query Language) (sql)



The query that you place in the FROM clause must have a table alias. In this example, we used the t as the table alias for the subquery.  To come up with the final result SQL Server carries the following steps:

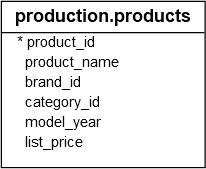
* Execute the subquery in the FROM clause.
* Use the result of the subquery and execute the outer query.

**CORELATED SUBQUERY**

A correlated subquery is a [subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/) that uses the values of the outer query. In other words, it depends on the outer query for its values. Because of this dependency, a correlated subquery cannot be executed independently as a simple subquery.

Moreover, a correlated subquery is executed repeatedly, once for each row evaluated by the outer query. The correlated subquery is also known as a repeating subquery.

Consider the following products table from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/):



The following example finds the products whose list price is equal to the highest list price of the products within the same category:

SELECT

product\_name,

list\_price,

category\_id

FROM

production.products p1

WHERE

list\_price IN (

SELECT

MAX (p2.list\_price)

FROM

production.products p2

WHERE

p2.category\_id = p1.category\_id

GROUP BY

p2.category\_id

)

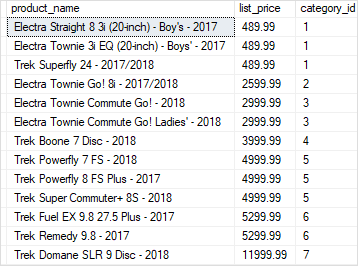
ORDER BY

category\_id,

product\_name;

Code language: SQL (Structured Query Language) (sql)

Here is the result:



In this example, for each product evaluated by the outer query, the subquery finds the highest price of all products in its category. If the price of the current product is equal to the highest price of all products in its category, the product is included in the result set. This process continues for the next product and so on.

As you can see, the correlated subquery is executed once for each product evaluated by the outer query.