SQL Server Subquery

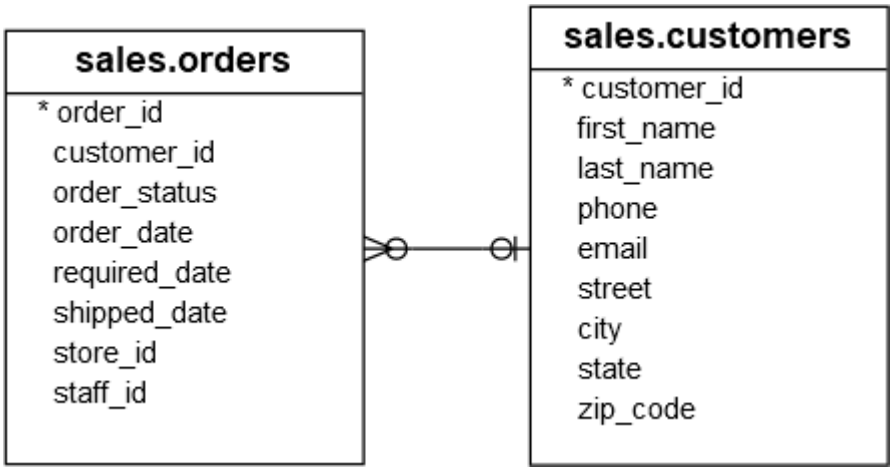
Summary: in this tutorial, you will learn about the SQL Server subquery and how to use the subquery for querying data.

Introduction to SQL Server subquery

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;
```

Here is the result:

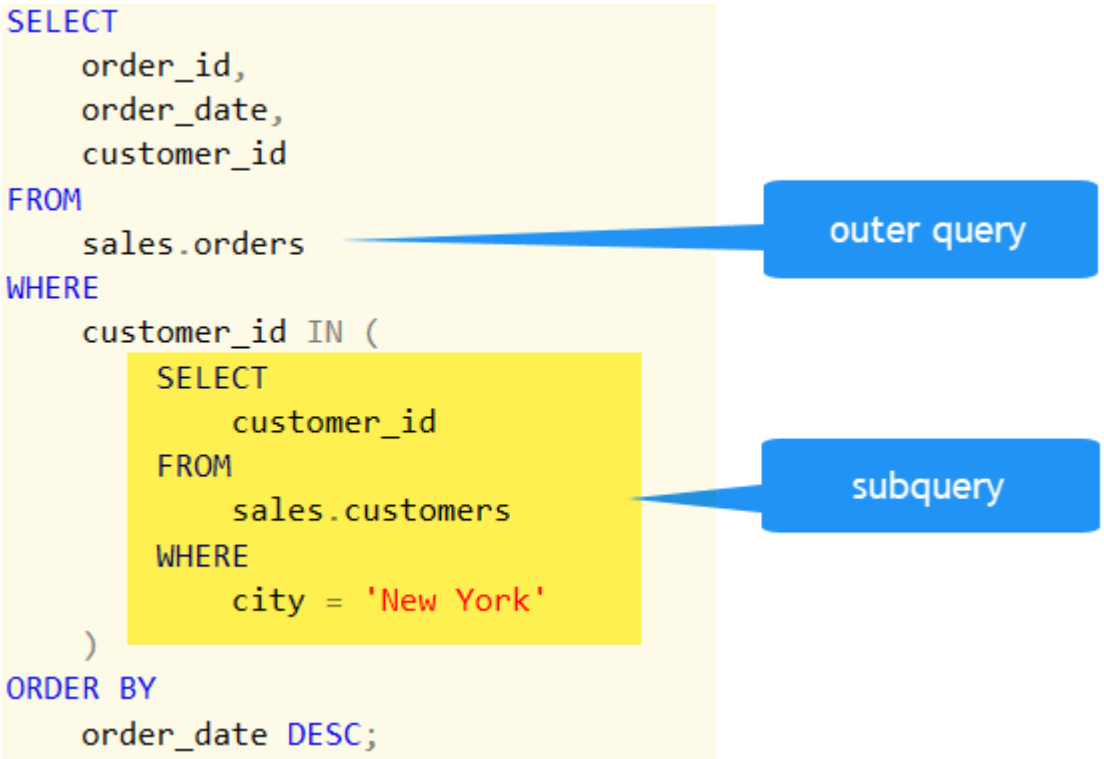
order_id	order_date	customer_id
1510	2018-04-09	16
1351	2018-01-16	1016
1020	2017-07-23	16
572	2016-11-24	178
514	2016-10-19	927
352	2016-08-03	16
274	2016-06-17	411
182	2016-04-18	854
120	2016-03-14	327

In this example, the following statement is a subquery:

```
SELECT
    customer_id
FROM
    sales.customers
WHERE
    city = 'New York'
```

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'
```

customer_id
16
178
327
411
854
927
1016

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;
```

product_name	list_price
Surly Karate Monkey 27.5+ Frameset - 2017	2499.99
Trek Fuel EX 7 29 - 2018	2499.99
Surly Krampus Frameset - 2018	2499.99
Surly Troll Frameset - 2018	2499.99
Trek Domane SL 5 Disc Women's - 2018	2499.99
Trek 1120 - 2018	2499.99
Trek Domane SL 5 Disc - 2018	2499.99
Heller Bloodhound Trail - 2018	2599.00
Heller Shagamaw GX1 - 2018	2599.00
Trek Domane S 5 Disc - 2017	2599.99
Electra Townie Go! 8i Ladies' - 2018	2599.99
Electra Townie Go! 8i - 2017/2018	2599.99
Electra Townie Go! 8i - 2017/2018	2599.99
Electra Townie Go! 8i Ladies' - 2018	2599.99
Electra Townie Go! 8i - 2017/2018	2599.99
Trek Domane S 6 - 2017	2699.99
Trek Lift+ - 2018	2799.99
Trek Conduit+ - 2018	2799.99
Trek Neko+ - 2018	2799.99

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';
```

brand_id
6
9

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)
```

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;
```

order_id	order_date	max_list_price
1615	2018-12-28	2499.99
1614	2018-11-28	2299.99
1613	2018-11-18	4999.99
1612	2018-10-21	1559.99
1611	2018-09-06	3199.99
1610	2018-08-25	3199.99
1609	2018-08-23	749.99
1608	2018-07-12	529.99
1607	2018-07-11	999.99
1606	2018-07-10	659.99
1605	2018-07-01	4499.99
1604	2018-06-17	209.99
1602	2018-04-30	899.99
1603	2018-04-30	229.99
1598	2018-04-29	3499.99
1599	2018-04-29	481.99

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'
    );
```

product_id	product_name
1	Trek 820 - 2016
2	Ritchey Timberwolf Frameset - 2016
3	Surly Wednesday Frameset - 2016
4	Trek Fuel EX 8 29 - 2016
5	Heller Shagamaw Frame - 2016
6	Surly Ice Cream Truck Frameset - 2016
7	Trek Slash 8 27.5 - 2016
8	Trek Remedy 29 Carbon Frameset - 2016
27	Surly Big Dummy Frameset - 2017
28	Surly Karate Monkey 27.5+ Frameset - 2017
29	Trek X-Caliber 8 - 2017
30	Surly Ice Cream Truck Frameset - 2017
31	Surly Wednesday - 2017

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)
```

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
    )
```

product_name	list_price
Trek 820 - 2016	379.99
Ritchey Timberwolf Frameset - 2016	749.99
Surly Wednesday Frameset - 2016	999.99
Trek Fuel EX 8 29 - 2016	2899.99
Heller Shagamaw Frame - 2016	1320.99
Surly Ice Cream Truck Frameset - 2016	469.99
Trek Slash 8 27.5 - 2016	3999.99
Trek Remedy 29 Carbon Frameset - 2016	1799.99
Trek Conduit+ - 2016	2999.99
Surly Straggler - 2016	1549.00
Surly Straggler 650b - 2016	1680.99
Electra Townie Original 21D - 2016	549.99
Electra Cruiser 1 (24-Inch) - 2016	269.99
Electra Girl's Hawaii 1 (16-inch) - 2015/2016	269.99
Electra Moto 1 - 2016	529.99

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)
```

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
    )
```

product_name	list_price
Trek Fuel EX 8 29 - 2016	2899.99
Trek Slash 8 27.5 - 2016	3999.99
Trek Conduit+ - 2016	2999.99
Trek Fuel EX 9.8 29 - 2017	4999.99
Trek Fuel EX 9.8 27.5 Plus - 2017	5299.99
Trek Remedy 9.8 - 2017	5299.99
Trek Domane SL 6 - 2017	3499.99
Trek Silque SLR 7 Women's - 2017	5999.99
Trek Silque SLR 8 Women's - 2017	6499.99
Trek Domane SL Disc Frameset - 2017	3199.99
Trek Domane S 6 - 2017	2699.99
Trek Domane SLR 6 Disc - 2017	5499.99
Trek Madone 9.2 - 2017	4999.99
Trek Domane S 5 Disc - 2017	2599.99
Trek Powerfly 8 FS Plus - 2017	4999.99

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)
```

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;
```

customer_id	first_name	last_name	city
75	Abby	Gamble	Amityville
1224	Abram	Copeland	Harlingen
673	Adam	Henderson	Los Banos
1023	Adena	Blake	Ballston Spa
1412	Adrien	Hunter	Rego Park
769	Agatha	Melton	Springfield Gardens
771	Agnes	Sims	Buffalo
1181	Agustina	Lawrence	Brooklyn
735	Aide	Franco	Atwater
384	Aimee	Merritt	Flushing
1093	Alejandrina	Hodges	Deer Park
534	Alejandro	Haney	Wantagh
562	Alejandro	Norman	Upland

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;
```

customer_id	first_name	last_name	city
1174	Aaron	Knapp	Yonkers
338	Abbey	Pugh	Forest Hills
1085	Adam	Thornton	Central Islip
195	Addie	Hahn	Franklin Square
1261	Adelaida	Hancock	San Pablo
22	Adelle	Larsen	East Northport
574	Adriene	Rivera	Encino
1252	Adriene	Rollins	Plainview
527	Afton	Juarez	Coram
1353	Agatha	Daniels	South El Monte
1322	Ai	Forbes	Franklin Square
937	Aida	Koch	West Hempstead

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;
```

staff_id	order_count
9	86
3	184
6	553
7	540
2	164
8	88

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;
```

average_order_count_by_staff
269

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.

In this tutorial, you have learned about the SQL Server subquery concept and how to use various subquery types to query data.