-Tạo Thêm các bảng và dữ liệu như sau:

NhanVien

	MaNV	HotenNV	GT	NS	MaNVQL
1	0000000001	Viên Thanh Nhã	Nam	1981-07-16	0000000002
2	0000000002	Trần Thanh An	Nam	1978-08-20	NULL
3	0000000003	Nguyễn Tâm Như	Nữ	1990-10-10	0000000002

KhachHang

	MaKH	HoKH	TenKH	Phone	Email
1	001	Nguyễn Văn	An	098899999	an@gmail.com
2	002	Lưu Bình	Nguyên	0978977777	nguyen@gmail.com

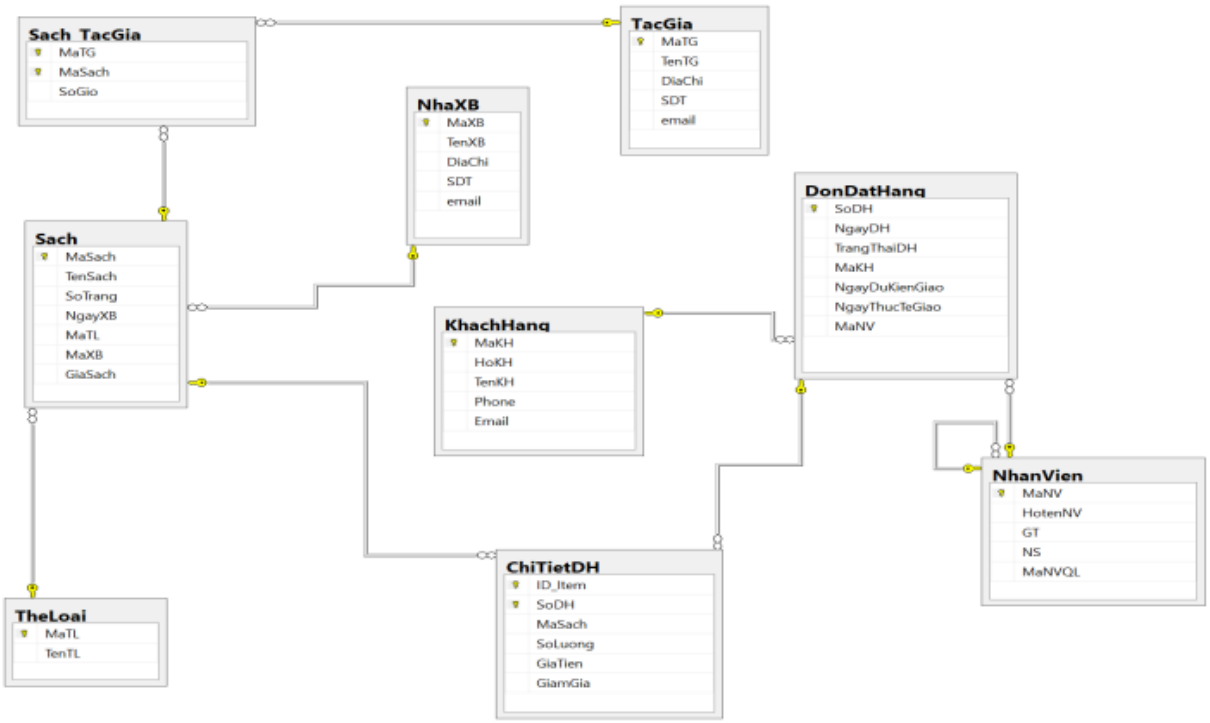
DonDatHang

	SoDH	NgayDH	TrangThaiDH	MaKH	NgayDuKienGiao	NgayThucTeGiao	MaNV
1	001	2020-10-20 00:00:00.000	1	001	2020-10-20 00:00:00.000	NULL	NULL
2	002	2020-11-20 00:00:00.000	0	002	2020-12-20 00:00:00.000	NULL	NULL

ChitietDH

	ID_Item	SoDH	MaSach	SoLuong	GiaTien	GiamGia
1	1	001	KTDC	10	30000	0.1
2	2	001	KTTM	5	20000	0.2
3	3	002	KTVM	2	40000	0.5
4	4	002	THGT	1	35000	0.4

--Bổ sung thêm cột giá sách Mô hình Diagram như sau:



-Function trong SQL

Trong hướng dẫn này, bạn sẽ tìm hiểu mọi thứ bạn cần biết về function do người dùng tự định nghĩa trong SQL Server, bao gồm scalar-valued function (hàm vô hướng) trả về một giá trị đơn và table-valued function (hàm bảng) trả về các bản ghi dữ liệu.

Function do người dùng tự định nghĩa trong SQL Server giúp bạn đơn giản hóa công việc lập trình của mình bằng cách đóng gói các logic nghiệp vụ phức tạp để sử dụng lại trong mọi truy vấn.

Scalar function trong SQL Server

Trong phần này, bạn sẽ tìm hiểu về các scalar function (hàm vô hướng) trong SQL Server và cách sử dụng chúng để đóng gói các công thức hoặc logic nghiệp vụ và sử dụng lại chúng trong các truy vấn.

Scalar function là gì?

Scalar function (hàm vô hướng) trong SQL Server yêu cầu một hoặc nhiều tham số và trả về một giá trị đơn.

Scalar function giúp bạn đơn giản hóa mã của mình. Ví dụ: bạn có thể có một phép tính phức tạp xuất hiện trong nhiều truy vấn SELECT. Thay vì thêm công thức trong mọi truy vấn, bạn có thể tạo một scalar function đóng gói công thức này và sử dụng nó trong mỗi truy vấn.

Tạo scalar function trong SQL Server

Để tạo một scalar function, bạn sử dụng câu lệnh `CREATE FUNCTION` như sau:

CREATE FUNCTION [schema_name.]function_name (parameter_list)

RETURNS data_type AS

BEGIN statements RETURN value

END

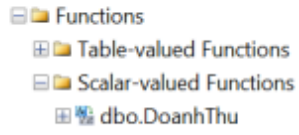
Trong cú pháp này:

- Đầu tiên, chỉ định tên của function sau các từ khóa `CREATE FUNCTION`. Tên lược đồ là tùy chọn. Nếu bạn không chỉ định rõ ràng, SQL Server sẽ sử dụng lược đồ `dbo` theo mặc định.
- Thứ hai, chỉ định một danh sách các tham số được bao quanh bởi cặp dấu ngoặc đơn sau tên function.
- Thứ ba, chỉ định kiểu dữ liệu của giá trị trả về trong câu lệnh `RETURNS`.
- Cuối cùng, thêm một câu lệnh `RETURN` để trả về một giá trị bên trong phần thân của function.

> ***Lưu ý**: kiểu dữ liệu trả về trong thân của function phải giống với kiểu dữ liệu được khai báo sau từ khóa `RETURNS`.*

Ví dụ sau đây tạo ra một function tính toán doanh thu thuần dựa trên số lượng, giá niêm yết và chiết khấu:

```
--Scale function
CREATE FUNCTION DoanhThu
(
    @quantity INT,
    @price DEC(10,2),
    @discount DEC(4,2)
)
RETURNS DEC(10,2)
AS
BEGIN
    RETURN @quantity * @price * (1 - @discount);
END;
```




```
--Gọi hàm scale như sau:
SELECT dbo.DoanhThu(10,100,0.1) as danhthu;
```

	danhthu
1	900.00

```sql
--Gọi hàm tính doanh thu của đơn hàng sử dụng hàm scale
SELECT dbo.DoanhThu(10,100,0.1) as danhthu;
--Tính doanh thu của don dat hang
SELECT
    SoDH,
    SUM(dbo.DoanhThu(SoLuong, GiaTien, GiamGia)) DoanhThu
FROM
    ChiTietDH
GROUP BY
    SoDH
ORDER BY
    DoanhThu DESC;
```

	SoDH	DoanhThu
1	001	350000.00
2	002	61000.00

Chỉnh sửa scalar function trong SQL Server

Để chỉnh sửa một scalar function, bạn sử dụng từ khóa `ALTER` thay vì `CREATE` như sau:

```sql
ALTER FUNCTION [schema_name.]function_name
(
    parameter_list
)
RETURN data_type AS
BEGIN
    statements
    RETURN value
END
```

> ***Lưu ý**: bạn có thể sử dụng câu lệnh `CREATE OR ALTER` để tạo function nếu nó không tồn tại hoặc để chỉnh sửa function hiện có.*

```sql
CREATE OR ALTER FUNCTION [schema_name.]function_name
(
    parameter_list
)
RETURN data_type AS
BEGIN
    statements
    RETURN value
END
```

Xóa scalar function trong SQL Server

Để xóa một scalar function (hàm vô hướng) hiện có, bạn sử dụng câu lệnh `DROP FUNCTION` như sau:

```sql
DROP FUNCTION [schema_name.]function_name;
```

Biến kiểu bảng trong SQL Server

Trong phần này, bạn sẽ tìm hiểu về biến kiểu bảng (table variable) trong SQL Server để lưu trữ các bản ghi dữ liệu.

Biến kiểu bảng là gì?

Biến kiểu bảng là loại biến cho phép bạn lưu trữ các bản ghi dữ liệu, tương tự như các bảng tạm.

Cách khai báo biến kiểu bảng trong SQL Server

Để khai báo một biến kiểu bảng, bạn sử dụng câu lệnh `DECLARE` như sau:

```
DECLARE @table_variable_name TABLE
(
    column_list
);
```

Trong cú pháp này, bạn chỉ định tên của biến kiểu bảng ở giữa từ khóa `DECLARE` và `TABLE`. Tên của các biến kiểu bảng phải bắt đầu bằng ký tự `@`.

Theo sau từ khóa `TABLE`, bạn định nghĩa cấu trúc của biến kiểu bảng tương tự như cấu trúc của bảng thông thường bao gồm định nghĩa các cột, kiểu dữ liệu, kích thước, ràng buộc tùy chọn, v.v.

Phạm vi của các biến kiểu bảng trong SQL Server

Tương tự như các biến cục bộ, biến kiểu bảng sẽ không còn tồn tại sau khi kết thúc khối lệnh.

Nếu bạn định nghĩa một biến kiểu bảng trong một stored procedure hoặc function, biến kiểu bảng sẽ không còn tồn tại sau khi stored procedure hoặc function kết thúc.

```
---
--Khai báo biến kiểu table
DECLARE @sach_table TABLE
(
    TenSach VARCHAR(100) NOT NULL,
    SoTrang INT NOT NULL,
    GiaSach DECIMAL(11,2)
);
--Chèn insert dữ liệu vào biến table
INSERT INTO @sach_table
SELECT
    TenSach,
    SoTrang,
    GiaSach
FROM


    Sach
WHERE
    SoTrang = 30;

--Truy vấn xem kết quả của biến kiểu bảng
select * from @sach_table
```

	TenSach	SoTrang	GiaSach
1	Tin h?c van phòng cho công trình	30	NULL
2	Tin h?c gi?i thu?t	30	NULL

Lưu ý: chạy cả 3 mục

Table function trong SQL Server

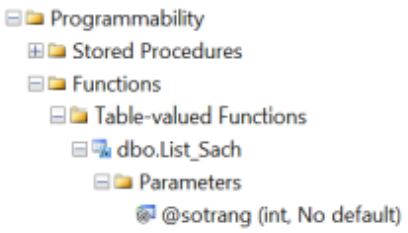
Trong phần này, bạn sẽ tìm hiểu cách sử dụng table function trong SQL Server bao gồm table function đơn câu lệnh và table function đa câu lệnh.

Table function trong SQL Server là gì

Table function (hàm bảng) là function do người dùng tự định nghĩa trả về dữ liệu kiểu bảng. Kiểu trả về của table function là một bảng, do đó bạn có thể sử dụng table function giống như bạn sẽ sử dụng bảng.

Tạo table function trong SQL Server

```
--Tạo table Function trong SQL server
--Table Function
--Danh sách các sách có số trang là tham số đầu vào
CREATE FUNCTION List_Sach
(
    @sotrang INT
)
RETURNS TABLE
AS
RETURN
    SELECT
        TenSach,
        SoTrang,
        NgayXB
    FROM
        Sach
    WHERE
        SoTrang = @sotrang;
```



Thực thi table function trong SQL Server

Để thực thi một table function, bạn sử dụng nó trong mệnh đề `FROM` của câu lệnh `SELECT` như sau:

```
---Thực thi hàm List_Sach
SELECT
    *
FROM
    List_Sach(30);
```

	TenSach	SoTrang	NgayXB
1	Tin học văn phòng cho công trình	30	2019-11-24
2	Tin học giải thuật	30	2020-10-10

```
Có thể chỉ định số cột cần lấy
SELECT TenSach,NgayXB

FROM
    List_Sach(30);
```

	TenSach	NgayXB
1	Tin học văn phòng cho công trình	2019-11-24
2	Tin học giải thuật	2020-10-10

```
--Sửa đổi Function tale
--Liệt kê sách từ số trang bắt đầu đến số trang kết thúc
ALTER FUNCTION List_Sach (
    @start_sotrang INT,
    @end_sotrang INT
)
RETURNS TABLE
AS
RETURN
   SELECT
        TenSach,
        SoTrang,
        NgayXB
```

```
    FROM
        Sach

    WHERE
        SoTrang BETWEEN @start_sotrang AND @end_sotrang
--Gọi hàm với số trang từ 20 đến 50 trang như sau:
SELECT
    *
FROM
    List_Sach(20,50)
ORDER BY
    TenSach;
```

	TenSach	SoTrang	NgayXB
1	Cấp thoát nước	40	2017-08-30
2	Kỹ thuật môi trường cơ bản	20	2017-08-20
3	Tin học đại cương	20	2020-01-01
4	Tin học giải thuật	30	2020-10-10
5	Tin học văn phòng	50	2020-12-20
6	Tin học văn phòng cho công trình	30	2019-11-24

Table function đa câu lệnh trong SQL Server

Table function đa câu lệnh hoặc MSTVF là function có nhiều câu lệnh và trả về giá trị kiểu bảng.

Table function rất hữu ích vì bạn có thể thực hiện nhiều truy vấn trong function và tổng hợp kết quả vào bảng được trả về.

Để định nghĩa table function, bạn sử dụng biến kiểu bảng làm giá trị trả về. Bên trong function, bạn thực hiện một hoặc nhiều truy vấn và chèn dữ liệu vào biến kiểu bảng này.

```
--Function sau kết hợp bảng nhân viên và bảng khách hàng vào 1 danh sách duy nhất.

CREATE FUNCTION List_LienHe()
    RETURNS @contacts TABLE (
        Ma VARCHAR(10),

        KieuLH VARCHAR(20)
    )
AS
BEGIN
    INSERT INTO @contacts
    SELECT
        MaNV,

        'NhanVien'
    FROM
```

```
        NhanVien;

    INSERT INTO @contacts
    SELECT
        MaKH,

        'KhachHang'
    FROM
        KhachHang;
    RETURN;
END;

--Gọi hàm List_LienHe như sau:
SELECT
    *
FROM
    List_LienHe();
```

Results | Messages

	Ma	KieuLH
1	0000000001	NhanVien
2	0000000002	NhanVien
3	0000000003	NhanVien
4	001	KhachHang
5	002	KhachHang

Xóa function trong SQL Server

Trong phần này, bạn sẽ tìm hiểu cách xóa function do người dùng tự định nghĩa bằng cách sử dụng câu lệnh `DROP FUNCTION` trong SQL Server.

Cú pháp:

```
DROP FUNCTION [ IF EXISTS ] [ schema_name. ] function_name;
```

Trong cú pháp này:

- Tùy chọn `IF EXISTS` cho phép bạn xóa function nếu nó tồn tại. Nếu function không tồn tại thì câu lệnh sẽ không làm gì cả. Nếu bạn cố gắng xóa một function không tồn tại mà không chỉ định tùy chọn `IF EXISTS`, bạn sẽ gặp lỗi.
- Tùy chọn `schema_name` chỉ định tên của lược đồ mà function do người dùng tự định nghĩa thuộc về. Tên lược đồ là tùy chọn.
- `function_name` là tên của function mà bạn muốn xóa.

Ghi chú:

Nếu function mà bạn muốn xóa được tham chiếu trong view hoặc function khác được tạo bằng tùy chọn `WITH SCHEMABINDING` thì câu lệnh `DROP FUNCTION` sẽ thất bại.

Ngoài ra, nếu có các ràng buộc như `CHECK` hoặc `DEFAULT` và các cột được tính toán liên quan đến function, câu lệnh `DROP FUNCTION` cũng sẽ thất bại.

Để xóa nhiều function do người dùng tự định nghĩa, bạn chỉ định danh sách tên function được phân tách bằng dấu phẩy sau mệnh đề `DROP FUNCTION` như sau:

```
IF EXISTS (
  SELECT * FROM sysobjects WHERE id = object_id(N'function_name')
  AND xtype IN (N'FN', N'IF', N'TF')
)
  DROP FUNCTION function_name
GO
```

Ghi chú:

- FN = Scalar Function
- IF = Inlined Table Function
- TF = Table Function

Ví dụ:

```
IF EXISTS (
    SELECT * FROM sysobjects WHERE id = object_id(N'abc')
    AND xtype IN (N'FN', N'IF', N'TF')
)
    DROP FUNCTION abc
```

Hoặc có thể dùng

```
IF object_id(N'abc', N'FN') IS NOT NULL
    DROP FUNCTION abc
GO
```

Ghi chú

Nếu function mà bạn muốn xóa được tham chiếu bởi các view hoặc các function khác được tạo bằng tùy chọn WITH SCHEMABINDING, thì chức năng DROP FUNCTION sẽ không thành công.

Ngoài ra, nếu có các ràng buộc như CHECK hoặc DEFAULT và các cột được tính tham chiếu đến hàm, câu lệnh DROP FUNCTION cũng sẽ bị lỗi.

```
--Ví dụ xóa hàm với WITH SHCEMABIDING
CREATE FUNCTION DoanhThuGiamGia
(
    @quantity INT,
    @list_price DEC(10,2),
    @discount DEC(4,2)
)
RETURNS DEC(10,2)
WITH SCHEMABINDING
AS
BEGIN
    RETURN @quantity * @list_price * @discount
END
```

```
--Tạo view sử dụng DoanhThuGiamGia
CREATE VIEW dbo.discounts
WITH SCHEMABINDING
AS
SELECT
    SoDH,
    SUM(dbo.DoanhThuGiamGia(
        SoLuong,
        GiaTien,
        GiamGia
    )) AS So_Tien_Giam_Gia
FROM
    dbo.ChiTietDH
GROUP BY
    SoDH;

--Drop sẽ gặp lỗi

DROP FUNCTION DoanhThuGiamGia;
```

```
Messages

Msg 3729, Level 16, State 1, Line 1
Cannot DROP FUNCTION 'DoanhThuGiamGia' because it is being referenced by object 'discounts'.
```

```
--Để xóa ta thực hiện xóa View sau đó xóa hàm

--Xóa view
drop view discounts;
--Xóa Hàm
drop function DoanhThuGiamGia;
```

Aggregate Functions

An aggregate function performs a calculation one or more values and returns a single value. The aggregate function is often used with the GROUP BY clause and HAVING clause of the SELECT statement.

The following table shows the SQL Server aggregate functions:

The following table shows the SQL Server aggregate functions:

Aggregate function	Description
AVG	The AVG() aggregate function calculates the average of non-NULL values in a set.
CHECKSUM_AGG	The CHECKSUM_AGG() function calculates a checksum value based on a group of rows.
COUNT	The COUNT() aggregate function returns the number of rows in a group, including rows with NULL values.
COUNT_BIG	The COUNT_BIG() aggregate function returns the number of rows (with BIGINT data type) in a group, including rows with NULL values.
MAX	The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.
MIN	The MIN() aggregate function returns the lowest value (minimum) in a set of non-NULL values.
STDEV	The STDEV() function returns the statistical standard deviation of all values provided in the expression based on a sample of the data population.
STDEVP	The STDEVP() function also returns the standard deviation for all values in the provided expression, but does so based on the entire data population.

STDEVP	The `STDEVP()` function also returns the standard deviation for all values in the provided expression, but does so based on the entire data population.
SUM	The `SUM()` aggregate function returns the summation of all non-NULL values a set.
VAR	The `VAR()` function returns the statistical variance of values in an expression based on a sample of the specified population.
VARP	The `VARP()` function returns the statistical variance of values in an expression but does so based on the entire data population.

SQL Server aggregate function syntax

The following illustrates the syntax of an aggregate function:

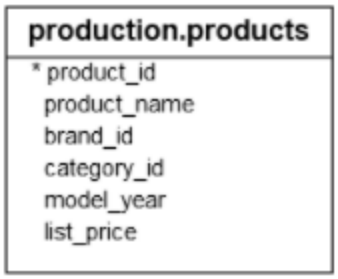
```
aggregate_function_name(DISTINCT | ALL expression)
```

In this syntax;

- First, specify the name of an aggregate function that you want to use such as `AVG`, `SUM`, and `MAX`.
- Second, use `DISTINCT` if you want only distinct values are considered in the calculation or `ALL` if all values are considered in the calculation. By default, `ALL` is used if you don't specify any modifier.
- Third, the `expression` can be a column of a table or an expression that consists of multiple columns with arithmetic operators.

SQL Server aggregate function examples

We will use the `products` table from the sample database for the demonstration.



AVG example

The following statement use the `AVG()` function to return the average list price of all products in the products table:

```
SELECT
    AVG(list_price) avg_product_price
FROM
    production.products;
```

The following shows the output:

avg_product_price
1520.591401

Because the list price in USD, it should have two decimal places at most. Therefore, you need to round the result to a number with two decimal places. To do this, you use the `ROUND` and `CAST` functions as shown in the following query:

```
SELECT
    CAST(ROUND(AVG(list_price),2) AS DEC(10,2))
    avg_product_price
FROM
    production.products;
```

avg_product_price
1520.59

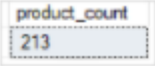
First, the `ROUND` function returns the rounded average list price. And then the `CAST` function converts the result to a decimal number with two decimal places.

COUNT example

The following statement uses the `COUNT()` function to return the number of products whose price is greater than 500:

```
SELECT
    COUNT(*) product_count
FROM
    production.products
WHERE
    list_price > 500;
```

The following shows the output:



In this example:

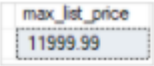
- First, the `WHERE` clause gets products whose list price is greater than 500.
- Second, the `COUNT` function returns the number of products with list prices greater than 500.

MAX example

The following statement uses the MAX() function to return the highest list price of all products:

```
SELECT
    MAX(list_price) max_list_price
FROM
    production.products;
```

The following picture shows the output:

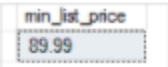


MIN example

Similarly, the following statement uses the MIN() function to return the lowest list price of all products:

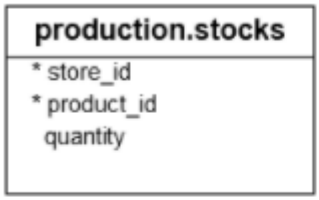
```
SELECT
    MIN(list_price) min_list_price
FROM
    production.products;
```

The output is:



SUM example

To demonstrate the `SUM()` function, we will use the `stocks` table from the sample database.



The following statement uses the SUM() function to calculate the total stock by product id in all warehouses:

```
SELECT
    product_id,
    SUM(quantity) stock_count
FROM
    production.stocks
GROUP BY
    product_id
ORDER BY
    stock_count DESC;
```

Here is the output:

product_id	stock_count
188	86
64	82
109	79
196	79
61	78
182	77
166	77

Here is how the statement works:

- First, the `GROUP BY` clause summarized the rows by product id into groups.
- Second, the `SUM()` function calculated the sum of quantity for each group.

STDEV example

The following statement uses the `STDEV()` function to calculate the statistical standard deviation of all list prices:

```
SELECT
    CAST(ROUND(STDEV(list_price),2) as DEC(10,2)) stdev_list_price
FROM
    production.products;
```

In this tutorial, you have learned about the SQL Server aggregate functions and how to use them to calculate aggregates.

Introduction to SQL Server AVG() function

SQL Server `AVG()` function is an aggregate function that returns the average value of a group.

The following illustrates the syntax of the `AVG()` function:

```
AVG([ALL | DISTINCT] expression)
```

In this syntax:

- `ALL` instructs the `AVG()` function to take all values for calculation. `ALL` is used by default.
- `DISTINCT` instructs the `AVG()` function to operate only on unique values.
- `expression` is a valid expression that returns a numeric value.

The `AVG()` function ignores `NULL` values.

SQL Server AVG() function: ALL vs. DISTINCT

The following statements create a new table, insert some values into the table, and query data against it:

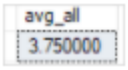
```
CREATE TABLE t(
    val dec(10,2)
);
INSERT INTO t(val)
VALUES(1),(2),(3),(4),(4),(5),(5),(6);

SELECT
    val
FROM
    t;
```

The following statement uses the `AVG()` function to calculate the average of all values in the `t` table:

```
SELECT
    AVG(ALL val)
FROM
    t;
```

The following picture shows the output:



In this example, we used `ALL` modifier, therefore, the average function considers all eight values in the `val` column in the calculation:

```
(1 + 2 + 3 + 4 + 4 + 5 + 5 + 6) /  8 = 3.75
```

The following statement uses the `AVG()` function with `DISTINCT` modifier:

```
SELECT
    AVG(DISTINCT val)
FROM
    t;
```

Here is the result:

avg_distinct
3.500000

Because of the `DISTINCT` modifier, the `AVG()` function performs the calculation on distinct values:

```
(1 + 2 + 3 + 4 + 5 + 6) / 6 = 3.5
```

SQL Server AVG() function examples

Let's take some examples to see how the `AVG()` function works.

A) SQL Server AVG() simple example

The following example returns the average list price of all products:

```
SELECT
    AVG(list_price)
FROM
    production.products;
```

In this example, the `AVG()` function returned a single value for the whole table.

Here is the output:

avg_list_price
1520.591401

B) SQL Server AVG() with GROUP BY example

If you use the `AVG()` function with a `GROUP BY` clause, the `AVG()` function returns a single value for each group instead of single value for the whole table.

The following example returns the average list price for each product category:

```
SELECT
    category_name,
    CAST(ROUND(AVG(list_price),2) AS DEC(10,2))
    avg_product_price
FROM
    production.products p
    INNER JOIN production.categories c
        ON c.category_id = p.category_id
GROUP BY
    category_name
ORDER BY
    category_name;
```

The following picture shows the output:

category_name	avg_product_price
Children Bicycles	287.79
Comfort Bicycles	682.12
Cruisers Bicycles	730.41
Cyclocross Bicycles	2542.79
Electric Bikes	3281.66
Mountain Bikes	1649.76

C) SQL Server AVG() in HAVING clause example

See the following example:

```
SELECT
    brand_name,
    CAST(ROUND(AVG(list_price),2) AS DEC(10,2))
    avg_product_price
FROM
    production.products p
    INNER JOIN production.brands c ON c.brand_id = p.brand_id
GROUP BY
    brand_name
HAVING
    AVG(list_price) > 500
ORDER BY
    avg_product_price;
```

Here is the output:

brand_name	avg_product_price
Sun Bicycles	524.47
Haro	621.99
Ritchey	749.99
Electra	761.01
Surly	1331.75
Heller	2173.00

In this example:

- First, the `GROUP BY` clause divides the products by brands into groups.
- Second, the `AVG()` function calculates average list price for each group.
- Third, the `HAVING` clause removes the brand whose average list price is less than 500.

In this tutorial, you have learned how to use the SQL Server `AVG()` function to calculate the average value from a set of values.

Link tham khảo :

SQL Server COUNT Function By Practical Examples (sqlservertutorial.net)