Hàm (Function)

A. Khái niệm

Tương tự với định nghĩa Stored Procedure – SP (thủ tục), hàm là một tập các lệnh T-SQL và một số cấu trúc điều khiển, được lưu với một tên và được thực thi như là một đơn vị công việc đơn (single unit of work). Trong các ngôn ngữ khác như C, pascal, Basic, một hàm thông thường là một tập các câu lệnh với mục đích hoàn tất một mục đích nào đó và có thể được gọi từ một chương trình như là một lệnh đơn.

Hàm được dùng trong:

- Danh sách chọn của một câu lệnh Select để trả về một giá trị.
- Một điều kiện tìm kiếm của mệnh đề Where trong các câu lệnh T-SQL

B. Phân loại hàm

Thông thường hàm được chia ra làm 2 dạng: hàm hệ thống và hàm do người dùng tự định nghĩa

I. Hàm hệ thống (Build-in functions)

Định nghĩa: Là tập hợp các hàm được định nghĩa sẵn, đi kèm với HQT CSDL cài đặt. Những hàm này hoạt động như là một định nghĩa trong T-SQL và không thể hiệu chỉnh. Hàm hệ thống có thể chỉ được tham chiếu trong các câu lệnh T-SQL. Giá trị trả về của hàm có thể là một Rowset (tập các dòng), Argergate (giá trị tổng hợp) và Scalar (vô hướng) – nghĩa là các giá trị đơn.

Các hàm do hệ thống định nghĩa được chia thành 5 nhóm hàm chính như sau: nhóm hàm xử lý chuỗi (String functions), nhóm hàm về ngày giờ (Date functions), nhóm hàm toán học (Mathematical functions), nhóm hàm kết hợp (aggregate Functions), nhóm hàm hệ thống khác (System functions).

1. Nhóm hàm xử lý chuỗi (String functions)

Thao tác với dữ liệu kiểu ký tự.

- Hàm **CHARINDEX (***string1, string2, start_position***)**: Tìm vị trí bắt đầu của chuỗi ký tự chỉ định string1 trong chuỗi string2 và bắt đầu tìm ở vị trí start_position trong chuỗi string2.

Ví dụ: `SELECT CHARINDEX ('test', 'This is a test', 1)` → Hàm sẽ trả về giá trị 11, vị trí bắt đầu của chuỗi 'test' trong chuỗi 'This is a test'.

- Hàm **LEFT (***string, number_of_characters***)**: Trả về chuỗi gồm number_of_characters ký tự tính từ trái sang của chuỗi string.

Ví dụ: `SELECT LEFT ('This is a test', 4)` → Hàm sẽ trả về chuỗi 'This'

- Hàm **LEN** (*string)*: Xác định độ dài của chuỗi ký tự *string.*

Ví dụ : `SELECT LEN ('This is a test')` → Hàm sẽ trả về giá trị 14

- Hàm **LOWER (***string***)**: Hàm trả về chuỗi ký tự thường.

Ví dụ: `SELECT LOWER ('This is a TEST')` → Hàm sẽ trả về chuỗi 'this is a test'

- Hàm **LTRIM (***string***)**: Cắt bỏ các ký tự trắng bên trái của chuỗi.

Ví dụ: `SELECT LTRIM (' This is a test ')` → Hàm sẽ trả về chuỗi 'This is a test '

- Hàm **RIGHT** (*string, number_of_characters*): trả về chuỗi gồm number_of_characters ký tự tính từ phải sang của chuỗi string.

Ví dụ: `SELECT RIGHT ('This is a test', 4)` → Hàm sẽ trả về chuỗi 'test'

- Hàm **RTRIM** (*string*): Cắt bỏ các ký tự trắng bên phải của chuỗi.

Ví dụ: `SELECT RTRIM(' This is a test ')` → Hàm sẽ trả về chuỗi ' This is a test'

❖ Hàm **SUBSTRING** (*expression, start, length*): Hàm trả về chuỗi con gồm length ký tự của expression tính từ vị trí start.

Ví dụ: `SELECT x = SUBSTRING ('abcdef', 2, 3)` → Hàm sẽ trả về chuỗi 'bcd'

❖ Hàm **UPPER** (*string)*: Chuyển đổi các ký thường thành chữ hoa.

Ví dụ: `SELECT UPPER ('This is a TEST')` → Hàm sẽ trả về chuỗi 'THIS IS A TEST'

2. Nhóm hàm về ngày giờ (Date functions)

Là các hàm làm việc với dữ liệu kiểu datetime. Một số hàm làm việc với các kiểu thông tin đặc biệt được gọi là **datepart**. Trước khi đi vào các hàm, ta xét một số ký tự đặc tả của **datepart** cho trong bảng sau:

Datepart	**Abbreviation**	**Values**
Hour	hh	0 - 23
Minute	Mi	0 - 59
Second	Ss	0 - 59
Millisecond	Ms	0 - 999
Day of Year	Dy	1 - 366
Day	Dd	1 - 31
Week	wk	1 - 53
Weekday	dw	1 - 7

Month	mm	1 - 12
Quarter	qq	1 - 4
Year	yy	1753 - 9999

- Hàm **DATEADD** (*datepart, amount, date*): cộng thêm một số amount thời gian thành phần datepart của date.

Ví dụ: `SELECT DATEADD (year, 1, GETDATE()))` → Hàm sẽ trả về ngày hiện tại cộng thêm một năm.

- Hàm **DATEDIFF** (*datepart, date1, date2*): so sánh chênh lệch giữa hai ngày bằng việc sử dụng tham số datepart.

Ví dụ: `SELECT DATEDIFF (hour, '1/1/2008 12:00:00', '1/1/2008 16:00:00')` → trả về giá trị 4 → hai ngày chênh nhau 4 giờ.

- Hàm **DATEPART** (*datepart, date*): trả về giá trị của thành phần datepart trong date.

Ví dụ: `SELECT DATEPART (month, '1/1/2008 16:00:00')` → trả về giá trị tháng 1.

- Hàm **DAY** (*date*): Xác định số ngày của tháng trong dữ liệu ngày giờ date.

Ví dụ: `SELECT DAY ('7/22/1979 00:04:00')` → trả về giá trị ngày là 22

- Hàm **GETDATE ():** Trả về giá trị ngày hiện tại của hệ thống.
- Hàm **MONTH** (*date*): Trả về tháng của dữ liệu ngày giờ.
- Hàm **YEAR** (*date*): Trả về năm của dữ liệu ngày giờ.

3. Nhóm hàm toán học (Mathematical functions)

- Hàm **ABS** (*number)*: Trả về giá trị tuyệt đối của số number.
- Hàm **CEILING** *(number):* Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng number.

- Hàm **FLOOR** (*number*): Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng number.
- Hàm **ROUND** (*number, precision*): Hàm làm tròn số number lấy precision chữ số sau dấu thập phân.
- Hàm **SQUARE** (*number*): Hàm trả về giá trị bình phương số number.
- Hàm **SQRT** (number): Hàm trả về giá trị căn bậc hai số number.

4. Nhóm hàm kết hợp (aggregate Functions)

Các hàm tập hợp thực hiện tính toán trên một tập hợp các giá trị và trả về một giá trị đơn. Ngoại trừ hàm COUNT, hàm tập hợp bỏ qua các giá trị NULL.

Các hàm tập hợp thường sử dụng với mệnh đề GROUP BY trong khối câu lệnh SELECT. Hàm tập hợp được phép dùng như là các biểu thức trong trường hợp:

- Trong danh sách select của khối câu lệnh SELECT.
- Trong mệnh đề COMPUTE hoặc COMPUTE BY .
- Trong mệnh đề HAVING

Sau đây là một số hàm tập hợp hay được sử dụng:

- Hàm **AVG** *([ALL|DISTINCT] expression*): trả về giá trị trung bình của tập các giá trị trong một nhóm.

  Trong đó:

  - **ALL**: Áp dụng cho các hàm tập hợp để chỉ định cho tất cả các giá trị. ALL là từ khóa mặc định.
  - **DISTINCT**: Chỉ định chỉ lấy một thể hiện duy nhất của một giá trị. Nghĩa là trong tập hợp có nhiều phần tử có cùng một giá trị thì chỉ lấy một giá trị đại diện cho nó.

Ví dụ: Hãy tìm mức lương trung bình của các phòng

```
USE         QUANLYDEANCONGTY

SELECT      DISTINCT AVG (LUONG)

FROM        NHANVIEN
```

```
GROUP BY    PHG
```

→ Có nhiều phòng có mức lương trung bình bằng nhau thì kết quả chỉ trả về 1 giá trị TB chung (ví dụ tồn tại 2 phòng đều có giá trị lương trung bình là 3000$ thì chỉ trả về 1 dòng 3000$)

- Hàm **COUNT** ({[ALL|DISTINCT] expression]|*}): trả về kiểu int số các phần tử của một nhóm.

Lưu ý: Khi sử dụng hàm COUNT

- **COUNT (*):** Trả về số các phần tử trong một nhóm bao gồm cả giá trị NULL và giá trị duplicates.
- **COUNT (**ALL *expression*): xác định giá trị cho *expression* tại mỗi dòng trong nhóm và trả về số các giá trị không NULL
- **COUNT** (DISTINCT *expression*): xác định giá trị cho *expression* tại mỗi dòng trong nhóm và trả về số các giá trị không trùng và không NULL

- Hàm **COUNT_BIG**({[ALL|DISTINCT] expression}|*): Trả về số các phần tử trong một nhóm. Hàm COUNT_BIG làm việc như hàm COUNT. Điểm khác nhau giữa chúng là hàm COUNT trả về giá trị kiểu int còn hàm COUNT_BIG trả về giá trị kiểu bigint.
- Hàm **MAX** ([ALL|DISTINCT] expression): Trả về giá trị lớn nhất trong biểu thức expression.
- Hàm **MIN** ([ALL|DISTINCT] expression): Trả về giá trị lớn nhất trong biểu thức expression
- Hàm **SUM** ([ALL|DISTINCT] expression): Trả về tổng của tất cả các giá trị của biểu thức hoặc tổng các giá trị DISTINCT của biểu thức expression. Hàm SUM chỉ áp dụng cho các cột kiểu số. Các giá trị NULL được bỏ qua.

5. Nhóm hàm hệ thống khác (System functions)

Các hàm hệ thống là các hàm lấy thông tin hệ thống về các đối tượng và đã thiết lập trong SQL Server.

- Hàm **CONVERT** (data_type, expression): Chuyển đổi biểu thức expression thành kiểu dữ liệu data_type

Ví dụ: `SELECT CONVERT (VARCHAR (5), 12345)` → trả về chuỗi '12345'.

- Hàm **CAST** (expression AS data_type): Chuyển đổi biểu thức expression thành kiểu dữ liệu data_type.
- Hàm **CURRENT_USER**: Trả về người sử dụng hiện tại

Ví dụ: `SELECT CURRENT_USER`

- Hàm **DATALENGTH** (expression): Trả về số byte được sử dụng trong biểu thức expression.
- Hàm **HOST_NAME** (): trả về tên máy tính mà người sử dụng hiện tại đang login.

Ví dụ : `SELECT HOST_NAME()`

- Hàm **SYSTEM_USER**: Hàm trả về tên của các User đang login hệ thống.

Ví dụ: `SELECT SYSTEM_USER`

- Hàm **USER_NAME():** Hàm trả về username khi đưa số user ID.

Ví dụ:

```
SELECT name
FROM sysobjects
WHERE USER_NAME (uid) = 'dbo'
```

II. Hàm do người dùng định nghĩa (User-defined function - UDFs)

UDFs: là những hàm này do người dùng tự định nghĩa để đáp ứng một mục tiêu nào đó. Một số hạn chế so với thủ tục là các tham số truyền vào không được mang thuộc tính OUTPUT, nghĩa là giá trị của tham số không được truyền ra bên ngoài hàm UDFs, thay vào đó ta phải sử dụng giải pháp là trả về giá trị cho hàm bằng phát biểu RETURN. Giá trị trả về của hàm có thể là một giá trị vô hướng (Scalar valued) hoặc bảng (Table-valued)

UDFs thường được chia thành 3 dạng cơ bản:

Scalar Functions:dạng hàm vô hướng trả về một giá trị đơn và có thể được sử dụng như một biến/giá trị trong một biểu thức (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

Table-valued Functons: dạng hàm có giá trị bảng trả về một tập kết quả và có thể được sử dụng như một bảng dữ liệu hay view. Hàm giá trị bảng có thể được tham chiếu trong mệnh đề FROM của câu lệnh SELECT.

Multistatement Table-valued: dạng hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT

C. Tạo lập và sử dụng hàm

Cú pháp chung

CREATE FUNCTION Function_name ([<parameter> [, …n]**)**

RETURNS (data_type)

AS

sql_statement [...]

Trong đó:

- **Parameter:** Các tham số cách nhau bởi dấu phẩy. Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần:tên tham số được bắt đầu bởi dấu @, kiểu dữ liệu của tham số;
- **Data_type:** kiểu dữ liệu trả về của hàm;
- **Sql_statement:** tập hợp các câu lệnh sử dụng trong nội dung hàm. Các câu lệnh này có thể đặt trong cặp từ khoá BEGIN...END hoặc có thể không.

Dựa trên cơ sở phân loại UDFs thành 3 loại: Scalar Functions, Table-valued Functons, Multistatement Table-valued. Tương ứng có các cú pháp xây dựng hàm như sau

I. Scalar Functions

Đặc điểm: Dạng hàm vô hướng trả về một giá trị đơn và có thể được sử dụng như một biến/giá trị trong một biểu thức (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

Cú pháp

CREATE FUNCTION Function_name ([<parameter> [, …n]**)**

RETURNS (data_type)

AS

BEGIN

<sql_statement [...]>

RETURN (result)

END

VD1: Viết hàm trả về thứ của một ngày trong tuần.

```
GO
CREATE FUNCTION fun_Thu (@ngay DATETIME)
RETURNS NVARCHAR(10)
AS
     BEGIN
          DECLARE @st NVARCHAR(10)
          SELECT @st = CASE DATEPART(DW, @ngay)
                       WHEN 1 THEN 'Chủ nhật'
                       WHEN 2 THEN 'Thứ hai'
                       WHEN 3 THEN 'Thứ ba'
                       WHEN 4 THEN 'Thứ tư'
                       WHEN 5 THEN 'Thứ năm'
```

```
                    WHEN 6 THEN 'Thứ sáu'
                    ELSE 'Thứ bảy'
                END
        RETURN (@st) /* Trị trả về của hàm */
    END
GO
```

Lưu ý: một hàm khi đã được định nghĩa có thể được sử dụng như các hàm do HQT CSDL cung cấp (thông thường trước tên hàm ta phải chỉ định thêm tên của người sở hữu hàm)

Câu lệnh SELECT dưới đây sử dụng hàm đã được định nghĩa ở ví dụ trước:

```
SELECT    HONV, TENLOT, TENNV, DBO. THU (NGAYSINH)
FROM      NHANVIEN
    WHERE     PHG = 5
```

VD2: Viết thủ tục cập nhật lương cho nhân viên theo công thức

Lương = Lương cơ bản + Phụ cấp chức vụ + Phụ cấp giờ làm

Trong đó: Lương cơ bản = Hệ số lương * 200, Hệ số lương = 2.34,

Phụ cấp chức vụ = 1000 nếu nhân viên đó là trưởng phòng,

Phụ cấp giờ làm = 500 nếu có tham gia đề án với thời gian trên 20 giờ

Yêu cầu có sử dụng hàm

```
GO
CREATE PROC sp_CapNhatLuong
AS
BEGIN
    UPDATE NHANVIEN
    SET LUONG = 2.34 * 200 + dbo.fun_PhuCapQuanLy(MANV) +
```

```
          dbo.fun_PhuCapGioLam(MANV)
END
GO
--Hàm tính phụ cấp quản lý
GO
CREATE FUNCTION fun_PhuCapQuanLy(@MANV NVARCHAR(9))
RETURNS INT
AS
BEGIN
     IF EXISTS (SELECT MAPHG
                FROM PHONGBAN
                WHERE TRPHG = @MANV)
          RETURN 1000
     RETURN 0
END
GO
--Hàm tính phụ cấp giờ làm
GO
CREATE FUNCTION fun_PhuCapGioLam(@MANV NVARCHAR(9))
RETURNS INT
AS
BEGIN
     IF EXISTS (SELECT MADA
                FROM PHANCONG
                WHERE MA_NVIEN = @MANV
                GROUP BY MADA, MA_NVIEN
                HAVING SUM(THOIGIAN) > 20)
          RETURN 500
     RETURN 0
```

```
END
GO
```

VD3: Viết thủ tục tăng lương cho trưởng phòng, với mỗi nhân viên thuộc phòng ban của trưởng phòng đó tham gia 2 đề án trở lên, trưởng phòng đó được tăng 200 lương

```
GO
CREATE PROC fun_TangLuongTrongPhong
AS
BEGIN
     UPDATE NHANVIEN
     SET LUONG = LUONG +dbo.fun_TinhluongSoNVThamGia2DA(PHG)
     WHERE exists (SELECT MAPHG
                   FROM PHONGBAN
                   WHERE TRPHG = MANV)
END
GO
--Hàm tính lương sau khi đếm số nhân viên tham gia 2 đề án
GO
CREATE FUNCTION fun_TinhluongSoNVThamGia2DA(@MAPHG INT)
RETURNS INT
AS
BEGIN
     IF EXISTS (SELECT MA_NVIEN
                FROM PHANCONG
                WHERE EXISTS (SELECT MANV
                              FROM NHANVIEN
                              WHERE MA_NVIEN = MANV
                                 AND PHG = @MAPHG))
          RETURN (SELECT COUNT(MA_NVIEN)
                  FROM PHANCONG
```

```
                    WHERE EXISTS (SELECT MANV
                                  FROM NHANVIEN
                                  WHERE MA_NVIEN = MANV
                                        AND PHG = @MAPHG)
                    GROUP BY MA_NVIEN
                    HAVING COUNT(MADA) > 2) * 200
    RETURN 0
END
GO
```

II. Table-valued Functons

Đặc điểm: Dạng hàm vô hướng trả về một giá trị đơn và có thể được sử dụng như một biến/giá trị trong một biểu thức (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

Cú pháp

CREATE FUNCTION Function_name ([<parameter> [, …n]**)**

RETURNS TABLE

AS

BEGIN

RETURN (SELECT_statement)

END

Trong đó:

- Kiểu trả về của hàm phải được chỉ định bởi mệnh đề RETURNS TABLE.
- Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

VD1: Hãy trả về danh sách các nhân viên ở phòng có tên theo yêu cầu người truy vấn

```
GO
CREATE FUNCTION fun_XemNV (@tenphong nvarchar(30))
RETURNS TABLE
AS
     RETURN(SELECT HONV, TENLOT, TENNV
            FROM NHANVIEN INNER JOIN PHONGBAN ON PHG = MAPH
            WHERE TENPHG = @tenphong)
GO
```

Với hàm được định nghĩa như trên, để biết danh sách các nhân viên phòng nghiên cứu, ta sử dụng câu lệnh như sau:

```
SELECT * FROM dbo.func_XemNV ('Nghiên cứu')
```

VD2: Viết hàm cho biết danh sách các nhân viên thuộc phòng ban Nghiên cứu tham gia 3 đề án trở lên

```
GO
CREATE FUNCTION fun_DSNVNghienCuu3DeAn()
RETURNS TABLE
AS
     RETURN
     (SELECT MANV, HONV + ' ' + TENLOT + ' ' + TENNV 'HO
TEN'
      FROM NHANVIEN INNER JOIN PHANCONG ON MANV = MA_NVIEN
          INNER JOIN PHONGBAN ON PHG = MAPHG
      WHERE TENPHG LIKE N'%Nghiên cứu%'
      GROUP BY MANV, HONV, TENLOT, TENNV
      HAVING COUNT(MADA) >= 3)
GO
```

Xuất kết quả

```
SELECT * FROM dbo.fun_DSNVNghienCuu3DeAn()
```

VD3: Viết hàm cho biết danh sách các phòng ban với các thông tin mã phòng, tên phòng, số nhân viên nữ, số nhân viên nam

```
GO
CREATE FUNCTION fun_DSPhongBan()
RETURNS TABLE
as
     RETURN(SELECT  MAPHG,  TENPHG,  dbo.fun_DemSoNVNu(MAPHG)
'SO NV NU', dbo.fun_DemSoNVNam(MAPHG) 'SO NV NAM'
               FROM PHONGBAN)
GO
--Hàm đếm sô nhân viên nữ
GO
CREATE FUNCTION fun_DemSoNVNu(@MAPHG INT)
RETURNS INT
AS
BEGIN
     DECLARE @SoNVNu INT
     SELECT @SoNVNu = COUNT(NHANVIEN.MANV)
     FROM NHANVIEN
     WHERE PHG = @MAPHG AND PHAI LIKE N'%Nữ%'
     RETURN @SoNVNu
END
GO
--Hàm đếm sô nhân viên nam
GO
CREATE FUNCTION fun_DemSoNVNam(@MAPHG INT)
```

```
RETURNS INT
AS
BEGIN
     DECLARE @SoNVNam INT
     SELECT @SoNVNam = COUNT(NHANVIEN.MANV)
     FROM NHANVIEN
     WHERE PHG = @MAPHG AND PHAI LIKE N'%Nam%'
     RETURN @SoNVNam
END
GO
```

Xuất kết quả

```
SELECT * FROM dbo.fun_DSPhongBan()
```

III. Multistatement Table-valued

Đặc điểm: dạng hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT

Cú pháp

```
CREATE FUNCTION [owner_name.]function_name
([{@parameter_name [AS] data_type [=default]} [, … n]])
RETURNS @TABLE_NAME
TABLE ({column_definition | table_constraint} [, … n])
[WITH {ENCRYPTION |SCHEMABINDING} [[,] ... n]]
[AS]
BEGIN
     < sql_statement [...]>
     RETURN
END
```

Khi định nghĩa hàm dạng này cần lưu ý một số điểm sau:

- Cấu trúc của bảng trả về bởi hàm được xác định dựa vào định nghĩa của bảng trong mệnh đề RETURNS. Biến *@biến_bảng* trong mệnh đề RETURNS có phạm vi sử dụng trong hàm và được sử dụng như là một tên bảng.
- Câu lệnh RETURN trong thân hàm không chỉ định giá trị trả về. Giá trị trả về của hàm chính là các dòng dữ liệu trong bảng có tên là *@biếnbảng* được định nghĩa trong mệnh đề RETURNS

VD1: Hãy viết hàm tạo biến bảng thống kê, biến bảng này mang giá trị là một bảng dữ liệu gồm có 3 cột: mã phòng, tên phòng, tổng số nhân viên của phòng đó.

```
GO
CREATE FUNCTION fun_TongNV (@phong INT)
RETURNS @bangthongke
TABLE
(
     MaPhong INT,
     TenPhong NVARCHAR (50),
     TongSoNV INT
)
AS
BEGIN
     IF @phong = 0
          INSERT INTO @bangthongke
          SELECT MAPHG, TENPHG, COUNT (MANV)
          FROM NHANVIEN INNER JOIN PHONGBAN ON PHG = MAPHG
          GROUP BY MAPHG, TENPHG
     ELSE
          INSERT INTO @bangthongke
          SELECT MAPHG, TENPHG, COUNT (MANV)
          FROM NHANVIEN INNER JOIN PHONGBAN ON PHG = MAPHG
          WHERE PHG = @phong
          GROUP BY MAPHG, TENPHG
     RETURN /*Trả kết quả về cho hàm*/
END
GO
```

- Để lập bảng thống kê số lượng nhân viên của tất cả các phòng, ta sử dụng câu lệnh:

```
SELECT * FROM dbo.fun_TongNV(0) ORDER BY TongSoNV
```

- Để lập bảng thống kê số lượng nhân viên của một phòng bất kỳ khi biết mã số phòng chẳng hạn như phòng 5, ta sử dụng câu lệnh như sau:

```
SELECT * FROM dbo.fun_TongNV(5)
```

VD2: Xuất ra danh sách các phòng ban có số nhân viên nam lớn hơn số nhân viên nữ, nêu không có thì xuất ra dánh sách tất cả các phòng ban. Thông tin bao gồm mã phòng, tên phòng, số nv nam, số nv nữ, lương trưởng phòng và lương trung bình.

```
GO
CREATE FUNCTION fun_DSPhongBan2()
RETURNS @bangthongke TABLE
(
     MaPhg INT,
     TenPhg NVARCHAR(15),
     SoNVNu INT,
     SoNVNam INT,
     LuongTP INT,
     LuongTB INT
)
AS
BEGIN
     IF EXISTS (SELECT MAPHG
                FROM PHONGBAN
                WHERE dbo.fun_DemSoNVNam(MAPHG) -
                     dbo.fun_DemSoNVNu(MAPHG) > 0)
          INSERT INTO @bangthongke
          SELECT MAPHG, TENPHG, dbo.fun_DemSoNVNu(MAPHG),
```

```
            dbo.fun_DemSoNVNam(MAPHG),
            dbo.fun_TinhLuongTruongPhong(MAPHG),
            dbo.fun_TinhLuongTrungBinh(MAPHG)
        FROM PHONGBAN
        WHERE dbo.fun_DemSoNVNam(MAPHG) -
            dbo.fun_DemSoNVNu(MAPHG) > 0
    ELSE
        INSERT INTO @bangthongke
        SELECT MAPHG, TENPHG, dbo.fun_DemSoNVNu(MAPHG),
            dbo.fun_DemSoNVNam(MAPHG),
            dbo.fun_TinhLuongTruongPhong(MAPHG),
            dbo.fun_TinhLuongTrungBinh(MAPHG)
        FROM PHONGBAN
    RETURN
END
GO
--Hàm tính lương trưởng phòng
GO
CREATE FUNCTION fun_TinhLuongTruongPhong(@MAPHG INT)
RETURNS INT
AS
BEGIN
    DECLARE @LuongTP INT
    SELECT @LuongTP = LUONG
    FROM NHANVIEN
    WHERE MANV = (SELECT TRPHG
                  FROM PHONGBAN
                  WHERE MAPHG = @MAPHG)
    RETURN @LuongTP
```

```
END
GO
--Hàm tính lương trung bình phòng
GO
CREATE FUNCTION fun_TinhLuongTrungBinh(@MAPHG INT)
RETURNS INT
AS
BEGIN
     DECLARE @LuongTB INT
     SELECT @LuongTB = AVG(LUONG)
     FROM NHANVIEN
     WHERE PHG = @MAPHG
     RETURN @LuongTB
END
GO
```

Xuất kết quả

```
SELECT * FROM dbo.fun_DSPhongBan2()
```

VD3: Viết hàm cho biết danh sách các đề án có nhân viên nữ tham gia, thêm thuộc tính LuuY (1: số nhân viên nữ lớn hơn 3, 0: số nhân viên nữ nhỏ hơn hoặc bằng 3)

```
GO
CREATE FUNCTION fun_DSDeAn()
RETURNS @bangthongke TABLE
(
     MaDA INT,
     TenDA NVARCHAR(15),
     SoNVNu INT,
     SoNV INT,
     LuuY INT
```

```
)
AS
BEGIN
     IF EXISTS (SELECT MADA
                FROM PHANCONG
                WHERE EXISTS (SELECT MANV
                              FROM NHANVIEN
                              WHERE MANV = MA_NVIEN
                                 AND PHAI LIKE N'%Nữ%'))
          INSERT INTO @bangthongke
          SELECT MADA, TENDA,
               dbo.fun_DemSoNVNuThamGiaDA(MADA),
               dbo.fun_DemSoNVThamGiaDA(MADA),
               dbo.fun_LuuY(MADA)
          FROM DEAN
          WHERE EXISTS(SELECT MADA
                       FROM PHANCONG
                       WHERE DEAN.MADA = PHANCONG.MADA
                         AND EXISTS(SELECT MANV
                                    FROM NHANVIEN
                                    WHERE MANV = MA_NVIEN
                                       AND PHAI LIKE N'%Nữ%'))
     ELSE
          INSERT INTO @bangthongke
          SELECT MADA, TENDA, 0,
               dbo.fun_DemSoNVThamGiaDA(MADA), 0
          FROM DEAN
     RETURN
END
```

```
GO
--Hàm đếm số nhân viên nữ tham gia đề án
GO
CREATE FUNCTION fun_DemSoNVNuThamGiaDA(@MADA INT)
RETURNS INT
AS
BEGIN
     DECLARE @SoNVNu INT
     SELECT @SoNVNu = COUNT(MA_NVIEN)
     FROM PHANCONG
     WHERE MADA = @MADA
          AND EXISTS (SELECT MANV
                      FROM NHANVIEN
                      WHERE MANV = MA_NVIEN
                          AND PHAI LIKE N'%Nữ%')
     RETURN @SoNVNu
END
GO
--Hàm đếm số nhân viên tham gia đề án
GO
CREATE FUNCTION fun_DemSoNVThamGiaDA(@MADA INT)
RETURNS INT
AS
BEGIN
     DECLARE @SoNV INT
     SELECT @SoNV = count(MA_NVIEN)
     FROM PHANCONG
     WHERE MADA = @MADA
     RETURN @SoNV
```

```
END
GO
--Hàm trả về giá trị cột lưu ý
GO
CREATE FUNCTION fun_LuuY(@MADA INT)
RETURNS INT
AS
BEGIN
     IF(dbo.fun_DemSoNVNuThamGiaDA(@MADA) > 3)
          RETURN 1
     RETURN 0
END
GO
```

Xuất kết quả

```
SELECT * FROM dbo.fun_DSDeAn()
```