Sử dụng hàm cửa sổ (window functions) trong truy vấn SQL

1. Hàm cửa sổ là gì?

Khi thực hiện các truy vấn thống kê dữ liệu sử dụng mệnh đề GROUP BY, các biểu thức/hàm thống kê sẽ được tính toán trên từng nhóm dữ liệu. Kết quả của câu truy vấn cho chúng ta biết được giá trị thống kê trên từng nhóm, tuy nhiên chi tiết các dòng dữ liệu trong mỗi nhóm là gì thì chúng ta không quan sát được.

Trong nhiều trường hợp, chúng ta cần phải thực hiện các yêu cầu truy vấn dữ liệu mà kết quả của truy vấn vừa cho biết được giá trị thống kê, vừa cho biết được chi tiết của các dòng dữ liệu. Hãy hình dung yêu cầu này qua ví dụ trên bảng NhanVien có cấu trúc và dữ liệu được tạo bởi các lệnh sau:

```
CREATE TABLE NhanVien
(
  MaNV nvarchar(10) primary key,
  HoTen nvarchar(50) not null,
  ChucVu nvarchar(50) not null,
  TienLuong money not null,
  DonVi nvarchar(50) not null
)
GO

INSERT INTO NhanVien(MaNV, HoTen, ChucVu, TienLuong, DonVi)
VALUES
('NV01', N'Lê Thanh An', N'Trưởng phòng', 3000, N'Nhân sự'),
('NV02', N'Mai Thị Hạnh', N'Nhân viên', 1000, N'Nhân sự'),
('NV03', N'Trần Văn Hữu', N'Phó phòng', 1200, N'Nhân sự'),
('NV04', N'Nguyễn Thị Hoa', N'Nhân viên', 800, N'Nhân sự'),
('NV05', N'Trần Chí Hiếu', N'Trưởng phòng', 2500, N'Kinh doanh'),
('NV06', N'Vũ Thanh Lê', N'Nhân viên', 500, N'Kinh doanh'),
('NV07', N'Hoàng Mai Hương', N'Nhân viên', 800, N'Kinh doanh'),
('NV08', N'Nguyễn Ngọc Tùng', N'Nhân viên', 1200, N'Kinh doanh'),
('NV09', N'Trần Thanh Toàn', N'Nhân viên', 1000, N'Kế toán'),
('NV10', N'Nguyễn Văn Bắc', N'Nhân viên', 800, N'Nhân sự'),
('NV11', N'Trần Thanh Phong', N'Nhân viên', 1000, N'Kinh doanh'),
```

```
('NV12', N'Vũ Thị Hoa', N'Nhân viên', 500, N'Nhân sự'),
('NV13', N'Lương Chí Hữu', N'Trưởng phòng', 2000, N'Kế toán'),
('NV14', N'Nguyễn Hữu Cảnh', N'Nhân viên', 800, N'Kế toán'),
('NV15', N'Nguyễn Quang Linh', N'Nhân viên', 850, N'Kinh doanh'),
('NV16', N'Nguyễn Hoàng Vũ', N'Phó phòng', 1500, N'Nhân sự'),
('NV17', N'Lương Thị Hải', N'Phó phòng', 1800, N'Kế toán'),
('NV18', N'Vũ Văn Vương', N'Phó phòng', 1600, N'Nhân sự'),
('NV19', N'Bạch Hải Châu', N'Phó phòng', 1700, N'Kế toán'),
('NV20', N'Nguyễn Văn Hùng', N'Phó phòng', 1400, N'Kinh doanh');
GO
```

Dữ liệu trong bảng như sau:

```
SELECT * FROM NhanVien
```

MaNV	HoTen	ChucVu	TienLuong	DonVi
NV01	Lê Thanh An	Trưởng phòng	3000.00	Nhân sự
NV02	Mai Thị Hạnh	Nhân viên	1000.00	Nhân sự
NV03	Trần Văn Hữu	Phó phòng	1200.00	Nhân sự
NV04	Nguyễn Thị Hoa	Nhân viên	800.00	Nhân sự
NV05	Trần Chí Hiếu	Trưởng phòng	2500.00	Kinh doanh
NV06	Vũ Thanh Lê	Nhân viên	500.00	Kinh doanh
NV07	Hoàng Mai Hương	Nhân viên	800.00	Kinh doanh
NV08	Nguyễn Ngọc Tùng	Nhân viên	1200.00	Kinh doanh
NV09	Trần Thanh Toàn	Nhân viên	1000.00	Kế toán
NV10	Nguyễn Văn Bắc	Nhân viên	800.00	Nhân sự
NV11	Trần Thanh Phong	Nhân viên	1000.00	Kinh doanh
NV12	Vũ Thị Hoa	Nhân viên	500.00	Nhân sự
NV13	Lương Chí Hữu	Trưởng phòng	2000.00	Kế toán
NV14	Nguyễn Hữu Cảnh	Nhân viên	800.00	Kế toán
NV15	Nguyễn Quang Linh	Nhân viên	850.00	Kinh doanh
NV16	Nguyễn Hoàng Vũ	Phó phòng	1500.00	Nhân sự
NV17	Lương Thị Hải	Phó phòng	1800.00	Kế toán
NV18	Vũ Văn Vương	Phó phòng	1600.00	Nhân sự
NV19	Bạch Hải Châu	Phó phòng	1700.00	Kế toán
NV20	Nguyễn Văn Hùng	Phó phòng	1400.00	Kinh doanh

Truy vấn sau đây cho biết mức lương trung bình của mỗi đơn vị:

```
SELECT DonVi, AVG(TienLuong) as LuongTB
FROM NhanVien
GROUP BY DonVi;
```

DonVi	LuongTB
Kế toán	1460.00
Kinh doanh	1178.5714
Nhân sự	1300.00

Kết quả của truy vấn cho chúng ta biết được lương trung bình của mỗi đơn vị (tức là mỗi nhóm sẽ được hiển thị là một dòng trong kết quả truy vấn), tuy nhiên chúng ta không thể biết được cụ thể lương của mỗi nhân viên là bao nhiêu.

Giả sử, chúng ta cần phải so sánh mức lương của mỗi nhân viên với lương trung bình của đơn vị mà nhân viên đó làm việc, ta có thể sử dụng truy vấn trên như là một truy vấn con và thực hiện truy vấn như sau:

```
SELECT  nv.MaNV,
        nv.HoTen,
        nv.DonVi,
        nv.TienLuong,
        ltb.LuongTB
FROM    NhanVien AS nv
        JOIN
        (
              SELECT DonVi, AVG(TienLuong) as LuongTB
              FROM NhanVien
              GROUP BY DonVi
        ) AS ltb ON nv.DonVi = ltb.DonVi
ORDER BY nv.DonVi
```

Kết quả của truy vấn trên sẽ cho chúng ta có được thông tin để có thể so sánh giữa mức lương của mỗi nhân viên với mức lương trung bình của đơn vị mà nhân viên đó làm việc:

MaNV	HoTen	DonVi	TienLuong	LuongTB
NV09	Trần Thanh Toàn	Kế toán	1000.00	1460.00
NV13	Lương Chí Hữu	Kế toán	2000.00	1460.00
NV14	Nguyễn Hữu Cảnh	Kế toán	800.00	1460.00
NV17	Lương Thị Hải	Kế toán	1800.00	1460.00
NV19	Bạch Hải Châu	Kế toán	1700.00	1460.00
NV20	Nguyễn Văn Hùng	Kinh doanh	1400.00	1178.5714
NV15	Nguyễn Quang Linh	Kinh doanh	850.00	1178.5714
NV11	Trần Thanh Phong	Kinh doanh	1000.00	1178.5714
NV05	Trần Chí Hiếu	Kinh doanh	2500.00	1178.5714
NV06	Vũ Thanh Lê	Kinh doanh	500.00	1178.5714
NV07	Hoàng Mai Hương	Kinh doanh	800.00	1178.5714
NV08	Nguyễn Ngọc Tùng	Kinh doanh	1200.00	1178.5714
NV01	Lê Thanh An	Nhân sự	3000.00	1300.00
NV02	Mai Thị Hạnh	Nhân sự	1000.00	1300.00
NV03	Trần Văn Hữu	Nhân sự	1200.00	1300.00
NV04	Nguyễn Thị Hoa	Nhân sự	800.00	1300.00
NV12	Vũ Thị Hoa	Nhân sự	500.00	1300.00
NV10	Nguyễn Văn Bắc	Nhân sự	800.00	1300.00
NV16	Nguyễn Hoàng Vũ	Nhân sự	1500.00	1300.00
NV18	Vũ Văn Vương	Nhân sự	1600.00	1300.00

Thay vì phải sử dụng truy vấn con như câu lệnh ở trên, ta có thể sử dụng hàm cửa sổ (window function) để viết lại câu truy vấn trên một cách ngắn gọn như sau:

```
SELECT      MaNV,
            HoTen,
            DonVi,
            TienLuong,
            AVG(TienLuong) OVER(PARTITION BY DonVi) AS LuongTB
FROM    NhanVien
```

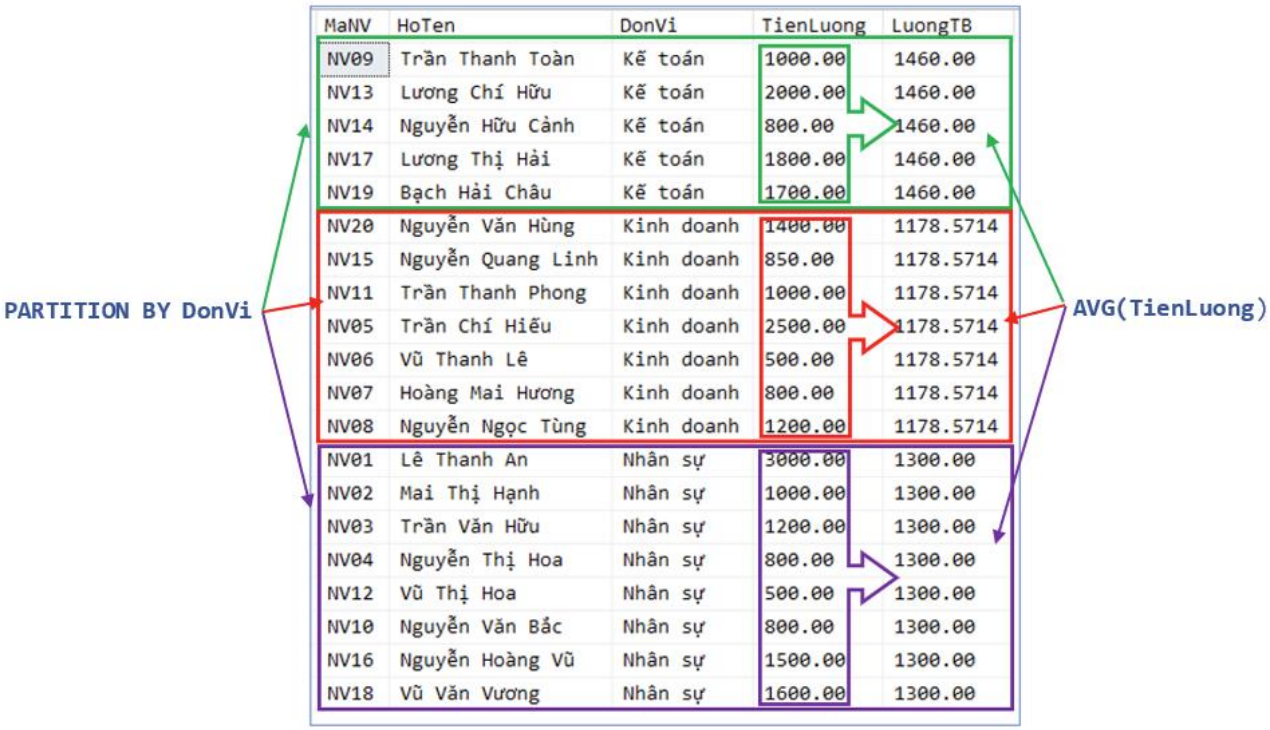
Hàm AVG sử dụng trong câu lệnh trên được gọi là hàm cửa sổ.

Trong SQL, hàm cửa sổ (window function) là một hàm được sử dụng để thực hiện tính toán trên một tập hợp dữ liệu được phân vùng bởi "cửa sổ" (window) trong kết quả của một truy vấn. Các hàm cửa sổ thực hiện việc tính toán trên một tập hợp các dòng và trả về một giá trị tổng hợp cho mỗi dòng. Hiểu một cách đơn giản, "cửa sổ" là một tập con của dữ liệu trong kết quả truy vấn và được xác định bằng cách sử dụng mệnh đề OVER().

Trong câu truy vấn trên, hàm cửa sổ:

```
AVG(TienLuong) OVER(PARTITION BY DonVi)
```

phân chia dữ liệu thành ba phân vùng (ba "cửa sổ") dựa trên cột DonVi và thực hiện việc tính giá trị trung bình cột TienLuong (hàm AVG) trên mỗi phân vùng. Giá trị thống kê được hiển thị tại mỗi dòng, như minh họa ở hình dưới đây:



MaNV	HoTen	DonVi	TienLuong	LuongTB
NV09	Trần Thanh Toàn	Kế toán	1000.00	1460.00
NV13	Lương Chí Hữu	Kế toán	2000.00	1460.00
NV14	Nguyễn Hữu Cảnh	Kế toán	800.00	1460.00
NV17	Lương Thị Hải	Kế toán	1800.00	1460.00
NV19	Bạch Hải Châu	Kế toán	1700.00	1460.00
NV20	Nguyễn Văn Hùng	Kinh doanh	1400.00	1178.5714
NV15	Nguyễn Quang Linh	Kinh doanh	850.00	1178.5714
NV11	Trần Thanh Phong	Kinh doanh	1000.00	1178.5714
NV05	Trần Chí Hiếu	Kinh doanh	2500.00	1178.5714
NV06	Vũ Thanh Lê	Kinh doanh	500.00	1178.5714
NV07	Hoàng Mai Hương	Kinh doanh	800.00	1178.5714
NV08	Nguyễn Ngọc Tùng	Kinh doanh	1200.00	1178.5714
NV01	Lê Thanh An	Nhân sự	3000.00	1300.00
NV02	Mai Thị Hạnh	Nhân sự	1000.00	1300.00
NV03	Trần Văn Hữu	Nhân sự	1200.00	1300.00
NV04	Nguyễn Thị Hoa	Nhân sự	800.00	1300.00
NV12	Vũ Thị Hoa	Nhân sự	500.00	1300.00
NV10	Nguyễn Văn Bắc	Nhân sự	800.00	1300.00
NV16	Nguyễn Hoàng Vũ	Nhân sự	1500.00	1300.00
NV18	Vũ Văn Vương	Nhân sự	1600.00	1300.00

Hàm cửa sổ cho phép thực hiện các tính toán so sánh, tích luỹ, hoặc tính toán trên các hàng trong cửa sổ dựa trên các giá trị trong cửa sổ đó. Chúng thường được sử dụng trong các tình huống như tính các giá trị tổng hợp như: tỷ lệ phần trăm (percentiles), xếp hạng (ranking), tính tổng tích luỹ (cumulative sums), và nhiều tính toán khác mà liên quan đến việc so sánh dữ liệu tổng hợp với các hàng khác trong cửa sổ.

2. Cú pháp của hàm cửa sổ

Cú pháp tổng quát của hàm cửa sổ có dạng như sau:

```
window_function([ALL] expression)
OVER
(
      [PARTITION BY partition_columns]
      [ORDER BY sort_columns [ASC|DESC]]
      [ROWS|RANGE BETWEEN start_position AND end_position]
)
```

Trong đó:

- **`window_function`**

  Là tên của hàm cửa sổ như AVG, SUM, RANK, ROW_NUMBER,...

- **`ALL`**

  Từ khóa ALL trong hàm cửa sổ được sử dụng để xác định rằng hàm cửa sổ sẽ thực hiện tính toán trên tất cả các hàng trong cửa sổ xác định, bất kể giá trị trong cửa sổ có trùng nhau hay không. Nếu không sử dụng từ khóa ALL, hàm cửa sổ mặc định sẽ thực hiện tính toán trên các giá trị duy nhất trong cửa sổ.

- **`expression`**

  Là tên cột hoặc biểu thức được sử dụng để hàm cửa sổ tính toán trên đó.

- **`PARTITION BY partition_columns`**

  Danh sách các cột được sử dụng để phân vùng dữ liệu, mỗi phân vùng được gọi là một "cửa sổ"(window). Hàm cửa sổ sẽ thực hiện việc tính toán giá trị tổng hợp trên riêng từng phân vùng và khởi động lại quá trình tính toán khi qua phân vùng khác.

  Nếu PARTITION BY không được chỉ định thì toàn bộ dữ liệu sẽ được hiểu là nằm trong cùng một phân vùng.

- **`ORDER BY sort_columns`**

  Chỉ định danh sách các cột dùng để xác định thứ tự sắp xếp logic của các dòng khi thực hiện tính toán hàm cửa sổ.

- Nếu mệnh đề ORDER BY không được chỉ định, thứ tự sắp xếp mặc định sẽ là tăng dần (ASC) và hàm cửa sổ sẽ sử dụng tất cả các dòng trong phân vùng để tính toán.
- Trong trường hợp có chỉ định mệnh đề ORDER BY mà không chỉ định phạm vi tính toán (tức là không sử dụng ROWS hoặc RANGE cho hàm cửa sổ) thì phạm vi tính toán mặc định sẽ là UNBOUNDED PRECEDING AND CURRENT ROW (Nếu hàm cửa sổ có cho phép sử dụng tùy chọn ROWS/RANGE trong cú pháp của hàm).

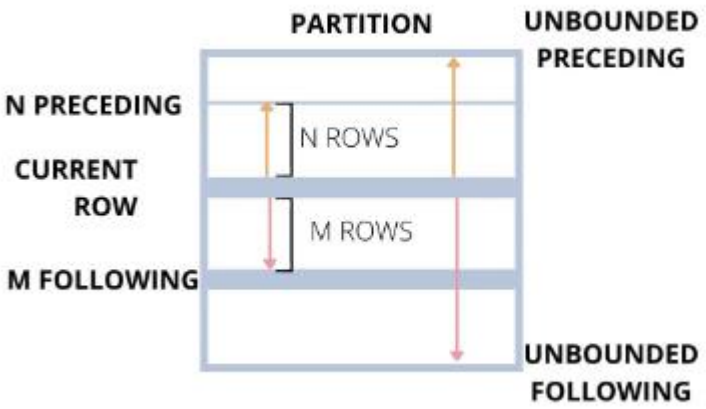
Lưu ý rằng, mệnh đề ORDER BY trong hàm cửa sổ có mục đích sử dụng khác với mệnh đề ORDER BY trong câu lệnh SELECT, vì nó chỉ ảnh hưởng đến cách tính toán của hàm cửa sổ, chứ không ảnh hưởng đến cách sắp xếp khi hiển thị kết quả truy vấn.

- `ROWS|RANGE BETWEEN start_position AND end_position`

Xác định phạm vi tính toán của hàm cửa sổ, tức là giới hạn các dòng được sử dụng để tính toán trong phân vùng bằng cách chỉ định điểm bắt đầu và điểm kết thúc trong phân vùng. Mệnh đề ORDER BY bắt buộc phải được sử dụng trong trường hợp có sử dụng đến ROWS hoặc RANGE trong hàm cửa sổ.

`start_position` và `end_position` có thể sử dụng một trong số các giá trị sau:

Giá trị	Ý nghĩa
UNBOUNDED PRECEDING	Dòng đầu tiên của phân vùng
N PRECEDING	N dòng trước dòng hiện tại
CURRENT ROW	Dòng hiện tại
N FOLLOWING	N dòng sau dòng hiện tại
UNBOUNDED FOLLOWING	Dòng cuối cùng của phân vùng



Một số ví dụ minh họa ở các phần tiếp theo sẽ giúp chúng ta hiểu rõ hơn về cách sử dụng hàm cửa sổ trong SQL.

3. Ví dụ minh họa

Trong phần này, chúng ta sẽ cùng tìm hiểu cụ thể hơn về cách sử dụng hàm cửa sổ thông qua một ví dụ minh họa. Trong ví dụ này, chúng ta sẽ sử dụng dữ liệu của bảng **ThongKeDoanhThu** có cấu trúc và dữ liệu được tạo bởi các lệnh như sau:

```
CREATE TABLE ThongKeDoanhThu
(
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Nam INT,
    MatHang NVARCHAR(255),
    LoaiHang NVARCHAR(255),
    DoanhThu MONEY
)
GO

INSERT INTO ThongKeDoanhThu
VALUES (2010, N'Gạo', N'Thực phẩm', 5),
       (2010, N'Đường', N'Thực phẩm', 7),
       (2010, N'Beer', N'Đồ uống', 10),
       (2010, N'Cafe', N'Đồ uống', 2),
       (2011, N'Gạo', N'Thực phẩm', 8),
       (2011, N'Đường', N'Thực phẩm', 3),
       (2011, N'Beer', N'Đồ uống', 4),
       (2011, N'Cafe', N'Đồ uống', 11),

       (2012, N'Gạo', N'Thực phẩm', 6),
       (2012, N'Đường', N'Thực phẩm', 10),
       (2012, N'Cafe', N'Đồ uống', 15),
       (2013, N'Gạo', N'Thực phẩm', 5),
       (2013, N'Đường', N'Thực phẩm', 12),
       (2013, N'Beer', N'Đồ uống', 14),
       (2013, N'Cafe', N'Đồ uống', 5),
       (2014, N'Gạo', N'Thực phẩm', 5),
```

```
(2014, N'Đường', N'Thực phẩm', 10),
(2014, N'Beer', N'Đồ uống', 5),
(2015, N'Beer', N'Đồ uống', 14),
(2015, N'Cafe', N'Đồ uống', 5);
```

Dữ liệu trong bảng như sau:

Id	Nam	MatHang	LoaiHang	DoanhThu
1	2010	Gạo	Thực phẩm	5
2	2010	Đường	Thực phẩm	7
3	2010	Beer	Đồ uống	10
4	2010	Cafe	Đồ uống	2
5	2011	Gạo	Thực phẩm	8
6	2011	Đường	Thực phẩm	3
7	2011	Beer	Đồ uống	4
8	2011	Cafe	Đồ uống	11
9	2012	Gạo	Thực phẩm	6
10	2012	Đường	Thực phẩm	10
11	2012	Cafe	Đồ uống	15
12	2013	Gạo	Thực phẩm	5
13	2013	Đường	Thực phẩm	12
14	2013	Beer	Đồ uống	14
15	2013	Cafe	Đồ uống	5
16	2014	Gạo	Thực phẩm	5
17	2014	Đường	Thực phẩm	10
18	2014	Beer	Đồ uống	5
19	2015	Beer	Đồ uống	14
20	2015	Cafe	Đồ uống	5

Dữ liệu trong bảng trên cho biết số liệu tổng hợp doanh thu bán hàng của các mặt hàng theo từng năm. Dữ liệu như trên có thể tạo ra các phân vùng như sau:

- Năm thống kê bao gồm các năm từ 2010 đến 2015.
- Có 4 mặt hàng là Gạo, Đường, Beer và Cafe.
- Các mặt hàng được phân thành 2 loại là Thực phẩm và Đồ uống.

Câu lệnh sau đây:

```
SELECT MatHang, Nam, DoanhThu,
       SUM(DoanhThu) OVER() AS TongTatCa,
       SUM(DoanhThu) OVER
       (
            PARTITION BY MatHang
       ) AS TongTheoMatHang,
       SUM(DoanhThu) OVER
       (
```

```
            PARTITION BY MatHang
            ORDER BY Nam
            ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
      ) AS TongLuyKe,
      SUM(DoanhThu) OVER
      (
            PARTITION BY MatHang
            ORDER BY Nam
            ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
      ) AS TongLanCan
FROM ThongKeDoanhThu
```

Có kết quả như sau:

	MatHang	Nam	DoanhThu	TongTatCa	TongTheoMatHang	TongLuyKe	TongLanCan
1	Beer	2010	10.00	156.00	47.00	10.00	14.00
2	Beer	2011	4.00	156.00	47.00	14.00	28.00
3	Beer	2013	14.00	156.00	47.00	28.00	23.00
4	Beer	2014	5.00	156.00	47.00	33.00	33.00
5	Beer	2015	14.00	156.00	47.00	47.00	19.00
6	Cafe	2010	2.00	156.00	38.00	2.00	13.00
7	Cafe	2011	11.00	156.00	38.00	13.00	28.00
8	Cafe	2012	15.00	156.00	38.00	28.00	31.00
9	Cafe	2013	5.00	156.00	38.00	33.00	25.00
10	Cafe	2015	5.00	156.00	38.00	38.00	10.00
11	Đường	2010	7.00	156.00	42.00	7.00	10.00
12	Đường	2011	3.00	156.00	42.00	10.00	20.00
13	Đường	2012	10.00	156.00	42.00	20.00	25.00
14	Đường	2013	12.00	156.00	42.00	32.00	32.00
15	Đường	2014	10.00	156.00	42.00	42.00	22.00
16	Gạo	2010	5.00	156.00	29.00	5.00	13.00
17	Gạo	2011	8.00	156.00	29.00	13.00	19.00
18	Gạo	2012	6.00	156.00	29.00	19.00	19.00
19	Gạo	2013	5.00	156.00	29.00	24.00	16.00
20	Gạo	2014	5.00	156.00	29.00	29.00	10.00

Trong kết quả của truy vấn như ở trên có 4 cột là kết quả tính toán của các hàm cửa sổ (hàm SUM). Chúng ta có thể hình dung cách tính của các hàm này như sau:

- Cột **TongTatCa** là kết quả của hàm:

  ```
  SUM(DoanhThu) OVER()
  ```

  Hàm này không chỉ định mệnh để PARTITION BY nên tất cả dữ liệu trong bảng đều nằm trong cùng một phân vùng. Hàm cửa sổ sẽ thực hiện tính toán tổng của

cột **DoanhThu** trên tất cả các dòng dữ liệu hiện có trong bảng. Như chúng ta có thể thấy, giá trị của hàm này giống nhau ở tất cả các dòng.

- Cột **TongTheoMatHang** là kết quả của hàm:

```
SUM(DoanhThu) OVER
(
      PARTITION BY MatHang
 )
```

Trong hàm này dữ liệu được phân thành các phân vùng dựa trên MatHang (có tất cả 4 phân vùng). Hàm cửa sổ thực hiện tính tổng các dòng dữ liệu trong mỗi phân vùng. Chính vì vậy, chúng ta thấy rằng các dòng dữ liệu nằm trong cùng một phân vùng có cùng giá trị như nhau.

	MatHang	Nam	DoanhThu	TongTheoMatHang	
1	Beer	2010	10.00	47.00	
2	Beer	2011	4.00	47.00	
3	Beer	2013	14.00	47.00	= 10 + 4 + 14 + 5 + 14
4	Beer	2014	5.00	47.00	
5	Beer	2015	14.00	47.00	
6	Cafe	2010	2.00	38.00	
7	Cafe	2011	11.00	38.00	
8	Cafe	2012	15.00	38.00	= 2 + 11 + 15 + 5 + 5
9	Cafe	2013	5.00	38.00	
10	Cafe	2015	5.00	38.00	
11	Đường	2010	7.00	42.00	
12	Đường	2011	3.00	42.00	
13	Đường	2012	10.00	42.00	= 7 + 3 + 10 + 12 + 10
14	Đường	2013	12.00	42.00	
15	Đường	2014	10.00	42.00	
16	Gạo	2010	5.00	29.00	
17	Gạo	2011	8.00	29.00	
18	Gạo	2012	6.00	29.00	= 5 + 8 + 6 + 5 + 5
19	Gạo	2013	5.00	29.00	
20	Gạo	2014	5.00	29.00	

Chúng ta cũng sẽ có kết quả tương tự nếu sử dụng cách viết hàm cửa sổ như sau:

```
SUM(DoanhThu) OVER
(
      PARTITION BY MatHang
      ORDER BY Nam
```

```
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    )
```

- Cột **TongLuyKe** được tính bởi hàm cửa sổ:

```
SUM(DoanhThu) OVER
    (
        PARTITION BY MatHang
        ORDER BY Nam
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS TongLuyKe
```

Hàm này cũng sử dụng cột MatHang để phân chia dữ liệu thành các phân vùng. Trong hàm này, mệnh đề:

```
ORDER BY Nam
```

chỉ định thứ tự logic để tính toán trong mỗi phân vùng là theo thứ tự tăng dần của Nam, còn mệnh đề:

```
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

xác định cách tính tổng tại mỗi dòng là từ dòng đầu tiên của phân vùng (UNBOUNDED PRECEDING) cho đến dòng hiện tại (CURRENT ROW).

Có thể hình dung cách tính toán của hàm cửa sổ này như hình minh họa bên dưới:

	MatHang	Nam	DoanhThu	TongLuyKe	
1	Beer	2010	10.00	10.00	= 10
2	Beer	2011	4.00	14.00	= 10 + 4
3	Beer	2013	14.00	28.00	= 10 + 4 + 14
4	Beer	2014	5.00	33.00	= 10 + 4 + 14 + 5
5	Beer	2015	14.00	47.00	= 10 + 4 + 14 + 5 + 14
6	Cafe	2010	2.00	2.00	= 2
7	Cafe	2011	11.00	13.00	= 2 + 11
8	Cafe	2012	15.00	28.00	= 2 + 11 + 15
9	Cafe	2013	5.00	33.00	= 2 + 11 + 15 + 5
10	Cafe	2015	5.00	38.00	= 2 + 11 + 15 + 5 + 5

- Kết quả ở cột **TongLanCan** được tính bởi hàm:

```
SUM(DoanhThu) OVER
(
    PARTITION BY MatHang
    ORDER BY Nam
    ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
)
```

Hàm này cũng sử dụng cách phân vùng dữ liệu và chỉ định thứ tự logic tính toán như ở trường hợp trên. Điểm khác biệt là phạm vi tính toán được xác định bởi mệnh đề:

```
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
```

Tức là, tại mỗi dòng, giá trị tổng sẽ được tính trên phạm vi từ dòng trước dòng hiện tại 1 dòng (1 PRECEDING) cho đến dòng phía sau dòng hiện tại 1 dòng (1 FOLLOWING). Lưu ý rằng, dòng đầu tiên (và dòng cuối cùng) trong phân vùng không có dòng trước (và dòng sau) nên tổng sẽ được tính dựa trên 2 giá trị thay vì 3 giá trị như ở các dòng còn lại.

Có thể hình dung cách tính toán của hàm này như hình minh họa sau:

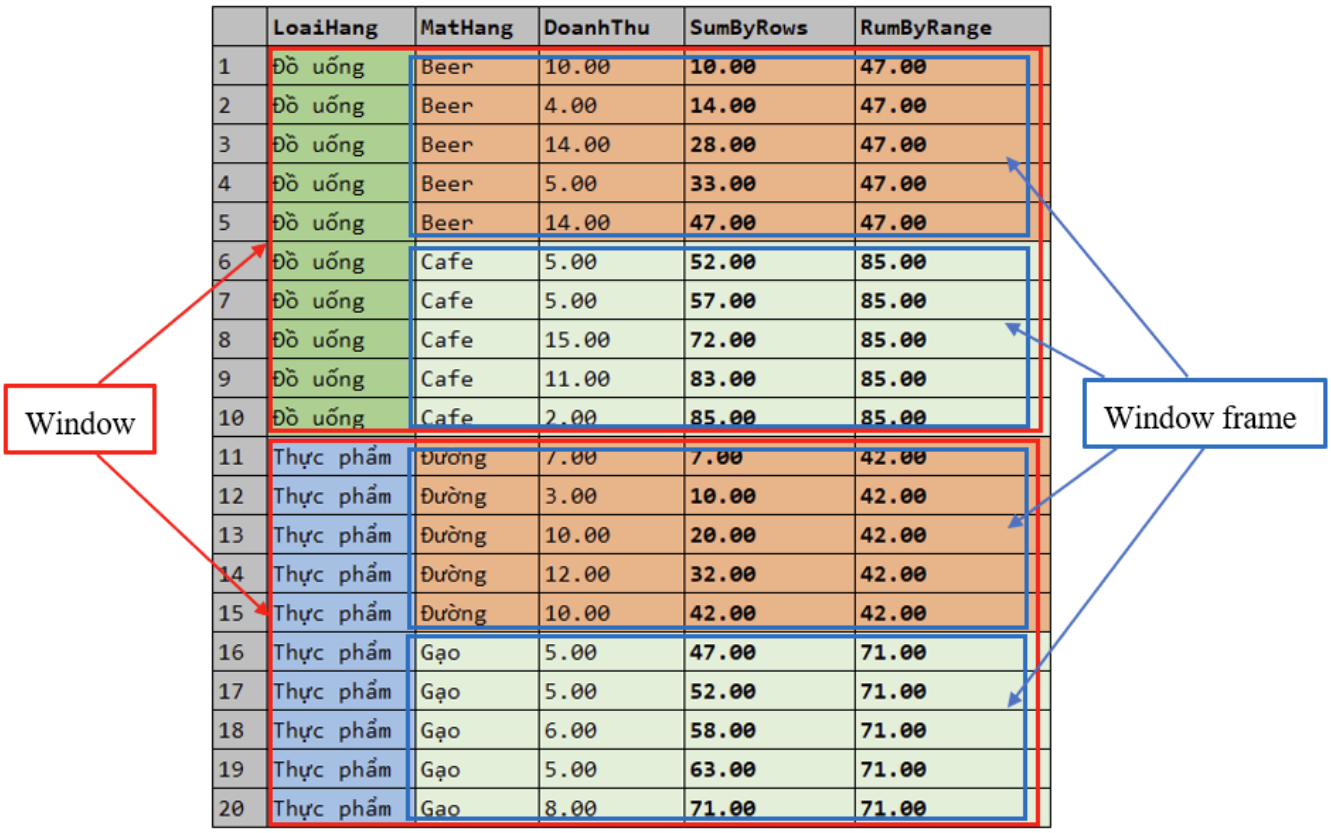
	MatHang	Nam	DoanhThu	TongLanCan	
1	Beer	2010	10.00	14.00	= 10 + 4
2	Beer	2011	4.00	28.00	= 10 + 4 + 14
3	Beer	2013	14.00	23.00	= 4 + 14 + 5
4	Beer	2014	5.00	33.00	= 14 + 5 + 14
5	Beer	2015	14.00	19.00	= 5 + 14

4. Sự khác biệt giữa ROWS và RANGE

Trong ví dụ ở phần 3, chúng ta đã sử dụng ROWS để giới hạn phạm vi tính toán của hàm cửa sổ. Để giới hạn phạm vi tính toán của hàm cửa sổ, chúng ta có thể sử dụng ROWS hoặc RANGE. Vậy sự khác biệt giữa ROWS và RANGE là gì? Chúng ta cùng phân biệt chúng thông qua ví dụ ở câu lệnh dưới đây:

```
SELECT LoaiHang, MatHang, DoanhThu,
        SUM(DoanhThu) OVER
        (
            PARTITION BY LoaiHang
            ORDER BY MatHang
            ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
        ) AS SumByRows,
        SUM(DoanhThu) over
        (
             PARTITION BY LoaiHang
             ORDER BY MatHang
             RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
       ) AS RumByRange
FROM   ThongKeDoanhThu
```

Kết quả của truy vấn trên như sau:



	LoaiHang	MatHang	DoanhThu	SumByRows	RumByRange
1	Đồ uống	Beer	10.00	10.00	47.00
2	Đồ uống	Beer	4.00	14.00	47.00
3	Đồ uống	Beer	14.00	28.00	47.00
4	Đồ uống	Beer	5.00	33.00	47.00
5	Đồ uống	Beer	14.00	47.00	47.00
6	Đồ uống	Cafe	5.00	52.00	85.00
7	Đồ uống	Cafe	5.00	57.00	85.00
8	Đồ uống	Cafe	15.00	72.00	85.00
9	Đồ uống	Cafe	11.00	83.00	85.00
10	Đồ uống	Cafe	2.00	85.00	85.00
11	Thực phẩm	Đường	7.00	7.00	42.00
12	Thực phẩm	Đường	3.00	10.00	42.00
13	Thực phẩm	Đường	10.00	20.00	42.00
14	Thực phẩm	Đường	12.00	32.00	42.00
15	Thực phẩm	Đường	10.00	42.00	42.00
16	Thực phẩm	Gạo	5.00	47.00	71.00
17	Thực phẩm	Gạo	5.00	52.00	71.00
18	Thực phẩm	Gạo	6.00	58.00	71.00
19	Thực phẩm	Gạo	5.00	63.00	71.00
20	Thực phẩm	Gạo	8.00	71.00	71.00

Trong truy vấn trên, cả 2 hàm cửa sổ đều sử dụng mệnh đề:

```
PARTITION BY LoaiHang
```

để phân vùng dữ liệu dựa trên cột **LoaiHang**. Chúng ta có thể thấy như hình trên, dữ liệu được chia thành hai phân vùng (cửa sổ/window). Trong mỗi phân vùng, dữ liệu lại tiếp tục được phân thành các phân vùng con (khung cửa sổ/window frame) dựa vào **MatHang** thông qua mệnh đề:

```
ORDER BY MatHang
```

Hiểu một cách đơn giản, nếu mệnh đề PARTITION BY chia dữ liệu thành các phân vùng gọi là cửa sổ (window), thì mệnh đề ORDER BY sẽ tạo ra các phân vùng con của mỗi phân vùng gọi là khung cửa sổ (window frame).

Cả hai hàm cửa sổ trong câu lệnh trên đều thực hiện việc tính tổng của cột doanh thu với phạm vi được chỉ định là:

```
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

Tuy nhiên, chúng ta có thể thấy sự khác biệt về kết quả tính toán của hai hàm này tùy thuộc vào tùy chọn phạm vi áp dụng là ROWS hay RANGE như sau:

- **Đối với tùy chọn phạm vi là ROWS:** Phạm vi tính toán được xác định chính xác theo dòng, từ dòng đầu tiên của phân vùng (cửa sổ/window) cho đến dòng hiện tại.

- **Đối với tùy chọn phạm vi là RANGE:** Phạm vi tính toán sẽ được xác định là theo từng phân vùng con (khung cửa sổ/window frame); Và như vậy, kết quả tính toán sẽ là tổng của từ phân vùng con đầu tiên cho đến phân vùng con tương ứng với dòng hiện tại.

5. Các hàm cửa sổ thông dụng

5.1. Các hàm tổng hợp (aggregate functions)

Hàm	Mô tả
SUM(expression)	Tính tổng các giá trị
COUNT(expression)	Đếm số lượng giá trị khác NULL
MIN(expression)	Lấy giá trị nhỏ nhất
MAX(expression)	Lấy giá trị lớn nhất
AVG(expression)	Tính giá trị trung bình

5.2. Các hàm xếp hạng (ranking functions)

Lưu ý: Khi sử dụng các hàm xếp hạng, mệnh đề ORDER BY bắt buộc phải được chỉ định

Hàm	Mô tả
ROW_NUMBER()	Đánh số thứ tự cho các dòng dữ liệu trong mỗi phân vùng
RANK()	Xếp vị thứ cho mỗi dòng trong phân vùng dựa vào giá trị của các cột được chỉ định trong mệnh đề ORDER BY. Các dòng có giá trị bằng nhau sẽ xếp cùng vị thứ. Vị thứ tiếp theo sẽ bỏ qua nếu như đã có các dòng có cùng vị thứ.
DENSE_RANK()	Xếp vị thứ cho mỗi dòng trong phân vùng dựa vào giá trị của các cột được chỉ định trong mệnh đề ORDER BY. Các dòng có giá trị bằng nhau sẽ xếp cùng vị thứ. Vị thứ tiếp theo sẽ không bị bỏ qua cho dù đã có các dòng có cùng vị thứ.

NTILE(N)	Phân chia các dòng hiện tại trong phân vùng thành N nhóm. Mỗi nhóm sẽ được đánh số thứ tự lần lượt từ 1 đến N.

5.3. Các hàm tính giá trị (value function)

Lưu ý: Khi sử dụng các hàm tính giá trị, mệnh đề ORDER BY bắt buộc phải được chỉ định

Hàm	**Mô tả**
FIRST_VALUE(expression)	Lấy giá trị đầu tiên trong tập các giá trị được sắp xếp
LAST_VALUE(expression)	Lấy giá trị cuối cùng trong tập các giá trị được sắp xếp
LEAD(expression, N)	Lấy giá trị của dòng thứ N sau dòng hiện tại (nếu không có thì trả về NULL)
LAG(expression, N)	Lấy giá trị của dòng thứ N trước dòng hiện tại (nếu không có thì trả về NULL).