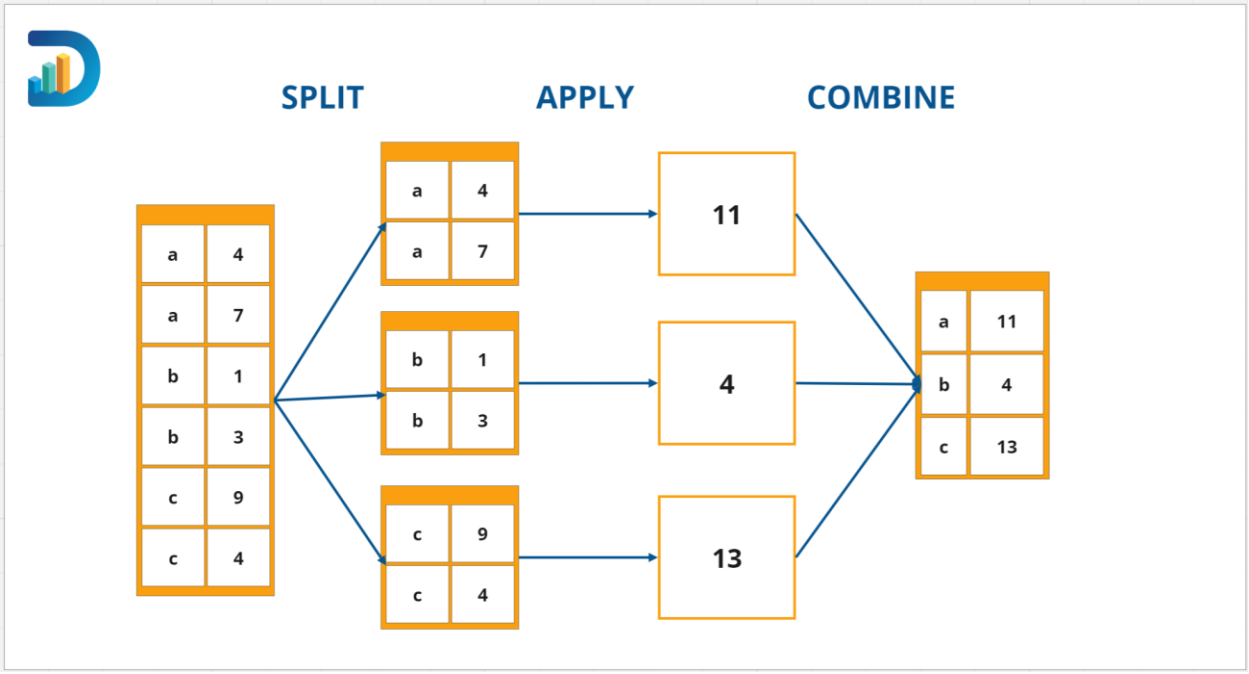
Chapter 5: GROUP BY & HAVING.

5.1. Nhóm các hàng bằng GROUP BY

GROUP BY được sử dụng để nhóm các dòng dữ liệu dựa trên giá trị của một hoặc nhiều cột. Mục đích chính của GROUP BY là thực hiện các phép tổng hợp hoặc hàm tính toán trên các nhóm dữ liệu.

Sau đây là cách GROUP BY được thực thi trong câu lệnh.



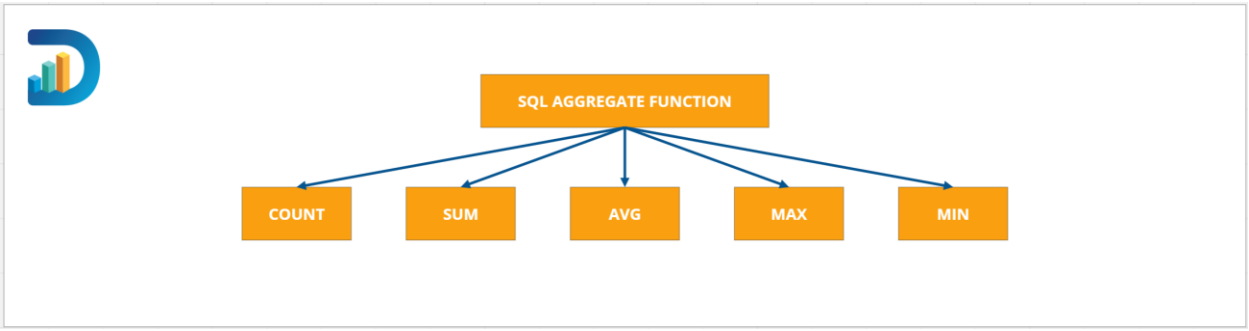
Cú pháp của GROUP BY:

```
SELECT
    Col_1,
    Col_2,
    Aggregate_function(Col_3)
FROM Table
GROUP BY
    Col_1,
    Col_2
```

- Col_1, Col_2: Cột truy vấn, cột được gộp từ mệnh đề GROUP BY.
- Aggregate_function(Col_3): Hàm tính toán được áp dụng với Col_3.

- Col_3: Cột được áp dụng hàm tính toán
- Table: Bảng dữ liệu truy vấn.

Các hàm tính toán phổ biến



Ví dụ: Từ bảng Product thuộc bộ dữ liệu adventureworks, truy vấn các cột sau: ProductCategoryID, tổng giá tiền của sản phẩm, giá tiền trung bình, đếm số ProductID, số ProductID riêng biệt, giá tiền tối đa, giá tiền tối thiểu của giá tiền từng sản phẩm. Nhóm các trường có chung ProductIDCategoryID thành các nhóm riêng biệt.

```
167  USE adventureworks
168  SELECT
169      ProductCategoryID,
170      SUM(ListPrice) AS sum_listprice,
171      AVG(ListPrice) AS avg_listprice,
172      COUNT(ProductID) AS count_productid,
173      COUNT(DISTINCT ProductModelID) AS count_distinct_productmodelid,
174      MAX(ListPrice) AS max_listprice,
175      MIN(ListPrice) AS min_listprice
176  FROM SalesLT.Product
177  GROUP BY ProductCategoryID
```

Results Messages

	ProductCategoryID	sum_listprice	avg_listprice	count_productid	count_distinct_productmodelid	max_listprice	min_listprice
1	5	53867.68	1683.365	32	5	3399.99	539.99
2	6	68690.35	1597.45	43	7	3578.27	539.99
3	7	31355.46	1425.2481	22	3	2384.07	742.35
4	8	591.12	73.89	8	8	120.27	44.54
5	9	276.72	92.24	3	3	121.49	53.99

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu được truy vấn từ bảng SalesLT.Product.
- GROUP BY: Nhóm các đơn hàng có chung ProductCategoryID vào với nhau.
- SELECT: Truy vấn các cột ProductCategoryID, tổng giá tiền của sản phẩm, giá tiền trung bình, đếm số ProductID, số ProductID riêng biệt, giá tiền tối đa, giá tiền tối thiểu của từng sản phẩm.
    - Hàm SUM: Tính toán tổng giá tiền của từng nhóm sản phẩm.
    - Hàm AVG: Tính toán giá tiền trị trung bình của từng sản phẩm.
    - Hàm COUNT: Đếm số ProductID của từng nhóm sản phẩm.

- Hàm COUNT(DISTINCT): Đếm số ProductID riêng biệt của từng nhóm sản phẩm.
- Hàm MAX: Tính toán giá tiền lớn nhất của từng nhóm sản phẩm.
- Hàm MIN: Tính toán giá tiền nhỏ nhất của từng nhóm sản phẩm.

5.2. Chỉ định điều kiện các nhóm bằng mệnh đề HAVING

HAVING là một điều kiện được sử dụng trong SQL sau mệnh đề GROUP BY. Nó được sử dụng để lọc các nhóm dữ liệu dựa trên hàm tính toán (aggregate function). Mục đích chính của HAVING là lọc các nhóm dữ liệu sau khi đã thực hiện tính toán.

Cú pháp của HAVING:

```
SELECT
    Col_1,
    Col_2,
    Aggregate_function(Col_3)
FROM Table
GROUP BY
    Col_1,
    Col_2
HAVING Aggregate_condition
```

- Col_1, Col_2: Cột truy vấn, cột được gộp từ mệnh đề GROUP BY.
- Aggregate_function(Col_3): Hàm tính toán được áp dụng với Col_3.
- Col_3: Cột được áp dụng hàm tính toán
- Table: Bảng dữ liệu truy vấn.
- Aggregate_condition: Lọc nhóm dữ liệu đã được tính toán thoả mãn điều kiện đề ra.

Ví dụ: Từ bảng Product thuộc bộ dữ liệu adventureworks, truy vấn các cột sau: ProductCategoryID, cân nặng trung bình của sản phẩm được lưu dưới tên avg_weight, giá trung bình của sản phẩm được lưu dưới tên avg_list_price. Chỉ hiển thị ProductCategoryID thoả mãn cân nặng trung bình lớn hơn 1000.

```sql
USE adventureworks
SELECT
    ProductCategoryID,
    AVG(Weight) AS avg_weight,
    AVG(ListPrice) AS avg_list_price
FROM SalesLT.Product
GROUP BY ProductCategoryID
HAVING AVG(Weight) > 1000
```

Results Messages

	ProductCategoryID	avg_weight	avg_list_price
1	5	11366.399375	1683.365
2	6	7914.302325	1597.45
3	7	12555.784090	1425.2481
4	16	1288.195000	678.2535
5	18	1046.555757	780.0436
6	20	1396.048888	631.4155

Ln 9, Col 1 Spaces: 4 UTF-8 CRLF SQL 6 rows M

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu được truy vấn từ bảng SalesLT.Product.
- GROUP BY: Nhóm các đơn hàng có chung ProductCategoryID vào với nhau.
- HAVING: Lọc nhóm dữ liệu đã được tính toán đã được tính toán thoả mãn điều kiện cân nặng trung bình của sản phẩm lớn hơn 1000.
- SELECT: Truy vấn các cột ProductCategoryID, avg_weight, avg_list_price.

5.3. WHERE và HAVING khác nhau ở đâu?

WHERE sử dụng để lọc các hàng của dữ liệu gốc trong khi HAVING sử dụng để lọc các hàng của kết quả sau GROUP BY.

Các hàm tính toán (Aggregate Functions) ví dụ: SUM, AVG, MIN, MAX, … chỉ có thể được sử dụng trong câu lệnh HAVING.

Ví dụ: Từ bảng thuộc SalesOrderDetail, truy vấn các cột sau SalesOrderID, tổng số đơn hàng, tổng doanh thu và doanh thu trung bình. Lọc điều kiện thoả mãn UnitPriceDiscount = 0 và tổng số đơn hàng lớn hơn bằng 100.

Không sử dụng được hàm tính toán (Aggregate Function) trong câu lệnh WHERE nên khi thực hiện truy vấn câu lệnh sẽ báo lỗi.

```
61  USE adventureworks
62  SELECT
63      SalesOrderID,
64      SUM(OrderQty) AS num_items,
65      SUM(LineTotal) AS total_revenue,
66      SUM(LineTotal) / SUM(OrderQty) AS avg_revenue
67  FROM SalesLT.SalesOrderDetail
68  WHERE
69      UnitPriceDiscount = 0 AND
70      SUM(OrderQty) >= 100
```

Messages

```
11:44:26 AM    Started executing query at Line 61
               Msg 147, Level 15, State 1, Line 2
               An aggregate may not appear in the WHERE clause un
               less it is in a subquery contained in a HAVING cla
               use or a select list, and the column being aggrega
               ted is an outer reference.
               Total execution time: 00:00:00.026
```

Vì vậy ta phải sử dụng từ khóa (Keyword) HAVING để thực hiện truy vấn.

```
61  USE adventureworks
62  SELECT
63      SalesOrderID,
64      SUM(OrderQty) AS num_items,
65      SUM(LineTotal) AS total_revenue,
66      SUM(LineTotal) / SUM(OrderQty) AS avg_revenue
67  FROM SalesLT.SalesOrderDetail
68  WHERE UnitPriceDiscount = 0
69  GROUP BY SalesOrderID
70  HAVING SUM(OrderQty) >= 100
71  ORDER BY total_revenue
```

Results Messages

	SalesOrderID	num_items	total_revenue	avg_revenue
1	71783	168	32149.776000	191.367714
2	71782	140	33319.986000	237.999900
3	71902	159	58945.026000	370.723433
4	71797	177	63145.374000	356.753525
5	71784	198	67933.572000	343.098848
6	71938	167	74160.228000	444.073221

0 Choose SQL Language 00:00:00 sql-st.datapot.edu.vn : adventureworks

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu được lấy từ bảng SalesLT.SalesOrderDetail.
- WHERE: Lọc các dòng dữ liệu thoả mãn đơn vị chiết khấu bằng “0”.
- GROUP BY: Nhóm các hàng của cột SalesOrderID có cùng giá trị trong một cột thành các nhóm riêng biệt.
- HAVING: Lọc nhóm dữ liệu đã được tính toán thoả mãn điều kiện tổng số lượng sản phẩm lớn hơn hoặc bằng 100.
- SELECT: Truy vấn các cột SalesOrderID, tổng số lượng sản phẩm, tổng doanh thu, doanh thu trung bình.
- ORDER BY: Dữ liệu được sắp xếp theo tổng doanh thu giảm dần.

5.4. Hàm cửa sổ (Window Functions)

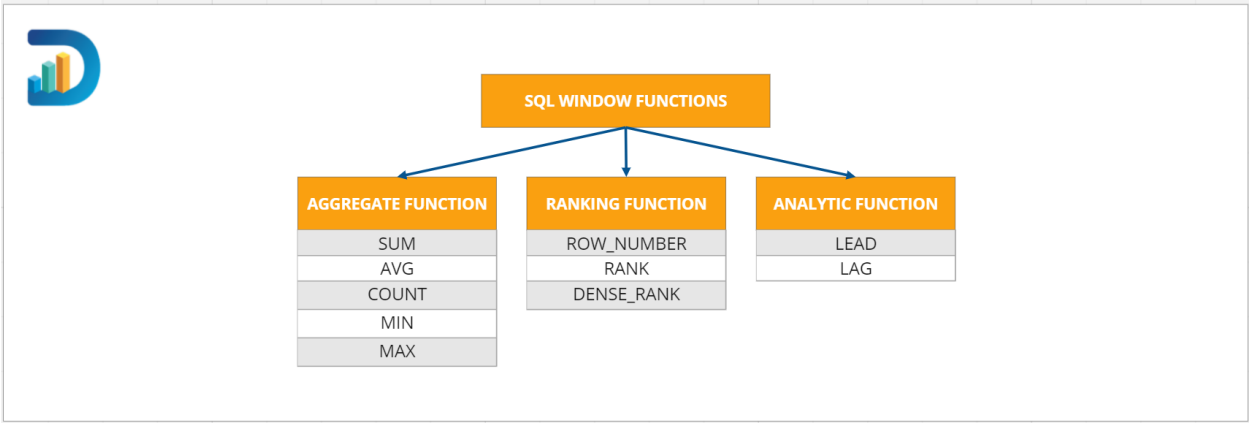
Hàm cửa sổ (Window Functions) trong SQL được sử dụng để thực hiện các phép tính toán các dòng có liên quan đến dòng hiện tại.

Lệnh truy vấn:

```
SELECT
    Col_1,
    {Window_function} (Col_2)
    OVER([PARTITION BY Col_1 ] [ORDER BY Col_3])
FROM Table_name
```

- Col_1: Tên cột đầu tiên muốn chọn.
- Hàm cửa sổ:
  - {Window Functions}: Tên của hàm tổng hợp như SUM, AVG,…
  - Col_2: Tên của cột mà áp dụng Window Functions.
  - OVER: Xác định khung cửa sổ, bao gồm PARTITION BY(Nếu có), ORDER BY(Sắp xếp dữ liệu trong cửa sổ)
  - PARTITION BY(Nếu có): Nhóm các hàng liên quan đến nhau để thực hiện tính toán.
  - ORDER BY: Sắp xếp các hàng có trong từng cửa sổ.
  - Col_3: Cột để sắp xếp trong từng cửa sổ.
- New_col: Tên bạn muốn đặt cho dữ liệu mới.
- Table_name: Tên bảng dữ liệu.

Các hàm cửa sổ phổ biến:



Trong phạm vi Textbook của khoá DP080 sẽ tập trung giới thiệu về Ranking Function, hai nhóm hàm còn lại là Aggregate Function và Analytic Function được đưa ra để tham khảo thêm.

5.4.1. Hàm xếp hạng (Ranking Functions)

Một số hàm xếp hạng phổ biến thường được thấy như:

- ROW_NUMBER

Dùng để xếp hạng các dòng dữ liệu, nhưng không quan tâm đến giá trị giống nhau.

Ví dụ: Từ bảng Product thuộc bộ dữ liệu adventurewoks, truy vấn tên và đánh số thứ tự theo StandardCost.

```
90  USE adventureworks
91  SELECT
92      Name,
93      StandardCost,
94      ROW_NUMBER() OVER(ORDER BY StandardCost) AS ProductRank
95  FROM SalesLT.Product
```

Results Messages

	Name	StandardCost	ProductRank
1	Patch Kit/8 Patches	0.8565	1
2	Road Tire Tube	1.4923	2
3	Touring Tire Tube	1.8663	3
4	Mountain Tire Tube	1.8663	4
5	Water Bottle - 30 oz.	1.8663	5
6	Bike Wash - Dissolver	2.9733	6
7	Racing Socks, M	3.3623	7

⚠ 0 00:00:00 sql-st.datapot.edu.vn : adventureworks

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu truy vấn từ bảng Product.
- SELECT: Truy vấn các cột Name, StandardCost và đánh số thứ tự tăng dần bằng hàm ROW_NUMBER dựa trên thứ tự của StandardCost.

Ví dụ: Từ bảng Product thuộc bộ dữ liệu adventurewoks, truy vấn tên, gom nhóm theo màu sắc và đánh số thứ tự, sắp xếp thứ tự theo StandardCost.

```
210  USE adventureworks
211  SELECT
212      Name,
213      Color,
214      StandardCost,
215      ROW_NUMBER()OVER(PARTITION BY Color ORDER BY StandardCost) AS ProductRank
216  FROM SalesLT.Product
```

Results Messages

	Name	Color	StandardCost	ProductRank
1	Patch Kit/8 Patches	*NULL*	0.8565	1
2	Road Tire Tube	*NULL*	1.4923	2
3	Touring Tire Tube	*NULL*	1.8663	3
4	Mountain Tire Tube	*NULL*	1.8663	4
5	Water Bottle - 30 oz.	*NULL*	1.8663	5

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu truy vấn từ bảng Product.
- SELECT: Truy vấn các cột Name, StandardCost và đánh số thứ tự tăng dần bằng hàm ROW_NUMBER dựa trên thứ tự của StandardCost.
  - Hàm cửa sổ với hàm ROW_NUMBER: Gom nhóm theo màu sắc sau đó đánh số thứ tự theo cột StandardCost, sắp xếp theo cột StandardCost.
  - Partition By sẽ tạo ra các tập dữ liệu nhỏ. Sau đó đánh số thứ tự trong từng trường.

- Hàm RANK

Thường được sử dụng để xếp hạng các dòng dữ liệu dựa trên một hoặc nhiều cột.

Cú pháp của hàm RANK():

SELECT

Col_1,

Col_2,

RANK() OVER (PARTITION BY Col_1 ORDER BY Col_2) AS RANK

FROM Table_name

- Col_1, Col_2: Cột thực hiện truy vấn.

- RANK(): Xếp hạng các hàng dữ liệu.
- OVER(PARTITION BY Col_1): Xác định phạm vi của cửa sổ dữ liệu mà hàm sẽ được áp dụng. Ở đây là đảm bảo hàm RANK() chỉ được áp dụng cho các hàng có cùng giá trị của Col_1.

  (PARTITION BY) là không bắt buộc.

- Table_name: Bảng dữ liệu được sử dụng.

Ví dụ: Từ bảng dữ liệu Product thuộc bộ dữ liệu Product. Truy vấn Name, StandardCost và xếp hạng dựa trên StandardCost.

```
90    USE adventureworks
91    SELECT
92        Name,
93        StandardCost,
94        RANK() OVER(ORDER BY StandardCost) AS ProductRank
95    FROM SalesLT.Product
```

Results Messages

	Name	StandardCost	ProductRank
1	Patch Kit/8 Patches	0.8565	1
2	Road Tire Tube	1.4923	2
3	Touring Tire Tube	1.8663	3
4	Mountain Tire Tube	1.8663	3
5	Water Bottle - 30 oz.	1.8663	3
6	Bike Wash - Dissolver	2.9733	6
7	Racing Socks, M	3.3623	7

0 00:00:00 sql-st.datapot.edu.vn : adventureworks

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu truy vấn từ bảng Product.
- SELECT: Truy vấn các cột Name, StandardCost và xếp hạng bằng hàm RANK dựa trên StandardCost.

  Đối với các sản phẩm Touring Tire Tube, Mountain Tire Tube, Water Bottle cùng chung 1 xếp hạng là 3 do có chung cùng một mức

StandardCost là 1.8863. Khác với Row_Number sẽ đánh lần thứ tự bất kể StandardCost có chung giá trị.

- DENSE_RANK

Hàm DENSE_RANK xếp hạng các dòng dữ liệu, nhưng không bỏ qua thứ hạng nếu có các giá trị giống nhau.

Ví dụ: Nếu có 2 giá trị lớn nhất, cả hai được xếp hạng là 1 và giá trị tiếp theo xếp hạng thứ 2.

Cú pháp của hàm DENSE_RANK:

```
SELECT
    Col_1,
    Col_2,
DENSE_RANK() OVER (PARTITION BY Col_1 ORDER BY Col_2) AS RANK
FROM Table_name
```

- Col_1, Col_2: Cột thực hiện truy vấn.
- DENSE_RANK(): Xếp hạng các hàng dữ liệu và không bỏ qua thứ hạng nếu có các giá trị giống nhau.
- OVER(PARTITION BY Col_1): Xác định phạm vi của cửa sổ dữ liệu mà hàm sẽ được áp dụng. Ở đây là đảm bảo hàm DENSE_RANK() chỉ được áp dụng cho các hàng có cùng giá trị của Col_1.

  (PARTITION BY) là không bắt buộc.
- Table_name: Bảng dữ liệu được sử dụng.

Ví dụ: Từ bảng dữ liệu Product thuộc bộ dữ liệu Product. Truy vấn Name, StandardCost và xếp hạng dựa trên StandardCost.

```
90  USE adventureworks
91  SELECT
92      Name,
93      StandardCost,
94      DENSE_RANK() OVER(ORDER BY StandardCost) AS ProductRank
95  FROM SalesLT.Product
```

Results Messages

	Name	StandardCost	ProductRank
1	Patch Kit/8 Patches	0.8565	1
2	Road Tire Tube	1.4923	2
3	Touring Tire Tube	1.8663	3
4	Mountain Tire Tube	1.8663	3
5	Water Bottle - 30 oz.	1.8663	3
6	Bike Wash - Dissolver	2.9733	4
7	Racing Socks, M	3.3623	5

⚠ 0 00:00:00 sql-st.datapot.edu.vn : adventureworks

Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu truy vấn từ bảng Product.
- SELECT: Truy vấn các cột Name, StandardCost và xếp hạng bằng hàm DENSE_RANK dựa trên StandardCost.
  Đối với các sản phẩm Touring Tire Tube, Mountain Tire Tube, Water Bottle cùng chung 1 xếp hạng là 3 do có chung cùng một mức StandardCost là 1.8863 và xếp hạng được đánh tiếp theo sẽ là 4 thay vì bị bỏ qua như ROW_NUMBER.

Tương tự như ROW_NUMBER thì DENSE_RANK cũng có thể sử dụng kèm với PARTITION BY để phân nhóm dữ liệu và đánh thứ hạng trong từng nhóm.

- So sánh 3 hàm RANK, DENSE_RANK và ROW_NUMBER.

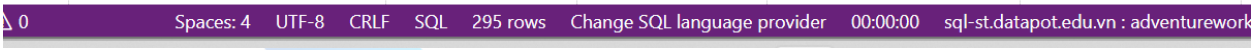
Để phân biệt rõ ràng hơn, ta cùng xem ví dụ sau:

Từ bảng SalesLT.Product thuộc bộ dữ liệu adventureworks, truy vấn các cột Name, StandardCost, xếp hạng thứ tự theo hàm RANK, DENSE_RANK và ROW_NUMBER.

```
102  USE adventureworks
103  SELECT
104      Name,
105      StandardCost,
106      RANK() OVER(ORDER BY StandardCost) AS Product_Rank,
107      DENSE_RANK() OVER(ORDER BY StandardCost) AS Product_Dense_Rank,
108      ROW_NUMBER() OVER(ORDER BY StandardCost) AS Row_Number_Rank
109  FROM SalesLT.Product
```

Results Messages

	Name	StandardCost	Product_Rank	Product_Dense_Rank	Row_Number_Rank
1	Patch Kit/8 Patches	0.8565	1	1	1
2	Road Tire Tube	1.4923	2	2	2
3	Touring Tire Tube	1.8663	3	3	3
4	Mountain Tire Tube	1.8663	3	3	4
5	Water Bottle - 30 oz.	1.8663	3	3	5
6	Bike Wash - Dissolver	2.9733	6	4	6
7	Racing Socks, M	3.3623	7	5	7
8	Racing Socks, L	3.3623	7	5	8
9	Road Bottle Cage	3.3623	7	5	9
10	Mountain Bike Socks, M	3.3963	10	6	10
11	Mountain Bike Socks, L	3.3963	10	6	11
12	Mountain Bottle Cage	3.7363	12	7	12



Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu được truy vấn từ bảng Product.
- SELECT: Truy vấn các cột Name, StandardCost, xếp hạng thứ tự theo hàm RANK, DENSE_RANK và ROW_NUMBER.
  - Hàm ROW_NUMBER: Đánh số thứ tự sau khi sắp xếp theo cột dữ liệu đề ra.
  - Hàm RANK: Hàm đánh xếp hạng, khi xếp hạng đến giá trị giống nhau, những giá trị này được xếp cùng chung 1 hạng và bỏ qua xếp hạng tiếp theo.
  - Hàm DENSE_RANK: Xếp hạng hàng dữ liệu theo thứ tự, khi xếp hạng đến các giá trị giống nhau, những giá trị này được xếp cùng chung 1 hạng và không bỏ qua xếp hạng tiếp theo.

Bạn có thể tham khảo các hàm xếp hạng khác tại Microsoft Document.

5.4.2. Một số hàm tính toán trong hàm cửa sổ (Window Functions) khác

5.4.2.1. Hàm tính toán (Aggregate Functions)

- Hàm SUM

Hàm SUM được sử dụng để tính tổng của một cột.

Cú pháp của hàm SUM():

SUM(expression)

- Expression: Biểu thức cần tính tổng.

Ví dụ: Từ bảng dữ liệu dbo.FactInternetSales và dbo.DimSalesTerritory thuộc bộ dữ liệu AdventureWorksDW2019, kết hợp chung bảng FactInternetSales và DimSalesTerritory. Truy vấn ProductKey, SalesTerritoryCountry, OrderQuantity. Tạo hàm cửa sổ để tính tổng OrderQuantity, sắp xếp theo thứ tự giảm dần của ProductKey.

```
180  USE AdventureWorksDW2019
181  SELECT
182      ProductKey,
183      SalesTerritoryCountry,
184      OrderQuantity,
185      SUM(OrderQuantity) OVER(PARTITION BY SalesTerritoryCountry ORDER BY ProductKey) AS sum_orderqty
186  FROM dbo.FactInternetSales AS FIS
187  INNER JOIN dbo.DimSalesTerritory AS DST
188  ON FIS.SalesTerritoryKey = DST.SalesTerritoryKey
```

Results Messages

	ProductKey	SalesTerritoryCountry	OrderQuantity	sum_orderqty
104	214	United Kingdom	1	289
105	214	United Kingdom	1	289
106	214	United Kingdom	1	289
107	214	France	1	244
108	214	France	1	244

Giải thích câu lệnh truy vấn:

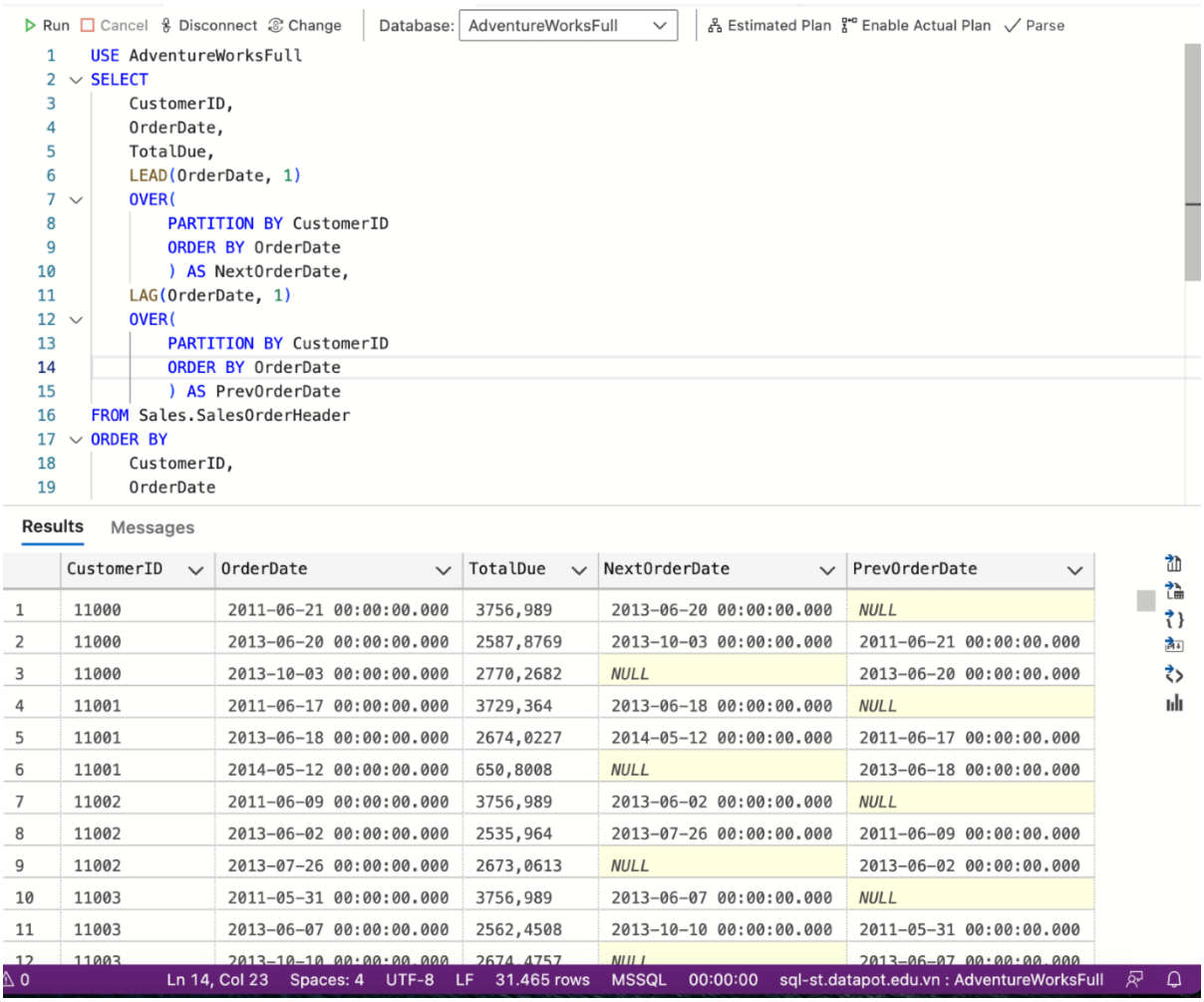
- FROM: Dữ liệu được truy vấn từ bảng SalesOrderHeader.
- INNER JOIN: Kết hợp điểm chung bảng FactInternetSales được gán tên FIS với bảng DimSalesTerritory được gán tên DST.
- ON: Khai báo điều kiện kết hợp bảng từ cột khoá chính SalesTerritoryKey trong bảng FactInternetSales và khoá ngoại SalesTerritoryKey trong bảng DimSalesTerritory.
- SELECT: Truy vấn các cột ProductKey, SalesTerrritoryCountry, OrderQuantity.
    - Hàm cửa sổ với hàm SUM: Gom nhóm theo các quốc gia sau đó tính tổng, sắp xếp theo thứ tự giảm dần của ProductKey và gán tên sum_orderqty.

5.4.2.2. Hàm thống kê (Analytic Functions)

Hàm thống kê (Analytic Functions) thường thấy phổ biến như:

- LEAD(Col): sử dụng để so sánh giữa giá trị hiện tại và giá trị của dòng tiếp theo.
- LAG(Col): sử dụng để so sánh giữa giá trị hiện tại và giá trị của dòng trước đó.

Ví dụ: Từ bảng SalesOrderHeader thuộc bộ dữ liệu AdventureWork, truy vấn ngày Order trước và sau của từng đơn hàng so với ngày hạn



Giải thích câu lệnh truy vấn:

- FROM: Dữ liệu được truy vấn từ bảng SalesOrderHeader.
- SELECT: Truy vấn các cột CustomerID, OrderDate, TotalDate, ngày order trước so với ngày hạn và ngày order sau so với ngày hạn.

Tài liệu tham khảo:

- Mệnh đề GROUP BY và HAVING trong SQL
- Window Function và những ứng dụng của nó trong SQL
- So sánh sự khác nhau của HAVING và WHERE trong SQL
- Phân biệt Window Functions và Group By trong SQL