**How to Calculate Totals, Subtotals and Grand Total**

Asking about totals, subtotals and grand total is a popular ad hoc or regular request by sales or accounting department. In this tutorial, we'll see how to calculate these summary values in MySQL by using a few different techniques.

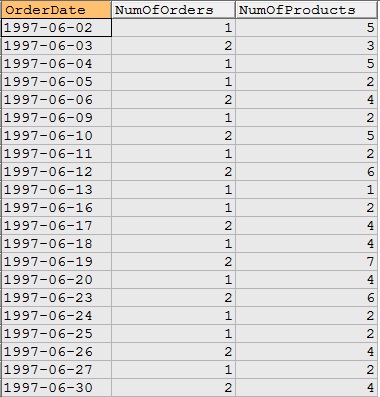
**1. Calculating Totals**

Suppose we need to find out the number of orders and products sold for each date in June 1997 and also the total number of orders and products sold for the entire month in June 1997. With normal SQL techniques, we'll write 2 queries and then union the results, see below.

This query uses GROUP BY to calculate the number of orders and products on each order date.

-- Query 1  
select date(t2.OrderDate) as OrderDate,  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate;

Query result set - 21 rows returned



The following query calculates total number of orders and products for the month in June 1997.

-- Query 2  
select 'Total',  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6;

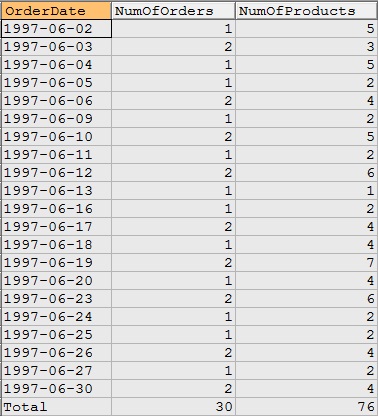
Query result set - 1 rows returned

https://www.geeksengine.com/img/tutorial/mysql/problem-solving/rollup-2.jpg

The following query uses UNION to append Query 2's result to the end of Query 1's result.

-- Query 3  
select date(t2.OrderDate) as OrderDate,  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate  
UNION  
-- Query 2  
select 'Total',  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6;

Query result set - 22 rows returned



**WITH ROLLUP modifier**

In MySQL, there is a better way to accomplish the total summary values calculation - use the WITH ROLLUP modifier in conjunction with GROUP BY clause.

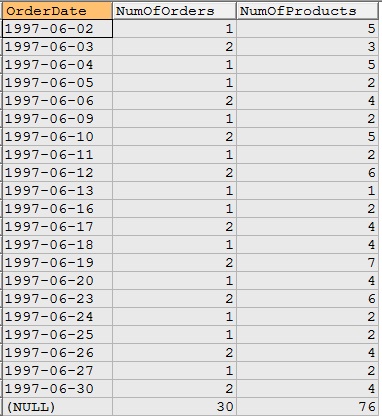
The WITH ROLLUP modifier allows a summary output that represents a higher-level or super-aggregate summarized value for the GROUP BY column(s). If there are multiple columns in the GROUP BY clause, ROLLUP can answer summary questions at multiple aggregated levels over all rows.

A NULL value is produced as a column identifier to indicate the super-aggregated total.

In query 4 below, the output shows the number of orders aggregated for each order date and a total for the month is calculated and placed at the end of the result set. This is achieved by adding the keyword WITH ROLLUP after GROUP BY and produces an identical result to the UNION query showing above (except that NULL is displayed as the label for total).

-- Query 4: ROLLUP  
select date(t2.OrderDate) as OrderDate,  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate WITH ROLLUP;

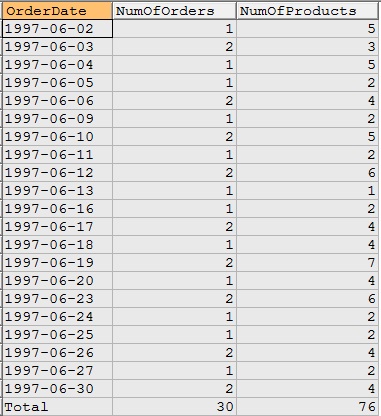
Query result set - 22 rows returned



In the query result above, ROLLUP identifier placed NULL value in the OrderDate column for the month total. To show a more meaningful label, in the query below, we converted the NULL value to the word "Total". Note that comparison operator <=> is NULL-safe comparison operator.

-- Query 4.1: converts NULL to 'Total'  
select if(t2.OrderDate <=> null, 'Total', date(t2.OrderDate)) as OrderDate,  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate WITH ROLLUP;

Query result set - 22 rows returned



**GROUPING() function**

So far we have seen the super-aggregate row uses NULL value as the indicator for the aggregated total. In order to differentiate the NULL values for super-aggregate rows from NULL values in regular grouped rows, MySQL 8.0 introduced GROUPING() function.

As MySQL states:

Each argument to GROUPING() must be an expression that exactly matches an expression in the GROUP BY clause.

For each expression, GROUPING() produces 1 if the expression value in the current row is a NULL representing a super-aggregate value. Otherwise, GROUPING() produces 0, indicating that the expression value is a NULL for a regular result row or is not NULL.

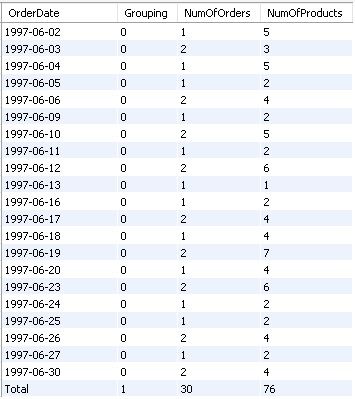
GROUPING() function can be used in the SELECT list or HAVING clause.

In query below, GROUPING function is used to convert NULL value to the word "Total".

Note that, when running the following query, if your MySQL server is not version 8.0 or above, you'll get this error instead "Error Code : 1305 FUNCTION northwind.GROUPING does not exist".

-- Query 5: this query returns identical result as query 4.1 above.  
select if(GROUPING(t2.OrderDate)=1, "Total", date(t2.OrderDate)) as OrderDate,  
 GROUPING(t2.OrderDate) as "Grouping",  
 count(distinct t1.OrderID) as NumOfOrders,  
 count(t1.ProductID) as NumOfProducts   
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate WITH ROLLUP;

Query result set - 22 rows returned



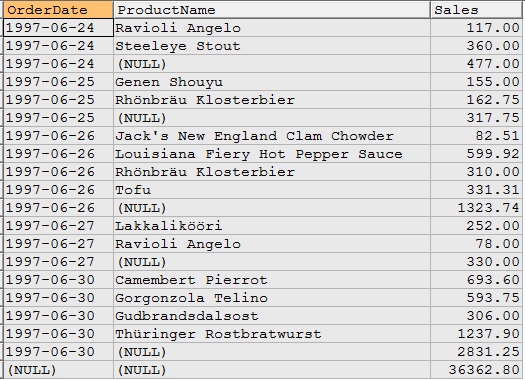
**2. Calculating Subtotals and Grand Total**

Query below has 2 columns in the GROUP BY clause. In this case, the WITH ROLLUP modifier outputs a subtotal value for each order date and a grand total value at the end for all sales in June.

The hierarchy of summary is in the GROUP BY clause. In query here, each individual order date is grouped together to calculate Subtotal, and GRAND TOTAL is calculated at the end for the entire month.

-- Query 6  
select date(t2.OrderDate) as OrderDate,  
 t3.ProductName,  
 round(sum(t1.Quantity\*t1.UnitPrice\*(1-t1.Discount)),2) as Sales  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate, t3.ProductName WITH ROLLUP;

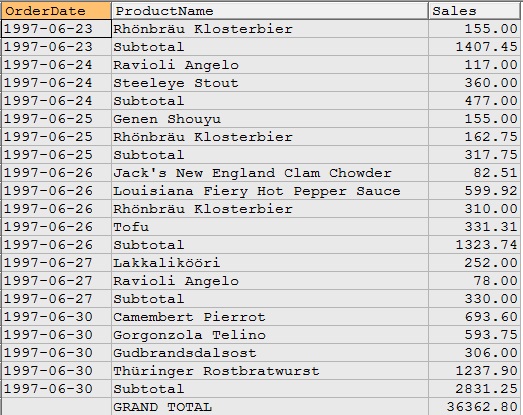
Query result set - 98 rows returned



Below is a similar query but the difference to the query above is that the NULL values are converted to the word "Subtotal" and "GRAND TOTAL".

select if(t2.OrderDate <=> null, '', date(t2.OrderDate)) as OrderDate,  
 if(t2.OrderDate <=> null, 'GRAND TOTAL',   
 if(t3.ProductName <=> null, 'Subtotal',t3.ProductName)) as ProductName,  
 round(sum(t1.Quantity\*t1.UnitPrice\*(1-t1.Discount)),2) as Sales  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate,   
 t3.ProductName   
WITH ROLLUP;

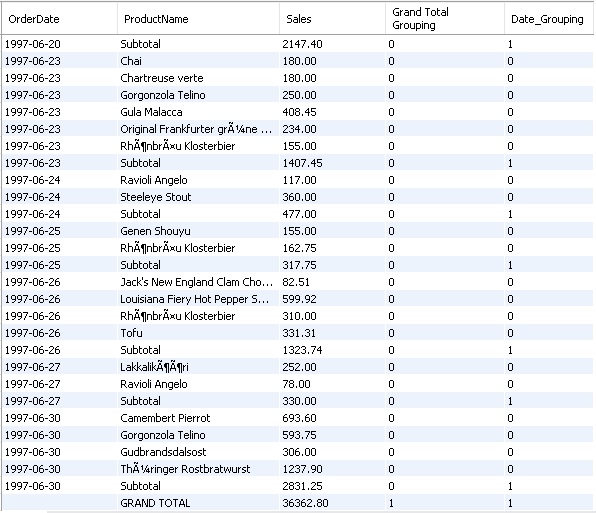
Query result set - 98 rows returned



In query below, two GROUPING functions are used as separate columns on CategoryName and ProductName.

select if(GROUPING(t2.OrderDate)=1, '', date(t2.OrderDate)) as OrderDate,  
 if(GROUPING(t2.OrderDate)=1, 'GRAND TOTAL',   
 if(GROUPING(t3.ProductName)=1, 'Subtotal',t3.ProductName)) as ProductName,  
 round(sum(t1.Quantity\*t1.UnitPrice\*(1-t1.Discount)),2) as Sales,  
 GROUPING(t2.OrderDate) as "Grand Total Grouping",  
 GROUPING(t3.ProductName) as "Date\_Grouping"  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
where year(t2.OrderDate)=1997 and month(t2.OrderDate)=6  
group by t2.OrderDate,   
 t3.ProductName   
WITH ROLLUP;

Query result set - 98 rows returned



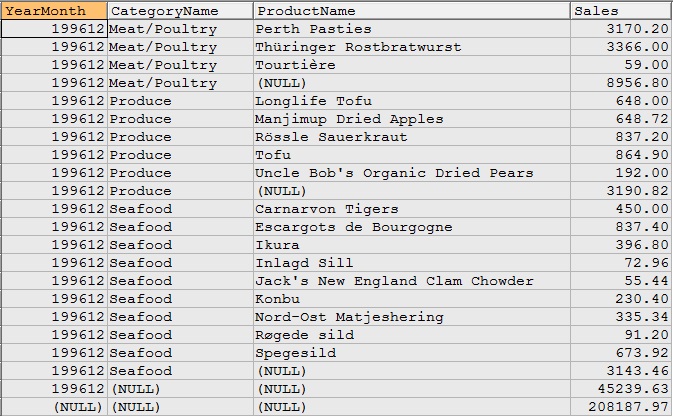
**3. Calculating two Subtotals and a final Grand Total**

Query below shows three levels of hierarchy from low to high.

Level 1 (lowest): total product sales for a category in a month.  
Level 2 (middle): total product sales for a month.  
Level 3 (highest): product sales for all months in the year.

select EXTRACT(YEAR\_MONTH FROM t2.OrderDate) as YearMonth,  
 t4.CategoryName,  
 t3.ProductName,  
 round(sum(t1.Quantity\*t1.UnitPrice\*(1-t1.Discount)),2) as Sales  
from order\_details as t1  
join orders as t2 on t1.OrderID=t2.OrderID  
join products as t3 on t1.ProductID=t3.ProductID  
join categories as t4 on t4.CategoryID=t3.CategoryID  
where year(t2.OrderDate)=1996  
group by EXTRACT(YEAR\_MONTH FROM t2.OrderDate),   
 t4.CategoryName,  
 t3.ProductName  
with ROLLUP;

Query result set



Here GROUPING is applied.

select if(All\_Months\_Grouped=1 and Category\_Grouped=1 and Product\_Grouped=1,   
 "GRAND TOTAL", YearMonth) as YearMonth,  
 if(Category\_Grouped=1 and Product\_Grouped=1,   
 "MONTH TOTAL", CategoryName) as CategoryName,  
 if(Product\_Grouped=1, "CATEGORY TOTAL", ProductName) as ProductName,  
 Sales  
from  
(   
 select EXTRACT(YEAR\_MONTH FROM t2.OrderDate) as YearMonth,  
 t4.CategoryName,  
 t3.ProductName,  
 round(sum(t1.Quantity\*t1.UnitPrice\*(1-t1.Discount)),2) as Sales,  
 GROUPING(EXTRACT(YEAR\_MONTH FROM t2.OrderDate)) as All\_Months\_Grouped,  
 GROUPING(t4.CategoryName) as Category\_Grouped,  
 GROUPING(t3.ProductName) as Product\_Grouped  
 from order\_details as t1  
 join orders as t2 on t1.OrderID=t2.OrderID  
 join products as t3 on t1.ProductID=t3.ProductID  
 join categories as t4 on t4.CategoryID=t3.CategoryID  
 where year(t2.OrderDate)=1996  
 group by EXTRACT(YEAR\_MONTH FROM t2.OrderDate),   
 t4.CategoryName,  
 t3.ProductName  
 with ROLLUP  
) as a;

Query result set

