Aggregate Data for the Last Day of the Month

**By** [**Adam Aspin**](http://www.sqlservercentral.com/Authors/Articles/Adam_Aspin/335295/)**, 2018/08/16**

**FROM:** [**http://www.sqlservercentral.com/articles/T-SQL/173372/?utm\_source=SSC&utm\_medium=pubemail**](http://www.sqlservercentral.com/articles/T-SQL/173372/?utm_source=SSC&utm_medium=pubemail)

This is the seventh article in a short series of SQL Server query challenges to test and expand your SQL Server querying skills. While none of these short articles is of overwhelming complexity, they will hopefully either teach you something new or help you to reactivate potentially dormant SQL skills in preparation for your next real-world challenge. You can see the rest of the articles on [the Query Answers with SQL Server page](http://www.sqlservercentral.com/articles/Query+Answers+with+SQL+Server/174515/).

In this article we will once again imagine that you are working for a small classic car retailer called Prestige Cars Ltd, using the data that is in the company database.

The Challenge

The sales director of Prestige Cars Ltd. is still convinced that some days deliver better sales than others, so now she wants to see the sales for the last day of each month in 2016. Your challenge is to come up with the SQL to satisfy her request. Here is one possible solution:

;

WITH TallyTable\_CTE

AS

(

SELECT ROW\_NUMBER() OVER (ORDER BY StockCode) AS ID

FROM Data.Stock

)

,LastDayOfMonth\_CTE

AS

(

SELECT EOMONTH(DATEFROMPARTS(2016, ID, 1)) AS LastDayDate

FROM TallyTable\_CTE

WHERE ID <= 12

)

SELECT CTE.LastDayDate

,SUM(SLS.SalePrice) AS TotalDailySales

FROM Data.SalesByCountry SLS

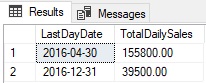
INNER JOIN LastDayOfMonth\_CTE CTE

ON CTE.LastDayDate = CAST(SLS.SaleDate AS DATE)

GROUP BY CTE.LastDayDate

ORDER BY CTE.LastDayDate

If you run this query, you see the output that is shown in Figure 1.



**Figure 1. Showing aggregate data for the last day of the month**

So, in fact, only two months in the entire year have sales on the last day of the month!

How It Works

This query is composed of three fundamental elements.

A tally CTE

That delivers a list of incremental numbers. This CTE uses the ROW\_NUMBER() function to create the list of numbers, starting with 1. The row number field is aliased as ID. The table that the set of numbers is based on is completely irrelevant, as it only serves as a basis for the records that are created, and none of the actual data in the underlying table is required.

A second CTE

That uses the tally CTE as the source for its data. This second CTE extracts 12 records from the tally CTE (because there are 12 months in the year). It does this by applying a WHERE clause to the ID field of the tally CTE that sets the maximum ID to 12. Then the second CTE creates a list of the 12 months of a year (2016 in this case) by applying the DATEFROMPARTS() function to the tally list. Since the tally list ID field is defined as being the second parameter of the DATEFROMPARTS() function (the month in other words), the second CTE generates a list of the 12 months in the year. The last day of each month is then extracted from the month date using the EOMONTH() function.

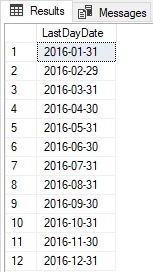
A query

That uses the second CTE as its data source. This query joins to the SalesByCountry view (which contains a list of all the sales from the underlying database tables to prevent you having to carry out complex table joins) on the SaleDate field so that any sales that occurred on the last day of any month are returned. Once again a CAST() function is applied to ensure that the fields used to join the two tables are only using date elements for the join and that any time elements of the data are not used.

This sequence of datasets is probably easier to understand if you display the output from the second CTE. You can do this by placing the following piece of code under the second CTE and deleting the final query.

SELECT \* FROM LastDayOfMonth\_CTE

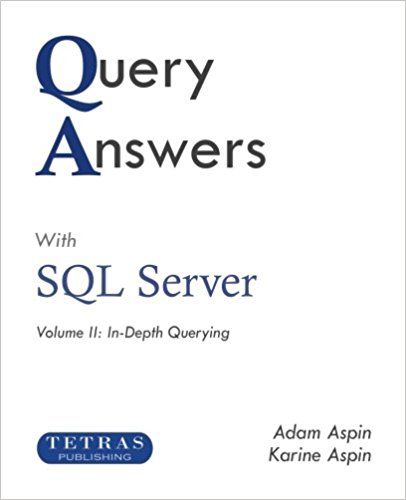
Figure 2 shows you the output from the second of two CTEs.



**Figure 2. Displaying the last day of the month**

That is it – you have seen a simple example of how to aggregate data for the last day of the month. Keep watching SQL Server Central.com ready for the next article in this series.

This article is adapted from the book “Query Answers with SQL Server Volume II: In-Depth Querying” by Adam and Karine Aspin. Available as both a paperback and an eBook from all good resellers including [Amazon](https://www.amazon.com/Query-Answers-SQL-Server-Depth/dp/1999617223), Kobo and iBooks, as well as all good bookstores.

[](http://www.amazon.com/Query-Answers-SQL-Server-Depth/dp/1999617223)

The sample data is available either in the accompanying material for this article, or on the Tetras Publishing website at: [www.tetraspublishing.com/sqlserver](http://www.tetraspublishing.com/sqlserver).

The Series

There are a number of articles in this series. You can see them all on the [Query Answers page](http://www.sqlservercentral.com/articles/Query+Answers+with+SQL+Server/174515/).

Resources:

[SQLQueriesSampleData.zip](http://www.sqlservercentral.com/Files/SQLQueriesSampleData.zip/34901.zip)