**A DateRange Table Valued function**

[Jonathan Roberts](https://www.sqlservercentral.com/author/jonathan-ac-roberts), 2019-08-29 (first published: 2019-08-20)

FROM: <https://www.sqlservercentral.com/scripts/a-daterange-table-valued-function>

Introduction

SQL developers often need to make use of a Calendar table to get date ranges for their queries, this often involves creating a permanent table on the database and populating it with years of data then having to remember to update the Calender table to add new dates and maybe delete old dates as time expires to keep the size of the table under control. This function can be a replacement for a simple calendar table and also has other uses for selecting a range of any datetime parameter.

This is a fast table-valued function that will return a table of datetime2 rows in any date range desired.

The function has an inline tally table to generate a row for each date value required, it  will return enough rows (up to *2*32) to cope with any number the SQL Server DATEDIFF function returns (up to *2*31-1), it is a fast function though you might be waiting a while for it to generate that many rows!

Syntax

SELECT value

FROM dbo.DateRange(startdate, enddate, datepart, interval)

Arguments

**@StartDate** - parameter indicating the value of the first row returned.

This can be any of data types:

* date
* datetime
* datetimeoffset
* datetime2
* smalldatetime
* time

**@EndDate** - parameter indicating the value of the last row returned. note if the datepart is not a perfect divider it may return a higher or lower value depending on the datepart chosen and the interval.

This can be any of:

* date
* datetime
* datetimeoffset
* datetime2
* smalldatetime
* time

**@DatePart** any of 'ns' ,'mcs', 'ms', 'ss' , 'mi', 'hh', 'dd', 'ww', 'mm', 'qq', 'yy' (see [https://docs.microsoft.com/en-us/sql/t-sql/functions/dateadd-transact-sql](https://docs.microsoft.com/en-us/sql/t-sql/functions/dateadd-transact-sql?view=sql-server-2017" \t "_blank))

Note: this parameter must be enclosed in single quotes (unlike datepart parameter to the DATEDIFF function).

Return Types

Returns a single-column table of type datetime2 with column-name [value].

Usage

The function is an inline table-valued function (or sometimes called a parameterised view) so it can be used just like a view or a table but needs parameters. It generates a single-column table of datetime2 with rows differing by any multiple of any of the dateparts that SQL Server provides.

1. Used to get a list of times of all seconds in a given day:

SELECT Value

FROM dbo.DateRange('2018-09-20','2018-09-21','ss',1)

Will return 86,401 rows of every second between 2018-09-20 00:00:00 and 2018-09-21 00:00:00

If you enter a startdate that's greater than the enddate it will return the list in descending order, of course as it returns a table you can sort it how you like with an ORDER BY.

SELECT Value

FROM dbo.DateRange('2018-09-21','2018-09-20','mi',10)

Will return times in reverse order of 10 seconds intervals from '2018-09-21' to '2018-09-20'

The return order can be overridden:

SELECT value

FROM dbo.DateRange('2018-09-21','2018-09-20','mi',10)

ORDER BY value ASC

2. Used to get a contiguous list of dates between two date variables for left joining on a table that has missing dates (if you want all dates to be shown whether or not the joining table has them.

SELECT dr.value, SUM(s.SalesRevenue) Revenue

FROM dbo.DateRange('2018-09-01','2018-09-02','dd',1) dr

LEFT JOIN dbo.Sales s on s.Date = dr.Date

GROUP BY dr.Date

ORDER BY dr.Date

Uses

There are many uses of this function, it won't return business or national specific dates like Bank Holiday dates and business financial dates but it can be used instead of a simple calendar table. It will return a table with a range of datetimes for any datepart (or muliple of) that SQL Server provides.Its uses include replacing a simple calendar table, to select a full range of times when some of the times are not present of the table being queried.

SELECT dr1.Value Date, DATENAME(weekday,dr1.Value),x.time

FROM [dbo].[DateRange] ('20181001','20201101','dd',1) dr1

CROSS APPLY(SELECT CONVERT(time,[Value]) time FROM [dbo].[DateRange] ('20181001','20181002','hh',1) dr2 WHERE convert(time,dr2.Value) between convert(time,'07:00') AND convert(time,'16:00')) x

WHERE DATEPART(dw,dr1.Value) NOT IN (1,2)

ORDER BY 1,3

IF OBJECT\_ID('[dbo].[DateRange]','IF') IS NULL BEGIN

PRINT 'CREATE FUNCTION [dbo].[DateRange]'

EXEC ('CREATE FUNCTION [dbo].[DateRange] () RETURNS TABLE AS RETURN SELECT 1 X')

END

GO

/\*-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: DateRange

-- Returns a table of datetime values based on the parameters

-- Parameters:

-- @StartDate :Start date of the series

-- @EndDate :End date of the series

-- @DatePart :The time unit for @interval

-- ns : nanoseconds

-- mcs : microseconds

-- ms : milliseconds

-- ss : seconds

-- mi : minutes

-- hh : hours

-- dd : days

-- ww : weeks

-- mm : months

-- qq : quarters

-- yy : years

-- @Interval :The number of dateparts between each value returned

--

-- Sample Calls:

-- SELECT \* FROM [dbo].[DateRange]('2011-01-01 12:24:35', '2011-02-01 12:24:35', 'ss', 2)

-- SELECT COUNT(\*) FROM [dbo].[DateRange]('2018-01-01 00:00:00', '2018-01-25 20:31:23.646', 'ms', default)

-- SELECT \* FROM [dbo].[DateRange]('2011-01-01', '2012-02-03', default, default)

-- SELECT \* FROM [dbo].[DateRange]('2012-02-03', '2011-01-01', 'dd', 7)

-- SELECT DATEDIFF(ns,'2018-01-01 00:00:00.000', value),Value,\* FROM [dbo].[DateRange]('2018-01-01 00:00:00.000', '2018-01-01 00:00:00.00001', 'ns', 100)

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ALTER FUNCTION [dbo].[DateRange]

(

@StartDate datetime2,

@EndDate datetime2,

@DatePart nvarchar(3)='dd',

@Interval int=1

)

RETURNS TABLE AS RETURN

WITH A(A) AS (SELECT 0 FROM (VALUES (0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0)) A(A)),

B(RowNum) AS (SELECT TOP(ABS(CASE @DatePart

WHEN 'ns' THEN DATEDIFF(ns, @EndDate, @StartDate)/@Interval

WHEN 'mcs' THEN DATEDIFF(mcs,@EndDate, @StartDate)/@Interval

WHEN 'ms' THEN DATEDIFF(ms, @EndDate, @StartDate)/@Interval

WHEN 'ss' THEN DATEDIFF(ss, @EndDate, @StartDate)/@Interval

WHEN 'mi' THEN DATEDIFF(mi, @EndDate, @StartDate)/@Interval

WHEN 'hh' THEN DATEDIFF(hh, @EndDate, @StartDate)/@Interval

WHEN 'dd' THEN DATEDIFF(dd, @EndDate, @StartDate)/@Interval

WHEN 'ww' THEN DATEDIFF(ww, @EndDate, @StartDate)/@Interval

WHEN 'mm' THEN DATEDIFF(mm, @EndDate, @StartDate)/@Interval

WHEN 'qq' THEN DATEDIFF(qq, @EndDate, @StartDate)/@Interval

WHEN 'yy' THEN DATEDIFF(yy, @EndDate, @StartDate)/@Interval

ELSE DATEDIFF(dd, IIF(@StartDate < @EndDate, @StartDate, @EndDate), IIF(@StartDate < @EndDate, @EndDate, @StartDate))/@Interval

END) + 1)

ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) - 1

FROM A A, A B, A C, A D, A E, A F, A G, A H) -- A maximum of 16^8 (or 2^32) rows could be returned from this inline tally

SELECT CASE @DatePart

WHEN 'ns' THEN DATEADD(ns, T.AddAmount, @StartDate)

WHEN 'mcs' THEN DATEADD(mcs,T.AddAmount, @StartDate)

WHEN 'ms' THEN DATEADD(ms, T.AddAmount, @StartDate)

WHEN 'ss' THEN DATEADD(ss, T.AddAmount, @StartDate)

WHEN 'mi' THEN DATEADD(mi, T.AddAmount, @StartDate)

WHEN 'hh' THEN DATEADD(hh, T.AddAmount, @StartDate)

WHEN 'dd' THEN DATEADD(dd, T.AddAmount, @StartDate)

WHEN 'ww' THEN DATEADD(ww, T.AddAmount, @StartDate)

WHEN 'mm' THEN DATEADD(mm, T.AddAmount, @StartDate)

WHEN 'qq' THEN DATEADD(qq, T.AddAmount, @StartDate)

WHEN 'yy' THEN DATEADD(yy, T.AddAmount, @StartDate)

ELSE DATEADD(dd, T.AddAmount, @StartDate)

END [Value]

FROM B

CROSS APPLY(VALUES (IIF(@StartDate<@EndDate, @interval\*RowNum, @interval\*-RowNum))) T(AddAmount)

GO

Recent comments

SSC Enthusiast

Points: 145

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 20, 2019 at 11:30 pm

[#3672967](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3672967)

Danke schön!  Nicely done -- it's in my toolbox now.

We have a project coming up where I'll have the chance to alter or replace the following proc (which runs in prod now):

if object\_id('dbo.api\_delivery\_dates\_view\_get') is not null  
 drop proc dbo.api\_delivery\_dates\_view\_get;  
go  
create proc dbo.api\_delivery\_dates\_view\_get  
 @qtr\_id int  
as  
declare   
 @start\_dt date,  
 @frequency int,  
 @end\_dt date;  
  
select   
 @start\_dt=delivery\_dt,  
 @frequency=freq\_days  
from  
 calendars  
where  
 qtr\_id=@qtr\_id;  
  
select @end\_dt=dateadd(qq, datediff(qq, 0, @start\_dt)+1, -1);  
  
with  
calendar\_range\_cte(delivery\_dt) as (  
 select @start\_dt  
 union all  
 select dateadd(d, @frequency, delivery\_dt)  
 from calendar\_range\_cte   
 where delivery\_dt < @end\_dt),  
calendar\_reschedules\_cte(old\_delivery\_dt, new\_delivery\_dt) as (  
 select old\_delivery\_dt, new\_delivery\_dt  
 from calendar\_reschedules  
 where qtr\_id=@qtr\_id)  
select  
 coalesce(crec.new\_delivery\_dt, crc.delivery\_dt) delivery\_dt  
from  
 calendar\_range\_cte crc  
 left join  
 calendar\_reschedules\_cte crec on crc.delivery\_dt=crec.old\_delivery\_dt  
where  
 crc.delivery\_dt < @end\_dt  
except  
select  
 cancel\_delivery\_dt  
from  
 calendar\_cancellations  
where  
 qtr\_id=@qtr\_id  
option (maxrecursion 0)  
for json path;  
go

Your function slides right in like:

drop proc if exists api\_delivery\_dates\_view\_get\_test;  
go  
create proc api\_delivery\_dates\_view\_get\_test  
 @qtr\_id int  
as  
declare   
 @start\_dt date,  
 @frequency int,  
 @end\_dt date;  
  
select   
 @start\_dt=delivery\_dt,  
 @frequency=freq\_days  
from  
 calendars  
where  
 qtr\_id=@qtr\_id;  
  
select @end\_dt=dateadd(qq, datediff(qq, 0, @start\_dt)+1, -1);  
  
with  
calendar\_range\_cte(delivery\_dt) as (  
 select [value] as delivery\_dt from dbo.daterange(@start\_dt, @end\_dt, 'dd', @frequency)),  
calendar\_reschedules\_cte(old\_delivery\_dt, new\_delivery\_dt) as (  
 select old\_delivery\_dt, new\_delivery\_dt  
 from calendar\_reschedules  
 where qtr\_id=@qtr\_id)  
select  
 coalesce(crec.new\_delivery\_dt, crc.delivery\_dt) delivery\_dt  
from  
 calendar\_range\_cte crc  
 left join  
 calendar\_reschedules\_cte crec on crc.delivery\_dt=crec.old\_delivery\_dt  
where  
 crc.delivery\_dt < @end\_dt  
except  
select  
 cancel\_delivery\_dt  
from  
 calendar\_cancellations  
where  
 qtr\_id=@qtr\_id  
for json path;

No need to set max recursions anymore.  No need for the where clause with inequality comparison on date.

The only change to the function I made was to make it d.i.e. (drop if exists) and make it all lower case.  Since Sql 2016 most objects can d.i.e.

print 'create function [dbo].[daterange]'   
drop function if exists dbo.daterange;  
go   
/\*-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
-- function: daterange  
-- returns a table of datetime values based on the parameters  
-- parameters:   
-- @startdate :start date of the series   
-- @enddate :end date of the series   
-- @datepart :the time unit for @interval  
-- ns : nanoseconds   
-- mcs : microseconds   
-- ms : milliseconds   
-- ss : seconds  
-- mi : minutes  
-- hh : hours  
-- dd : days  
-- ww : weeks  
-- mm : months  
-- qq : quarters  
-- yy : years  
-- @interval :the number of dateparts between each value returned  
--  
-- sample calls:  
-- select \* from [dbo].[daterange]('2011-01-01 12:24:35', '2011-02-01 12:24:35', 'ss', 2)  
-- select count(\*) from [dbo].[daterange]('2018-01-01 00:00:00', '2018-01-25 20:31:23.646', 'ms', default)  
-- select \* from [dbo].[daterange]('2011-01-01', '2012-02-03', default, default)  
-- select \* from [dbo].[daterange]('2012-02-03', '2011-01-01', 'dd', 7)  
-- select datediff(ns,'2018-01-01 00:00:00.000', value),value,\* from [dbo].[daterange]('2018-01-01 00:00:00.000', '2018-01-01 00:00:00.00001', 'ns', 100)  
-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
create function dbo.daterange  
(  
 @startdate datetime2,   
 @enddate datetime2,   
 @datepart nvarchar(3)='dd',   
 @interval int=1  
)  
returns table as return   
with  
a(a) as (  
 select 0 from (values (0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0),(0)) a(a)),  
b(rownum) as (  
 select top(abs(case @datepart  
 when 'ns' then datediff(ns, @enddate, @startdate)/@interval  
 when 'mcs' then datediff(mcs,@enddate, @startdate)/@interval  
 when 'ms' then datediff(ms, @enddate, @startdate)/@interval  
 when 'ss' then datediff(ss, @enddate, @startdate)/@interval  
 when 'mi' then datediff(mi, @enddate, @startdate)/@interval  
 when 'hh' then datediff(hh, @enddate, @startdate)/@interval  
 when 'dd' then datediff(dd, @enddate, @startdate)/@interval  
 when 'ww' then datediff(ww, @enddate, @startdate)/@interval  
 when 'mm' then datediff(mm, @enddate, @startdate)/@interval  
 when 'qq' then datediff(qq, @enddate, @startdate)/@interval  
 when 'yy' then datediff(yy, @enddate, @startdate)/@interval  
 else datediff(dd, iif(@startdate < @enddate, @startdate, @enddate), iif(@startdate < @enddate, @enddate, @startdate))/@interval  
 end) + 1)  
 row\_number() over (order by (select null)) - 1  
 from a a, a b, a c, a d, a e, a f, a g, a h) -- a maximum of 16^8 (or 2^32) rows could be returned from this inline tally  
select case @datepart   
 when 'ns' then dateadd(ns, t.addamount, @startdate)  
 when 'mcs' then dateadd(mcs,t.addamount, @startdate)  
 when 'ms' then dateadd(ms, t.addamount, @startdate)  
 when 'ss' then dateadd(ss, t.addamount, @startdate)  
 when 'mi' then dateadd(mi, t.addamount, @startdate)  
 when 'hh' then dateadd(hh, t.addamount, @startdate)  
 when 'dd' then dateadd(dd, t.addamount, @startdate)  
 when 'ww' then dateadd(ww, t.addamount, @startdate)  
 when 'mm' then dateadd(mm, t.addamount, @startdate)  
 when 'qq' then dateadd(qq, t.addamount, @startdate)  
 when 'yy' then dateadd(yy, t.addamount, @startdate)  
 else dateadd(dd, t.addamount, @startdate)  
 end [value]  
 from b  
 cross apply(values (iif(@startdate<@enddate, @interval\*rownum, @interval\*-rownum))) t(addamount)  
go

Cheers!

[Jonathan AC Roberts](https://www.sqlservercentral.com/forums/user/jonathan-ac-roberts)

SSCoach

Points: 16809

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 11:45 am

[#3673065](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673065)

Hi scdecade,

Thanks for the feedback, I'm glad you've found a use for the function.

I didn't use "drop if exists" as this is a feature that was only introduced in SQL Server 2016, so it wouldn't work on earlier versions. Also, I find it good to alter a stored procedure or function instead of drop and create as the security settings for users won't be lost.

I think you can squeeze your entire SP into one SQL query:

SELECT COALESCE(crec.new\_delivery\_dt, crc.delivery\_dt) delivery\_dt  
 FROM (SELECT TOP(1) c.delivery\_dt start\_dt,   
 dateadd(qq, datediff(qq, 0, c.delivery\_dt) + 1, -1) end\_dt,  
 freq\_days frequency  
 FROM calendars c  
 WHERE c.qtr\_id = @qtr\_id) AS parms  
 CROSS APPLY (SELECT [value] delivery\_dt  
 FROM dbo.daterange(parms.start\_dt, parms.end\_dt, 'dd', parms.frequency) crc   
 WHERE crc.[value] < parms.end\_dt) crc   
 LEFT JOIN calendar\_reschedules crec  
 ON crec.old\_delivery\_dt = crc.delivery\_dt  
 AND crec.qtr\_id = @qtr\_id  
 WHERE NOT EXISTS (SELECT \*   
 FROM calendar\_cancellations cc  
 WHERE cc.qtr\_id = @qtr\_id   
 AND cc.cancel\_delivery\_dt = COALESCE(crec.new\_delivery\_dt, crc.delivery\_dt))  
 FOR JSON PATH;

[Jeff Moden](https://www.sqlservercentral.com/forums/user/jeff-moden)

SSC Guru

Points: 994313

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 11:50 am

[#3673067](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673067)

Nice function, Jonathan.  Well done!

--Jeff Moden

RBAR is pronounced "ree-bar" and is a "Modenism" for Row-By-Agonizing-Row.  
First step towards the paradigm shift of writing Set Based code:  
\_\_\_\_\_\_\_\_Stop thinking about what you want to do to a row... think, instead, of what you want to do to a column.  
"If you think its expensive to hire a professional to do the job, wait until you hire an amateur."--Red Adair  
"Change is inevitable... change for the better is not."  
When you put the right degree of spin on it, the number 3|8 is also a glyph that describes the nature of a DBAs job.

Helpful Links:  
[How to post code problems](https://www.sqlservercentral.com/articles/forum-etiquette-how-to-post-datacode-on-a-forum-to-get-the-best-help)  
[Create a Tally Function (fnTally)](https://www.sqlservercentral.com/scripts/create-a-tally-function-fntally)

[Jonathan AC Roberts](https://www.sqlservercentral.com/forums/user/jonathan-ac-roberts)

SSCoach

Points: 16809

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 12:07 pm

[#3673071](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673071)

Jeff Moden wrote:

Nice function, Jonathan.  Well done!

Thank you Jeff, that's a great compliment coming from someone as experienced as you.

[scdecade](https://www.sqlservercentral.com/forums/user/scdecade)

SSC Enthusiast

Points: 145

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 1:48 pm

[#3673128](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673128)

Your proc produced the correct output.  I'm still trying to get my head around cross apply.  It's kind of magical.  The last part, where not exists, I was really sure that wasn't going to work but it did!

A couple of things:

qtr\_id is primary key of the 'calendars' table, so top(1) was not necessary (no way you could've known)

"WHERE crc.[value] < parms.end\_dt" was not necessary at all so I deleted it.  Your function obviated that necessity.  I wrote that but then I pasted it in with the code.  Sorry my butterfingers!

The dates in our tables (calendar, calendar\_reschedules, and calendar\_cancellations) are all 'date' column types.  The output of the function was datetime so I cast it back to date.

The proc looks like this now:

drop proc if exists api\_delivery\_dates\_view\_get\_test;  
go  
create proc api\_delivery\_dates\_view\_get\_test  
 @qtr\_id int  
as  
set nocount on;  
  
with  
parms\_cte(start\_dt, end\_dt, frequency) as (  
 select  
 delivery\_dt start\_dt,   
 dateadd(qq, datediff(qq, 0, delivery\_dt) + 1, -1) end\_dt,  
 freq\_days frequency  
 from  
 calendars  
 where  
 qtr\_id = @qtr\_id),  
cr\_cte(old\_delivery\_dt, new\_delivery\_dt) as (  
 select  
 old\_delivery\_dt,  
 new\_delivery\_dt  
 from  
 calendar\_reschedules  
 where  
 qtr\_id=@qtr\_id),  
cc\_cte(delivery\_dt) as (  
 select  
 cancel\_delivery\_dt  
 from  
 calendar\_cancellations  
 where  
 qtr\_id=@qtr\_id)  
select   
 coalesce(cc.new\_delivery\_dt, cal.delivery\_dt) delivery\_dt  
from  
 parms\_cte pc  
 cross apply  
 (select cast([value] as date) delivery\_dt from dbo.daterange(pc.start\_dt, pc.end\_dt, 'dd', pc.frequency)) cal  
 left join  
 cr\_cte cc on cc.old\_delivery\_dt = cal.delivery\_dt  
where   
 not exists (select \* from cc\_cte where delivery\_dt = coalesce(cc.new\_delivery\_dt, cal.delivery\_dt))  
for json path;  
go  
  
--exec api\_delivery\_dates\_view\_get\_test 4

As far as making objects d.i.e remember when this happened?

<https://siliconangle.com/2019/05/19/salesforce-recovers-outage-caused-faulty-database-script/>

This put the fear of the almighty into management here.  I was told to make sure this is as close to impossible as possible.  So we have a process that happens before users are assigned rights to objects/procedures.  I wonder if the dba survived with job intact.  Does anybody know?

Thanks again Jonathan!

[Jonathan AC Roberts](https://www.sqlservercentral.com/forums/user/jonathan-ac-roberts)

SSCoach

Points: 16809

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 3:39 pm

[#3673166](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673166)

scdecade wrote:

"WHERE crc.[value] < parms.end\_dt" was not necessary at all so I deleted it.  Your function obviated that necessity.  I wrote that but then I pasted it in with the code

I'm not sure that you should remove that as the original code had it in.

These two queries produce different results, @EndDate is not included if you add the where clause.

declare @StartDate datetime2 = '20190821',  
 @EndDate datetime2 = '20190828';  
  
select cast([value] as date) delivery\_dt  
 from dbo.DateRange(@StartDate,@EndDate,'dd',1);  
  
select cast([value] as date) delivery\_dt  
 from dbo.DateRange(@StartDate,@EndDate,'dd',1)  
 where [value] < @EndDate;

An alternative way of doing this without putting it in the where clause would be to subtract 1 day from @EndDate within the DateRange parameters:

select cast([value] as date) delivery\_dt   
 from dbo.DateRange(@StartDate,DATEADD(dd,-1,@EndDate),'dd',1)

[rVadim](https://www.sqlservercentral.com/forums/user/rvadim)

Hall of Fame

Points: 3899

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 3:48 pm

[#3673170](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673170)

Thanks Jonathan, nice function. Which versions of SQL Server it will work with? It doesn't seem to work on 2008R2. I replaced IIF with CASE but now getting this:

Msg 535, Level 16, State 0, Line 33  
The datediff function resulted in an overflow. The number of dateparts separating two date/time instances is too large. Try to use datediff with a less precise datepart.

--Vadim R.

[Jonathan AC Roberts](https://www.sqlservercentral.com/forums/user/jonathan-ac-roberts)

SSCoach

Points: 16809

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 3:55 pm

[#3673172](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673172)

rVadim wrote:

Thanks Jonathan, nice function. Which versions of SQL Server it will work with? It doesn't seem to work on 2008R2. I replaced IIF with CASE but now getting this:

Msg 535, Level 16, State 0, Line 33  
The datediff function resulted in an overflow. The number of dateparts separating two date/time instances is too large. Try to use datediff with a less precise datepart.

As it's written with IIF it would only work with SS 2012 or higher.

datetime2 was introduced in SS 2008, so if you've changed the IIF to CASE I think it should work with 2008R2.

From the error message, it looks like you have called it with a date range that would result in the datediff function used within the daterange function returning a value outside the range of (-2,147,483,648 to +2,147,483,647).

What values were you using to call the DateRange function?

[rVadim](https://www.sqlservercentral.com/forums/user/rvadim)

Hall of Fame

Points: 3899

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 3:58 pm

[#3673173](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673173)

Sample calls from function comment. The last one works but 4 prior produce that error.

--Vadim R.

[Jonathan AC Roberts](https://www.sqlservercentral.com/forums/user/jonathan-ac-roberts)

SSCoach

Points: 16809

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 4:13 pm

[#3673194](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673194)

rVadim wrote:

Sample calls from function comment. The last one works but 4 prior produce that error.

I don't have access to a 2008 server, but maybe you could try it with the Start/End date parameters closer together to see if you can get it to return some results?

[scdecade](https://www.sqlservercentral.com/forums/user/scdecade)

SSC Enthusiast

Points: 145

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 6:08 pm

[#3673215](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673215)

Jonathan AC Roberts wrote:

scdecade wrote:

"WHERE crc.[value] < parms.end\_dt" was not necessary at all so I deleted it.  Your function obviated that necessity.  I wrote that but then I pasted it in with the code

I'm not sure that you should remove that as the original code had it in.

These two queries produce different results, @EndDate is not included if you add the where clause.

declare @StartDate datetime2 = '20190821',  
 @EndDate datetime2 = '20190828';  
  
select cast([value] as date) delivery\_dt  
 from dbo.DateRange(@StartDate,@EndDate,'dd',1);  
  
select cast([value] as date) delivery\_dt  
 from dbo.DateRange(@StartDate,@EndDate,'dd',1)  
 where [value] < @EndDate;

An alternative way of doing this without putting it in the where clause would be to subtract 1 day from @EndDate within the DateRange parameters:

Jonathan AC Roberts wrote:

scdecade wrote:

"WHERE crc.[value] < parms.end\_dt" was not necessary at all so I deleted it.  Your function obviated that necessity.  I wrote that but then I pasted it in with the code

I'm not sure that you should remove that as the original code had it in.

These two queries produce different results, @EndDate is not included if you add the where clause.

declare @StartDate datetime2 = '20190821',  
 @EndDate datetime2 = '20190828';  
  
select cast([value] as date) delivery\_dt  
 from dbo.DateRange(@StartDate,@EndDate,'dd',1);  
  
select cast([value] as date) delivery\_dt  
 from dbo.DateRange(@StartDate,@EndDate,'dd',1)  
 where [value] < @EndDate;

An alternative way of doing this without putting it in the where clause would be to subtract 1 day from @EndDate within the DateRange parameters:

select cast([value] as date) delivery\_dt   
 from dbo.DateRange(@StartDate,DATEADD(dd,-1,@EndDate),'dd',1)

select cast([value] as date) delivery\_dtfrom dbo.daterange(@startdate,dateadd(dd,-1,@enddate),'dd',1)

The last one (but in lower case)

This reply was modified 1 week ago by  [scdecade](https://www.sqlservercentral.com/forums/user/scdecade" \o "View scdecade's profile).

[scdecade](https://www.sqlservercentral.com/forums/user/scdecade)

SSC Enthusiast

Points: 145

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 7:24 pm

[#3673241](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673241)

Have a look at this code:

declare  
 @start\_dt date,  
 @frequency int,  
 @end\_dt date;  
  
select  
 @start\_dt='2019-09-01',  
 @frequency=14;  
  
/\* the end\_dt is always set to the last day of the quarter (for any arbitrary start\_dt)\*/  
select @end\_dt=dateadd(qq, datediff(qq, 0, @start\_dt)+1, -1);  
  
select @end\_dt end\_dt;  
-- returns --  
/\*  
end\_dt  
2019-09-30  
\*/  
  
/\* create recursive cte with strict date inequality \*/  
with  
calendar\_range\_cte(delivery\_dt) as (  
 select @start\_dt  
 union all  
 select dateadd(d, @frequency, delivery\_dt)  
 from calendar\_range\_cte   
 where delivery\_dt < @end\_dt)  
select \* from calendar\_range\_cte;  
  
-- returns --  
/\*  
delivery\_dt  
2019-09-01  
2019-09-15  
2019-09-29  
2019-10-13  
\*/  
  
/\* create recursive cte with strict date inequality AND strict date inequality in accessor \*/  
with  
calendar\_range\_cte(delivery\_dt) as (  
 select @start\_dt  
 union all  
 select dateadd(d, @frequency, delivery\_dt)  
 from calendar\_range\_cte   
 where delivery\_dt < @end\_dt)  
select \* from calendar\_range\_cte  
where  
 delivery\_dt <= @end\_dt;  
  
-- returns --  
/\*  
delivery\_dt  
2019-09-01  
2019-09-15  
2019-09-29  
\*/

It returns 3 results:

the end date of the quarter

A cte with a strict inequality on end date within the definition of the cte

A cte with a strict inequality in the cte and within the select accessor.

The accessor of the cte always delivers the next value in the recursion.  #2 returns '2019-10-13' which is the next value after the inequality inside the cte.  It is Sql Server's behavior obviously but I guess I never understood why this was the case.  When I saw your code it seemed to click I could ditch the second where clause.

[Jonathan AC Roberts](https://www.sqlservercentral.com/forums/user/jonathan-ac-roberts)

SSCoach

Points: 16809

[Follow](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function#follow)

August 21, 2019 at 8:36 pm

[#3673303](https://www.sqlservercentral.com/forums/topic/a-daterange-table-valued-function/#post-3673303)

scdecade wrote:

The accessor of the cte always delivers the next value in the recursion.  #2 returns '2019-10-13' which is the next value after the inequality inside the cte.  It is Sql Server's behavior obviously but I guess I never understood why this was the case.  When I saw your code it seemed to click I could ditch the second where clause.

Yes, I see now, I didn't look at your original code too carefully