Generate Dates between date ranges

Asked 11 years, 2 months ago Modified 1 month ago Viewed 134k times

62

I need to populate a table that will store the date ranges between 2 given dates: 09/01/11 - 10/10/11

So in this case the table would start from 09/01/11 and store each day till it got to 10/10/11 I was wondering if there was a slick way of doing this in SQL Server - I am currently using SQL Server 2008. Thanks

sql sql-server tsql sql-server-2008 range

Share Edit Follow

edited Oct 19, 2011 at 16:47
sll
60.5k 21 104 155

asked Oct 19, 2011 at 16:39
Nate Pet
42.9k 120 264 406

Related: Select Consecutive Numbers in SQL – KyleMit ♦ Jan 2, 2020 at 20:36

13 Answers

Sorted by:
Highest score (default)

80

Easy on SQL 2005+; easier if you have a numbers or tally table. I faked it below:

```
DECLARE @StartDate DATE = '20110901'
  , @EndDate DATE = '20111001'

SELECT  DATEADD(DAY, nbr - 1, @StartDate)
FROM    ( SELECT    ROW_NUMBER() OVER ( ORDER BY c.object_id ) AS nbr
          FROM      sys.columns c
        ) nbrs
WHERE   nbr - 1 <= DATEDIFF(DAY, @StartDate, @EndDate)
```

If you have a tally table, replace the subquery with the table. No recursion.

EDIT: Since folks seem to have questions about the tally table, let me rewrite this using a zero-based tally table. First, here's some code to create and populate a table.

```
CREATE TABLE [dbo].[nbrs](
    [nbr] [INT] NOT NULL
) ON [PRIMARY]
GO


CREATE UNIQUE CLUSTERED INDEX [clidx] ON [dbo].[nbrs]
(
    [nbr] ASC
)
GO

INSERT INTO dbo.nbrs (nbr)
```

```
SELECT nbr-1
FROM ( SELECT    ROW_NUMBER() OVER ( ORDER BY c.object_id ) AS nbr
          FROM      sys.columns c
        ) nbrs
GO
```

Now, that you have the numbers table as a permanent object in your database, you can reuse it for the query INSTEAD of the subquery. The query has also been edited to use a zero-based calculation.

```
DECLARE @StartDate DATE = '20110901'
      , @EndDate DATE = '20111001'

SELECT  DATEADD(DAY, nbr, @DateStart)
FROM    nbrs
WHERE   nbr <= DATEDIFF(DAY, @DateStart, @DateEnd)
```

Performant, and no recursion.

Share Edit Follow

edited Jan 7, 2022 at 2:56

answered Oct 19, 2011 at 16:55

Stuart Ainsworth
12.7k 41 46

Quite useful when prototyping in Spark JDBC and CTEs can not be used cause everything is wrapped as a subquery. – Răzvan Flavius Panda May 27, 2016 at 9:38

1 Your database must have a lot of columns in. The best solution would be to use a table of numbers (or a tally table). The sub query is there as a fake. Replace the sub query with the tally table. – Stuart Ainsworth Jan 6, 2022 at 22:51

1 Just to make the warning explicit to everyone looking at this answer: the subquery `SELECT ROW_NUMBER() OVER ( ORDER BY c.object_id ) AS nbr FROM sys.columns c` puts a limit on how many dates are returned. E.g for me the max dates is 1399, You can calculate this with `select count(*) from sys.columns` . I imagine this will catch a lot of people by surprise. – br3nt Jun 7, 2022 at 3:14

1 It's limited to the number of columns you have in your database, so it will vary accordingly. It's a poor proxy for a numbers table, but for mocking up this scenario it works. If you need to populate a much larger numbers table, you can do a cross join between sys.columns and sys.columns (in your case 1399 x 1399). – Stuart Ainsworth Jun 7, 2022 at 22:40

1 To get the series at Day and Hour Level. `SELECT  DATEADD(HOUR, nbr - 1, @FromDate) as DayHour FROM (SELECT  ROW_NUMBER() OVER ( ORDER BY c.object_id ) AS nbr  FROM  sys.columns c ) nbrs WHERE  nbr - 1 <= DATEDIFF(HOUR, @FromDate, @ToDate)` – Shekar Gurram Jul 28, 2022 at 17:14

55

Try this if you are using SQL Server 2005 or newer:

```
WITH Dates AS (
        SELECT
         [Date] = CONVERT(DATETIME,'09/01/2011')
        UNION ALL SELECT
         [Date] = DATEADD(DAY, 1, [Date])
        FROM
         Dates
        WHERE
         Date < '10/10/2011'
) SELECT
 [Date]
```

```
FROM
 Dates
 OPTION (MAXRECURSION 45)
```

A good example of cool stuff you can do with a CTE.

Share Edit Follow

answered Oct 19, 2011 at 16:43

Abe Miessler
81.2k 97 302 480

16

-- **Declarations**

```
DECLARE @dates TABLE(dt DATE)
DECLARE @dateFrom DATE
DECLARE @dateTo DATE

SET @dateFrom = '2001/01/01'
SET @dateTo = '2001/01/12'
```

-- **Query:**

```
WHILE(@dateFrom <= @dateTo)
BEGIN
   INSERT INTO @dates
   SELECT @dateFrom

   SELECT @dateFrom = DATEADD(day, 1, @dateFrom)
END
```

-- **Output**

```
SELECT * FROM @dates
```

Share Edit Follow

edited Sep 27, 2022 at 12:20

Rychu
535 3 17

answered Oct 19, 2011 at 16:45

sll
60.5k 21 104 155

1 While it probably doesn't matter much in this case, I prefer that people don't get in the habit of using them when a set based solution is available because it can lead to performance problems in the wrong situation. – Abe Miessler Oct 19, 2011 at 16:51

15 @AbeMiessler - To be fair a recursive CTE is still a loop. – Martin Smith Oct 19, 2011 at 17:00

11

Here is a solution that does not require recursion, and at the same time, this table-valued function is re-usable in many queries without the need to repeat the declaration of boilerplate variables again. This is the only alternative, for those who don't want recursion.

Create this simple function:

```sql
CREATE FUNCTION [dbo].[GenerateDateRange]
(@StartDate AS DATE,
 @EndDate AS   DATE,
 @Interval AS  INT
)
RETURNS @Dates TABLE(DateValue DATE)
AS
BEGIN
    DECLARE @CUR_DATE DATE
    SET @CUR_DATE = @StartDate
    WHILE @CUR_DATE <= @EndDate BEGIN
        INSERT INTO @Dates VALUES(@CUR_DATE)
        SET @CUR_DATE = DATEADD(DAY, @Interval, @CUR_DATE)
    END
    RETURN;
END;
```

And then select by:

```sql
select *
from dbo.GenerateDateRange('2017-01-03', '2017-12-01', 1)
```

Share Edit Follow

edited Feb 11, 2017 at 8:03
Stephen Rauch ♦
46.3k 31 109 129

answered Feb 11, 2017 at 3:14
sken130
301 2 9

The difference is, this little solution does not require recursion, and at the same time, is re-usable in many queries without the need to repeat the declaration of boilerplate variables again. My solution is only an alternative, for those who don't want recursion. I never say anything about being the one. – sken130 Feb 11, 2017 at 7:11

The solution is really good and easy to implement. Thanks for this. – zulqadar idrishi Mar 11, 2021 at 13:09

8

I realize that this is an old thread, but I have to admit my dismay at the overabundance of recursive and looping solutions given here. I wonder just how many folks realize that recursion is nothing more than a very expensive loop? I understand the desire to create a Table-Valued Function, but I suggest that the following is far more efficient as it is set-based, without looping, recursion, or repeated single insert statements:

```
CREATE FUNCTION dbo.GenerateDateRange(@StartDate AS DATE, @EndDate AS DATE)
RETURNS TABLE WITH SCHEMABINDING AS
    WITH e1(n) AS (VALUES (1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),
(1),(1),(1)) AS x(n)) -- 16 records
        ,e2(n) AS (SELECT 1 FROM e1 a CROSS JOIN e1 b) -- 16^2 or 256 records
(16*16)
        ,cteTally(n) AS (SELECT ROW_NUMBER() over (ORDER BY 1) AS n FROM e2 a
CROSS JOIN e2 b) -- 16^4 or 65,536 records (256*256)
    SELECT DATEADD(DAY, n-1, @StartDate)
    FROM cteTally
    WHERE n <= DATEDIFF(DAY, @StartDate, @EndDate) + 1;
GO
```

Share Edit Follow

edited Aug 21, 2019 at 15:34

answered Aug 21, 2019 at 15:23

SQL RV
83 1 6

2

Use MVJ's F_TABLE_DATE function, it is purely awesome:

http://www.sqlteam.com/forums/topic.asp?TOPIC_ID=61519

Once you implement this just pass in start and end date and you can insert all dates between.

Share Edit Follow

answered Oct 19, 2011 at 16:41

JonH
32.4k 12 85 144

2

This is an old thread, but in case it helps anyone, this is what I use in modern versions of SQL Server that support CTE's. This also gives you the Day of the Week and it can be tweaked to give other values you may need (i.e. Quarter, Month, etc.).

```
DECLARE @StartDate datetime
DECLARE @EndDate datetime
SET @StartDate = '1/1/2020'
SET @EndDate = '12/31/2020'
DECLARE @DayTable Table(theDate date, theDayOfWeek nvarchar(50));
WITH DayTable AS (SELECT CAST(@StartDate AS DATETIME) theDate, DATENAME(dw,
@StartDate) theDayOfWeek UNION ALL SELECT DATEADD(dd, 1, theDate),
DATENAME(dw,DATEADD(dd, 1, theDate)) FROM DayTable s  WHERE DATEADD(dd, 1,
theDate) <= CAST(@EndDate AS DATETIME))
INSERT INTO @DayTable(theDate, theDayOfWeek) SELECT theDate, theDayOfWeek FROM
DayTable OPTION (MAXRECURSION 365);
SELECT * FROM @DayTable
```

Share Edit Follow

edited May 20, 2020 at 13:50

answered May 20, 2020 at 13:45

DanielG
1,606 1 12 24

1

If for some reason you can't `declare` variables, such as when using derived tables in Looker, you can go like this:

```
select
  dateadd(day, nbr - 1, convert(date, '2017-01-01')) as d
from (
  select row_number() over (order by c.object_id) as nbr from sys.columns c
) nbrs
where
  nbr - 1 <= datediff(
    day,
    convert(date, '2017-01-01'),
    convert(date, '2018-12-31')
  )
```

By the way, this is how your *date series* view could look like in LookerML:

```
view: date_series {
  derived_table: {
    sql:
      select
        dateadd(day, nbr - 1, convert(date, '2017-01-01')) as d
      from (
        select row_number() over (order by c.object_id) as nbr from sys.columns
c
      ) nbrs
      where
        nbr - 1 <= datediff(day, convert(date, '2017-01-01'), convert(date,
'2018-12-31')) ;;
  }

  dimension: date {
    primary_key: yes
    type: date
    sql: ${TABLE}.d ;;
  }
}
```

Share Edit Follow

edited Feb 11, 2018 at 19:17

answered Dec 21, 2017 at 9:07

Lars Blumberg
18k 11 86 116

1

Try Following CODE:

```
DECLARE @DateStart DATE = '2021-01-20' , @DateEnd DATE = '2021-01-29';
with Extract_Dates_CTE (MyDate) as (
    select @DateStart
    Union ALL
    select DATEADD(day, 1, MyDate)
    from Extract_Dates_CTE
    where MyDate < @DateEnd
)
select ROW_NUMBER() OVER(ORDER BY a.MyDate) AS RowDateID, a.MyDate AS
```

```
ExtractedDates
from Extract_Dates_CTE a;
```

```
DECLARE @DateStart DATE = '2021-01-20' , @DateEnd DATE = '2021-01-29';
with Extract_Dates_CTE (MyDate) as (
    select @DateStart
    Union ALL
    select DATEADD(day, 1, MyDate)
    from Extract_Dates_CTE
    where MyDate < @DateEnd
)
select ROW_NUMBER() OVER(ORDER BY a.MyDate) AS RowDateID, a.MyDate AS ExtractedDates
from Extract_Dates_CTE a;
```

%

Results | Messages

RowDateID	ExtractedDates
1	2021-01-20
2	2021-01-21
3	2021-01-22
4	2021-01-23
5	2021-01-24
6	2021-01-25
7	2021-01-26
8	2021-01-27
9	2021-01-28
10	2021-01-29

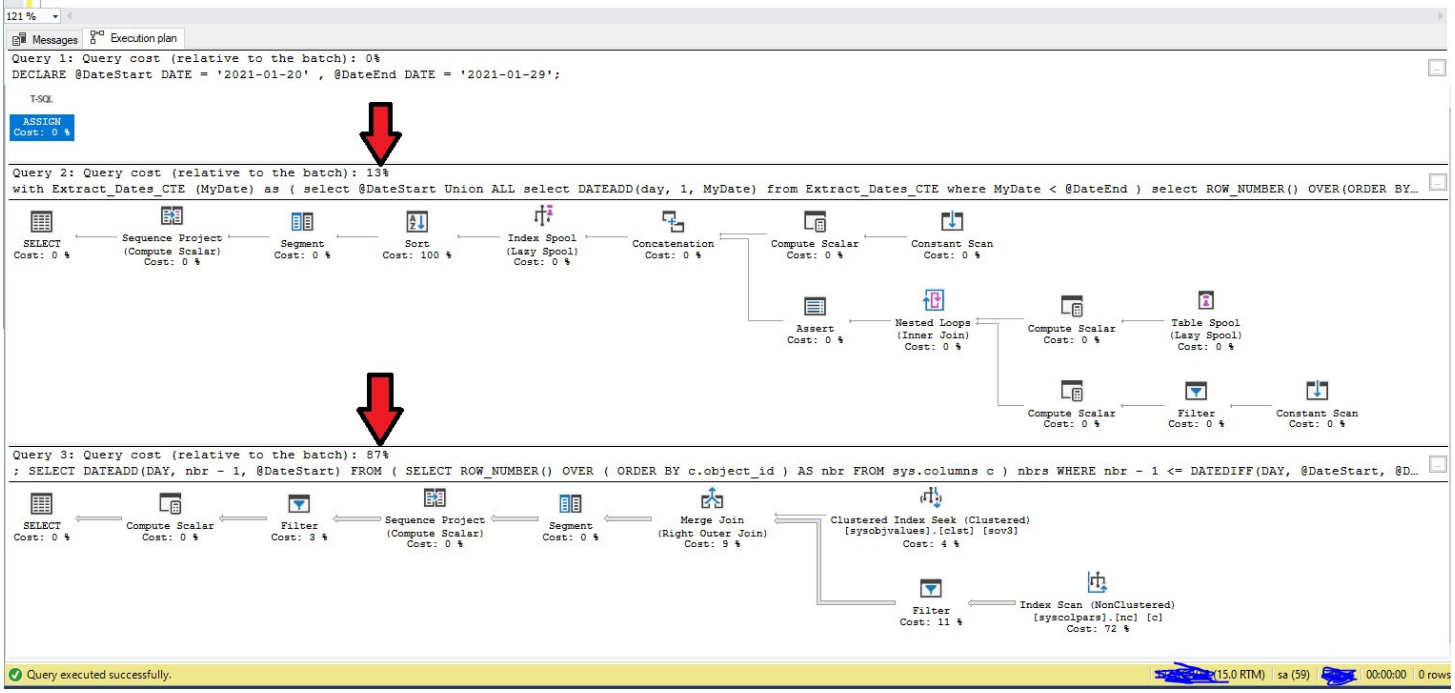
Examining the performance, I found that using the CTE method has a better performance that I have shown in the figure. For this purpose, I used two queries and displayed the performance using the SQL Server tool.

```
DECLARE @DateStart DATE = '2021-01-20' , @DateEnd DATE = '2021-01-29';
with Extract_Dates_CTE (MyDate) as (
    select @DateStart
    Union ALL
    select DATEADD(day, 1, MyDate)
    from Extract_Dates_CTE
    where MyDate < @DateEnd
)
select ROW_NUMBER() OVER(ORDER BY a.MyDate) AS RowDateID, a.MyDate AS
ExtractedDates
from Extract_Dates_CTE a;

SELECT  DATEADD(DAY, nbr - 1, @DateStart)
FROM    ( SELECT    ROW_NUMBER() OVER ( ORDER BY c.object_id ) AS nbr
          FROM      sys.columns c
        ) nbrs
WHERE   nbr - 1 <= DATEDIFF(DAY, @DateStart, @DateEnd)
```

```sql
DECLARE @DateStart DATE = '2021-01-20' , @DateEnd DATE = '2021-01-29';
with Extract_Dates_CTE (MyDate) as (
    select @DateStart
    Union ALL
    select DATEADD(day, 1, MyDate)
    from Extract_Dates_CTE
    where MyDate < @DateEnd
)
select ROW_NUMBER() OVER(ORDER BY a.MyDate) AS RowDateID, a.MyDate AS ExtractedDates
from Extract_Dates_CTE a;

SELECT  DATEADD(DAY, nbr - 1, @DateStart)
FROM    ( SELECT    ROW_NUMBER() OVER ( ORDER BY c.object_id ) AS nbr
          FROM      sys.columns c
        ) nbrs
WHERE   nbr - 1 <= DATEDIFF(DAY, @DateStart, @DateEnd)
```



Share Edit Follow

edited Jan 5, 2022 at 12:29

answered Jan 5, 2022 at 12:10

AliNajafZadeh
1,168 2 10 21

Try it with a fixed table instead of a calculated table. See my edits to my answer above. – Stuart Ainsworth Jan 7, 2022 at 1:50

0

Recursive query is a good alternative when we cannot create functions in the database.

MySQL 8+ & MariaDB 10.2.2+

```sql
WITH RECURSIVE dates AS (
  SELECT '2022-01-01' AS _day -- Your start date
  UNION ALL
  SELECT DATE_ADD(_day, INTERVAL 1 DAY)
  FROM dates
  WHERE _day < '2022-10-12'   -- Your end date
)
```

Postgres 11+

```sql
WITH RECURSIVE dates AS (
  SELECT DATE('2022-01-01') AS _day -- Your start date
  UNION ALL
```

```
    SELECT DATE(_day + INTERVAL '1 day')
    FROM dates
    WHERE _day < '2022-10-12'   -- Your end date
  )
```

To join these dates in your SELECT statement, you can use a `JOIN dates ON true` to replicate your rows for each date in your date range.

```
[WITH statement according to your database]
SELECT col1, col2, _day
FROM my_table
JOIN dates ON true
```

Share Edit Follow

answered Nov 14, 2022 at 19:35

Octávio Lage
21 7

0

Using @Abe Miesler's answer, for other's convenience I built it into a TVF for SQL Server 2008 onwards. It may help others - I had to find a way to include the CTE inside the TVF!

```
    --Generate a range of dates with interval option, courtesy of Abe Miessler
for the core query here!
ALTER FUNCTION [dbo].[DateRange]
(@startDate AS DATE,
 @EndDate AS   DATE,
 @interval AS  INT
)
RETURNS @Dates TABLE(dateValue DATE)
AS
     BEGIN
         WITH Dates
              AS (
              SELECT [Date] = CONVERT( DATETIME, @startDate)
              UNION ALL
              SELECT [Date] = DATEADD(DAY, ISNULL(@interval, 1), [Date])
              FROM Dates
              WHERE Date < @EndDate)
              INSERT INTO @Dates
                     SELECT [Date]
                     FROM Dates
                     OPTION(MAXRECURSION 900);
         RETURN;
     END;
```

Share Edit Follow

answered Aug 12, 2016 at 10:07

Richard Griffiths
678 1 11 23

-1

```
CREATE table #ProductSales (ProjectID Int, ProjectName varchar(100),
TotalBillableFees Money, StartDate Date, EndDate Date, DataDate Date)

  Insert into #ProductSales
  Values
  (373104,'Product Sales - Flex Creation Test',40000.00,'2019-04-01','2020-06-
```

```
01','2019-08-01'),
  (375111,'Product Sales - SMART',40000.00,'2019-04-01','2019-09-01','2019-08-
01')

  ;WITH Dates AS (
        SELECT ProjectiD

,Convert(decimal(10,2),TotalBillableFees/IIF(DATEDIFF(MONTH,StartDate,EndDate)=0,1
AS BillableFeesPerMonths,EndDate
         ,[Date] = CONVERT(DATETIME,EOMONTH(StartDate))
         FROM #ProductSales
        UNION ALL SELECT ProjectiD,BillableFeesPerMonths,EndDate,
         [Date] = DATEADD(MONTH, 1, [Date])
        FROM
         Dates
        WHERE
         Date < EOMONTH(EndDate)
) SELECT ProjectID,BillableFeesPerMonths,
 CAST([Date] as Date) Date
FROM
 Dates
 OPTION (MAXRECURSION 45)
```

Share Edit Follow

edited Aug 8, 2019 at 6:16
realr
3,493 6 22 34

answered Aug 8, 2019 at 0:50
Mukehp
11 2

-1

```
Declare @StartDate datetime = '2015-01-01'
Declare @EndDate datetime = '2016-12-01'
declare @DaysInMonth int
declare @tempDateRange Table
(
DateFrom datetime,
DateThru datetime
);

While @StartDate<=@EndDate
begin
    SET
@DaysInMonth=DAY(DATEADD(DD,-1,DATEADD(MM,DATEDIFF(MM,-1,@StartDate),0)))

    IF DAY(@StartDate)=1
        SET @EndDate=DATEADD(DAY,14,@StartDate)
    ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=30
        SET @EndDate=DATEADD(DAY,14,@StartDate)
    ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=31
        SET @EndDate=DATEADD(DAY,15,@StartDate)
    ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=28
        SET @EndDate=DATEADD(DAY,12,@StartDate)
    ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=29
        SET @EndDate=DATEADD(DAY,13,@StartDate)

    INSERT INTO @tempDateRange (DateFrom,DateThru)
    VALUES
     (
        @StartDate,
        @EndDate
     )

    SET @StartDate=DATEADD(DAY,1,@EndDate)

    IF @EndDate< '2016-12-31'
```

```
      IF DAY(@StartDate)=1
         SET @EndDate=DATEADD(DAY,14,@StartDate)
      ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=30
         SET @EndDate=DATEADD(DAY,14,@StartDate)
      ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=31
         SET @EndDate=DATEADD(DAY,15,@StartDate)
      ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=28
         SET @EndDate=DATEADD(DAY,12,@StartDate)
      ELSE IF DAY(@StartDate)=16 AND @DaysInMonth=29
         SET @EndDate=DATEADD(DAY,13,@StartDate)
end ;

select * from @tempDateRange

++++++++++++++++++++++++++++
Result:
DateFrom |DateThru
```

Share Edit Follow

edited Jan 13, 2017 at 19:25
Donald Duck
7,999 22 73 93

answered Jan 13, 2017 at 16:56
SanH
19 2

While this code may answer the question, providing additional context regarding why and/or how this code answers the question improves its long-term value. – Donald Duck Jan 13, 2017 at 17:17