Using sql function generate_series() in redshift

Asked 9 years, 8 months ago Modified 1 month ago Viewed 46k times Part of AWS Collective

26

I'd like to use the generate series function in redshift, but have not been successful.

The redshift documentation says it's not supported. The following code does work:

```
select *
from generate_series(1,10,1)
```

outputs:

```
1
2
3
...
10
```

I'd like to do the same with dates. I've tried a number of variations, including:

```
select *
from generate_series(date('2008-10-01'),date('2008-10-10 00:00:00'),1)
```

kicks out:

```
 ERROR: function generate_series(date, date, integer) does not exist
 Hint: No function matches the given name and argument types.
 You may need to add explicit type casts. [SQL State=42883]
```

Also tried:

```
select *
from generate_series('2008-10-01 00:00:00'::timestamp,
'2008-10-10 00:00:00'::timestamp,'1 day')
```

And tried:

```
select *
from generate_series(cast('2008-10-01 00:00:00' as datetime),
cast('2008-10-10 00:00:00' as datetime),'1 day')
```

both kick out:

```
ERROR: function generate_series(timestamp without time zone, timestamp without
time zone, "unknown") does not exist
Hint: No function matches the given name and argument types.
You may need to add explicit type casts. [SQL State=42883]
```

If not looks like I'll use this code from another post:

```
SELECT to_char(DATE '2008-01-01'
+ (interval '1 month' * generate_series(0,57)), 'YYYY-MM-DD') AS ym
```

PostgreSQL generate_series() with SQL function as arguments

sql amazon-redshift generate-series

Share Edit Follow

edited May 23, 2017 at 10:31
Community Bot
1 1

asked Jun 24, 2013 at 18:20
Elm
1,355 6 22 33

2 Run `SELECT version()` against your database to retrieve your version of Postgres. – Erwin Brandstetter Jun 24, 2013 at 19:05

For a workaround see: stackoverflow.com/a/34167753/3019685 – systemjack Jun 22, 2016 at 17:37

I have posted a workaround in a similar question here stackoverflow.com/questions/22759980/… – AlexYes Jun 22, 2017 at 10:45

10 Answers

Sorted by: Highest score (default)

25

Amazon Redshift seems to be based on PostgreSQL 8.0.2. The timestamp arguments to generate_series() were added in 8.4.

Something like this, which sidesteps that problem, *might* work in Redshift.

```
SELECT current_date + (n || ' days')::interval
from generate_series (1, 30) n
```

It works in PostgreSQL 8.3, which is the earliest version I can test. It's documented in 8.0.26.

Later . . .

It seems that generate_series() is unsupported in Redshift. But given that you've verified that `select * from generate_series(1,10,1)` *does* work, the syntax above at least gives you a fighting chance. (Although the interval data type is also documented as being unsupported on Redshift.)

Still later . . .

You could also create a table of integers.

```
create table integers (
  n integer primary key
);
```

Populate it however you like. You might be able to use generate_series() locally, dump the table, and load it on Redshift. (I don't know; I don't use Redshift.)

Anyway, you can do simple date arithmetic with that table without referring directly to generate_series() or to interval data types.

```
select (current_date + n)
from integers
where n < 31;
```

That works in 8.3, at least.

Share Edit Follow

edited Jun 24, 2013 at 19:32

answered Jun 24, 2013 at 19:03

Mike Sherrill 'Cat Recall'
89.8k 17 120 179

21

Using Redshift today, you can generate a range of dates by using datetime functions and feeding in a number table.

```
select (getdate()::date - generate_series)::date from generate_series(1,30,1)
```

Generates this for me

```
date
2015-11-06
2015-11-05
2015-11-04
2015-11-03
2015-11-02
2015-11-01
2015-10-31
2015-10-30
2015-10-29
2015-10-28
2015-10-27
2015-10-26
2015-10-25
2015-10-24
2015-10-23
2015-10-22
2015-10-21
2015-10-20
2015-10-19
2015-10-18
2015-10-17
2015-10-16
2015-10-15
2015-10-14
2015-10-13
2015-10-12
2015-10-11
2015-10-10
2015-10-09
```

```
2015-10-08
```

Share Edit Follow

answered Nov 7, 2015 at 0:51
Gabe Brown
1,418 3 17 22

8 while this generates a series, I have found no way (CTE, subquery, or inserting into a table or temporary table to join it to another table for filtering). – cfeduke Jan 25, 2016 at 18:06

4 @cfeduke you could wrap this into a temp table as `with series AS ( select * from generate_series(1,10,1) ) select * from series` – blotto Mar 31, 2016 at 22:54

1 @blotto - thank you! Your comment and the last example in the question solved the problem for me. – GeekyDeaks Apr 20, 2016 at 8:32

16 The query, `with series AS ( select * from generate_series(1,10,1) ) select * from series` , does work. However, as soon as join it with another table, `with series AS ( select * from generate_series(1,10,1) ) select * from series join mytable on true` , then I still get the error `Function "generate_series(integer,integer,integer)" not supported.` – Slobodan Pejic Apr 24, 2017 at 16:21

1 I'm having the same result as @SlobodanPejic: bizarrely: -this works perfectly to generate a date range CTE, -and I can run that code in Redshift and get the expected date range, -but if I try to join onto that date range elsewhere, I get `Function "generate_series(bigint,bigint)" not supported.` – JCMontalbano Jul 17, 2021 at 5:20

13

The `generate_series()` function is not fully supported by Redshift. See the **Unsupported PostgreSQL functions** section of the developer guide.

UPDATE

generate_series is working with Redshift now.

```
SELECT CURRENT_DATE::TIMESTAMP  - (i * interval '1 day') as date_datetime
FROM generate_series(1,31) i
ORDER BY 1
```

This will generate last 30 days date

Ref: generate_series function in Amazon Redshift

Share Edit Follow

edited Dec 5, 2018 at 12:39

answered Apr 1, 2014 at 9:54
DJo
2,093 2 29 46

8 It appears that `generate_series()` works occasionally, but reliably fails if and when used in queries joining to actual tables. My guess is that these example queries are running on the leader node and succeeding, but fail when running on other nodes. From your link to the unsupported functions docs: "some unsupported functions will not return an error when run on the leader node" – ryantuck Feb 21, 2019 at 17:27

As of writing this, `generate_series()` on our instance of Redshift (1.0.33426) could not be used to, for example, create a table:

4

```
# select generate_series(1,100,1);
1
2
...

# create table normal_series as select generate_series(1,100,1);
INFO: Function "generate_series(integer, integer, integer) not supported.
ERROR: Specified types or functions (one per INFO message) not supported on
Redshift tables.
```

However, `with recursive` works:

```
# create table recursive_series as with recursive t(n) as (select 1::integer
union all select n+1 from t where n < 100) select n from t;
SELECT

-- modify as desired, here is a date series:
# select getdate()::date + n from recursive_series;
2021-12-18
2021-12-19
...
```

Share Edit Follow

answered Dec 17, 2021 at 9:52

crdb
41 1

This is a really nice answer - for reference, recursive CTE's are supported in Redshift since about a year ago: aws.amazon.com/about-aws/whats-new/2021/04/... – Karl Anka Feb 10, 2022 at 10:31

2

I needed to do something similar, but with 5 minutes intervals over 7 days. So here's a CTE based hack (ugly but not too verbose)

```
INSERT INTO five_min_periods
WITH
periods  AS (select 0 as num UNION select 1 as num UNION select 2 UNION select
3 UNION select 4 UNION select 5 UNION select 6 UNION select 7 UNION select 8
UNION select 9 UNION select 10 UNION select 11),
hours    AS (select num from periods UNION ALL select num + 12 from periods),
days     AS (select num from periods where num <= 6),
rightnow AS (select CAST( TO_CHAR(GETDATE(), 'yyyy-mm-dd hh24') || ':' ||
trim(TO_CHAR((ROUND((DATEPART (MINUTE, GETDATE()) / 5), 1) * 5 ),'09')) AS
TIMESTAMP) as start)
select
  ROW_NUMBER() OVER(ORDER BY d.num DESC, h.num DESC, p.num DESC) as idx
  , DATEADD(minutes, -p.num * 5, DATEADD( hours, -h.num, DATEADD( days, -d.num,
n.start ) ) ) AS period_date
from days d, hours h, periods p, rightnow n
```

Should be able to extend this to other generation schemes. The trick here is using the Cartesian product join (i.e. no JOIN/WHERE clause) to multiply the hand-crafted CTE's to produce the necessary increments and apply to an anchor date.

Share Edit Follow

edited Sep 6, 2017 at 11:29

answered Sep 6, 2017 at 10:33

El Jeffe
21 3

1

Redshift's generate_series() function is a leader node only function and as such you cannot use it for downstream processing on the compute nodes. This can be replace by a recursive CTE (or keep a "dates" table on your database). I have an example of such in a recent answer:

Cross join Redshift with sequence of dates

One caution I like to give in answers like this is to be careful with inequality joins (or cross joins or any under-qualified joins) when working with VERY LARGE tables which can happen often in Redshift. If you are joining with a moderate Redshift table of say 1M rows then things will be fine. But if you are doing this on a table of 1B rows then the data explosion will likely cause massive performance issues as the query spills to disk.

I've written a couple of white papers on how to write this type of query in a data space sensitive way. This issue of massive intermediate results is not unique to Redshift and I first developed my approach solving a client's HIVE query issue. "First rule of writing SQL for Big Data - don't make more"

Share Edit Follow

answered Dec 17, 2021 at 16:43
Bill Weiner
7,672 2 5 17

0

Per the comments of @Ryan Tuck and @Slobodan Pejic `generate_series()` does not work on Redshift when joining to another table.

The workaround I used was to write out every value in the series in the query:

```
SELECT
'2019-01-01'::date AS date_month
UNION ALL
SELECT
'2019-02-01'::date AS date_month
```

Using a Python function like this:

```
import arrow

def generate_date_series(start, end):
    start = arrow.get(start)
    end = arrow.get(end)

    months = list(
        f"SELECT '{month.format('YYYY-MM-DD')}'::date AS date_month"
        for month in arrow.Arrow.range('month', start, end)
    )

    return "\nUNION ALL\n".join(months)
```

Share Edit Follow

answered Sep 18, 2019 at 15:20
Derek Hill

5,737 2 53 72

0

perhaps not as elegant as other solutions, but here's how I did it:

```
drop table if exists #dates;
create temporary table #dates as
with recursive cte(val_date) as
    (select
        cast('2020-07-01' as date) as val_date
    union all
    select
        cast(dateadd(day, 1, val_date) as date) as val_date
    from
        cte
    where
        val_date <= getdate()
    )
select
    val_date as yyyymmdd
from
    cte
order by
    val_date
;
```

Share Edit Follow

edited Jun 27, 2022 at 22:08

answered Jun 27, 2022 at 21:02

Tom Renish
388 4 7

0

For five minute buckets i would do the following:

```
select date_trunc('minute', getdate()) - (i || ' minutes')::interval
from generate_series(0, 60*5-1, 5) as i
```

You could replace 5 by any given interval, and 60 by the number of rows you want.

Share Edit Follow

answered Aug 18, 2022 at 21:19

Guilherme Santos
1

-1

```
SELECT CURRENT_DATE::TIMESTAMP - (i * interval '1 day') as date_datetime
FROM generate_series(1,(select datediff(day,'01-Jan-2021',now()::date))) i
ORDER BY 1
```

Share Edit Follow

edited Jan 13 at 10:12

Sydney_dev
833 2 12 23

answered Jan 5 at 22:39

Raghav Sharma
1

1 Your answer could be improved with additional supporting information. Please edit to add further details, such as citations or documentation, so that others can confirm that your answer is correct. You can find more information on how

to write good answers in the help center. – Community Bot Jan 6 at 17:58