SQL TIPS AND TRICKS

PART 15

Building Calendar (50 Years) Dimension Table IN MYSQL From Scratch

MAYURI DANDEKAR

To build Calendar Table, We will be using Recursive CTE.
A recursive common table expression is one having a subquery that refers to its own name.

Example –

```
WITH RECURSIVE cte (n) AS
(
SELECT 1                    (-- returns initial row set --)
UNION ALL
SELECT n + 1 FROM cte WHERE n < 5                    (-- returns additional row set --)
)
SELECT * FROM cte;
```

It is important for recursive CTEs that the recursive SELECT part include a condition to terminate recursion. As a development technique to guard against a runaway recursive CTE, you can force termination by placing a limit on execution time. By default, cte_max_recursion_depth has a value of 1000, causing the CTE to terminate when it recurses past 1000 levels. Applications can change the session value to adjust for their requirements

Example—

```
SET SESSION cte_max_recursion_depth = 1000000;
```

STEP 1–

Build a hardcoded 1 row table from Date functions.

Increase or Decrease the no. of columns as per the requirement.

[check the date functions as per your database]

```
3 •   SELECT CAST('2000-01-01' as date) as cal_date,
4     (YEAR ('2000-01-01')) as cal_year,
5     (DAYOFYEAR ('2000-01-01')) as cal_year_day,
6     (QUARTER ('2000-01-01')) as cal_quarter,
7     (MONTH ('2000-01-01')) as cal_month,
8     (MONTHNAME ('2000-01-01')) as cal_month_name,
9     (DAY ('2000-01-01')) as cal_month_day,
10    (WEEK ('2000-01-01')) as cal_week,
11    (DAYOFWEEK ('2000-01-01')) as cal_weekday,
12    (DAYNAME ('2000-01-01')) as cal_dayname;
13
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cal_date	cal_year	cal_year_day	cal_quarter	cal_month	cal_month_name	cal_month_day	cal_week	cal_weekday	cal_dayname
2000-01-01	2000	1	1	1	January	1	0	7	Saturday

STEP 2–

After creating 1date, we need to build remaining dates till 2050.

To get those dates, we will be using RECURSIVE CTE.
Recursive CTE works in 2 parts—
1st part SELECT statement returns the initial row set i.e STEP 1
2nd part SELECT statement returns additional row sets by adding 1
Both SELECT statements are combined by UNION ALL.

WHERE clause shows the condition till when to print the row sets.
Finally select everything from the cte.

```
14  WITH RECURSIVE cte AS(
15    SELECT 1 AS id
16    UNION ALL
17    SELECT id + 1 AS id
18    FROM cte
19    WHERE id < 10
20    )
21  SELECT * FROM cte;
```

Result Grid | Filter Rows:

id
1
2
3
4
5
6
7
8
9
10

STEP 3–

TO get the actual result, combine STEP 1 & STEP 2 query

```
-- recursive cte.
WITH RECURSIVE cte AS(
SELECT CAST('2000-01-01' as date) as cal_date,
(YEAR ('2000-01-01')) as cal_year,
(DAYOFYEAR ('2000-01-01')) as cal_year_day,
(QUARTER ('2000-01-01')) as cal_quarter,
(MONTH ('2000-01-01')) as cal_month,
(MONTHNAME ('2000-01-01')) as cal_month_name,
(DAY ('2000-01-01')) as cal_month_day,
(WEEK ('2000-01-01')) as cal_week,
(DAYOFWEEK ('2000-01-01')) as cal_weekday,
(DAYNAME ('2000-01-01')) as cal_dayname
UNION ALL
SELECT DATE_ADD(cal_date, INTERVAL 1 DAY) AS cal_date,
(YEAR (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_year,
(DAYOFYEAR (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_year_day,
(QUARTER (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_quarter,
(MONTH (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_month,
(MONTHNAME (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_month_name,
(DAY (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_month_day,
(WEEK (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_week,
(DAYOFWEEK (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_weekday,
(DAYNAME (DATE_ADD(cal_date, INTERVAL 1 DAY))) as cal_dayname
FROM cte
WHERE cal_date <= cast( '2050-12-30' AS date)
)
```

STEP 4–

To print the final result, use row_number() to get the primary key as Id and display all the required columns or * to select everything.
By default, maximum recursion value is 1000. Our table contains more than 1000 rows so set the recursion depth value as per requirement.

```
49    SELECT  /*+ SET_VAR(cte_max_recursion_depth = 20000) */
50    ROW_NUMBER() OVER(ORDER BY cal_date ASC) AS id,
51    cal_date, cal_year, cal_year_day, cal_quarter, cal_month, cal_month_name, cal_month_day, cal_week, cal_weekday, cal_dayname
52    FROM cte
53    ;
54
```

```
24 ● ⊕ WITH RECURSIVE cte AS(
49      SELECT  /*+ SET_VAR(cte_max_recursion_depth = 20000) */
50      ROW_NUMBER() OVER(ORDER BY cal_date ASC) AS id,
51      cal_date, cal_year, cal_year_day, cal_quarter, cal_month, cal_month_name, cal_month_day, cal_week, cal_weekday, cal_dayname
52      FROM cte ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

id	cal_date	cal_year	cal_year_day	cal_quarter	cal_month	cal_month_name	cal_month_day	cal_week	cal_weekday	cal_dayname
1	2000-01-01	2000	1	1	1	January	1	0	7	Saturday
2	2000-01-02	2000	2	1	1	January	2	1	1	Sunday
3	2000-01-03	2000	3	1	1	January	3	1	2	Monday
4	2000-01-04	2000	4	1	1	January	4	1	3	Tuesday
5	2000-01-05	2000	5	1	1	January	5	1	4	Wednesday
6	2000-01-06	2000	6	1	1	January	6	1	5	Thursday
7	2000-01-07	2000	7	1	1	January	7	1	6	Friday
8	2000-01-08	2000	8	1	1	January	8	1	7	Saturday
9	2000-01-09	2000	9	1	1	January	9	2	1	Sunday
10	2000-01-10	2000	10	1	1	January	10	2	2	Monday
11	2000-01-11	2000	11	1	1	January	11	2	3	Tuesday
12	2000-01-12	2000	12	1	1	January	12	2	4	Wednesday
13	2000-01-13	2000	13	1	1	January	13	2	5	Thursday
14	2000-01-14	2000	14	1	1	January	14	2	6	Friday
15	2000-01-15	2000	15	1	1	January	15	2	7	Saturday
16	2000-01-16	2000	16	1	1	January	16	3	1	Sunday
17	2000-01-17	2000	17	1	1	January	17	3	2	Monday
18	2000-01-18	2000	18	1	1	January	18	3	3	Tuesday
19	2000-01-19	2000	19	1	1	January	19	3	4	Wednesday
20	2000-01-20	2000	20	1	1	January	20	3	5	Thursday
21	2000-01-21	2000	21	1	1	January	21	3	6	Friday
22	2000-01-22	2000	22	1	1	January	22	3	7	Saturday

```
51      cal_date, cal_year, cal_year_day, cal_quarter, cal_month, cal_month_name, cal_month_day, cal_week, cal_weekday,
52      FROM cte ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

id	cal_date	cal_year	cal_year_day	cal_quarter	cal_month	cal_month_name	cal_month_day	cal_week	cal_weekday	cal_dayname
18607	2050-12-10	2050	344	4	12	December	10	49	7	Saturday
18608	2050-12-11	2050	345	4	12	December	11	50	1	Sunday
18609	2050-12-12	2050	346	4	12	December	12	50	2	Monday
18610	2050-12-13	2050	347	4	12	December	13	50	3	Tuesday
18611	2050-12-14	2050	348	4	12	December	14	50	4	Wednesday
18612	2050-12-15	2050	349	4	12	December	15	50	5	Thursday
18613	2050-12-16	2050	350	4	12	December	16	50	6	Friday
18614	2050-12-17	2050	351	4	12	December	17	50	7	Saturday
18615	2050-12-18	2050	352	4	12	December	18	51	1	Sunday
18616	2050-12-19	2050	353	4	12	December	19	51	2	Monday
18617	2050-12-20	2050	354	4	12	December	20	51	3	Tuesday
18618	2050-12-21	2050	355	4	12	December	21	51	4	Wednesday
18619	2050-12-22	2050	356	4	12	December	22	51	5	Thursday
18620	2050-12-23	2050	357	4	12	December	23	51	6	Friday
18621	2050-12-24	2050	358	4	12	December	24	51	7	Saturday
18622	2050-12-25	2050	359	4	12	December	25	52	1	Sunday
18623	2050-12-26	2050	360	4	12	December	26	52	2	Monday
18624	2050-12-27	2050	361	4	12	December	27	52	3	Tuesday
18625	2050-12-28	2050	362	4	12	December	28	52	4	Wednesday
18626	2050-12-29	2050	363	4	12	December	29	52	5	Thursday
18627	2050-12-30	2050	364	4	12	December	30	52	6	Friday
18628	2050-12-31	2050	365	4	12	December	31	52	7	Saturday

THANK YOU

MAYURI DANDEKAR