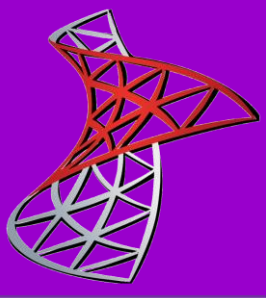


عنوان مقاله: بهبود T-SQL Windowing در SQL Server 2022

نویسنده مقاله: تیم فنی نیک‌آموز

تاریخ انتشار: شهریورماه ۱۴۰۱

منبع:-https://nikamooz.com/windowing-improvements-sql-server

مایکروسافت اخیرا اولین پیش نمایش عمومی SQL Server 2022 را منتشر کرده است. این نسخه بهبودهایی در T-SQL داشته است. در این مقاله من بر روی بهبودهای Windowing و NULL treatment تمرکز می‌کنم. این عبارت‌های جدید شامل عبارت Window و عبارت windowing NULL treatment است.

من از نمونه پایگاه داده TSQLV6 در مثال‌های این مقاله استفاده خواهم کرد. این نمونه پایگاه داده را می‌توانید از لینک زیر دانلود کنید.

http://tsql.lucient.com/SampleDatabases/TSQLV6.zip

عبارت WINDOW

عبارت WINDOW بخشی از استاندارد ISO/IEC SQL است. این عبارت به شما این امکان را می‌دهد تا قسمت‌هایی از یک WINDOW را نام‌گذاری کنید (یا حتی کل آن را) و سپس از این نام در عبارت OVER در توابع WINDOW در کوئری خود استفاده کنید. این عبارت به شما اجازه می‌دهد تا با اجتناب از تکرار قسمت‌های یکسان window specifications خود، کدتان را کوتاه‌تر کنید. این عبارت اکنون در Azure SQL Database و SQL Server 2022 موجود است، در صورتی که از سطح compatibility، ۱۶۰ یا بالاتر در پایگاه داده استفاده کنید.

عبارت WINDOW بین عبارت‌های HAVING و ORDER BY در کوئری قرار دارد:

SELECT
FROM
WHERE
GROUP BY
HAVING
WINDOW
ORDER BY

عبارت WINDOW دارای سینتکس زیر است:

```
WINDOW window_name AS ( [ reference_window_name ]
                [ <window partition clause> ]
                [ <window order clause> ]
                [ <window frame clause> ] )
```

به عنوان یک مثال که عبارت WINDOW می‌تواند در کوتاه کردن کد شما مفید باشد، کوئری زیر را در نظر بگیرید:

```
USE TSQLV6;

SELECT orderid, custid, orderdate, qty, val,
  SUM(qty) OVER (PARTITION BY custid
            ORDER BY orderdate, orderid
            ROWS UNBOUNDED PRECEDING) AS runsumqty,
  SUM(val) OVER (PARTITION BY custid
            ORDER BY orderdate, orderid
            ROWS UNBOUNDED PRECEDING) AS runsumval
FROM Sales.OrderValues
WHERE custid IN (1, 2)
ORDER BY custid, orderdate, orderid;
```

این کوئری خروجی زیر را تولید می‌کند:

```
orderid     custid      orderdate       qty         val       runsumqty   runsumval
--------    ----------- ----------      ----------  -------   ----------  ----------
10643       1           2021-08-25      38          814.50    38          814.50
10692       1           2021-10-03      20          878.00    58          1692.50
10702       1           2021-10-13      21          330.00    79          2022.50
10835       1           2022-01-15      17          845.80    96          2868.30
10952       1           2022-03-16      18          471.20    114         3339.50
11011       1           2022-04-09      60          933.50    174         4273.00
10308       2           2020-09-18      6           88.80     6           88.80
10625       2           2021-08-08      18          479.75    24          568.55
10759       2           2021-11-28      10          320.00    34          888.55
10926       2           2022-03-04      29          514.40    63          1402.95
```

در این کوئری می‌توانید دو تابع window را با استفاده از window specifications یکسان، شامل partitioning، ordering و framing مشاهده کنید. برای کوتاه کردن کوئری، می‌توانید از عبارت WINDOW برای

نام‌گذاری window specifications با هر سه آیتم استفاده کنید، مثلاً با W نام گزاری کنید، و سپس OVER W را در هر دو تابع window مشخص کنید. به صورت زیر خواهد بود:

```
SELECT orderid, custid, orderdate, qty, val,
  SUM(qty) OVER W AS runsumqty,
  SUM(val) OVER W AS runsumval
FROM Sales.OrderValues
WHERE custid IN (1, 2)
WINDOW W AS ( PARTITION BY custid
              ORDER BY orderdate, orderid
              ROWS UNBOUNDED PRECEDING )
ORDER BY custid, orderdate, orderid;
```

همان طور که می‌بینید، زمانی که نام window نمایانگر تمام window specification هایی است که مورد نیاز شما است (نه فقط بخشی از آن)، نام window را دقیقاً بعد از عبارت OVER بدون پرانتز اضافه کنید.

ممکن است متوجه شده باشید که در سینتکس عبارت WINDOW، window name specification می‌تواند ارجاع به نام WINDOW دیگری داشته باشد. این مورد به ویژه زمانی مفید است که کوئری شما دارای توابع مختلف WINDOW با window specification های مختلف باشد و یک window specification همانند بخشی از یک WINDOW دیگر باشد. کوئری زیر را به عنوان نمونه در نظر بگیرید:

```
SELECT orderid, custid, orderdate, qty, val,
  ROW_NUMBER() OVER( PARTITION BY custid
                     ORDER BY orderdate, orderid ) AS ordernum,
  MAX(orderdate) OVER( PARTITION BY custid ) AS maxorderdate,
  SUM(qty) OVER( PARTITION BY custid
                 ORDER BY orderdate, orderid
                 ROWS UNBOUNDED PRECEDING ) AS runsumqty,
  SUM(val) OVER( PARTITION BY custid
                 ORDER BY orderdate, orderid
                 ROWS UNBOUNDED PRECEDING ) AS runsumval
FROM Sales.OrderValues
WHERE custid IN (1, 2)
ORDER BY custid, orderdate, orderid;
```

این کوئری خروجی زیر را تولید می‌کند:

orderid	custid	orderdate	qty	val	ordernum	maxorderdate	runsumqty	runsumval
10643	1	2021-08-25	38	814.50	1	2022-04-09	38	814.50
10692	1	2021-10-03	20	878.00	2	2022-04-09	58	1692.50
10702	1	2021-10-13	21	330.00	3	2022-04-09	79	2022.50
10835	1	2022-01-15	17	845.80	4	2022-04-09	96	2868.30
10952	1	2022-03-16	18	471.20	5	2022-04-09	114	3339.50
11011	1	2022-04-09	60	933.50	6	2022-04-09	174	4273.00
10308	2	2020-09-18	6	88.80	1	2022-03-04	6	88.80
10625	2	2021-08-08	18	479.75	2	2022-03-04	24	568.55
10759	2	2021-11-28	10	320.00	3	2022-03-04	34	888.55
10926	2	2022-03-04	29	514.40	4	2022-03-04	63	1402.95

به موارد زیر توجه کنید:

- window specification برای تابع MAX فقط یک عبارت window partition دارد.
- window specification برای تابع ROW_NUMBER دارای یک عبارت window partition است که همانند تابع MAX است، به علاوه یک عبارت window order.
- تابع SUM دارای window partition و عبارت‌های order مشابه تابع ROW_NUMBER است، به علاوه یک عبارت window frame.

قابلیت بازگشتی سینتکس عبارت WINDOW به شما این امکان را می‌دهد کد کوئری را کوتاه کنید، مانند زیر:

```
SELECT orderid, custid, orderdate, qty, val,
  ROW_NUMBER() OVER PO AS ordernum,
  MAX(orderdate) OVER P AS maxorderdate,
  SUM(qty) OVER POF AS runsumqty,
  SUM(val) OVER POF AS runsumval
FROM Sales.OrderValues
WHERE custid IN (1, 2)
WINDOW P AS ( PARTITION BY custid ),
      PO AS ( P ORDER BY orderdate, orderid ),
      POF AS ( PO ROWS UNBOUNDED PRECEDING )
ORDER BY custid, orderdate, orderid;
```

window name definition ها در عبارت WINDOW زیاد نیست. به عنوان مثال، کد زیر به لحاظ سینتکس معتبر است و همان معنای کوئری بالا را دارد:

```
SELECT orderid, custid, orderdate, qty, val,
  ROW_NUMBER() OVER PO AS ordernum,
  MAX(orderdate) OVER P AS maxorderdate,
  SUM(qty) OVER POF AS runsumqty,
  SUM(val) OVER POF AS runsumval
FROM Sales.OrderValues
WHERE custid IN (1, 2)
WINDOW POF AS ( PO ROWS UNBOUNDED PRECEDING ),
      PO AS ( P ORDER BY orderdate, orderid ),
      P AS ( PARTITION BY custid )
ORDER BY custid, orderdate, orderid;
```

توجه داشته باشید، هر چند، شما نمی‌توانید از چندین window name reference در یک window name specification استفاده کنید. شما فقط به یک window name reference محدود شده‌اید. به عنوان مثال، کد زیر به این دلیل معتبر نیست:

```
SELECT orderid, custid, orderdate, qty, val,
  SUM(qty) OVER ( P O F ) AS runsumqty,
  SUM(val) OVER ( P O F ) AS runsumval
FROM Sales.OrderValues
WHERE custid IN (1, 2)
WINDOW P AS ( PARTITION BY custid ),
      O AS ( ORDER BY orderdate, orderid ),
      F AS ( ROWS UNBOUNDED PRECEDING )
ORDER BY custid, orderdate, orderid;
```

این کد خطای زیر را ایجاد می‌کند:

```
Msg 102, Level 15, State 1, Line 106
Incorrect syntax near 'O'.
```

شما مجاز به ترکیب یک window name و عناصر بیشتر، در window specification هستید، مانند زیر:

```
SELECT orderid, custid, orderdate, qty, val,
  ROW_NUMBER() OVER ( P ORDER BY orderdate, orderid ) AS ordernum,
  MAX(orderdate) OVER P AS maxorderdate
FROM Sales.OrderValues
WHERE custid IN (1, 2)
WINDOW P AS ( PARTITION BY custid )
ORDER BY custid, orderdate, orderid;
```

این کوئری خروجی زیر را تولید می‌کند

```
orderid     custid   orderdate       qty         val     ordernum    maxorderdate
--------    --------  ----------      --------    -------  ----------- ------------
10643       1         2021-08-25      38          814.50   1           2022-04-09
10692       1         2021-10-03      20          878.00   2           2022-04-09
10702       1         2021-10-13      21          330.00   3           2022-04-09
10835       1         2022-01-15      17          845.80   4           2022-04-09
10952       1         2022-03-16      18          471.20   5           2022-04-09
11011       1         2022-04-09      60          933.50   6           2022-04-09
10308       2         2020-09-18      6           88.80    1           2022-03-04
10625       2         2021-08-08      18          479.75   2           2022-03-04
10759       2         2021-11-28      10          320.00   3           2022-03-04
10926       2         2022-03-04      29          514.40   4           2022-03-04
```

همان طور که قبلاً اشاره کردم، وقتی window name تمام window specification را نشان می‌دهد، مانند تابع MAX در این کوئری، نام window را دقیقا بعد از عبارت OVER بدون پرانتز مشخص می‌کنید. اما هنگامی که نام window تنها بخشی از window name است، مانند تابع ROW_NUMBER در این کوئری، نام window را مشخص می‌کنید و بقیه عناصر window را در داخل پرانتز قرار می‌دهید.

در حال حاضر، شما می‌دانید که مجاز به تعریف بازگشتی یک نام window بر اساس دیگری هستید. با این حال، در صورتی که این روال بازگشتی واضح نباشد، ارجاعات سیکلی مجاز نیستند. به عنوان مثال، کوئری زیر معتبر است زیرا تعاریف نام پنجره واضح هستند و سیکلی نیستند:

```
SELECT 'This is valid'
WINDOW W1 AS (), W2 AS (W1), W3 AS (W2);
```

این کوئری خروجی زیر را تولید می‌کند:

```
-------------
This is valid
```

با این حال، کوئری زیر نامعتبر است زیرا تعاریف نام window به صورت سیکلی است:

```
SELECT 'This is invalid'
WINDOW W1 AS (W2), W2 AS (W3), W3 AS (W1);
```

این کوئری خروجی زیر را تولید می‌کند:

```
Msg 5365, Level 15, State 1, Line 108
Cyclic window references are not permitted.
```

در نهایت، به دامنه window name های تعریف شده دقت کنید. به عنوان مثال، اگر یک window name را در کوئری داخلی یک CTE، جدول مشتق شده، view یا به صورت inline table valued function تعریف کنید، کوئری بیرونی، window name داخلی را تشخیص نخواهد داد. به عنوان مثال، کوئری زیر به این دلیل نامعتبر است:

```
WITH C AS
(
  SELECT orderid, custid, orderdate, qty, val,
    SUM(qty) OVER W AS runsumqtyall
  FROM Sales.OrderValues
  WHERE custid IN (1, 2)
  WINDOW W AS ( PARTITION BY custid
                ORDER BY orderdate, orderid
                ROWS UNBOUNDED PRECEDING )
)
SELECT *,
  SUM(qty) OVER W AS runsumqty22
FROM C
WHERE orderdate >= '20220101';
```

این کوئری خروجی زیر را تولید می‌کند:

```
Msg 5362, Level 15, State 3, Line 172
Window 'W' is undefined.
```

شما باید window name ای را که می‌خواهید، در هر یک از دامنه‌هایی که می‌خواهید از آن استفاده کنید، تعریف کنید، مانند زیر:

```
WITH C AS
(
  SELECT orderid, custid, orderdate, qty, val,
    SUM(qty) OVER W AS runsumqtyall
  FROM Sales.OrderValues
  WHERE custid IN (1, 2)
  WINDOW W AS ( PARTITION BY custid
              ORDER BY orderdate, orderid
              ROWS UNBOUNDED PRECEDING )
)
SELECT *,
  SUM(qty) OVER W AS runsumqty22
FROM C
WHERE orderdate >= '20220101'
WINDOW W AS ( PARTITION BY custid
            ORDER BY orderdate, orderid
            ROWS UNBOUNDED PRECEDING );
```

این کوئری خروجی زیر را تولید می‌کند:

```
orderid     custid      orderdate  qty         val     runsumqtyall runsumqty22
----------- ----------- ---------- ----------- ------- ------------ -----------
10835       1           2022-01-15 17          845.80  96           17
10952       1           2022-03-16 18          471.20  114          35
11011       1           2022-04-09 60          933.50  174          95
10926       2           2022-03-04 29          514.40  63           29
```

هر یک از دامنه‌ها نام پنجره خود را W تعریف می‌کنند و لازم نیست آنها بر اساس مشخصات یکسانی باشند (اگرچه در این مثال هستند).

Windowing NULL Treatment

NULL Treatment بخشی از استاندارد ISO/IEC SQL است و برای افست توابع window، شامل FIRST_VALUE، LAST_VALUE، LAG و LEAD در دسترس است. این عبارت دارای سینتکس زیر است:

<function>(<scalar_expression>[, <other args>]) [IGNORE NULLS | RESPECT NULLS] OVER(<specification>)

گزینه RESPECT NULLS پیش فرض است و به این معنی است که شما می‌خواهید تابع، مقدار <scalar_expression> را برگرداند، خواه NULL یا non-NULL باشد.

گزینه IGNORE NULLS قابلیت جدیدی را معرفی می‌کند که برنامه نویسان مدت‌ها بود مشتاقانه منتظر اضافه شدن آن در T-SQL بودند. این گزینه بدان معنی است که شما می‌خواهید تابع، مقدار <scalar_expression> را در صورت non-NULL برگرداند. با این حال، اگر NULL باشد، می‌خواهید که تابع به روال خود ادامه داده تا زمانی که یک مقدار non-NULL پیدا شود. اگر مقدار non-NULL پیدا نشد، یک NULL برمی‌گرداند.

برای نشان دادن کاربرد این عبارت، از جدولی به نام T1 در ادامه مقاله استفاده می‌کنم. برای ایجاد و پر کردن T1 از کد زیر استفاده کنید:

```
DROP TABLE IF EXISTS dbo.T1;

CREATE TABLE dbo.T1
(
  id INT NOT NULL CONSTRAINT PK_T1 PRIMARY KEY,
  col1 INT NULL,
  col2 INT NULL
);
GO

INSERT INTO dbo.T1(id, col1, col2) VALUES
  ( 2, NULL,  200),
  ( 3,   10, NULL),
  ( 5,   -1, NULL),
  ( 7, NULL,  202),
  (11, NULL,  150),
  (13,  -12,   50),
  (17, NULL,  180),
  (19, NULL,  170),
  (23, 1759, NULL);
```

فرض کنید ستون id نشان دهنده ترتیب وقوع رویدادهای ثبت شده در T1 است. هر ردیف نشان دهنده رویدادی است که در آن یک یا چند مقدار attribute تغییر کرده است. NULL به این معنی است که attributeآخرین مقدار non-NULL را که تا آن لحظه داشته است را همچنان حفظ کرده است.

فرض کنید باید آخرین مقدار col1 یعنی lastknowncol1 را به صورت non-NULL در هر رویداد برگردانید. بدون استفاده از NULL Treatment، باید از یک تکنیک نسبتاً پیچیده مانند موارد زیر استفاده کنید:

```
WITH C AS
(
  SELECT id, col1,
    MAX(CASE WHEN col1 IS NOT NULL THEN id END)
      OVER(ORDER BY id
           ROWS UNBOUNDED PRECEDING) AS grp
  FROM dbo.T1
)
SELECT id, col1,
  MAX(col1) OVER(PARTITION BY grp
                 ORDER BY id
                 ROWS UNBOUNDED PRECEDING) AS lastknowncol1
FROM C;
```

اگر قبلاً با این تکنیک آشنا نیستید، درک منطق آن ممکن است کمی برایتان پیچیده باشد.

این کد خروجی زیر را تولید می‌کند:

```
id          col1        lastknowncol1
----        ------      ---------------
2           NULL         NULL
3           10            10
5           -1            -1
7           NULL        -1
11           NULL        -1
13           -12         -12
17           NULL        -12
19           NULL        -12
23           1759        1759
```

با استفاده از به NULL Treatment، می‌توانید به راحتی با استفاده از تابع LAST_VALUE با گزینه IGNORE NULLS به همان خروجی بالا، دست پیدا کنید. کد به صورت زیر خواهد بود:

```
SELECT id, col1,
  LAST_VALUE(col1) IGNORE NULLS OVER( ORDER BY id ROWS UNBOUNDED PRECEDING ) AS
lastknowncol
FROM dbo.T1;
```

البته اگر بخواهید این منطق را برای چندین attribute اعمال کنید، تفاوت چشمگیرتر خواهد بود.

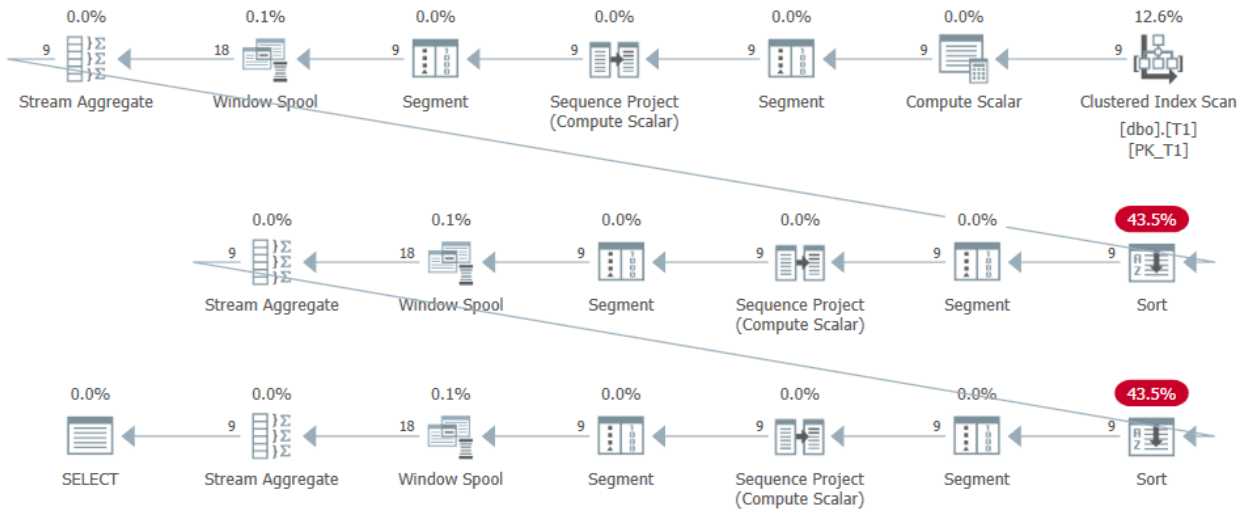
بدون استفاده از NULL Treatment، از کد زیر برای برگرداندن آخرین مقادیر ستون‌های col1 و col2 یعنی lastknowncol1 و lastknowncol2 استفاده می‌کنیم:

```
WITH C AS
(
  SELECT id, col1, col2,
    MAX(CASE WHEN col1 IS NOT NULL THEN id END)
      OVER(ORDER BY id
          ROWS UNBOUNDED PRECEDING) AS grp1,
    MAX(CASE WHEN col2 IS NOT NULL THEN id END)
      OVER(ORDER BY id
          ROWS UNBOUNDED PRECEDING) AS grp2
  FROM dbo.T1
)
SELECT id,
  col1,
  MAX(col1) OVER(PARTITION BY grp1
              ORDER BY id
              ROWS UNBOUNDED PRECEDING) AS lastknowncol1,
  col2,
  MAX(col2) OVER(PARTITION BY grp2
              ORDER BY id
              ROWS UNBOUNDED PRECEDING) AS lastknowncol2
FROM C;
```

این کد خروجی زیر را تولید می‌کند:

id	col1	lastknowncol1	col2	lastknowncol2
2	NULL	NULL	200	200
3	10	10	NULL	200
5	-1	-1	NULL	200
7	NULL	-1	202	202
11	NULL	-1	150	150
13	-12	-12	50	50
17	NULL	-12	180	180
19	NULL	-12	170	170
23	1759	1759	NULL	170

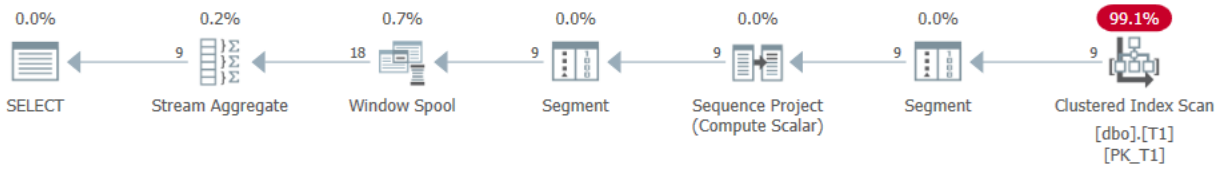
همچنین باید توجه داشته باشید که حتی اگر جدول T1 دارای یک ایندکس با id به عنوان کلید باشد، هر یک از lastknown های attribute در کوئری بالا منجر به یک عملگر explicit sorting در plan می‌شود. همان طور که در شکل زیر نشان داده شده است.



این واقعیت، باعث می‌شود که این راه حل با سربار بالایی همراه باشد. در ادامه از راه حل جایگزین، یعنی استفاده از NULL Treatment را خواهیم داشت:

```
SELECT id,
  col1, LAST_VALUE(col1) IGNORE NULLS OVER W AS lastknowncol1,
  col2, LAST_VALUE(col2) IGNORE NULLS OVER W AS lastknowncol2
FROM dbo.T1
WINDOW W AS ( ORDER BY id ROWS UNBOUNDED PRECEDING );
```

این راه‌حل بسیار کوتاه‌تر است و بهینه‌سازی توابع با این گزینه می‌تواند به اسکن مرتب‌سازی یک ایندکس تکیه کند و بنابراین از explicit sorting، اجتناب شود. همان طور که در plan این کوئری در شکل زیر نشان داده شده است.



همان طور که گفته شد، NULL treatment برای افست همه توابع window شامل (FIRST_VALUE، LAST_VALUE، LAG و LEAD) در دسترس است. در اینجا یک مثال با استفاده از LAG برای برگرداندن مقدار prevknowncol آورده شده است:

```
SELECT id, col1,
  LAG(col1) IGNORE NULLS OVER ( ORDER BY id ) AS prevknowncol1
FROM dbo.T1;
```

این کد خروجی زیر را تولید می‌کند:

```
id          col1        prevknowncol1
----------- ----------- -------------
2           NULL        NULL
3           10          NULL
5           -1          10
7           NULL        -10
11          NULL        -1
13          -12         -1
17          NULL        -12
19          NULL        -12
23          1759        -12
```

با توضیحات ارائه شده در این مقاله مطمئنم قصد ندارید این مثال را بدون استفاده از NULL treatment انجام دهید.

جمع‌بندی

در این مقاله به بهبودهای T-SQL در SQL Server 2022 در مورد توابع window و مدیریت NULL پرداخته شد. در این مقاله روی دو مفهوم زیر صحبت شد:

- استفاده مجدد بخشی از window definition و یا کل آن