عنوان: **محاسبه میانگین متحرک (moving average) در SQL Server 2012**
نویسنده: محمد سلیم آبادی
تاریخ: ۱۳۹۱/۱۱/۱۴ ۱۹:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server 2012, window function, querying, moving average

شرح مساله

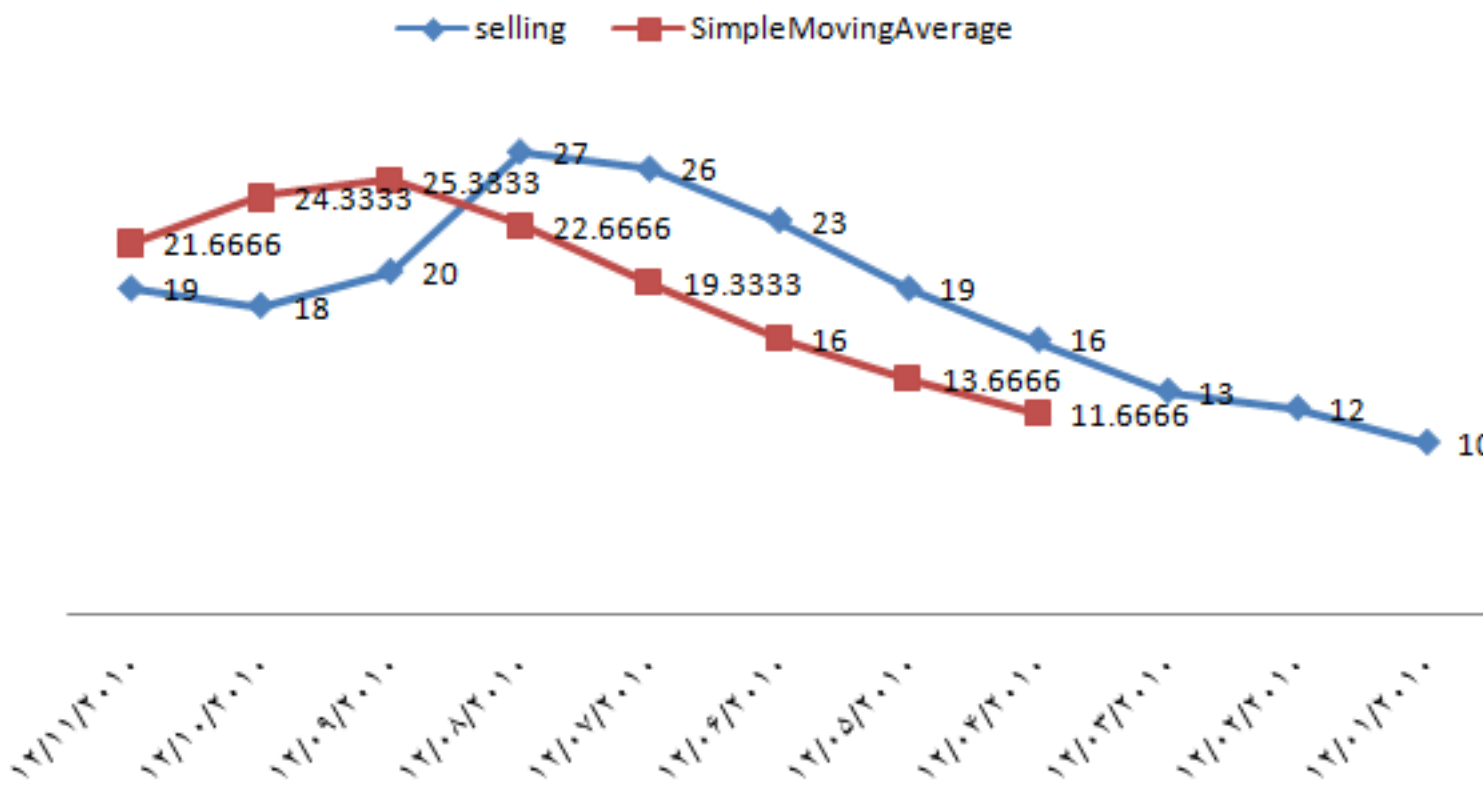
میانگین متحرک یا moving average به چند دسته تقسیم می‌شود که ساده‌ترین آنها میان متحرک ساده است.
برای محاسبه میانگین متحرک باید بازه زمانی مورد نظر را مشخص کنیم. مثلا میانگین فروش در 3 روز گذشته.

به جدول زیر توجه بفرمایید:

روز	فروش	میانگین متحرک 3 روزه	میانگین متحرک 4 روزه
1	10	***	***
2	12	***	***
3	13	***	***
4	16	11.6	***
5	19	13.6	12.7
6	23	16.0	15.0
7	26	19.3	17.7

میانگین متحرک فروش سه روز و چهار روز گذشته در جدول فوق قابل مشاهده است.
بطور مثال مقدار میانگین متحرک سه روزه برای روز چهارم برابر است با جمع فروش سه روز گذشته تقسیم بر سه. یعنی (13+12+10)/3
و برای روز ششم میانگین متحرک 4 روزه برابر است با جمع فروش چهار روز گذشته و تقسیم آنها بر چهار. یعنی 16+13+12+10 تقسیم بر 4 که برابر است با 12.7

در نمودار زیر، خط قرم رنگ مربوط به میانگین متحرک ساده (میانگین فروش سه روز گذشته) است و خط آبی رنگ نیز میزان فروش است



راه حل در SQL Server 2012

توسط توابع window این مساله را به سادگی می‌توانیم حل کنیم. همانطور که مشاهده می‌شود در تصویر زیر. کافیست ما به سطرهایی در بازه‌ی سه سطر قبل تا یک سطر قبل (برای میانگین متحرک سه روزه) دسترسی پیدا کرده و میانگین آن را بگیریم.

روز	فروش
1	10
2	12
3	13
4	16
5	19
6	23
7	26

ابتدا این جدول را ایجاد و تعدادی سطر برای نمونه در آن درج کنید:

```
CREATE TABLE Samples
(
[date] SMALLDATETIME,
selling SMALLMONEY
);

INSERT  Samples
VALUES
('2010-12-01 00:00:00', 10),
('2010-12-02 00:00:00', 12),
('2010-12-03 00:00:00', 13),
('2010-12-04 00:00:00', 16),
('2010-12-05 00:00:00', 19),
('2010-12-06 00:00:00', 23),
('2010-12-07 00:00:00', 26),
('2010-12-08 00:00:00', 27),
('2010-12-09 00:00:00', 20),
('2010-12-10 00:00:00', 18),
('2010-12-11 00:00:00', 19);
```

سپس برای محاسبه میانگین متحرک در بازه سه روز گذشته query زیر را اجرا کنید:

```
SELECT [date],
       selling,
       CASE WHEN rnk < 4 THEN NULL ELSE mv END AS SimpleMovingAverage
  FROM (SELECT *,
               AVG(selling) OVER(ORDER BY [date]
                                 ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING) AS mv,
               ROW_NUMBER() OVER(ORDER BY [date]) AS rnk
          FROM Samples
       ) AS d;
```

قلب query دستور ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING می‌باشد.

به این معنا که سطرهایی در بازه‌ی سه سطر قبل و یک سطر قبل را در Window انتخاب کرده و عمل میاگنین گیری را بر اساس مقادیر مورد نظر انجام بده.

راه حل در SQL Server 2005

به درخواست یکی از کاربران من راه حلی را پیشنهاد می‌کنم که جایگزین مناسبی برای روش قبلی است در صورت عدم استفاده از نسخه 2012. توابع window در اینگونه مسائل بهترین عملکرد را خواهند داشت.

```
SELECT S.[date], S.selling, CASE WHEN COUNT(*) < 3 THEN NULL ELSE AVG(s) END AS SimpleMovingAverage
  FROM Samples AS S
       OUTER APPLY (SELECT TOP(3) selling
                      FROM Samples
                     WHERE [date] < S.[date]
                     ORDER BY [date] DESC) AS D(s)
 GROUP BY S.[date], S.selling
 ORDER BY S.[date];
```

FOR FUN

توسط توابع Analytical ای چون LAG نیز می‌توان اینگونه مسائل را حل نمود. بطور مثال توسط تابع LAG به یک مقدار قبلی، دو مقدار قبلی و سه مقدار قبلی دسترسی پیدا کرده و آنها را با یکدیگر جمع نموده و تقسیم بر تعدادشان می‌کنیم یعنی:

```
select [date],
       selling,
       (
       lag(selling, 1) over(order by [date]) +
       lag(selling, 2) over(order by [date]) +
       lag(selling, 3) over(order by [date])
       ) / 3
  from Samples;
```

نظرات خوانندگان

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۱۵ ۱۸:۰

ممنون از شما. لطفا در صورت امکان راه حل بدون استفاده از توابع window را هم جهت مقایسه ارائه کنید.

با تشکر بسیار

نویسنده: محمد سلم ابادی
تاریخ: ۱۳۹۱/۱۱/۱۵ ۱۸:۳۷

یک راه حل جدید بدون کمک گرفتن از توابع Window به مقاله افزوده شد.

نویسنده: اسحق مهرجویی
تاریخ: ۱۳۹۲/۱۱/۱۸ ۱۳:۹

با سلام و تشکر از شما. برای محاسبه میانگین متحرک در این سایت به شیوه زیر عمل کرده. می‌تونید یه توضیحی راجع به اون بدهید.

```
SELECT
  T0.StockId
  ,T0.QuoteId
  ,T0.QuoteDay
  ,T0.QuoteClose
  , AVG (T0.QuoteClose) OVER (PARTITION BY T0.StockId ORDER BY T0.QuoteId ROWS 19 PRECEDING) AS MA20
FROM
  dbo.Quotes AS T0
```

نویسنده: محمد سلیم آبادی
تاریخ: ۱۳۹۲/۱۱/۱۸ ۱۵:۱۵

سلام اسحق،
در مثالی که من تهیه کردم، میانگین داده‌های مربوط به 3 سطر قبل محاسبه شده، بدون لحاظ مقدار جاری. اما در مساله مربوط به آن سایت میانگین داده‌های مرتبط به 19 سطر قبل و سطر جاری محاسبه شده.
و همچنین در بخش Specification مربوط به آن تابع میانگین، در مثال سایت از PARTITION استفاده شده آن هم به این خاطر که داده‌های جدول به گروه‌های مختلفی بر اساس مقادیر ستون StockId تقسیم شده است. و میخواسته میانگین مرتبط به هر StockId بطور مجزا محاسبه بشه. در واقع نتیجه را به تعداد StorckIdها بخش بندی کرده.

نام مستعاری که به جدول Quotes داده شده، غیر ضروری بوده، چرا که تنها یک جدول بیشتر در Query مشارکت نداشته و نیازی به ذکر نام جدول یا نام مستعار جدول نیست.
همچنین برای شفافیت بیشتر و ابهام زدایی، بهتر است قسمت Rows تابع تجمعی را کامل و صریح بنویسیم به این صورت:

```
SELECT StockId, QuoteId, QuoteDay, QuoteClose,
       AVG (QuoteClose) OVER (PARTITION BY StockId
                              ORDER BY QuoteId
                               ROWS BETWEEN 19 PRECEDING AND CURRENT ROW) AS MA20
FROM dbo.Quotes AS T0;
```

عنوان: **توابع Window و مساله های آماری running total و runnning average**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۱/۱۵ ۹:۵
آدرس: www.dotnettips.info
برچسب‌ها: sql server 2005, window function, running total, running average

مقدمه و شرح مساله

توسط ویژگی‌های جدیدی که در نسخه 2012 به بحث window افزوده شد می‌توانیم مسالهای running total و running average را به شکل بهینه ای حل کنیم.
ابتدا این دو مساله را بدون بکارگیری ویژگی‌های جدید، حل نموده و سپس سراغ توابع جدید خواهم رفت.

قبل از هر چیزی لازم است جدول زیر ساخته شود و داده‌های نمونه در آن درج شود:

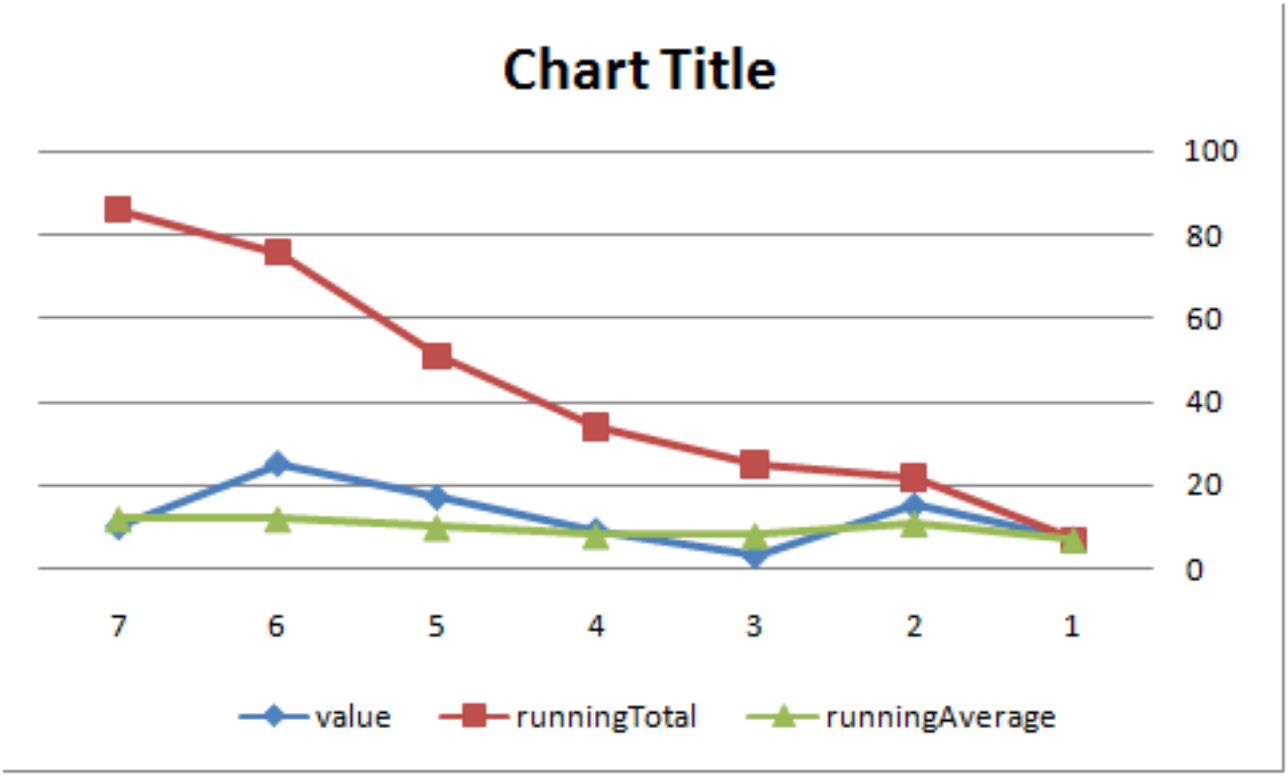
```
create table testTable
(
day_nbr integer not null primary key clustered,
value integer not null check (value > 0)
);
insert into testTable
values (10, 7), (20, 15), (30, 3), (40, 9), (50, 17), (60, 25), (70, 10);
```

مساله running total بسیار ساده است، یعنی جمع مقدار سطر جاری با مقادیر سطرهای قبلی (بر اساس یک ترتیب معین) running average هم مشابه به running total هست با این تفاوت که میانگین مقادیر سطرجاری وسطرهای قبلی محاسبه می‌شود.

Results | Messages | Execution plan

	day_nbr	value	runningTotal	runningAverage
1	10	7	7	7
2	20	15	22	11
3	30	3	25	8
4	40	9	34	8
5	50	17	51	10
6	60	25	76	12
7	70	10	86	12

و نتیجه به صورت نمودار:

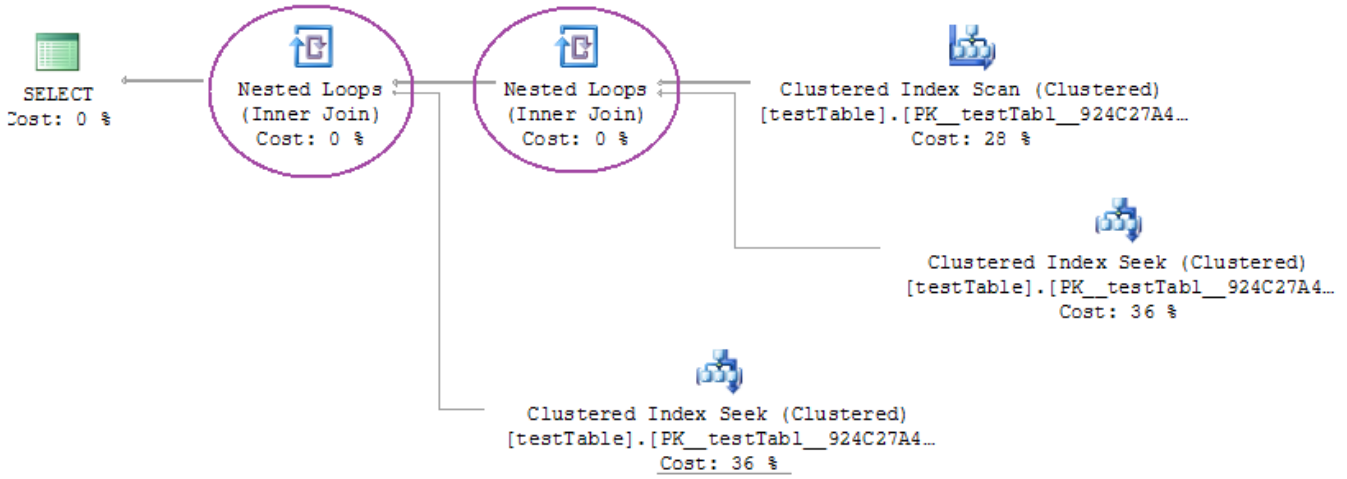


راه حل در SQL Server 2000

توسط دو correlated scalar subquery در ماده SELECT می‌توانیم مقادیر دو ستون مورد نظر با محاسبه کنیم:

```
select *,
       runningTotal = (select sum(value)
                         from testTable
                        where day_nbr <= t.day_nbr),
       runningAverage = (select avg(value)
                           from testTable
                          where day_nbr <= t.day_nbr)
  from testTable t;
```

اگر به نقشه اجرای این query نگاه کنید گره(عملگر) inner join دو بار بکار رفته است (به وجود دو subquery)، که این عدد در روش توابع تجمعی window به صفر کاهش پیدا خواهد کرد

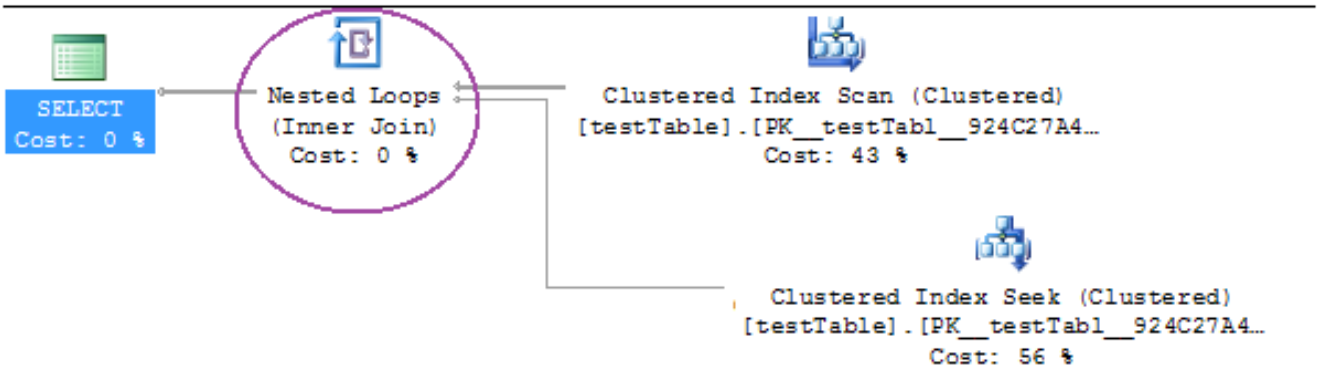


راه حل در SQL Server 2005

توسط cross apply به سادگی می‌توانیم دو subquery که در روش قبل بود را به یکی کاهش دهیم:

```
select *
  from testTable t
       cross apply (select sum(value) as runningTotal,
                           avg(value) as runningAverage
                      from testTable
                     where day_nbr <= t.day_nbr)d;
```

این بار تنها یک عملگر inner join در نقشه اجرای query مشاهده می‌شود:

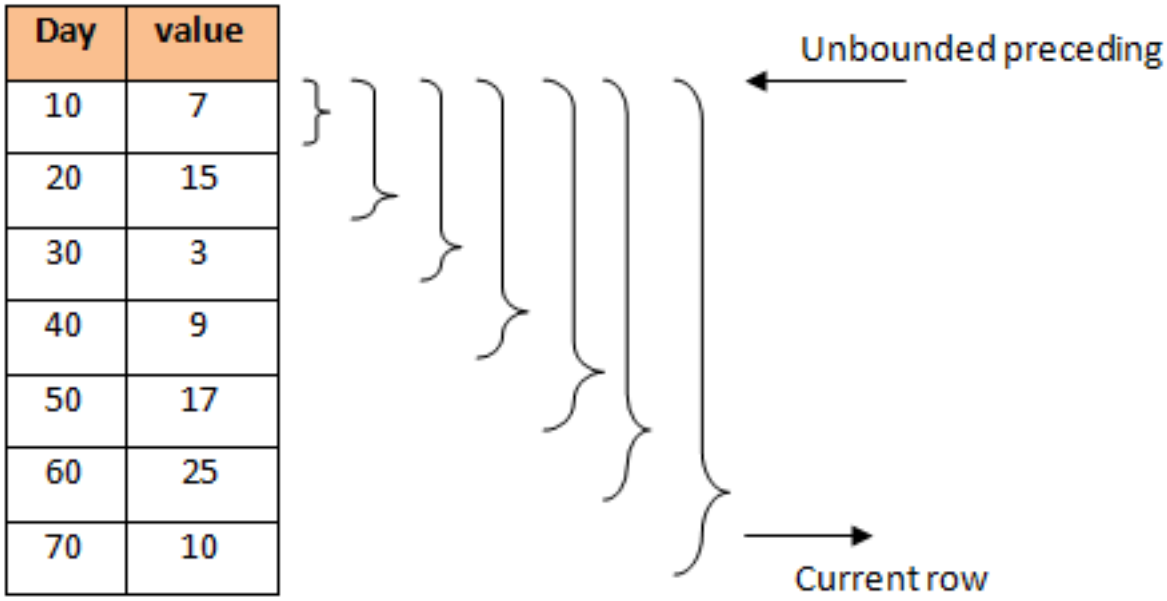


راه حل در SQL Server 2012

با اضافه شدن برخی از ویژگی‌های استاندارد به ماده OVER مثل rows و range شاهد بهبودی در عملکرد queryها هستیم.

یکی از کاربردهای توابع تجمعی window حل مساله running total و running average است.

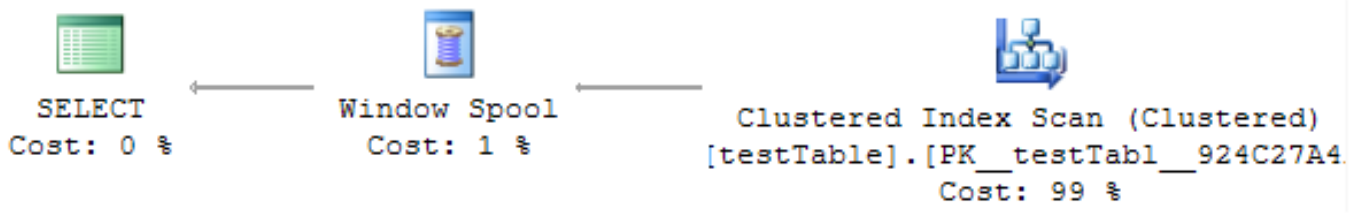
به تصویر زیر توجه کنید، همانطور که در قبل توضیح دادم ما به سطرجاری و سطرهای پیشین نیاز داریم تا اعمال تجمعی (جمع و میانگین) را روی مقادیر بدست آمده انجام دهیم. در تصویر زیر سطرجاری و سطرهای قبلی به ازای هر سطری به وضوح قابل مشاهده است، مثلا هنگامی که سطر جاری برابر با روز 30 است ما خود سطر جاری (current row) و تمام سطرهای پیشین و قبلی (unbounded preceding) را نیاز داریم.



و اکنون query مورد نظر

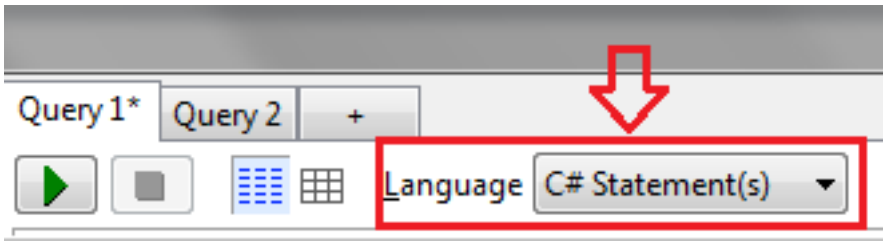
```
select *, sum(value) over(order by day_nbr rows between unbounded preceding and current row) as
runningTotal,
          avg(value) over(order by day_nbr rows between unbounded preceding and current row) as
runningAverage
from testTable
```

در نقشه اجرای این query دیگر خبری از عملگر inner join نخواهد بود که به معنای عملکرد بهتر query است.



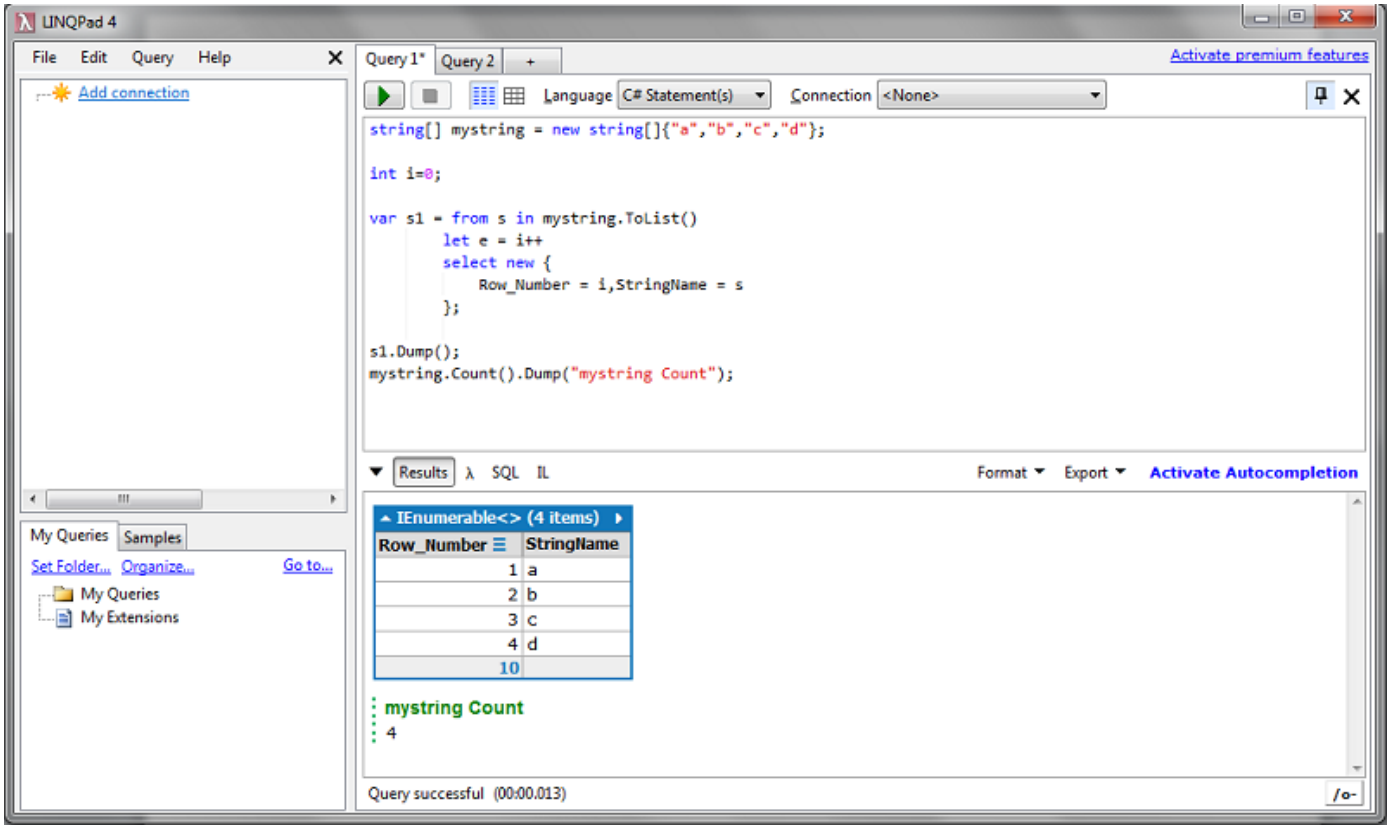
عنوان: **نحوه ایجاد شمارنده Row_Number() Sql Server در LINQ**
نویسنده: بهمن خلفی
تاریخ: ۱۳۹۲/۰۵/۱۹ ۱۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: LINQ, window function, linqpad

چند روز پیش برای انجام یک بخشی از کار پروژه خودم باید از توابع و window functionها در sql server استفاده میکردم که در سایت جاری آشنایی با Row_Number،Rank،Dense_Rank،NTILE و آشنایی با Window Functionها در SQL Server بصورت مفصل توضیح داده شده است.
حال اگر بخواهیم یکی از پرکاربردترین این توابع که Row_Number می‌باشد را در LINQ استفاده کنیم باید به چه صورت عمل کنیم.
من برای پیاده سازی از برنامه نیمه رایگان LINQPad استفاده کردم که میتوانید از سایت اصلی این نرم افزار دانلود نمائید.
پس از دانلود و اجرای آن ، در قسمت بالایی زبان linqpad را به C# Statement(s) تغییر دهید.

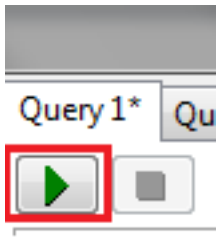


سپس کد زیر را به بخش query انتقال دهید.

```
string[] mystring = new string[]{"a","b","c","d"};

int i=0;

var s1 = from s in mystring.ToList()
let e = i++
select new {
Row_Number = i,StringName = s
};

s1.Dump();
mystring.Count().Dump("mystring Count");
```



سپس با زدن کلید F5 یا دکمه اجرای query نتیجه را مشاهده نمائید.



use-row_number-in-Linq.linq