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

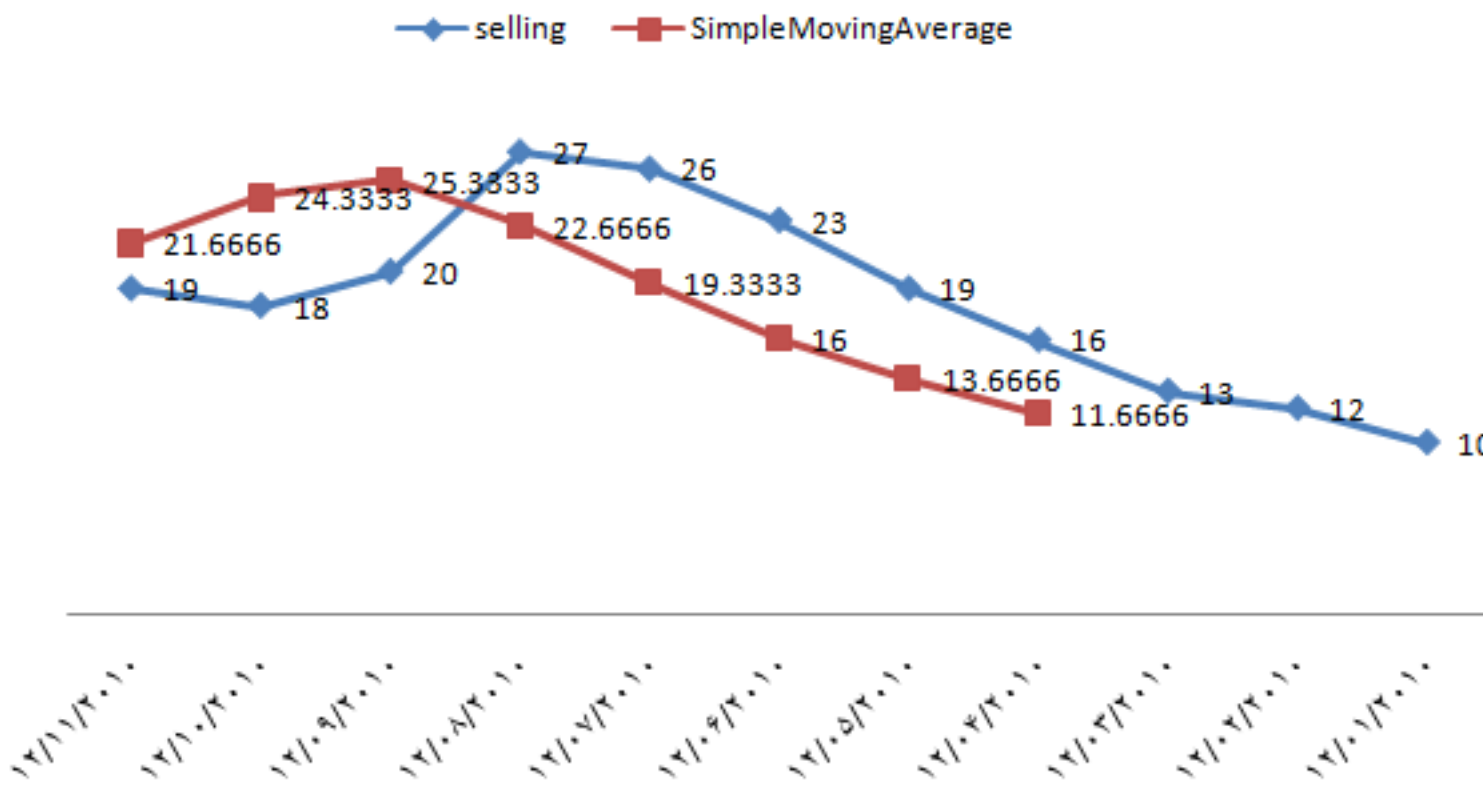
میانگین متحرک یا moving average به چند دسته تقسیم می‌شود که ساده‌ترین آنها میان متحرک ساده است.
برای محاسبه میانگین متحرک باید بازه زمانی مورد نظر را مشخص کنیم. مثلا میانگین فروش در 3 روز گذشته.

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: **بررسی مساله متداول Top N در نسخه های مختلف SQL Server**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۳:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, querying, Top N

مقدمه (شرح مساله)

چندی پیش در تالار T-SQL سوالی مطرح شد راجع به مساله ای که معروف است به top N per group.
تنها موضوعی که باعث شد من مطلبی راجع به آن بنویسم محدودیتی بود که کاربر مورد نظر داشت؛ که آن محدودیت چیزی نبود جز: query بایستی در نسخه 2000 جوابگو باشد.

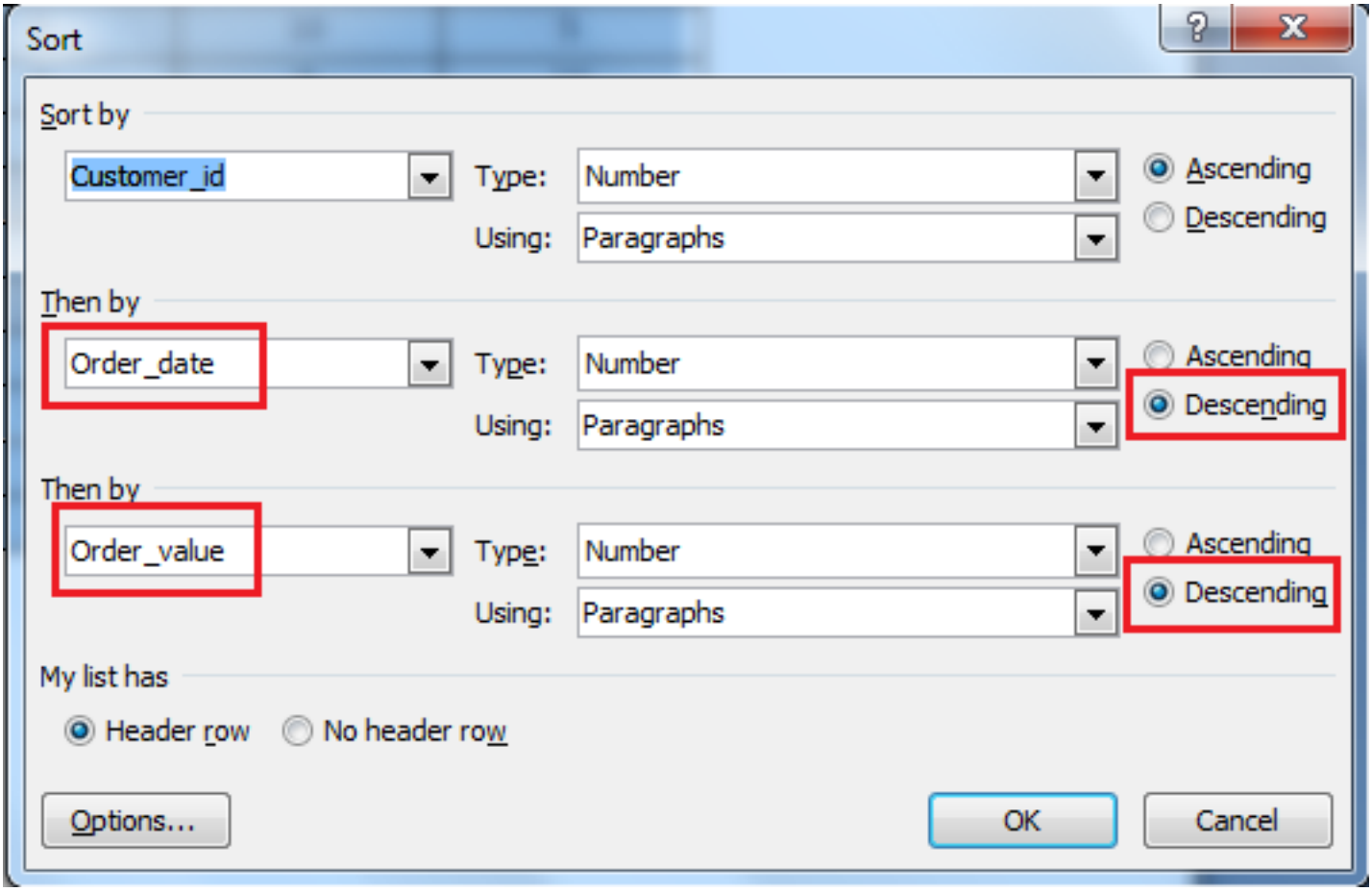
قطعا شده است که بخواهید مثلا به ازای هر مشتری آخرین سفارش آن را انتخاب کنید. این مساله Top N نامیده می‌شود.

فرض کنید جدولی داریم که حاوی سفارشات مشتریان می‌باشد. هر مشتری می‌تواند چندین سفارش داشته باشد؛ هر سفارش دارای حداقل دو مقدار "تاریخ سفارش" و "مبلغ سفارش است". هدف پیدا کردن آخرین سفارشات هر مشتری می‌باشد.
نکته: اگر چند تاریخ برای آخرین سفارش مشتری وجود داشت آنگاه بایستی بر اساس مبلغ سفارش مرتب سازی نزولی صورت بگیرد. یا به عبارت دیگر ابتدا باید مرتب سازی نزولی بر اساس ستون تاریخ سفارش انجام شود و سپس مرتب سازی نزولی بر اساس ستون مبلغ سفارش.

فرض می‌گیریم داده‌های جدول ما چیزیست شبیه به این:

Row_id	Customer_id	Order_date	Order_value
6	1	10	15
9	1	10	5
4	1	9	10
3	1	8	20
10	2	12	15
7	2	10	15
11	3	15	5
5	3	10	20
8	3	10	10
1	3	5	15
2	3	5	15

سطرهایی از جدول که رنگی شده اند سطرهای مورد نظر ما هستند که باید در خروجی ظاهر شوند.
داده‌های جدول با کمک قابلیت Sort نرم افزار word مرتب سازی شده اند، این تصویر را به این خاطر در اینجا قرار دادم چون که دیدم می‌تواند در شفاف سازی مساله به من کمک کند.
ابتدا مرتب سازی نزولی بر اساس ستون order_date انجام گرفته و سپس مرتب سازی نزولی بر اساس ستون order_value. و در پایان اولین سطر مربوط به هر مشتری به عنوان خروجی مورد نظر انتخاب می‌شوند.



راه حل ها

خب پر واضح است که در نسخه 2005 و بعد از آن ساده‌ترین و بهینه‌ترین راه حل استفاده از تابع row_number می‌باشد.

```
SELECT row_id, customer_id, order_date, order_value
  FROM (SELECT *,
               ROW_NUMBER() OVER(PARTITION BY customer_id
                                     ORDER BY order_date DESC, order_value DESC) AS rnk
          FROM table_name
       )t
 WHERE rnk = 1;
```

اما با محدودیتی که در نسخه 2000 وجود دارد راه حلی بهتر از این پیدا نخواهیم کرد:

```
SELECT *
  FROM table_name t
 WHERE row_id = (SELECT TOP 1 row_id
                   FROM table_name
                  WHERE customer_id = t.customer_id
                  ORDER BY order_date DESC, order_value DESC);
```

حالا چه میشود راه حلی بخواهیم مستقل از هر یک از نسخه‌های SQL Server:

```
SELECT MIN(row_id) AS row_id, customer_id, order_date, order_value
    FROM table_name t
   WHERE order_date =
```

```
        (SELECT MAX(order_date)
           FROM table_name
          WHERE customer_id = t.customer_id)
    AND order_value =
        (SELECT MAX(order_value)
           FROM table_name
          WHERE customer_id = t.customer_id
            AND order_date =
                (SELECT MAX(order_date)
                   FROM table_name
                  WHERE customer_id = t.customer_id))
                  GROUP BY customer_id, order_date, order_value;
```

عنوان: **استفاده از توابع Scalar بجای case**
نویسنده: فرهود جعفری
تاریخ: ۱۳۹۲/۱۱/۰۶ ۲۱:۲۵
آدرس: www.dotnettips.info
گروه‌ها: T-SQL, querying, SQL

گاهی از اوقات نیاز است در کوئری‌ها از بین چندین مقدار یکی انتخاب و بجای مقدار اصلی، رشته یا عبارتی جایگزین، نوشته شود. پر استفاده‌ترین راه حل پیشنهادی، استفاده از عبارت case در داخل کوئری هست که بر اساس موارد ممکن، عبارتهای برگشتی نوشته می‌شود. این راه حل خوبی به نظر می‌رسد اما اگر تعداد گزینه‌ها زیاد شود باعث شلوغ شدن متن کوئری و اشکال در بازبینی و نگهداری آن خواهد شد.
یک راه حل دیگر استفاده از توابع نوع Scalar می‌باشد؛ به این صورت که میتوان مقدار استخراج شده از جدول را به تابع تعریف شده فرستاد و در ازاء، مقدار بازگشتی مناسبی را در خروجی مشاهده کرد. حال به یک مثال توجه کنید:

```
Select Case Gen when 0 then 'مرد' when 1 then 'زن' end As Gen From Table
```

اکنون استفاده از تابع:

```
CREATE FUNCTION fcGenName
(
@Gen tinyint
)
RETURNS nvarchar(20)
AS
BEGIN
-- Declare the return variable here
DECLARE @gen nvarchar(20)

-- Add the T-SQL statements to compute the return value here
set @gen = (SELECT case @Gen when 0 then 'مرد' when 1 then 'زن' end as d)

-- Return the result of the function
RETURN @gen

END
```

و فراخوانی تابع در متن کوئری :

```
Select fcGenName(Gen) From Table
```

نظرات خوانندگان

نویسنده: م رحمانی
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۲:۲۲

سلام و تشکر
سؤال: این ارجاع به یک تابع تأثیر تو کارایی نداره؟

نویسنده: فرهود جعفری
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۴:۵۳

با سلام
ظاهراً در تعداد رکوردهای پایین مشکلی نداره اما در تعداد رکوردهای بالا احتمال کاهش سرعت اجرا دور از ذهن نیست. به هر حال من دقیق تست نکردم اما روی شیوه دیگه هم دارم کار می‌کنم که اون توابع برگشتی از نوع جدول هست که با این شیوه اساساً فرق داره

نویسنده: زاهدیان فرد
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۱:۴۴

استفاده از function خوبه مزیتش این که میشه جاهای مختلف استفاده کرد! ولی در تعداد رکورد پایین، چون در رکوردهای زیاد سرعت کوئری به شدت افت میکنه! روش اول بنظرم من بهتر

نویسنده: زاهدیان فرد
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۲:۱۰

در SQL server 2012 تابعی اضافه شده به اسم IIF که بجای

```
SELECT CASE @GEN WHEN 0 THEN 'Male' ELSE 'Woman' AS Gender
```

از این می‌توان استفاده کرد

```
SELECT IIF(Gen=0,'Male','Woman')
```

نویسنده: محمد سلیم آبادی
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۵:۳

در نسخه 2012 جهت سهولت در مهاجرت پایگاه داده‌های Access به SQL Server از توابع CHOOSE و IIF حمایت شده.

منتها تابع IIF چندان انعطاف پذیر نیست. مثلا اگر بخواهید به ازای چند حالت مشخص از داده‌های یک فیلد یک مقدار را برگردانید مجبورید چند تابع IIF تودرتو بنویسید. تودرتو بودن این تابع هم به 10 سطح محدود میشه.
اما CASE قابلیت‌ها و انعطاف پذیری بیشتری داره.
سوال میشه گاها کدام یک از این دو Performance یا کارایی بهتر دارد، در جواب میشه گفت هر دو برابر اند. در واقع IIF هنگام اجرا تبدیل به فرم CASE خواهد شد.
فرض کنید یک نظر سنجی تلوزیونی تنظیم کردیم که مردم از طریق پیامک نظر خودشان را به ما اعلام میکنند. شش گزینه هم داریم. برای انتخاب هر گزینه کافیست از اعداد 1 تا 6 استفاده کنیم. حال هنگام نمایش می‌خواهیم به جای اعداد مقدار متناظر ظاهر شود:

```
Use Tempdb
Go

CREATE TABLE [Sample] (value int);
INSERT INTO [Sample] VALUES (1),(2),(3),(4),(5),(6);
Go

--simple CASE Expression
SELECT value,
       CASE Value
        WHEN 1 THEN 'Very Bad'
        WHEN 2 THEN 'Bad'
        WHEN 3 THEN 'Not Bad'
        WHEN 4 THEN 'Good'
        WHEN 5 THEN 'Very Good'
        WHEN 6 THEN 'Excellent'
       ELSE NULL
END AS [Result]
FROM [Sample];

--CHOOSE Scalar Function
SELECT value,
       CHOOSE(value,'Very Bad','Bad','Not Bad','Good','Very Good','Excellent')
FROM [Sample];


--nested IIF Scalr Function
SELECT value,
IIF(value = 1, 'Very Bad',
  IIF(value = 2, 'Bad',
    IIF(value = 3, 'Not Bad',
      IIF(value = 4, 'Good',
        IIF(value = 5, 'Very Good', 'Excellent'
           )
        )
       )
     )
   )
FROM [Sample];
```