**بسمه تعالی**



برنامه نویسی بانک اطلاعاتی SQL Server

**نویسنده : محمد حسین فخرآوری**

## STATISTICS چیست - مقدمه

در sql server نام هایی مانند: دیتابیس، جدول، فیلد و ایندکس برای ما آشنا هستند و همواره از آن ها استفاده می کنیم. اما لازم است بدانیم که یک شیء بسیار کاربردی دیگر در sql server و در سطح جدول وجود دارد که غافل شدن از آن به معنای خداحافطی با اجرای سریع و مناسب کوئری ها است! این Object ریز نقش، **STATISTICS** نام دارد. در آموزش **STATISTICS چیست**در ابتدا با استفاده از مثالی عملی در دنیای واقعی، با مفهوم و کاربرد **STATISTICS**آشنا می شویم و سپس بحث را به sql server منتقل می کنیم و می بینیم

که **STATISTICS** چگونه نقشی تعیین کننده در سرعت اجرای کوئری ما دارد. با سه اصطلاح Selectivity، Density و Cardinality آشنا می شویم. اجزاء تشکیل دهنده یک **STATISTICS** و محتوبات هر کدام از این قسمت ها را می بینیم. در قسمت پایانی آموزش **STATISTICS چیست**به معرفی انواع، نحوه ایجاد و بروزرسانی  **STATISTICS**ها خواهیم پرداخت.

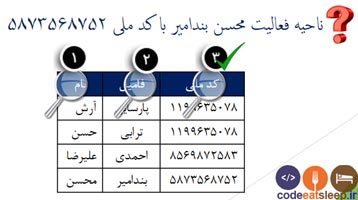
با توجه به گستردگی مطالب پیرامون **STATISTICS،**در**آموزش STATISTICS چیست**تنها به مباحث پایه ای پرداخته می شود و متناسب با موضوع، مابقی آموزش های مرتبط در مجموعه هایی مانند ترفندها و افزایش سرعت و کارایی مطرح خواهند شد.

## STATISTICS چیست - مفهوم و کاربرد

فرض کنید به شما یک لیست کاغذی داده می شود که در آن اطلاعات پرسنل شاغل در شرکت به همراه محل اشتغال آن ها نوشته شده است. در این لیست، نام، فامیل، کد ملی و محل اشتغال موجود است:



از شما خواسته می شود تا محل اشتغال فردی به نام محسن بندامیر با کد ملی 5873568752 را در لیست بالا پیدا کنید. دقت می کنید که شما برای جستجو کردن در لیست بالا، سه مقدار نام، فامیل و کد ملی را در اختیار دارید. اما همیشه از کدامیک برای این جستجو استفاده می کنیم؟ بله پاسخ کد ملی است:



در اینجا این سؤال مطرح می شود که چرا ما برای جستجوی بالا از کد ملی فرد استفاده کردیم؟ پاسخ این است که چون می دانیم به ازای هر فرد، تنها یک کد ملی(پونیک) وجود دارد و با جستجوی این مقدار در لیست، به سرعت فرد مورد نظر را پیدا و محل اشتغال او را استخراج می کنیم. خوب اگر جدولی با ساختار لیست کاغذی شکل بالا در sql server داشته باشیم و همین اطلاعات را در آن درج کنیم، برای رسیدن به پاسخ مورد نظر مان(محل اشتغال فردی به نام محسن بندامیر با کد ملی 5873568752) کوئری زیر را طراحی می کنیم:

|  |
| --- |
| select empsection from tblemployees where empname=N'' محسن  and empfamil=N'' بندامیر and empnatinalcode=5873568752 |

خوب ما انتظار داریم که sql server هم در اجرای کوئری بالا، از فیلد "کد ملی" جهت جستجو بین رکوردها استفاده کند! (که البته انتظار کاملا درستی است) اما همانطور که می دانید این قانون که به ازای هر فرد در کشور ما تنها یک کد ملی وجود دارد یک قرداد دنیای بیرون است و sql server از آن اطلاعی ندارد! پس چگونه این کار انجام می شود؟ پاسخ  **STATISTICS**است.

پس همین جا اولین تعریف ابتدایی از **STATISTICS**را ارائه می دهیم:

**STATISTICS**یک شیء(Object) در sql server است که تعداد تکرار(میزان پراکندگی) مقادیر موجود در فیلدهای یک جدول را نگهداری می کند و sql server در اجرای کوئری ها با استفاده از اطلاعات موجود در آن، تشخیص می دهد که از چه فیلدی برای جستجو استفاده کند.

## دید انتزاعی از Statistics

|  |  |
| --- | --- |
| فامیل | تعداد تکرار |
| بندامیر | 3 |
| جمشیدی | 1 |
| احمدی | 1 |
| ترابی | 1 |
| پارسایی | 1 |
| علیزاده | 1 |
| هاشمی | 1 |

برای شروع، فرض می کنیم که **STATISTICS**تعداد تکرار(میزان پراکندگی) مقادیر موجود در فیلدهای یک جدول را برای هر فیلد به شکل فوق نگهداری می کند و به تدریج در قسمت های بعدی، این شکل را به آنچه به صورت واقعی در **STATISTICS**نگهداری می شود تبدیل می کنیم.

## STATISTICS چیست - اصطلاحات

 جهت ادامه آموزش **STATISTICS چیست**لازم است که با سه اصطلاح Selectivity، Density و Cardinality آشنا شویم:

Selectivity: معرف تعداد یونیک مقادیر یک ستون در جدول. یک مفهوم و معیار برای انتخاب یک فیلد از بین چند فیلد، جهت جستجو است. هر چه مقادیر موجود در یک فیلد، کمتر تکرار شده باشند در نتیجه Selectivity آن مقدار بالاتر است و به عبارتی گزینه ای مناسب تر برای جستجو محسوب می شود. با تعریف بالا ممکنه این سؤال برای شما پیش بیاد که اگر دو شرط دارای Selectivity یکسان باشند، sql server کدام یک را جهت اجرای جستجو انتخاب می کند؟ دقت کنید که Selectivity، تنها یکی از معیارها برای انتخاب یک شرط جهت جستجو است و عواملی دیگری مانند: وجود ایندکس مناسب به ازای فیلد، به روز بودن **Statistics**و ... نیز در این مورد تأثیر گذار هستند.

نکته اول: هر چه Selectivity یک شرط بالاتر باشد، تعداد رکوردهای خروجی کمتر است. (در مثال قبل شرط empnatinalcode=5873568752 تنها یک خروجی داشت

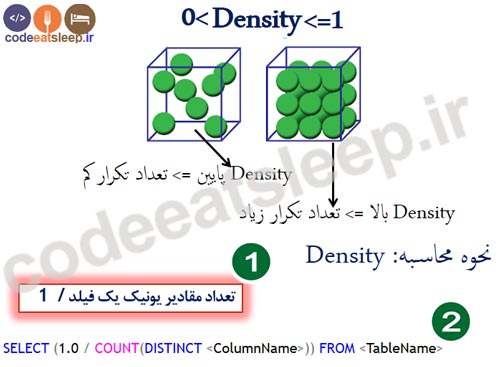


statistics-selectivity-notice2نکته دوم: در sql server بالاترین Selectivity را کلید کاندید(UK) و کلید اصلی(PK) دارند. در این دو کلید Selectivity برابر با یک است و هر شرطی روی این ستون ها، تنها یک رکورد بر می گرداند.



Density: بیان کننده تراکم و یا تعداد تکرار مقادیر یک فیلد است. به عنوان نمونه در مثال قبل Density مقدار "بندامیر" در فیلد فامیل برابر با 3 بود.

statistics-density-notice1نکته سوم: هر دو اصطلاح Density و Selectivity با تعداد تکرار مقادیر فیلدها مرتبط هستند. اما دارای مفهومی متضاد با یکدیگرند به گونه ای که Selectivity بالا به ازای یک مقدار، به معنای Density پایین برای آن مقدار است.

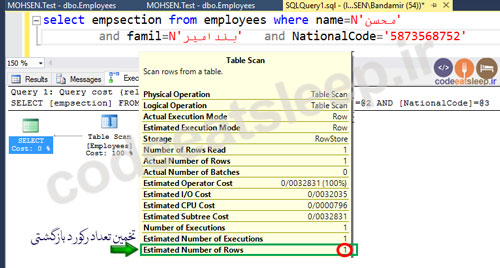


 Cardinality: به معنای تعداد رکوردهای بازگشتی، پس از اجرا شدن کوئری هستند. به عنوان مثال Cardinality کوئری مثال قبل برابر با یک بود.

statistics-cardinality-notice1نکته چهارم: Density و Selectivity به ازای فیلدها و قادیر اون ها تعریف می شوند و Cardinality به ازای کل کوئری(رکوردهای بازگشتی).

statistics-cardinality-notice1نکته پنجم: در Execution Plan مقدار Cardinality به دو صورت تخمینی(Estimated Number) و واقعی(Actual Number) نمایش داده می شود.

یک توضیح تکمیلی در نکته پنجم: قبل از اجرای واقعی یک کوئری sql server در واقع Optimizer باید بتواند تعداد رکوردهای بازگشتی یک کوئری را تخمین بزند. این تخمین با عنوان Estimated Number در tooltip عملگرهای Execution Plan نمایش داده می شود. نکته قابل توجه این است که این تعداد با استفاده از اطلاعات موجود در Statistics استخراج می شود.



با توضیحات ارائه شده، نتیجه می گیریم که هر چه Selectivity بالاتر باشد، تعداد رکوردهای خروجی کمتر خواهند بود(کاردینالیتی پایین) و در نتیجه کارایی و سرعت اجرای کوئری افزایش می یابد.



## STATISTICS چیست - اجزاء

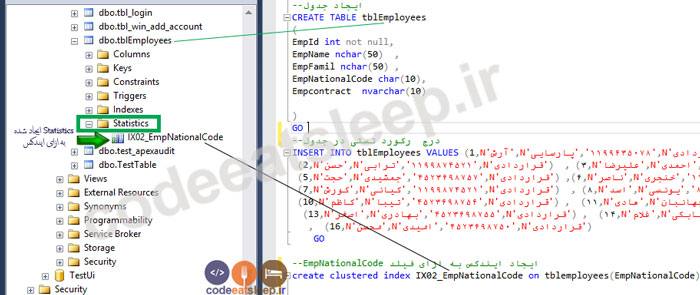
 در این قسمت می خواهیم ببینیم که در **Statistics**چه اطلاعاتی نگهداری می شود و به عبارتی قصد داریم تا **Statistics**را کالبدشکافی کنیم. برای این منظور جدول tblEmployees را ایجاد می کنیم و رکوردهایی در آن درج و  **Statistics**های این جدول را بررسی می کنیم(در مورد نحوه ایجاد **Statistics**در ادامه آموزش صحبت خواهیم کرد

|  |
| --- |
| CREATE TABLE tblEmployees (EmpId INT NOT NULL,  EmpName NCHAR(50),  EmpFamil NCHAR(50),  EmpNationalCode CHAR(10),  Empcontract NVARCHAR(10));  GO    INSERT INTO tblEmployees  VALUES(1, N'آرش', N'پارسایی', '1199635078', N'قراردادی'),  (2, N'حسن', N'ترابی', '1199874521', N'قراردادی'),  (3, N'علیرضا', N'احمدی', '8569872583', N'رسمی'),  (4, N'محسن', N'بندامیر', '5873568752', N'رسمی'),  (5, N'حجت', N'جمشیدی', '4523698757', N'قراردادی'),  (6, N'ناصر', N'خنجری', '1199635078', N'رسمی'),  (7, N'کورش', N'کیانی', '1199874521', N'قراردادی'),  (8, N'اسد', N'یونسی', '8569872589', N'رسمی'),  (9, N'محمد', N'ترکان', '5873568752', N'رسمی'),  (10, N'کاظم', N'تیبا', '4523698756', N'قراردادی'),  (11, N'هادی', N'جهانبان', '4523698752', N'رسمی'),  (12, N'شاهین', N'ترک', '4523698753', N'رسمی'),  (13, N'اصغر', N'بهادری', '4523698755', N'قراردادی'),  (14, N'غلام', N'بابکی', '4523698751', N'رسمی'),  (15, N'رضا', N'امانی', '4523698754', N'رسمی'),  (16, N'محسن', N'امیدی', '4523698750', N'قراردادی');  GO |

از آنجایی که ایجاد ایندکس روی جدول باعث ایجاد **Statistics**می شود، برای جدول tblEmployees یک ایندکس روی فیلد EmpNatinalCode ایجاد می کنیم:

|  |
| --- |
| create clustered index IX02\_EmpNationalCode on tblemployees(EmpNationalCode) |

نتیجه اجرای اسکریپت ها در sql server:



قبل از بررسی **Statistics**ایجاد شده(با نام IX02\_EmpNationalCode)، بهتر است بدانیم که یک **Statistics**از سه قسمت زیر تشکیل شده است:

     1. STAT\_HEADER

     2. Density Vector

     3. Histogram

با توجه به اینکه مهمترین قسمت در Histogram ، **Statistics**است. در ابتدا  این قسمت را بررسی می کنیم. در Histogram جزئیات کامل در مورد مقادیر فیلدها وجود داره به عبارتی در این قسمت است که تعداد تکرار(پراکندگی) مقادیر فیلدها نگهداری می شود. جهت مشاهده اطلاعات موجود در **Statistics**دو روش وجود دارد:

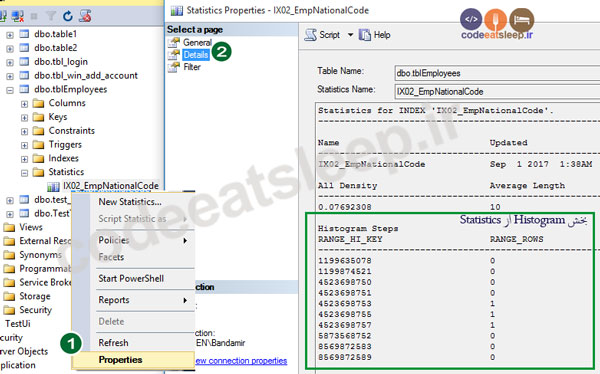
     1. مشاهده اطلاعات **Statistics**با استفاده از ویزارد راست کلیک و انتخاب Properties

     2. مشاهده اطلاعات **Statistics**با استفاده از DBCC Command بصورت (dbcc show\_statistics (Table-Name,**Statistics** -Name

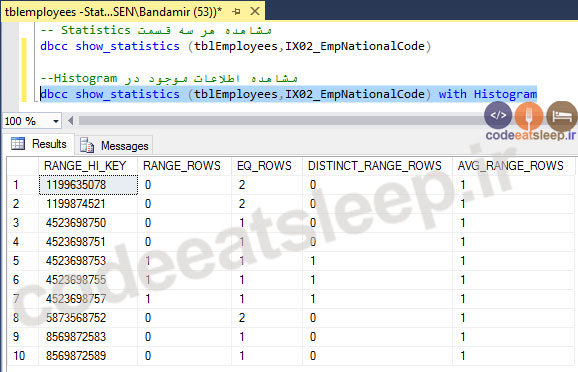
statistics-Histogram-notice1نکته اول: به هر سطر(رکورد) در Histogram، یک Step گفته می شود. هر Histogram حداکثر تا 200 عدد Step می تواند نگهداری کند.

با توجه به نکته بالا، زمانی که تعداد رکوردهای جدول ما زیاد باشد(بیش از 200 عدد)، در Histogram از رنج بندی استفاده می شود. به عبارتی در هر سطر (Step) از Histogram اطلاعات مربوط به چند مقدار نگهداری می شود.

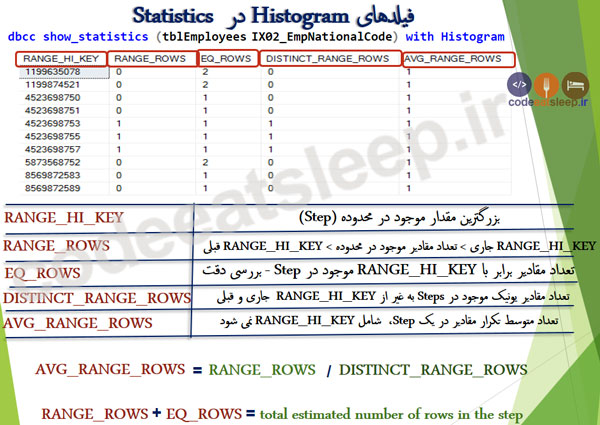
مشاهده اطلاعات **Statistics**با استفاده از ویزارد:



مشاهده اطلاعات **Statistics**با استفاده از DBCC Command:



 و اما معرفی هر کدام از فیلدهای موجود در Histogram:



برای اینکه نحوه محاسبه اعداد بالا رو بهتر متوجه شویم، در ابتدا محتویات جدول tblEmployees رو با هم می بینیم:



در اولین سطر(Step) از Histogram داریم:

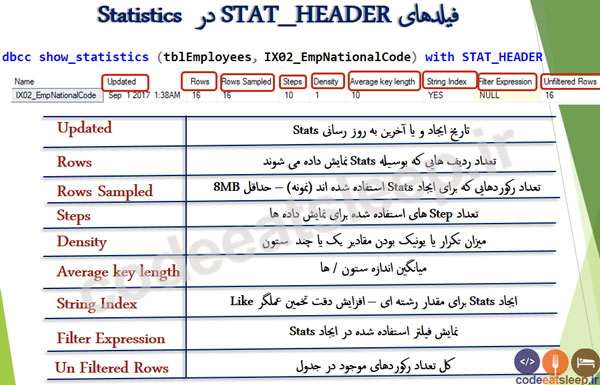
## Statistics - اولین سطر Histogram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AVG\_RANGE\_ROWS | DISTINCT\_RANGE\_ROWS | EQ\_ROWS | RANGE\_ROWS | RANGE\_HI\_KEY |
| 1 | 0 | 2 | 0 | 1199635078 |

بزرگترین مقدار موجود در این محدوده(RANGE\_HI\_KEY)، کد ملی 1199635078 است و دلیل اینکه در فیلد EQ\_ROWS(تعداد مقادیر برابر با بزرگترین مقدار موجود در محدوده) عدد 2 قرار دارد وجود دو فرد با این کد ملی(با کادر سبز رنگ مشخص شده اند) در جدول tblEmployees می باشد. مجموع دو مقدار RANGE\_ROWS و EQ\_ROWS هم تعداد رکوردهای موجود در این Step را نمایش می دهد که برابر با 2 است. یک نکته در مورد سطر پنجم(4523698753): در این سطر اگر دقت کنید، مجموع RANGE\_ROWS و EQ\_ROWS  برابر با 2 می شود به این معنا که در این سطر، اطلاعات دو رکورد نگهداری شده است(کدهای ملی 4523698753 و 4523698752) پس ایجاد محدوده(Range) در حالتی که رکوردها کمتر از 200 عدد باشند هم ممکن است رخ دهد. مابقی سطرها را به عنوان تمرین، بررسی و تحلیل کنید.

قسمت بعدی در STAT\_HEADER،  **Statistcs** است که اطلاعاتی مانند، آخرین زمان بروزرسانی، تعداد کل رکوردهای موجود، تعداد Stepها و ... در یک **Statistics** را نمایش می دهد:

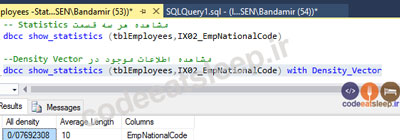
در شکل زیر، کلمه Stats مخفف  **Statistcs** است



 آخرین قسمت **Statistcs**  در واقع دومین قسمت آن Density Vector است که شامل سه فیلد: All density، Average Length و Columns می باشد. فیلد All density در واقع میزان density اطلاعات جدول را مشخص می کند که برابر است با: "تعداد مقادیر یونیک /  1" در جدول tblEmployees تعداد کل رکوردها برابر با 16 است و 3 عدد رکورد تکراری هم داریم، در نتیجه :

All density=1 / 13 = ۰/۰۷۶۹۲۳۰۷۶۹۲۳۰۷۶۹

فیلد Average Length هم میانگین طول فیلد/های **Statistcs** هستند و نهایتاً فیلد Columns هم نام فیلد/هایی است که **Statistcs** به ازای آن ها ایجاد شده است:

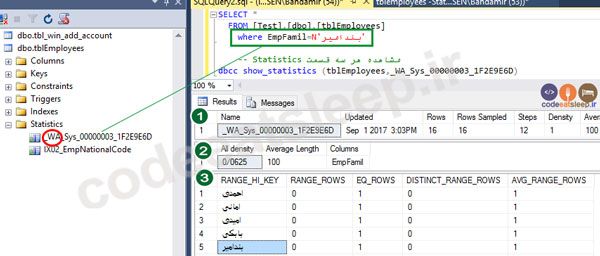


statistics-density-vector-notice1نکته دوم: مقدار موجود در فیلد All density، در شرایط خاصی، توسط sql server جهت تخمین تعداد رکوردهای بازگشتی استفاده می شود.

## STATISTICS چیست - ایجاد و بروزرسانی

زمانی که یک جدول را برا ی اولین بار ایجاد می کنیم(بدن کلید اصلی و ایندکس) هیچ **Statistcs** به ازای آن وجود ندارد. ایجاد شدن **Statistcs** در سه حالت زیر رخ می دهد:

1. به صورت خودکار توسط sql server در واقع optimizer  مثلا زمانی که ما یک کوئری داریم و در قسمت شرط آن فیلدی وجود دارد که از قبل برای آن **Statistcs** نداشته باشیم، sql server به صورت خودکار برای این فیلد یک **Statistcs** ایجاد می کند. نام این نوع **Statistcs**  با WA\_ شروع می شود.



statistics-creation-notice1نکته اول: حالت ایجاد **Statistcs** به صورت خودکار، تنها برای یک فیلد انجام می شود. پس در این حالت **Statistcs** شامل چند ستون multi column stats نداریم.

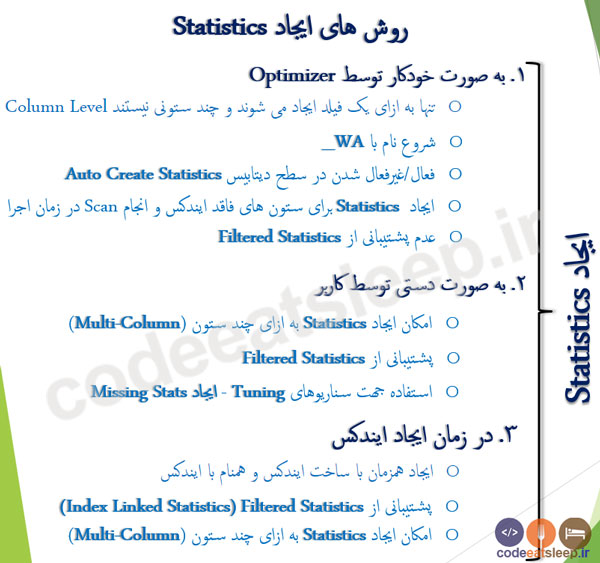
1. به صورت دستی توسط کاربر: در برخی مواقع، بویژه سناریوهای بهینه سازی، ممکن است لازم شود تا خودمان به صورت دستی یک **Statistcs** ایجاد کنیم. مثلا یک **Statistcs** چند ستونی

|  |
| --- |
| Create Statistics sf\_EmpId\_EmpFamil On dbo.tblEmployees(EmpId,EmpFamil) |

دستور بالا باعث ایجاد یک  **Statistcs** با نام sf\_EmpId\_EmpFamil روی فیلدهای EmpId , EmpFamil برای جدول tblEmployees می شود.

1. در زمان ایجاد ایندکس: همزمان با ایجاد ایندکس برای یک جدول، یک **Statistcs**  همنام با آن ایندکس و به ازای فیلد/های ایندکس ایجاد می شود.(این مورد را در مثال قبل دیدیم)

در شکل زیر خلاصه ای از ویژگی های این سه نوع **Statistcs** را مشاهده می کنید:



با مرور مطالب نحوه ایجاد **Statistcs،**این سؤال مطرح می شود که اگر فرض کنیم در زمان ایجاد ایندکس و ساخت  **Statistcs** مربوط به آن، جدول ما تنها 10 یا 20 رکورد داشته است و بعد از مدتی تعداد رکوردها به 1000 عدد برسد، اطلاعات وجود در **Statistcs** چه زمانی و چگونه بروزرسانی می شوند؟

بروزرسانی **Statistcs** دارای حد آستانه می باشد، به عبارتی متناسب با تعداد رکورد موجود در جدول در زمان ایجاد **Statistcs** ، باید تعداد تغییرات هرگونه عملیات DML شامل insert- update - delete) در جدول به یک حد آستانه (Threshold) برسد تا اطلاعات موجود در **Statistcs** بروزرسانی شود.

## بروزرسانی STATISTICS

|  |  |
| --- | --- |
| زمان بروز رسانی Statistics – حد آستانه | تعداد رکورد جدول در زمان ایجاد |
| با درج اولین رکورد در جدول | فاقد رکورد |
| رخ دادن بیش از 500 تغییر در جدول | کمتر از 500 عدد |
| رخ دادن 500 به علاوه 20% کل رکوردهای جدول،  تغییر در جدول | مساوی یا بیشتر از500 عدد |

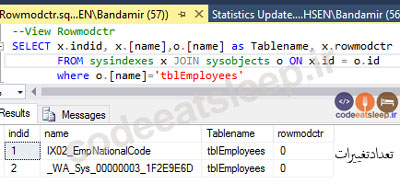
 به عنوان مثال اگر در زمان ایجاد ایندکس IX02\_EmpNationalCode، تعداد رکوردهای موجود در جدول برابر با 600 عدد بوده باشد، در نتیجه باید تعداد تغییرات در جدول به:

 620 = 500 +100/(600\*20) عدد برسد تا اطلاعات موجود در **Statistcs**مربوط به این ایندکس بروزرسانی شود. پس تا قبل از اینکه تغییرات به این عدد برسد، همچنان از اطلاعات قبلی موجود در **Statistcs**جهت تخمین تعداد رکورد بازگشتی استفاده می شود.

statistics-update-notice1نکته دوم: بروزرسانی **Statistics** به محض اجرای یک کوئری بعد از رسیدن تغییرات به حد آستانه (Threshold) انجام می شود.

statistics-update-notice1نکته سوم: تعداد تغییرات اعمال شده بعد از ایجاد هر **Statistics** در ستونی به نام Rowmodctr در ویوهای سیستمی نگهداری می شود:

|  |
| --- |
| --View Rowmodctr  SELECT x.indid, x.[name],o.[name] as Tablename, x.rowmodctr  FROM sysindexes x JOIN sysobjects o ON x.id = o.id  where o.[name]='tblEmployees' |



علاوه بر حد آستانه تغییرات، عوامل دیگری مانند Rebuild کردن ایندکس باعث بروزرسانی **Statistics**  می شود. لازم است به این نکته اشاره کنم که مبحث بروزسانی **Statistics** در آموزشی جداگانه مورد بررسی قرار می گیرد و کلیه نکات مربوط به آن به همراه اسکریپت های لازم جهت بروزرسانی، ارائه خواهند شد.

## بروز رسانی دستی

|  |
| --- |
| UPDATE STATISTICS [dbo].[ProductFavorite]  DECLARE @name NVARCHAR(MAX)    DECLARE Table\_Cursor CURSOR FOR  SELECT DISTINCT '['+TABLE\_SCHEMA+'].['+TABLE\_NAME+']' FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE='BASE TABLE'  OPEN Table\_Cursor FETCH NEXT FROM Table\_Cursor INTO @name  WHILE (@@FETCH\_STATUS = 0)  BEGIN  EXECUTE('UPDATE STATISTICS '+@name)    FETCH NEXT FROM Table\_Cursor INTO @name  END CLOSE Table\_Cursor DEALLOCATE Table\_Cursor |

## STATISTICS چیست - جمع بندی

همانطور که در ابتدای این آموزش اشاره شد، مبحث **Statistics** ها در sql server دو ویژگی پیچیدگی و وسعت مطلب را دارا هستند. با توجه به اهمیت شناخت **Statistics**ها در sql server و نیز کاربرد وسیع آن ها در آموزش های آتی ما، بویژه در مباحث افزایش سرعت و کارایی، سعی بر این است که اکثر مطالب و آموزش های لازم **Statistics** را پوشش دهیم.

در آموزش **STATISTICS چیست**تلاش شد تا با استفاده از مثالی عملی مفهوم **Statistics** و کاربد آن برای شما توضیح داده شود و به تدریج مباحثی تخصصی در رابطه با **Statistics** عنوان شد. نهایتاً با کالبدشکافی **Statistics** دیدیم که چه اطلاعاتی در آن ذخیره می شود و چگونه sql server جهت تخمین تعداد رکورد بازگشتی یک کوئری، از این اطلاعات استفاده می کند. در پایان نیز با سه روش ایجاد **Statistics** و نحوه بروزرسانی آن ها آشنایی پیدا کردیم.