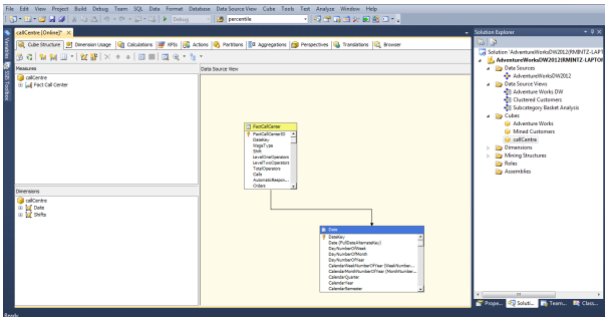
Richard Mintz's BI Blog

Percentiles like Excel to the 4th power (T-SQL, SQL CLR, MDX, ASSP)

JUNE 7, 2012 | RICHARD MINTZ
With the release of SQL 2012 additional analytic functions have been added to the SQL server T-SQL toolkit including 3 Percentile
functions: PERCENTILE_CONT; PERCENTILE_DISC; PERCENTILE_RANK, sadly no new MDX Percentile functions where added. So I thought I would share 4 ways to calculate percentiles just like Excel using: MDX; SSAS Stored Procedures; T-SQL and SQL CLR Aggregates.

The example queries in this post use the AdventureWorksDW2012 database and the AdventureWorksDW2012 UDM Analysis Services Database, with the primary tables being FactCallCenter and DimDate. The goal was to find the the 90th percentile for the 'average issue wait time' by shift. To ensure that the data used for SSAS and SQL examples was the same I created a very simple cube called callcentre in the AdventureWorksDW2012 model, see screen shot below, the cube contains one measure and two dimensions

- Measures
  - Average Time Per Issue
- Dimensions
  - Date (existing date dimension)
  - Shifts (degenerate from the shifts column in the FactCallCentre Table)



(https://richmintzbi.files.wordpress.com/2012/06/callcentre.png)

The source for percentile knowledge was Wikipedia, come on it's on the internet it must be true, both Excel percentile formulas percentile.inc and pecentile.exc are defined here (http://en.wikipedia.org/wiki/Percentile) under the **Alternative methods** heading.

Basically according to Wikipedia calculating an Excel like percentile comes down to two steps:

– Get the ordinal of the value with the following formulas

- **Exclusive**

$$n = \frac{P}{100}(N + 1)$$

- **Inclusive**

$$n = \frac{P}{100}(N - 1) + 1$$

– Calculate the percentile Value by splitting the ordinal into its integer component *k* and decimal component *d*,
such that $n = k + d$. Then v_P is calculated as:

$$v_P = \begin{cases} v_1, & \text{for } n = 1 \\ v_N, & \text{for } n = N \\ v_k + d(v_{k+1} - v_k), & \text{for } 1 < n < N \end{cases}$$

Here Goes...

T-SQL

the code snippet below displays the query used to calculate a percentile value using the exclusive method

```sql
/*steps
1. determine the index where percentile value is located
2. Calculate the percentile value
*/

declare @percentile numeric(19,2)
set @percentile = .90

;with percentile (rowIndex, rangePoint, [Shift], AverageTimePerIssue
(
select
 row_number() over (partition by [Shift] order by AverageTimePerIssu
 ((count(1) over (partition by [Shift])+1) * @percentile) as rangePo
 a.[Shift], AverageTimePerIssue from FactCallCenter a
where AverageTimePerIssue is not null
)

select [Shift],
case when max(rangePoint) % 1 <> 0 THEN
 CAST(ROUND(MIN(AverageTimePerIssue) +
 (
 (MAX(AverageTimePerIssue) - MIN(AverageTimePerIssue))
 * (max(rangePoint) % 1)),2) as Numeric(19,2))
ELSE
 CAST(round(max(AverageTimePerIssue),2) as numeric(19,2)) END as per
percentile
where rowIndex between Floor(rangePoint) and ceiling(rangePoint)
group by [shift],rangePoint
```

if you want to use the inclusive method you would modify the *((count(1) over (partition by [Shift])+1) * @percentile)* portion of the query to read *((count(1) over (partition by [Shift])-1) * .9)+1*

SQL CLR Aggregate

While a T-SQL script works it requires a lot of repetitive typing for multiple calculations and is not very reusable, I mean you could always bookmark this post or save the script as a stub for reuse, but I thought that a CLR Aggregate would be a better solution. As you can see selecting a percentile using an aggregate function requires much less typing and allows for reuse of the function without having to re write the implementation.

```sql
select [shift], [dbo].[percentiles]([AverageTimePerIssue], 90, 'EXC')
group by [shift]
```

The Aggregate Function takes three parameters:

1. The column to evaluate the percentile value from
2. The percentile to find as an integer
3. The calculation Method {‘EXC’, ‘INC’}

To create the CLR aggregate function you can create a new SQL Server Database Project if you are using Visual Studio 2010 and have installed the new SQL CLR Database Project template which allows you to deploy to SQL 2012 or a new Visual C# SQL CLR Database Project if you are using VS 2010 with SQL 2005 or 2008, these projects will deploy your function directly to

SQL Server. If you do not want to create a database project you can also simply created a new class library, then compile and register your assembly and function using SQL scripts (see script below). If you do not want to recreate the function you can download the compiled .dll here (http://sdrv.ms/LsIiKn) and execute the code below, assuming you saved the file to 'c:\Temp' , in the context of the database where you want to register the assembly and create the function

```
EXEC sp_configure 'show advanced options' , '1';
go
reconfigure;
go
EXEC sp_configure 'clr enabled' , '1'
go
reconfigure;
-- Turn advanced options back off
EXEC sp_configure 'show advanced options' , '0';
go

CREATE ASSEMBLY [sql.clr.percentiles] from 'c:\Temp\sql.clr.percenti
WITH PERMISSION_SET = SAFE
GO

CREATE AGGREGATE [dbo].[percentiles]
(@value [float], @percentile [smallint], @calcMethod [nvarchar](4000
RETURNS[float]
EXTERNAL NAME [sql.clr.percentiles].[Percentiles]
GO
```

Below you will find the code for the aggregate, the code basically performs the same operations as the SQL Equivalent, it finds the ordinal based on the previously defined formulas and then returns the percentile value using the formula as shown earlier.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.IO;
using Microsoft.SqlServer.Server;
[Serializable]
[Microsoft.SqlServer.Server.SqlUserDefinedAggregate(Format.UserDefi
public struct Percentiles : Microsoft.SqlServer.Server.IBinarySeria
{
 public void Init()
 {
 valueList = new List();
 }

public void Accumulate(SqlDouble value, SqlInt16 percentile, SqlStr
 {
 if (!value.IsNull)
 {
 valueList.Add(value.Value);
 }

if (!percentile.IsNull)
 {
 PercentileValue = percentile.Value > 100 ? (short)100 : percentile
 }
```

```
29   Method = calcMethod.ToString();
30   }
31
32  public void Merge(Percentiles Group)
33   {
34   foreach (double d in Group.valueList)
35   {
36   valueList.Add(d);
37   }
38   PercentileValue = Group.PercentileValue;
39   }
40
41  public SqlDouble Terminate()
42   {
43   double rangePoint = 0;
44   switch (Method)
45   {
46   case "EXC":
47   {
48   rangePoint = (((Convert.ToDouble(PercentileValue) / 100) * (valueL
49   break;
50   }
51   case "INC":
52   {
53   rangePoint = (((Convert.ToDouble(PercentileValue) / 100) * (valueL
54   break;
55   }
56   default:
57   {
58   return SqlDouble.Null;
59   }
60   }
61   valueList.Sort();
62
63   //if rangePoint is a whole number
64   if ( rangePoint % 1 == 0)
65   {
66   return valueList[(int)rangePoint];
67   }
68   else if ((int)Math.Ceiling(rangePoint) < valueList.Count)
69   {
70   return valueList[(int)Math.Floor(rangePoint)] +
71   (rangePoint % 1 * (valueList[(int)Math.Ceiling(rangePoint)] - valu
72   }
73   else
74   {
75   return SqlDouble.Null;
76   }
77   }
78
79  public void Read(BinaryReader binaryReader)
80   {
81   if (valueList == null)
82   {
83   valueList = new List();
84   }
85
86  Method = binaryReader.ReadString();
87
88   PercentileValue = binaryReader.ReadInt16();
89
```

```
long readerLength = binaryReader.BaseStream.Length;
 while (binaryReader.BaseStream.Position  {
 valueList.Add(binaryReader.ReadDouble());
 }
 }

public void Write(BinaryWriter binaryWriter)
 {
 binaryWriter.Write(Method);
 binaryWriter.Write(PercentileValue);
 foreach (double d in valueList)
 {
 binaryWriter.Write(d);
 }
 }

private string Method { get; set; }
 private short PercentileValue{ get; set; }
 private List valueList { get; set; }
}
```

MDX

calculating a percentile in MDX follows the same pattern; get the ordinal from your ordered set and calculate the value as seen in the code below. In an effort to keep the MDX readable, allow for easier troubleshooting and better performance I broke the calculation into several calculated measures with the final calculated measure ‘Percentile’ using the measures defined earlier in the script, thanks to Chris Webb (cwebbbi.wordpress.com) and Greg Galloway (http://www.artisconsulting.com/blogs/greggalloway/default.aspx) for their MDX optimization suggestions.

The code below provides the MDX used to generate a percentile value.

```
//number of items in the set
WITH MEMBER [measures].[Count]
as COUNT(
NONEMPTY(({[Shifts].currentmember}*
{[Date].[Day of Month].[Day of Month].members})
 ,[Measures].[Average Time Per Issue]))

//percentile index

MEMBER [Measures].[RangePoint] AS
(([measures].[Count]+1) *.90) -1 /* subtract 1 from total as item ar

MEMBER [Measures].[RangePoint_Int]
 AS
 INT([Measures].[RangePoint])

member Measures.[floor] as
 (BOTTOMCOUNT(NONEMPTY(({[Shifts].currentmember}*
{[Date].[Day of Month].[Day of Month].members})
 ,[Measures].[Average Time Per Issue]), [measures].[Count], [Measure
 item(([Measures].[RangePoint_int])),
 [Measures].[Average Time Per Issue])

member Measures.[Ceiling] as
(BOTTOMCOUNT(NONEMPTY(({[Shifts].currentmember}*
{[Date].[Day of Month].[Day of Month].members})
 ,[Measures].[Average Time Per Issue]), [measures].[Count], [Measure
 item(([Measures].[RangePoint_int]+1)),
 [Measures].[Average Time Per Issue])

MEMBER [Measures].[Percentiles] AS
IIF([Measures].[RangePoint] - [Measures].[RangePoint_Int] = 0,
//rangepoint is a whole number
(mySet.item([Measures].[RangePoint]-1),
 [Measures].[Average Time Per Issue]),
 //rangepoint is not a whole number
 Measures.[floor]
+
([Measures].[RangePoint] - [Measures].[RangePoint_Int])
*
(
Measures.[ceiling]
-
measures.[floor])
)
select {[Measures].[Percentiles]} on 0
,
([Shifts].[Shift].[Shift].members) on 1
from callcentre
```

To switch to the inclusive method you would modify the rangePoint calculated measure to read

```
MEMBER [Measures].[RangePoint] AS
((([measures].[Count]-1) *.90)+1) -1
```

SSAS Stored Procedures (ASSP)

Like T-SQL, the MDX solution requires lots of typing and reuse requires re-implementation; if you type anywhere like I do extra typing means lots of room for error. Fortunately SSAS provides a mechanism for creating reusable user defined functions, Analysis Services Stored Procedures.

To use the predefined Stored Procedure requires a fraction of the code and no need to re-implement the logic

```
1  with member measures.[90th percentile] as
2  [ssas storedprocedures].ValueAtPercentile(
3  NONEMPTY(([Date].[Day of Month].[day of month].members, [Measures].[A
4   [Measures].[Average Time Per Issue], .90, true, "EXC")
5
6  select measures.[90th percentile] on 0,
7   [Shifts].[Shift].[Shift].members on 1
8  from callcentre
```

The function takes 5 Parameters:

1. The set to evaluate
2. The measure to get the percentile of
3. The percentile to find as a decimal
4. The order the set{true, false}
5. The calculation Method{‘EXC’, ‘INC’}

It should be noted that ASSP’s used within calculated measures may not perform as well as straight MDX, however for me having both options available is really useful. I did some high level testing with these scripts (http://sdrv.ms/NNXaHs) using the SQL 2012 Adventure Works Cube and found the ASSP’s performed up to 14 time faster than the straight MDX scripts, bonus.

- Using my laptop, fully loaded with 3 instances of SQL Server including 2012 with all SSAS flavours, the ASSP executed in two seconds for both the Inclusive and Exclusive modes where the straight MDX took 28 seconds for Inclusive and 9 Seconds for Exclusive

My work on this stored procedure was heavily influenced by the Analysis Services Stored Procedure Project on Codeplex (//asstoredprocedures.codeplex.com), it was a huge help in the learning how to write ASSP’s and it provided an Order() function that I was able to reuse. I have submitted the function to the ASSP Project Authors in hopes that it may be included there. For those of you that do not want to create the ASSP you can download the compiled .dll here (http://sdrv.ms/MeFtkS) and follow the steps outlined at the end of this section to add the assembly to Analysis Services.

If you want to create your own ASSP Library you can follow the steps below.

- create a new C# Class Library project in Visual Studio
- add the following references
  - Microsoft.AnalysisServices (C:\Program Files (x86)\Microsoft SQL Server\110\SDK\Assemblies\Microsoft.AnalysisServices.DLL)
  - msmgdsrv (C:\Program Files (x86)\Microsoft Analysis Services\AS OLEDB\10\msmgdsrv.dll)
- Set the target framework project property to .Net Framework 3.5
- In project properties, Sign the assembly
- Rename the Class1.cs to something meaningful (in my solution it is called percentiles.cs)

- Add the following using statements

```
using Microsoft.AnalysisServices.AdomdServer;
using Microsoft.AnalysisServices;
using System.Collections.Generic;
```

Download the following file (http://sdrv.ms/NOmLQu) and add it to your solution (setOrdering Class from ASSP codeplex project)

- Replace all the code under the using statements with the following

```
namespace ssas.storedprocedures
{
 public class PercentileFunctions
 {
 [SafeToPrepare(true)]
 public static double RangePoint(Set inputSet, double percentileValu
 {
 switch (calcMethod)
 {
 case "EXC":
 {
 return ((percentileValue) * (inputSet.Tuples.Count + 1)) - 1;
 }
 case "INC":
 {
 return (((percentileValue) * (inputSet.Tuples.Count -1))+1) - 1;
 }
 default:
 {
 return ((percentileValue) * (inputSet.Tuples.Count + 1)) - 1;
 }
 }
 }

[SafeToPrepare(true)]
 public static double ValueAtPercentile(Set inputSet, Expression sor
 {
 //get position where percentile falls
 double Rank = RangePoint(inputSet, percentileValue, calcMethod);

 //order the set ascending using Codeplex SSAS Stored Procedure Func
 if (sortAscending)
 {
 Set s = setOrdering.Order(inputSet, sortExpression);

//if the Rank is a whole number
 if ((Rank % 1) == 0)
 {
 return sortExpression.Calculate(s.Tuples[Convert.ToInt32(Rank)]).To
 }
 //if Rank is a decimal
 else
 {
 return
 (sortExpression.Calculate(s.Tuples[(int)Math.Floor(Rank)]).ToDouble
 +
 (Rank % 1 *
 (sortExpression.Calculate(s.Tuples[(int)Math.Ceiling(Rank)]).ToDoub
 - sortExpression.Calculate(s.Tuples[(int)Math.Floor(Rank)]).ToDoubl
 );
```

```
51    /*(((sortExpression.Calculate(s.Tuples[Convert.ToInt32(Rank) + 1]).
52    - sortExpression.Calculate(s.Tuples[Convert.ToInt32(Rank)]).ToDoubl
53    }
54    }
55    else {
56    //if the Rank is a whole number
57    if ((Rank % 1) == 0)
58    {
59    return sortExpression.Calculate(inputSet.Tuples[Convert.ToInt32(Ran
60    }
61    //if Rank is a decimal
62    else
63    {
64    return
65    (sortExpression.Calculate(inputSet.Tuples[Convert.ToInt32(Rank)]).T
66    +
67    (((sortExpression.Calculate(inputSet.Tuples[Convert.ToInt32(Rank) +
68    - sortExpression.Calculate(inputSet.Tuples[Convert.ToInt32(Rank)]).
69    }
70    }
71    }
72  }
73  }
```

- Compile the Code
- Open SSMS, connect to Analysis Services
- Right click on the Assemblies folder and select New Assembly
- Browse to the compiled .dll
- check the include debug information if you want debug the function (how to debug link (http://msdn.microsoft.com/en-us/library/ms174763.aspx))
- use in a mdx query

Happy querying.

POSTED IN ANALYSIS SERVICES, SQL, SQL 2012, SQL CLR
CLR AGGREGATES MDX MDX PERCENTILES PERCENTILES SQL SQL 2012 SQL CLR PERCENTILES SSAS PERCENTILES SSAS STORED PROCEDURE PERCENTILES SSAS STORED PROCEDURES

19 thoughts on “Percentiles like Excel to the 4th power (T-SQL, SQL CLR, MDX, ASSP)”

1. **IGOR**
   SAYS:
   JULY 30, 2012 AT 11:32 AM
   Thanks Richard. This is a great example. And just in time as I was just looking for a percentile calculation in MDX.

   Reply
2. Pingback: Calcul de centile et quantile en MDX « Business Geek
3. Pingback: What can I say, I’m proud « Richard Mintz's BI Blog
4. **ZENTROPY**
   SAYS:
   DECEMBER 22, 2012 AT 10:53 AM
   This is exactly what I’ve been looking for. Unfortunately I can’t get it to work. When I run the following against my sample cube:

   with
   member [Measures].[Percentiles] AS
   [ASSP].ValueAtPercentile(
   NonEmpty([Dim Country].[Country ID].[Country ID].[Algeria], [Measures].[Base Salary])
   , [Measures].[Base Salary]
   , .90
   , true
   , “INC”
)
   select {[measures].[percentiles]} on 0
   from [DWtest];

   it simply gives me the sum. I get the same result no matter what I change the percentile number to

   Reply
   1. **ZENTROPY**
      SAYS:
      DECEMBER 22, 2012 AT 12:41 PM
      I also tried to debug the assembly to try to figure out what’s going on but I get nothing in VS when I execute the query. I guess it’s not hitting my breakpoints.

      Reply
   2. **RICHARD MINTZ**
      SAYS:
      DECEMBER 22, 2012 AT 4:18 PM
      Couple Questions
      – Are you able to send me a backup of the cube?
      – What happens if you add a dimension on Rows?

Richard

Reply

1. **ZENTROPY**
   SAYS:
   DECEMBER 24, 2012 AT 9:35 AM
   Yes I can, where should I send it? Adding dimensions makes no difference. I still get the sum.

2. **RICHARD MINTZ**
   SAYS:
   DECEMBER 24, 2012 AT 11:11 AM
   Hi,

   If you have a skydrive can you save the backup there and send me a link to it at richmintz@rogers.com

   Thanks,

3. **RICHARD MINTZ**
   SAYS:
   DECEMBER 27, 2012 AT 10:39 AM
   Hi,

   The Original Member that you created passes in a single member set to the sp and will only ever evaluate to the sum of the base salaries for Algeria, if you are interested in seeing the 90th percentile base salary for Algeria you would need the following query

```
with
member [Measures].[Percentiles] AS
[ASSP].ValueAtPercentile(
NonEmpty( ([Dim Country].[Country ID].[algeria],[Dim Jobs].[Job Name].[job
Name].members),
[Measures].[Base Salary])
, [Measures].[Base Salary]
, .90
, true
, “INC”
)
select {[measures].[percentiles]} on 0
from [DWtest];
```

   This Query Below will give you the 90th Percentile base Salary for jobs across countries

```
with
member [Measures].[Percentiles] AS
[ASSP].ValueAtPercentile(
NonEmpty( [Dim Country].[Country ID].[Country ID].members, [Measures].[Base
Salary])
, [Measures].[Base Salary]
, .90
, true
, “INC”
```

)
select {[measures].[percentiles]} on 0,
[Dim Jobs].[Job Name].[job Name].members on 1 from [DWtest];

I hope that helps.

Richard

Reply

1. **ZENTROPY**
   SAYS:
   DECEMBER 27, 2012 AT 10:54 AM
   That helps a great deal. I'm obviously just wrestling with my own limited knowledge of MDX. Thanks for taking the time to take a look at this and Happy Holidays!

5. **JCHIU**
   SAYS:
   FEBRUARY 15, 2013 AT 9:18 AM
   Most of my research on creating a percentile measure in cubes lead me back to this post. Your explanation and walkthrough was very helpful as I am still working on my MDX knowledge.

   I was hoping you might shed some light in this issue I keep running into after importing the value at percentile stored procedure.

   I was able to run this MDX query without error.

   with member measures.[90th percentile] as

   [assp].ValueAtPercentile(
   NONEMPTY([Patient Number].[Vw Patient Info].members, [Measures].[Est Paid Amt]),
   [Measures].[Est Paid Amt],.90, true, "EXC")

   select measures.[90th percentile] on 0
   from PremiumMedexp

   However, when I tried to create a calculated measure in the cube with

   [assp].ValueAtPercentile(
   NONEMPTY([Patient Number].[Vw Patient Info].members, [Measures].[Est Paid Amt]),
   [Measures].[Est Paid Amt],.90, true, "EXC")

   I kept getting a syntax error at '.90' from the MDX parser.
   I will really appreciate any thought on this.

   Reply

   1. **RICHARD MINTZ**
      SAYS:
      FEBRUARY 15, 2013 AT 9:34 AM
      Can you go to the script view in the calculations editor in BIDS and reply with the code that is creating your calc measure?

      Thanks

      Reply

6. **STEVEN THOMAS**
SAYS:
APRIL 1, 2014 AT 10:59 AM
Thanks. Very useful.

I got strange results using the 'INC' parameter.
Looking at the code for INC:

case "INC":
{
return (((percentileValue) * (inputSet.Tuples.Count -1))+1) – 1;

I would have expected:

case "INC":
{
return (((percentileValue) * (inputSet.Tuples.Count -1))+1) ;

For example if you have 17 numbers and want the .25 percentile your calculation gives (((.25)*(17-1))+1)-1 = 4 for the rangepoint

But the right rangepoint is 5 isn't it?

Thanks

Reply

1. **RICHARD MINTZ**
SAYS:
APRIL 1, 2014 AT 4:41 PM
The reason for subtracting 1 is because the item array is 0 based.

Reply

1. **STEVEN THOMAS**
SAYS:
APRIL 2, 2014 AT 3:03 AM
Of course. Thanks for the quick reply!

7. **MICHAEL**
SAYS:
JANUARY 9, 2015 AT 10:08 AM
I got this error in the SQL Server version even though the field that is passed to the percentile function IS NOT NULL. I am testing this on a record set of over 29 million records

Msg 6522, Level 16, State 1, Line 4
A .NET Framework error occurred during execution of user-defined routine or aggregate "percentiles":
System.ArgumentNullException: Value cannot be null.
Parameter name: value
System.ArgumentNullException:
at System.IO.BinaryWriter.Write(String value)
at Percentiles.Write(BinaryWriter binaryWriter)

Reply

1. **RICHARD MINTZ**
   SAYS:
   JANUARY 20, 2015 AT 1:15 PM
   Can you include your query?

   Reply
   1. **MICHAEL**
      SAYS:
      JANUARY 21, 2015 AT 9:46 AM
      select 'Cnty' GEO,'Month' Time,Cnty_cd Geographic, SLS_RCDED_PERIOD YearMonth, PropType

      , util.[dbo].[percentiles](TotalAssessedValue, 10, 'INC') AS ASSD_PCT10
      , util.[dbo].[percentiles](TotalAssessedValue, 20, 'INC') AS ASSD_PCT20
      , util.[dbo].[percentiles](TotalAssessedValue, 25, 'INC') AS ASSD_PCT25
      , util.[dbo].[percentiles](TotalAssessedValue, 30, 'INC') AS ASSD_PCT30
      , util.[dbo].[percentiles](TotalAssessedValue, 40, 'INC') AS ASSD_PCT40
      , util.[dbo].[percentiles](TotalAssessedValue, 50, 'INC') AS ASSD_PCT50
      , util.[dbo].[percentiles](TotalAssessedValue, 60, 'INC') AS ASSD_PCT60
      , util.[dbo].[percentiles](TotalAssessedValue, 70, 'INC') AS ASSD_PCT70
      , util.[dbo].[percentiles](TotalAssessedValue, 75, 'INC') AS ASSD_PCT75
      , util.[dbo].[percentiles](TotalAssessedValue, 80, 'INC') AS ASSD_PCT80
      , util.[dbo].[percentiles](TotalAssessedValue, 90, 'INC') AS ASSD_PCT90

      from
      Sand_HealthySales with(nolock)
      where PropType = 'ALL' and TotalAssessedValue is not null
      group by Cnty_cd, SLS_RCDED_PERIOD, PropType
      order by Cnty_cd, SLS_RCDED_PERIOD, PropType

      FYI: TotalAssessedValue is a decimal field

8. Pingback: Bandlines in SSRS