/\*

• CUME\_DIST()

• FIRST\_VALUE()

• LAST\_VALUE()

• LAG()

• LEAD()

• PERCENT\_RANK()

• PERCENTILE\_CONT()

• PERCENTILE\_DISC()

\*/

***Listing 3.1a – a Simple Example***

USE TEST

GO

DECLARE @CumDistDemo TABLE (

Col1 VARCHAR(8),

ColValue INTEGER

);

INSERT INTO @CumDistDemo VALUES

('AAA',1),

('BBB',2),

('CCC',3),

('DDD',4),

('EEE',5),

('FFF',6),

('GGG',7),

('HHH',8),

('III',9),

('JJJ',10)

SELECT Col1,ColValue,

CUME\_DIST() OVER(

ORDER BY ColValue

) AS CumeDistValue,

A.RowCountLE,

B.TotalRows,

CONVERT(DECIMAL(10,2),A.RowCountLE)

/ CONVERT(DECIMAL(10,2),B.TotalRows) AS MyCumeDist

FROM @CumDistDemo CDD

CROSS APPLY (

SELECT COUNT(\*) AS RowCountLE FROM @CumDistDemo

WHERE ColValue <= CDD.ColValue

) A

CROSS APPLY (

SELECT COUNT(\*) AS TotalRows FROM @CumDistDemo

) B

GO

***Listing 3.1b – The CUME\_DIST() function in action***

WITH CustSales (

SalesYear,SalesQuarter,SalesMonth,CustomerNo,StoreNo,CalendarDate,SalesTotal

)

AS

(

SELECT YEAR(CalendarDate) AS SalesYear

,DATEPART(qq,CalendarDate) AS SalesQuarter

,MONTH(CalendarDate) AS SalesMonth

,ST.CustomerNo,

ST.StoreNo,

ST.CalendarDate,

SUM(ST.UnitRetailPrice \* ST.TransactionQuantity) AS SalesTotal

FROM StagingTable.SalesTransaction ST

GROUP BY ST.CustomerNo

,ST.StoreNo

,ST.CalendarDate

,ST.UnitRetailPrice

,ST.TransactionQuantity

)

SELECT SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

,SUM(SalesTotal) AS MonthlySalesTotal

,CUME\_DIST() OVER (

PARTITION BY SalesYear

ORDER BY SUM(SalesTotal)

) AS CumeDist

FROM CustSales

WHERE SalesYear IN(2010,2011)

AND CustomerNo = 'C00000001'

GROUP BY SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

GO

***Listing 3.2 – A Simple Percent Rank Example***

DECLARE @CumeDistDemo TABLE (

Col1 VARCHAR(8),

ColValue DECIMAL(10,2)

);

INSERT INTO @CumeDistDemo VALUES

('AAA',1.0),

('BBB',2.0),

('CCC',3.0),

('DDD',4.0),

('EEE',5.0),

('FFF',6.0),

('GGG',7.0),

('HHH',8.0),

('III',9.0),

('JJJ',10.0)

SELECT Col1,ColValue,A.RowCountLTE AS MyRank,

RANK() OVER(

ORDER BY ColValue

) AS SQLRank,

PERCENT\_RANK() OVER(

ORDER BY ColValue

) AS PCTRank,

/\* current value rank - 1 /data sample total row count - 1 \*/

(RANK() OVER(

ORDER BY ColValue

) - 1.0) / CONVERT(DECIMAL(10,2),(

SELECT COUNT(\*) AS SampleRowCount

FROM @CumeDistDemo) - 1.0

) AS MyPctRank

FROM @CumeDistDemo CDD

CROSS APPLY (

SELECT COUNT(\*) AS RowCountLTE FROM @CumeDistDemo

WHERE ColValue <= CDD.ColValue

) A

GO

***Listing 3.3 – The PERCENT\_RANK() Function in Action***

WITH CustSales (

SalesYear,SalesQuarter,SalesMonth,CustomerNo,

StoreNo,CalendarDate,SalesTotal

)

AS

(

SELECT YEAR(CalendarDate) AS SalesYear

,DATEPART(qq,CalendarDate) AS SalesQuarter

,MONTH(CalendarDate) AS SalesMonth

,ST.CustomerNo

,ST.StoreNo

,ST.CalendarDate

,SUM(ST.TotalSalesAmount) AS SalesTotal

FROM StagingTable.SalesTransaction ST

GROUP BY ST.CustomerNo

,ST.StoreNo

,ST.CalendarDate

)

SELECT SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

,SUM(SalesTotal) AS MonthlySalesTotal

,CUME\_DIST() OVER (

PARTITION BY SalesYear

ORDER BY SUM(SalesTotal)

) AS CumeDist

,PERCENT\_RANK() OVER (

PARTITION BY SalesYear

ORDER BY SUM(SalesTotal)

) AS PctRank

FROM CustSales

WHERE SalesYear IN(2010,2011)

AND CustomerNo = 'C00000001'

GROUP BY SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

GO

**--Step 1: check compatibility level**.

***Listing 3.4 – Check Compatibility Level***

SELECT d.compatibility\_level

FROM sys.databases as d

WHERE d.name = Db\_Name();

GO

**--Step 2: set parameter MEMORY\_OPTIMIZED\_ELEVATE\_TO\_SNAPSHOT to ON.**

***Listing 3.5 – Set Memory Optimized Elevate to Snapshot***

ALTER DATABASE [APSAles]

SET MEMORY\_OPTIMIZED\_ELEVATE\_TO\_SNAPSHOT = ON;

GO

**--Step 3 -Next, add a dedicated file group for memory optimized data by running the**

**--following command.**

***Listing 3.6 – Add File Group for Memory Optimized Data***

ALTER DATABASE APSales

ADD FILEGROUP APSalesMemOptimized CONTAINS MEMORY\_OPTIMIZED\_DATA;

GO

**--Step 4: create a dedicated file for memory optimized tables.**

***Listing 3.7 – Add a file to the file group***

ALTER DATABASE APSales

ADD FILE (

name='APSalesMemoOptData',

filename=N'D:\APRESS\_DATABASES\AP\_SALES\MEMORYOPT\AP\_SALES\_MEMOPT.mdf'

)

TO FILEGROUP APSAlesMemOptimized

GO

**--Step 5: create the memory optimized table.**

***Listing 3.8 – Create Memory Optimized Table***

CREATE TABLE [SalesReports].[MemorySalesTotals](

[SalesTotalKey] INTEGER NOT NULL IDENTITY PRIMARY KEY NONCLUSTERED,

[SalesYear] [int] NOT NULL,

[SalesQuarter] [int] NOT NULL,

[SalesMonth] [int] NOT NULL,

[CustomerNo] [nvarchar](32) NOT NULL,

[StoreNo] [nvarchar](32) NULL,

[CalendarDate] [date] NOT NULL,

[SalesTotal] [decimal](21, 2) NULL

)

WITH (

MEMORY\_OPTIMIZED = ON,

DURABILITY = SCHEMA\_AND\_DATA

);

GO

**--Step 6: check that it was created.**

***Listing 3.9 – Check the Filegroups and Database\_files Tables***

SELECT g.name, g.type\_desc, f.physical\_name

FROM sys.filegroups g JOIN sys.database\_files f ON g.data\_space\_id = f.data\_space\_id

WHERE g.type = 'FX' AND f.type = 2

GO

**--Step 7: Load the Memory Optimized Table.**

***Listing 3.10 – Load the Memory Optimized Table***

INSERT INTO [SalesReports].[MemorySalesTotals]

SELECT YEAR(CalendarDate) AS SalesYear

,DATEPART(qq,CalendarDate) AS SalesQuarter

,MONTH(CalendarDate) AS SalesMonth

,ST.CustomerNo

,ST.StoreNo

,ST.CalendarDate

,SUM(ST.UnitRetailPrice \* ST.TransactionQuantity) AS SalesTotal

FROM StagingTable.SalesTransaction ST

GROUP BY ST.CustomerNo

,ST.StoreNo

,ST.CalendarDate

,ST.UnitRetailPrice

,ST.TransactionQuantity

GO

**--Step 8: Check estimated query plan.**

**--Step 9: Run the Query.**

***Listing 3.11 – Query the Memory Optimized Table***

DBCC dropcleanbuffers;

CHECKPOINT;

GO

SET STATISTICS IO ON

GO

SET STATISTICS TIME ON

GO

SELECT SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

,SUM(SalesTotal) AS MonthlySalesTotal

,CUME\_DIST() OVER (

PARTITION BY SalesYear

ORDER BY SUM(SalesTotal)

) AS CumeDist

,PERCENT\_RANK() OVER (

PARTITION BY SalesYear

ORDER BY SUM(SalesTotal)

) AS PctRank

FROM [SalesReports].[MemorySalesTotals]

WHERE SalesYear IN(2010,2011)

AND CustomerNo = 'C00000001'

GROUP BY SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

GO

***--Step 10: - create the suggested index.***

***Listing 3.12 – Create the Suggested Index***

ALTER TABLE SalesReports.MemorySalesTotals

ADD INDEX ieCustNoSaleYearMemTable

NONCLUSTERED (CustomerNo,SalesYear)

GO

**--Step 11: - Create a Second Estimated Index Plan**

**--Step 12: - Re-run the query and make sure all statistics are turned on**

***Listing 3.13 – The FIRST\_VALUE() & LAST\_VALUE() Function in Action***

SELECT SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

,SUM(SalesTotal) AS MonthlySalesTotal

,FIRST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

) AS SalesTotalFirstValue

,LAST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

) AS SalesTotalLastValue

,FIRST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

) -

LAST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

) AS Change

,CASE

WHEN (

FIRST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

) - -

LAST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

)

) > 0 THEN 'Sales Increase'

WHEN (

FIRST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

) -

LAST\_VALUE(SUM(SalesTotal)) OVER (

PARTITION BY SalesYear

ORDER BY SalesMonth

)

) < 0 THEN 'Sales Decrease'

ELSE 'No change'

END AS [Sales Performance]

FROM SaleReports.MemorySalesTotals

WHERE SalesYear IN(2010,2011)

AND CustomerNo = 'C00000001'

GROUP BY SalesYear

,SalesQuarter

,SalesMonth

,CustomerNo

GO

--Side comment, I added an index based on the following command:

ALTER TABLE [SalesReports].[MemorySalesTotals]

ADD INDEX ieSalesYearQuarterMonth(SalesYear,SalesQuarter,SalesMonth)

GO

***Listing 3.14 – The LAG() & LEAD() Function in Action***

SELECT

SalesYear,

SalesQuarter,

SalesMonth,

StoreNo,

ProductNo,

CustomerNo,

CalendarDate AS SalesDate,

SalesTotal,

LAG(SalesTotal) OVER (

PARTITION BY SalesYear,CustomerNo

ORDER BY SalesMonth,CustomerNo,CalendarDate

) AS LastMonthlySales,

LEAD(SalesTotal) OVER (

PARTITION BY SalesYear,CustomerNo

ORDER BY SalesMonth,CustomerNo,CalendarDate

) AS NextMonthylSales

FROM [SalesReports].[MemorySalesTotals]

WHERE StoreNO = 'S00005'

AND SalesYear = 2002

GO

***Listing 3.15 – Estimate Query Plan Tool Suggested Index***

ALTER TABLE [SalesReports].[MemorySalesTotals]

ADD INDEX ieSalesYearStoreNo

NONCLUSTERED ([SalesYear],[StoreNo])

GO

***Listing 3.16 The PERCENTILE\_CONT()& PERCENTILE\_DISC Function in Action***

DECLARE @ExampleValues TABLE (

TestKey VARCHAR(8) NOT NULL,

TheValue SMALLINT NOT NULL

);

INSERT INTO @ExampleValues VALUES

('ONE',1),('TWO',2),('THREE',3),('FOUR',4),('SIX',6),('SEVEN',7),('EIGHT',8),('NINE',9),('TEN',10),('TWELVE',12);

SELECT

TestKey,TheValue,

PERCENTILE\_CONT(.5)

WITHIN GROUP (ORDER BY TheValue)

OVER() AS PctCont, -- continuous

PERCENTILE\_DISC(.5)

WITHIN GROUP (ORDER BY TheValue)

OVER() AS PctDisc -- discrete

FROM @ExampleValues

GO

***Listing 3.17 – Percentile Continuous and Discrete Analysis***

WITH StoreSalesAnalysis (

SalesYear,SalesMonth,StoreNo,StoreName ,StoreTerritory,TotalSales

)

AS (

SELECT YEAR(CalendarDate) AS SalesYear

,MONTH(CalendarDate) AS SalesMonth

,StoreNo

,StoreName

,StoreTerritory

,SUM(TotalSalesAmount) AS TotalSales

FROM APSales.SalesReports.YearlySalesReport

GROUP BY YEAR(CalendarDate)

,MONTH(CalendarDate)

,StoreNo

,StoreName

,StoreTerritory

SELECT SalesYear

,SalesMonth

,StoreNo

,StoreName

,StoreTerritory

,FORMAT(TotalSales,'C') AS TotalSales

,FORMAT(PERCENTILE\_CONT(.5)

WITHIN GROUP (ORDER BY TotalSales)

OVER (

PARTITION BY SalesYear

),'C') AS PctCont

,FORMAT(PERCENTILE\_DISC(.5)

WITHIN GROUP (ORDER BY TotalSales)

OVER (

PARTITION BY SalesYear

),'C') AS PctDisc

FROM StoreSalesAnalysis

WHERE SalesYear IN(2010,2011)

AND StoreNo = 'S00004'

GO

***Listing 3.18 – Suggested Index***

/\*

Missing Index Details from chapter 03 - TSQL code - new - 10-06-2022.sql - DESKTOP-CEBK38L\GRUMPY2019I1.APSales (DESKTOP-CEBK38L\Angelo (52))

The Query Processor estimates that implementing the following index could improve the query cost by 87.5919%.

\*/

/\*

USE [APSales]

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]

ON [SalesReports].[YearlySalesReport] ([StoreNo])

INCLUDE ([StoreName],[StoreTerritory],[CalendarDate],[TotalSalesAmount])

GO

\*/

CREATE NONCLUSTERED INDEX ieStoreTerritoryDateTotalSales

ON [SalesReports].[YearlySalesReport] ([StoreNo])

INCLUDE ([StoreName],[StoreTerritory],[CalendarDate],[TotalSalesAmount])

GO

DROP TABLE IF EXISTS [SalesReports].[YearlySummaryReport]

GO

CREATE TABLE [SalesReports].[YearlySummaryReport](

[SalesYear] [int] NULL,

[SalesMonth] [int] NULL,

[StoreNo] [nvarchar](32) NOT NULL,

[StoreName] [nvarchar](64) NOT NULL,

[StoreTerritory] [nvarchar](64) NOT NULL,

[TotalSales] [decimal](10, 2) NULL

) ON [AP\_SALES\_FG]

GO

INSERT INTO APSales.SalesReports.YearlySummaryReport

SELECT YEAR(CalendarDate) AS SalesYear

,MONTH(CalendarDate) AS SalesMonth

,StoreNo

,StoreName

,StoreTerritory

,SUM(TotalSalesAmount) AS TotalSales

FROM APSales.SalesReports.YearlySalesReport

GROUP BY YEAR(CalendarDate)

,MONTH(CalendarDate)

,StoreNo

,StoreName

,StoreTerritory

GO

CREATE NONCLUSTERED INDEX [ieYearlySalesStoreTerritorySummary]

ON [SalesReports].[YearlySummaryReport] ([StoreNo],[SalesYear])

INCLUDE ([SalesMonth],[StoreName],[StoreTerritory],[TotalSales])

GO

***Listing 3.20 –Querying the Report Table***

DBCC dropcleanbuffers

CHECKPOINT;

GO

SET STATISTICS TIME ON

GO

SET STATISTICS IO ON

GO

SELECT SalesYear

,SalesMonth

,StoreNo

,StoreName

,StoreTerritory

,FORMAT(TotalSales,'C') AS TotalSales

,FORMAT(PERCENTILE\_CONT(.5)

WITHIN GROUP (ORDER BY TotalSales)

OVER (

PARTITION BY SalesYear

),'C') AS PctCont

,FORMAT(PERCENTILE\_DISC(.5)

WITHIN GROUP (ORDER BY TotalSales)

OVER (

PARTITION BY SalesYear

),'C') AS PctDisc

FROM APSales.SalesReports.YearlySummaryReport

WHERE SalesYear IN(2010,2011)

AND StoreNo = 'S00004

GO

SET STATISTICS TIME OFF

GO

SET STATISTICS IO OFF

GO

***Listing 3.21 – Index to Support Report Table.***

CREATE NONCLUSTERED INDEX [ieYearlySalesStoreTerritorySummary]

ON [SalesReports].[YearlySummaryReport] ([StoreNo],[SalesYear])

INCLUDE ([SalesMonth],[StoreName],[StoreTerritory],[TotalSales])

GO

ance improvement.