/\*

* RANK()
* PERCENT\_RANK()
* DENSE\_RANK()
* NTILE()
* ROW\_NUMBER()

\*/

***Listing 4.1 – Ranking Functions 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),

('FOUR',4),('SIX',6),('SEVEN',7),

('EIGHT',8),('NINE',9),('TEN',10);

SELECT

TestKey,

TheValue,

ROW\_NUMBER() OVER(ORDER BY TheValue) AS RowNo,

RANK() OVER(ORDER BY TheValue) AS ValueRank,

DENSE\_RANK() OVER(ORDER BY TheValue) AS DenseRank,

PERCENT\_RANK() OVER(ORDER BY TheValue) AS ValueRank,

FORMAT(PERCENT\_RANK() OVER(ORDER BY TheValue),'P') AS ValueRankAsPct

FROM @ExampleValues;

GO

***Listing 4.2 – Assigning Performance Buckets for Bonuses***

DECLARE @SalesPersonBonusStructure TABLE (

SalesPersonNo VARCHAR(4) NOT NULL,

SalesYtd MONEY NOT NULL

);

INSERT INTO @SalesPersonBonusStructure VALUES

('S001',2500.00),

('S002',2250.00),

('S003',2000.00),

('S004',1950.00),

('S005',1800.00),

('S006',1750.00),

('S007',1700.00),

('S008',1500.00),

('S009',1250.00),

('S010',1000.00);

-- Care must be taken how you sort (ASC or DESC)

SELECT SalesPersonNo

,SalesYtd

,NTILE(3) OVER(ORDER BY SalesYtd DESC) AS BonusBucket

,CASE

WHEN (NTILE(3) OVER(ORDER BY SalesYtd DESC)) = 1

THEN 'Award $500.00 Bonus'

WHEN (NTILE(3) OVER(ORDER BY SalesYtd DESC)) = 2

THEN 'Award $250.00 Bonus'

WHEN (NTILE(3) OVER(ORDER BY SalesYtd DESC)) = 3

THEN 'Award $150.00 Bonus'

END AS BonusAward

FROM @SalesPersonBonusStructure

GO

***Listing 4.3 – Rank versus Percent Rank***

WITH CustomerRanking (

CalendarYear,CalendarMonth,CustomerFullName,TotalSales

)

AS

(

SELECT CalendarYear

,CalendarMonth

,CustomerFullName

,SUM(TotalSalesAmount) AS TotalSales

FROM SalesReports.YearlySalesReport YSR

JOIN DimTable.Calendar C

ON YSR.CalendarDate = C.CalendarDate

GROUP BY C.CalendarYear

,C.CalendarMonth

,CustomerFullName

)

SELECT

CalendarYear

,CalendarMonth

,CustomerFullName

,FORMAT(TotalSales,'C') AS TotalSales

,RANK()

OVER (

-- PARTITION BY CalendarYear

ORDER BY TotalSales

) AS Rank

,PERCENT\_RANK()

OVER (

-- PARTITION BY CalendarYear

ORDER BY TotalSales

) AS PctRank

FROM CustomerRanking

WHERE CalendarYear = 2011

AND CalendarMonth = 1

ORDER BY

RANK() OVER (

PARTITION BY CalendarYear

ORDER BY TotalSales

) DESC

GO

***Listing 4.4 – Rank versus Dense Rank***

WITH CustomerRanking (

CalendarYear,CalendarMonth,CustomerFullName,TotalSales

)

AS

(

SELECT YEAR(CalendarDate)

,MONTH(CalendarDate)

,CustomerFullName

-- add one duplicate value on the fly

,CASE

WHEN CustomerFullName = 'Jim OConnel' THEN 17018.75

ELSE SUM(TotalSalesAmount)

END AS TotalSales

FROM SalesReports.YearlySalesReport

GROUP BY YEAR(CalendarDate)

,MONTH(CalendarDate)

,CustomerFullName

)

SELECT

CalendarYear

,CalendarMonth

,CustomerFullName

,FORMAT(TotalSales,'C') AS TotalSales

,RANK()

OVER (

ORDER BY TotalSales

) AS Rank

,DENSE\_RANK()

OVER (

ORDER BY TotalSales

) AS DenseRank

FROM CustomerRanking

WHERE CalendarYear = 2011

AND CalendarMonth = 1

ORDER BY

DENSE\_RANK() OVER (

PARTITION BY CalendarYear

ORDER BY TotalSales

) DESC

GO

***Listing 4.5 – Assigning Credit Analysts to Delinquent Accounts***

DECLARE @NumTiles INT;

SELECT @NumTiles = COUNT(DISTINCT [90DaysLatePaymentCount])

FROM Demographics.CustomerPaymentHistory

WHERE [90DaysLatePaymentCount] > 0;

SELECT CreditYear

,CreditQtr

,CustomerNo

,CustomerFullName

,SUM([90DaysLatePaymentCount]) AS Total90DayDelinquent

,NTILE(@NumTiles) OVER (

PARTITION BY CreditYear,CreditQtr

ORDER BY CreditQtr

) AS CreditAnaystBucket

,CASE NTILE(@NumTiles) OVER (

PARTITION BY CreditYear,CreditQtr

ORDER BY CreditQtr

)

WHEN 1 THEN 'Assign to Collection Analyst 1'

WHEN 2 THEN 'Assign to Collection Analyst 2'

WHEN 3 THEN 'Assign to Collection Analyst 3'

WHEN 4 THEN 'Assign to Collection Analyst 4'

WHEN 5 THEN 'Assign to Collection Analyst 5'

END AS CreditAnalystAssignment

FROM Demographics.CustomerPaymentHistory

WHERE [90DaysLatePaymentCount] > 0

GROUP BY CreditYear

,CreditQtr

,CustomerNo

,CustomerFullName

ORDER BY CreditYear

,CreditQtr

,SUM([90DaysLatePaymentCount]) DESC

GO

***Listing 4.6 – Rolling Sales Total By month***

WITH StoreProductAnalysis

(TransYear,TransMonth,TransQtr,StoreNo,ProductNo,ProductsBought)

AS

(

SELECT

YEAR(CalendarDate) AS TransYear

,MONTH(CalendarDate) AS TransMonth

,DATEPART(qq,CalendarDate) AS TransQtr

,StoreNo

,ProductNo

,SUM(TransactionQuantity) AS ProductsBought

FROM StagingTable.SalesTransaction

GROUP BY YEAR(CalendarDate,

,MONTH(CalendarDate)

,DATEPART(qq,CalendarDate)

,StoreNo

,ProductNo

)

SELECT

spa.TransYear

,spa.TransMonth

,spa.StoreNo

,spa.ProductNo

,p.ProductName

,spa.ProductsBought

,SUM(spa.ProductsBought) OVER(

PARTITION BY spa.StoreNo,spa.TransYear

ORDER BY spa.TransMonth

) AS RunningTotal

,ROW\_NUMBER() OVER(

PARTITION BY spa.StoreNo,spa.TransYear

ORDER BY spa.TransMonth

) AS EntryNoByMonth

,ROW\_NUMBER() OVER(

PARTITION BY spa.StoreNo,spa.TransYear,TransQtr

ORDER BY spa.TransMonth

) AS EntryNoByQtr

,ROW\_NUMBER() OVER(

ORDER BY spa.TransYear,spa.StoreNo

) AS EntryNoByYear

FROM StoreProductAnalysis spa

JOIN DimTable.Product p

ON spa.ProductNo = p.ProductNo

WHERE spa.TransYear IN(2011,2012)

AND spa.StoreNo IN ('S00009','S00010')

AND spa.ProductNo = 'P00000011129'

GO

***Listing 4.7a – Loading the SalesPersonLog***

USE TEST

GO

DROP TABLE IF EXISTS SalesPersonLog

GO

CREATE TABLE SalesPersonLog (

SalesPersonId VARCHAR(8),

SalesDate DATE,

SalesAmount DECIMAL(10,2),

IslandGapGroup VARCHAR(8)

);

TRUNCATE TABLE SalesPersonLog

GO

INSERT INTO SalesPersonLog

SELECT 'SP001'

,[CalendarDate]

,UPPER (

CONVERT(INT,CRYPT\_GEN\_RANDOM(1)

)) AS SalesAmount

,'ISLAND'

FROM APSales.[DimTable].[Calendar]

WHERE [CalendarYear] = 2010

AND [CalendarMonth] = 10

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Set up some gaps \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

UPDATE SalesPersonLog

SET SalesAmount = 0,

IslandGapGroup = 'GAP'

WHERE SalesDate BETWEEN '2010-10-5' AND '2010-10-6'

GO

UPDATE SalesPersonLog

SET SalesAmount = 0,

IslandGapGroup = 'GAP'

WHERE SalesDate BETWEEN '2010-10-11' AND '2010-10-16'

GO

UPDATE SalesPersonLog

SET SalesAmount = 0,

IslandGapGroup = 'GAP'

WHERE SalesDate BETWEEN '2010-10-22' AND '2010-10-23'

GO

-- Just in case the random sales value generator

-- set sales to 0 but the update labelled it as an ISLAND

UPDATE SalesPersonLog

SET IslandGapGroup = 'GAP'

WHERE SalesAmount = 0

GO

***Listing 4.7b – Generating the Gap and Island Report***

SELECT SalesPersonId,GroupName,SUM(SalesAmount) AS TotalSales

,MIN(StartDate) AS StartDate,MAX(StartDate) AS EndDate

,CASE

WHEN SUM(SalesAmount) <> 0 THEN 'Working, finally!'

ELSE 'Goofing off again!'

END AS Reason

FROM (

SELECT SalesPersonId,SalesAmount,

,IslandGapGroup + CONVERT(VARCHAR,(SUM(IslandGapGroupId)

OVER(ORDER BY StartDate) )) AS GroupName

,StartDate

,PreviousSalesDate AS EndDate

FROM

(

SELECT ROW\_NUMBER() OVER(ORDER BY SalesDate) AS RowNumber

,SalesPersonId

,SalesAmount

,IslandGapGroup

,SalesDate AS StartDate

,LAG(SalesDate)

OVER(ORDER BY SalesDate) AS PreviousSalesDate

,CASE

WHEN LAG(SalesDate) OVER(ORDER BY SalesDate) IS NULL

OR

(

LAG(SalesAmount) OVER(ORDER BY SalesDate) <> 0

AND SalesAmount = 0

) THEN ROW\_NUMBER() OVER(ORDER BY SalesDate)

WHEN (LAG(SalesAmount) OVER(ORDER BY SalesDate) = 0

AND SalesAmount <> 0)

THEN ROW\_NUMBER() OVER(ORDER BY SalesDate)

ELSE 0

END AS IslandGapGroupId

FROM SalesPersonLog

) T1

)T2

GROUP BY SalesPersonId,GroupName

ORDER BY StartDate

GO