# 3. Over

select A.EmployeeNumber, A.AttendanceMonth, A.NumberAttendance

,sum(A.NumberAttendance) over() as TotalAttendance,

convert(decimal(18,7),A.NumberAttendance) / sum(A.NumberAttendance) over() \* 100.0000 as PercentageAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

select sum(NumberAttendance) from tblAttendance

# 4. Partition by and Order by

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

sum(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber) as SumAttendance,

convert(money,A.NumberAttendance) /

sum(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber) \* 100 as PercentageAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

sum(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber, year(A.AttendanceMonth)

ORDER BY A.AttendanceMonth) as SumAttendance,

convert(money,A.NumberAttendance) /

sum(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber) \* 100 as PercentageAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

order by A.EmployeeNumber, A.AttendanceMonth

# 5. Range

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

SUM(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber ORDER BY A.AttendanceMonth ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) as RollingTotal

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

preceding + following can be unbounded (all by order)

CurrentRow = 0

# 6. Current Row and Unbounded

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

SUM(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber ORDER BY A.AttendanceMonth ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) as RollingTotal

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

SUM(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber ORDER BY A.AttendanceMonth ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as RollingTotal

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

SUM(A.NumberAttendance) over(PARTITION BY E.EmployeeNumber ORDER BY A.AttendanceMonth ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) as RollingTotal

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

# 7. Range versus Rows

select A.EmployeeNumber, A.AttendanceMonth, A.NumberAttendance

,sum(A.NumberAttendance)

over(partition by A.EmployeeNumber, year(A.AttendanceMonth)

order by A.AttendanceMonth

rows between current row and unbounded following) as RowsTotal

,sum(A.NumberAttendance)

over(partition by A.EmployeeNumber, year(A.AttendanceMonth)

order by A.AttendanceMonth

range between current row and unbounded following) as RangeTotal

from tblEmployee as E join (select \* from tblAttendance UNION ALL select \* from tblAttendance) as A

on E.EmployeeNumber = A.EmployeeNumber

--where A.AttendanceMonth < '20150101'

order by A.EmployeeNumber, A.AttendanceMonth

--unbounded preceding and current row

--current row and unbounded following

--unbounded preceding and unbounded following - RANGE and ROWS

Row is fast should should be used , use range when there is a ties

# 8. Omitting Range/Row?

select A.EmployeeNumber, A.AttendanceMonth, A.NumberAttendance

,sum(A.NumberAttendance) over() as TotalAttendance

--,convert(decimal(18,7),A.NumberAttendance) / sum(A.NumberAttendance) over() \* 100.0000 as PercentageAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

select sum(NumberAttendance) from tblAttendance

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

sum(A.NumberAttendance)

over(PARTITION BY E.EmployeeNumber, year(A.AttendanceMonth)

ORDER BY A.AttendanceMonth) as SumAttendance

from tblEmployee as E join (select \* from tblAttendance UNION ALL Select \* from tblAttendance) as A

on E.EmployeeNumber = A.EmployeeNumber

order by A.EmployeeNumber, A.AttendanceMonth

--range between unbounded preceding and unbounded following - DEFAULT where there is no ORDER BY

--range between unbounded preceding and current row - DEFAULT where there IS an ORDER BY

# 9. ROW\_NUMBER, RANK and DENSE\_RANK

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

ROW\_NUMBER() OVER(ORDER BY E.EmployeeNumber, A.AttendanceMonth) as TheRowNumber,

RANK() OVER(ORDER BY E.EmployeeNumber, A.AttendanceMonth) as TheRank,

DENSE\_RANK() OVER(ORDER BY E.EmployeeNumber, A.AttendanceMonth) as TheDenseRank

from tblEmployee as E join

(Select \* from tblAttendance union all select \* from tblAttendance) as A

on E.EmployeeNumber = A.EmployeeNumber

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

ROW\_NUMBER() OVER(PARTITION BY E.EmployeeNumber

ORDER BY A.AttendanceMonth) as TheRowNumber,

RANK() OVER(PARTITION BY E.EmployeeNumber

ORDER BY A.AttendanceMonth) as TheRank,

DENSE\_RANK() OVER(PARTITION BY E.EmployeeNumber

ORDER BY A.AttendanceMonth) as TheDenseRank

from tblEmployee as E join

(Select \* from tblAttendance union all select \* from tblAttendance) as A

on E.EmployeeNumber = A.EmployeeNumber

select \*, row\_number() over(order by (select null)) from tblAttendance

Incase of ties

The rank =>1,1,3,3,5,5

The dense rank=>1,1,2,2,3,3,

The row\_number=>1,2,3,4,5,6

# 10. NTILE –

# put it in bucket 10 means =22 = 22/10= 3,3,2,2,2,2,2,2,2,2

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

NTILE(10) OVER(PARTITION BY E.EmployeeNumber

ORDER BY A.AttendanceMonth) as TheNTile,

convert(int,(ROW\_NUMBER() OVER(PARTITION BY E.EmployeeNumber

ORDER BY A.AttendanceMonth)-1)

/ (count(\*) OVER(PARTITION BY E.EmployeeNumber

ORDER BY A.AttendanceMonth

ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)/10.0))+1 as MyNTile

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

where A.AttendanceMonth <'2015-05-01'

# 11. FIRST\_VALUE and LAST\_VALUE

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

first\_value(NumberAttendance)

over(partition by E.EmployeeNumber order by A.AttendanceMonth) as FirstMonth,

last\_value(NumberAttendance)

over(partition by E.EmployeeNumber order by A.AttendanceMonth

rows between unbounded preceding and unbounded following) as LastMonth

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

# 12. LAG and LEAD

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

lag(NumberAttendance, 1) over(partition by E.EmployeeNumber

order by A.AttendanceMonth) as MyLag,

lead(NumberAttendance, 1) over(partition by E.EmployeeNumber

order by A.AttendanceMonth) as MyLead,

NumberAttendance - lag(NumberAttendance, 1) over(partition by E.EmployeeNumber

order by A.AttendanceMonth) as MyDiff

--first\_value(NumberAttendance) over(partition by E.EmployeeNumber

-- order by A.AttendanceMonth

-- rows between 1 preceding and current row) as MyFirstValue,

--last\_value(NumberAttendance) over(partition by E.EmployeeNumber

-- order by A.AttendanceMonth

-- rows between current row and 1 following) as MyLastValue

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

LAG, Lead – give previous and preceeding value

Use secnd param to point lag or lead and third to fill null

To find difference

What is the order of the three arguments for the LAG and LEAD functions?

=>Field name, number of rows go forward/backward, value if null

I have a field in a table called "Amount".

What does LEAD(Amount) and LAG(Amount) do? (Please note - there is only one argument inside the brackets.)

Lag => previous row , lead=> next row

# 13. CUME\_DIST and PERCENT\_RANK

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

CUME\_DIST() over(partition by E.EmployeeNumber

order by A.AttendanceMonth) as MyCume\_Dist, -- tie take left

PERCENT\_RANK() over(partition by E.EmployeeNumber

order by A.AttendanceMonth) as MyPercent\_Rank, -- tie take right

cast(row\_number() over(partition by E.EmployeeNumber order by A.AttendanceMonth) as decimal(9,5))

/ count(\*) over(partition by E.EmployeeNumber) as CalcCume\_Dist,

cast(row\_number() over(partition by E.EmployeeNumber order by A.AttendanceMonth) - 1 as decimal(9,5))

/ (count(\*) over(partition by E.EmployeeNumber) - 1) as CalcPercent\_Rank

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

# 14. PERCENTILE\_CONT and PERCENTILE\_DISC

select A.EmployeeNumber, A.AttendanceMonth,

A.NumberAttendance,

CUME\_DIST() over(partition by E.EmployeeNumber

order by A.NumberAttendance) as MyCume\_Dist,

PERCENT\_RANK() over(partition by E.EmployeeNumber

order by A.NumberAttendance) \* 100 as MyPercent\_Rank

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

SELECT DISTINCT EmployeeNumber,

PERCENTILE\_CONT(0.4) WITHIN GROUP (ORDER BY NumberAttendance) OVER (PARTITION BY EmployeeNumber) as AverageCont,

PERCENTILE\_DISC(0.4) WITHIN GROUP (ORDER BY NumberAttendance) OVER (PARTITION BY EmployeeNumber) as AverageDisc

from tblAttendance

Cont-> Continues

Disc-> Discrete

# 15. Adding Totals

select E.Department, E.EmployeeNumber, A.AttendanceMonth as AttendanceMonth, sum(A.NumberAttendance) as NumberAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

group by E.Department, E.EmployeeNumber, A.AttendanceMonth

--order by Department, EmployeeNumber, AttendanceMonth

UNION

select E.Department, E.EmployeeNumber, null as AttendanceMonth, sum(A.NumberAttendance) as TotalAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

group by E.Department, E.EmployeeNumber

union

select E.Department, null, null as AttendanceMonth, sum(A.NumberAttendance) as TotalAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

group by E.Department

union

select null, null, null as AttendanceMonth, sum(A.NumberAttendance) as TotalAttendance

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

order by Department, EmployeeNumber, AttendanceMonth

# 16. ROLLUP, GROUPING and GROUPING\_ID

select E.Department, E.EmployeeNumber, A.AttendanceMonth as AttendanceMonth, sum(A.NumberAttendance) as NumberAttendance,

GROUPING(E.EmployeeNumber) AS EmployeeNumberGroupedBy,

GROUPING\_ID(E.Department, E.EmployeeNumber, A.AttendanceMonth) AS EmployeeNumberGroupedID

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

group by ROLLUP (E.Department, E.EmployeeNumber, A.AttendanceMonth)

order by Department, EmployeeNumber, AttendanceMonth

GROUPING=>give 1 if groupedby has employyeenumber -> so for department and employeenumber=1 and for attendance month =0

GROUPING\_ID = 1 for 1 column, 2 for second column=> 1+2=3 and for third column =4 =>1+2+4=7

If you want total of all possibility then use cube instead of rollup,

If you not want all permutation then use grouping set

# 17. GROUPING SETS

select E.Department, E.EmployeeNumber, A.AttendanceMonth as AttendanceMonth, sum(A.NumberAttendance) as NumberAttendance,

GROUPING(E.EmployeeNumber) AS EmployeeNumberGroupedBy,

GROUPING\_ID(E.Department, E.EmployeeNumber, A.AttendanceMonth) AS EmployeeNumberGroupedID

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

group by GROUPING SETS ((E.Department, E.EmployeeNumber, A.AttendanceMonth), (E.Department), ())

order by coalesce(Department, 'zzzzzzz'), coalesce(E.EmployeeNumber, 99999), coalesce(AttendanceMonth,'2100-01-01')

select E.Department, E.EmployeeNumber, A.AttendanceMonth as AttendanceMonth, sum(A.NumberAttendance) as NumberAttendance,

GROUPING(E.EmployeeNumber) AS EmployeeNumberGroupedBy,

GROUPING\_ID(E.Department, E.EmployeeNumber, A.AttendanceMonth) AS EmployeeNumberGroupedID

from tblEmployee as E join tblAttendance as A

on E.EmployeeNumber = A.EmployeeNumber

group by GROUPING SETS ((E.Department, E.EmployeeNumber, A.AttendanceMonth), (E.Department), ())

order by CASE WHEN Department IS NULL THEN 1 ELSE 0 END, Department,

CASE WHEN E.EmployeeNumber IS NULL THEN 1 ELSE 0 END, E.EmployeeNumber,

CASE WHEN AttendanceMonth IS NULL THEN 1 ELSE 0 END, AttendanceMonth

# 19. Geometry – Creating Point

BEGIN TRAN

CREATE TABLE tblGeom

(GXY geometry,

Description varchar(30),

IDtblGeom int CONSTRAINT PK\_tblGeom PRIMARY KEY IDENTITY(1,1))

INSERT INTO tblGeom

VALUES (geometry::STGeomFromText('POINT (3 4)', 0),'First point'),

(geometry::STGeomFromText('POINT (3 5)', 0),'Second point'),

(geometry::Point(4, 6, 0),'Third Point'),

(geometry::STGeomFromText('MULTIPOINT ((1 2), (2 3), (3 4))', 0), 'Three Points')

Select \* from tblGeom

ROLLBACK TRAN

# 20. Queries Points

begin tran

create table tblGeom

(GXY geometry,

Description varchar(20),

IDtblGeom int CONSTRAINT PK\_tblGeom PRIMARY KEY IDENTITY(1,1))

insert into tblGeom

VALUES (geometry::STGeomFromText('POINT (3 4)', 0),'First point'),

(geometry::STGeomFromText('POINT (3 5)', 0),'Second point'),

(geometry::Point(4, 6, 0),'Third Point'),

(geometry::STGeomFromText('MULTIPOINT ((1 2), (2 3), (3 4))', 0),'Three Points')

SELECT \* from tblGeom

select IDtblGeom, GXY.STGeometryType() as MyType

, GXY.STStartPoint().ToString() as StartingPoint

, GXY.STEndPoint().ToString() as EndingPoint

, GXY.STPointN(1).ToString() as FirstPoint

, GXY.STPointN(2).ToString() as SecondPoint

, GXY.STPointN(1).STX as FirstPointX

, GXY.STPointN(1).STY as FirstPointY

, GXY.STNumPoints() as NumberPoints

from tblGeom

DECLARE @g as geometry

DECLARE @h as geometry

select @g = GXY from tblGeom where IDtblGeom = 1

select @h = GXY from tblGeom where IDtblGeom = 3

select @g.STDistance(@h) as MyDistance

ROLLBACK TRAN

# 21. Defining LINESTRINGs, POLYGONs and CIRCULARSTRINGs

begin tran

create table tblGeom

(GXY geometry,

Description varchar(20),

IDtblGeom int CONSTRAINT PK\_tblGeom PRIMARY KEY IDENTITY(5,1))

insert into tblGeom

VALUES (geometry::STGeomFromText('LINESTRING (1 1, 5 5)', 0),'First line'),

(geometry::STGeomFromText('LINESTRING (5 1, 1 4, 2 5, 5 1)', 0),'Second line'),

(geometry::STGeomFromText('MULTILINESTRING ((1 5, 2 6), (1 4, 2 5))', 0),'Third line'),

(geometry::STGeomFromText('POLYGON ((4 1, 6 3, 8 3, 6 1, 4 1))', 0), 'Polygon'),

(geometry::STGeomFromText('CIRCULARSTRING (1 0, 0 1, -1 0, 0 -1, 1 0)', 0), 'Circle')

SELECT \* FROM tblGeom

rollback tran

# 22. Querying LINESTRINGs, POLYGONs and CIRCULARSTRINGs

begin tran

create table tblGeom

(GXY geometry,

Description varchar(20),

IDtblGeom int CONSTRAINT PK\_tblGeom PRIMARY KEY IDENTITY(5,1))

insert into tblGeom

VALUES (geometry::STGeomFromText('LINESTRING (1 1, 5 5)', 0),'First line'),

(geometry::STGeomFromText('LINESTRING (5 1, 1 4, 2 5, 5 1)', 0),'Second line'),

(geometry::STGeomFromText('MULTILINESTRING ((1 5, 2 6), (1 4, 2 5))', 0),'Third line'),

(geometry::STGeomFromText('POLYGON ((4 1, 6 3, 8 3, 6 1, 4 1))', 0), 'Polygon'),

(geometry::STGeomFromText('CIRCULARSTRING (1 0, 0 1, -1 0, 0 -1, 1 0)', 0), 'Circle')

SELECT \* FROM tblGeom

select IDtblGeom, GXY.STGeometryType() as MyType

, GXY.STStartPoint().ToString() as StartingPoint

, GXY.STEndPoint().ToString() as EndingPoint

, GXY.STPointN(1).ToString() as FirstPoint

, GXY.STPointN(2).ToString() as SecondPoint

, GXY.STPointN(1).STX as FirstPointX

, GXY.STPointN(1).STY as FirstPointY

, GXY.STBoundary().ToString() as Boundary

, GXY.STLength() as MyLength

, GXY.STNumPoints() as NumberPoints

from tblGeom

DECLARE @g as geometry

select @g = GXY from tblGeom where IDtblGeom = 5

select IDtblGeom, GXY.STIntersection(@g).ToString() as Intersection

, GXY.STDistance(@g) as DistanceFromFirstLine

from tblGeom

select GXY.STUnion(@g), Description

from tblGeom

where IDtblGeom = 8

rollback tran

# 23. Geography

begin tran

create table tblGeog

(GXY geography,

Description varchar(30),

IDtblGeog int CONSTRAINT PK\_tblGeog PRIMARY KEY IDENTITY(1,1))

insert into tblGeog

VALUES (geography::STGeomFromText('POINT (-73.993492 40.750525)', 4326),'Madison Square Gardens, NY'),

(geography::STGeomFromText('POINT (-0.177452 51.500905)', 4326),'Royal Albert Hall, London'),

(geography::STGeomFromText('LINESTRING (-73.993492 40.750525, -0.177452 51.500905)', 4326),'Connection')

select \* from tblGeog

DECLARE @g as geography

select @g = GXY from tblGeog where IDtblGeog = 1

select IDtblGeog, GXY.STGeometryType() as MyType

, GXY.STStartPoint().ToString() as StartingPoint

, GXY.STEndPoint().ToString() as EndingPoint

, GXY.STPointN(1).ToString() as FirstPoint

, GXY.STPointN(2).ToString() as SecondPoint

, GXY.STLength() as MyLength

, GXY.STIntersection(@g).ToString() as Intersection

, GXY.STNumPoints() as NumberPoints

, GXY.STDistance(@g) as DistanceFromFirstLine

from tblGeog

DECLARE @h as geography

select @g = GXY from tblGeog where IDtblGeog = 1

select @h = GXY from tblGeog where IDtblGeog = 2

select @g.STDistance(@h) as MyDistance

select GXY.STUnion(@g)

from tblGeog

where IDtblGeog = 2

ROLLBACK TRAN

select \* from sys.spatial\_reference\_systems

# 24. Spatial aggregates

begin tran

create table tblGeom

(GXY geometry,

Description varchar(20),

IDtblGeom int CONSTRAINT PK\_tblGeom PRIMARY KEY IDENTITY(5,1))

insert into tblGeom

VALUES (geometry::STGeomFromText('LINESTRING (1 1, 5 5)', 0),'First line'),

(geometry::STGeomFromText('LINESTRING (5 1, 1 4, 2 5, 5 1)', 0),'Second line'),

(geometry::STGeomFromText('MULTILINESTRING ((1 5, 2 6), (1 4, 2 5))', 0),'Third line'),

(geometry::STGeomFromText('POLYGON ((4 1, 6 3, 8 3, 6 1, 4 1))', 0), 'Polygon'),

(geometry::STGeomFromText('POLYGON ((5 2, 7 2, 7 4, 5 4, 5 2))', 0), 'Second Polygon'),

(geometry::STGeomFromText('CIRCULARSTRING (1 0, 0 1, -1 0, 0 -1, 1 0)', 0), 'Circle')

select \* from tblGeom

SELECT \* FROM tblGeom

where GXY.Filter(geometry::Parse('POLYGON((2 1, 1 4, 4 4, 4 1, 2 1))')) = 1

UNION ALL

SELECT geometry::STGeomFromText('POLYGON((2 1, 1 4, 4 4, 4 1, 2 1))', 0), 'Filter', 0

declare @i as geometry

select @i = geometry::UnionAggregate(GXY)

from tblGeom

Select @i as CombinedShapes

declare @j as geometry

select @j = geometry::CollectionAggregate(GXY)

from tblGeom

select @j

Select @i as CombinedShapes

--union all

select geometry::EnvelopeAggregate(GXY) as Envelope from tblGeom

--union all

select geometry::ConvexHullAggregate(GXY) as Envelope from tblGeom

ROLLBACK TRAN