**CODE SAMPLES**

**CROSS APPLY**

CROSS APPLY is your only option for "joining" table value functions and "expanding" xml documents

from #MultiPayWithBalance as m

left join LastDueDate as ldd

on ldd.PolicyID = m.PolicyID

join TransactionRange as tr

join PolicyVersionAttributes as pva

on pva.PolicyVersionID = tr.PolicyVersionID

on tr.PolicyID = m.PolicyID

join Policy as p

join States as s

on s.StateID = p.StateID

on p.PolicyID = m.PolicyID

left join Balances as b

on b.PolicyID = m.PolicyID

left join AssessedFees as af

on af.PolicyID = m.PolicyID

left join LastEndorsement as le

on le.PolicyID = m.PolicyID

left join PolicyCancellationAmount as pc

on pc.PolicyID = m.PolicyID

and @PolicyID is null

cross apply dbo.GetBillingStatus(tr.PolicyYearNum, pva.TotalPremium, m.AtRenewalBill, tr.EffectiveDate, s.StateCode, m.Arrangement, @BillDate, m.Balance + @CancellationAmount + isnull(pc.CancellationAmount, 0), isnull(af.PlanFeeCount, 0), isnull(af.SetupFeeCount, 0), le.CreatedDate,p.CompanyID) as bs

where @PolicyID is not null

**Split Function**

ALTER FUNCTION [dbo].[Split]

(

@String varchar(max)

,@Delimiter varchar

)

RETURNS @Results table

(

Ordinal int

,StringValue varchar(max)

)

as

begin

set @String = isnull(@String,'')

set @Delimiter = isnull(@Delimiter,'')

declare

@TempString varchar(max) = @String

,@Ordinal int = 0

,@CharIndex int = 0

set @CharIndex = charindex(@Delimiter, @TempString)

while @CharIndex != 0 begin

set @Ordinal += 1

insert @Results values

(

@Ordinal

,substring(@TempString, 0, @CharIndex)

)

set @TempString = substring(@TempString, @CharIndex + 1, len(@TempString) - @CharIndex)

set @CharIndex = charindex(@Delimiter, @TempString)

end

if @TempString != '' begin

set @Ordinal += 1

insert @Results values

(

@Ordinal

,@TempString

)

end

return

end

**CROSS APPLY**

CROSS APPLY is your only option for "joining" table value functions and "expanding" xml documents

**This code gets the distance in miles from point A to point B**

ALTER function [dbo].[GetDistance]

(

@latitude1 decimal(18,13)

,@longitude1 decimal(18,13)

,@latitude2 decimal(18,13)

,@longitude2 decimal(18,13)

)

returns float

as begin

declare @innerExpression float =

cos(radians(@latitude2))

\* cos(radians(@latitude1))

\* cos(radians(@longitude1)

- radians(@longitude2))

+ sin(radians(@latitude2))

\* sin(radians(@latitude1))

return case when @innerExpression < -1 or @innerExpression > 1 then null else acos(@innerExpression) \* 3959.0 end

end

**This code displays a 1,2,3 ect if there are duplicate claims with the same AdjusterID**

row\_number() over (partition by c.ClaimID, ca.AdjusterSystemUserID order by c.ClaimID) as DuplicateCount

**This code selects the MAX claim**

[Open] as (

select ass.ClaimID

from Assignments as ass

inner join dbo.ClaimStatusHistory as csh

inner join ( select max(ClaimStatusHistoryID) as ClaimStatusHistoryID

from dbo.ClaimStatusHistory

where CreatedDate <= @AsOfDate

group by ClaimID) as hid

on hid.ClaimStatusHistoryID = csh.ClaimStatusHistoryID

on ass.ClaimID = csh.ClaimID

where csh.ClaimStatusID = 1

),

**SUM(iff acks as a counter when pulling separate results from two separate temp tables**

SELECT

COUNT(ft.claimid) AS FastTrack

,count(tc.ClaimID) AS TotalAssignment

,sum(iif(ft.Closed = 0, 1, 0)) as OpenFastTrack

,sum(iif(ft.Closed = 1, 1, 0)) as ClosedFastTrack

,sum(iif(tc.Closed = 0, 1, 0)) as [Open]

,sum(iif(tc.Closed = 1, 1, 0)) as Closed

,su.SystemUserID

,su.FirstName + ' ' + su.LastName as [Adjuster]

FROM claim c

LEFT join #TotalClaims tc on c.ClaimID = tc.ClaimID

LEFT join #FTClaimsList ft on c.ClaimID = ft.ClaimID

join systemusers su on su.SystemUserID = tc.AdjusterSystemUserID

GROUP BY su.SystemUserID ,

su.FirstName + ' ' + su.LastName

**POPULATES a #TEMP TABLE with an EXISTING REPORT and sends parameters into the existing report. Also creates an index on a #TEMP table.**

if object\_id('tempdb.dbo.#Yesterday') is not null drop table #Yesterday

create table #Yesterday (

AccidentYear smallint,

Peril varchar(50),

ClaimNumber varchar(20),

LossDate date,

Losses money,

Outstanding money,

LAE money,

Incurred money,

PendingReserve money,

CloseDate date null,

WindType char(1),

[Status] varchar(20),

PolicyNumber varchar(25),

PED date,

Insured varchar(100),

PolicyForm varchar(20),

[State] varchar(2),

Building integer,

Contents integer,

CompanyCode varchar(20))

create clustered index idx\_ClaimNumber on #Yesterday (ClaimNumber)

insert into #Yesterday

exec [dbo].[reportExcessLosses]

@AsOfDate = @AsOfDate,

@XWindLimit = 0,

@WindLimit = 0,

@LossStartDate = @LossStartDate,

@LossEndDate = @LossEndDate,

@PendingReserves = 0

update #Yesterday

set CloseDate = null

where CloseDate = '1900-01-01'

DELETE FROM #Yesterday Where ClaimNumber IN (select c.ClaimNumber from dbo.Claim as c inner join dbo.GetBODCatastrophe() as bc on bc.CatastropheID = c.CatastropheID)

**Cast dates when comparing to a date parameter i.e. @StartDate**

cast(c.LossDate as date) **@StartDate**

cast(c.ReportedDate as date) **@Endate**

**CODE TO HANDLE DIVIDE BY ZERO error**

**COALESCE(ibs.TotalIncurred / NULLIF(cbs.ClosedClaimCount,0), 0) AS ClosedClaimsAVG,**

SELECT COALESCE(dividend / NULLIF(divisor,0), 0) FROM sometable

Select dividend / nullif(divisor, 0)

SELECT ISNULL(@num / NULLIF(@iter,0),@num);

1. SELECT @int1 / NULLIF(@int2,0) -- returns NULL

or

1. SELECT @int1 / ISNULL(NULLIF(@int2,0),1) -- returns @int1

SELECT ISNULL([Numerator] / NULLIF([Denominator], 0), 0) AS [Percentage]

FROM [Table1]

**Not Exists**

where ass.DuplicateCount = 1

and not exists ( select \*

from #AdjusterDetail as ad

where ad.ClaimID = ass.ClaimID)

**NEW ABOVE**

**QUERY TO CHECK FOR A UNIQUE KEY IN A TABLE**

SELECT EAEOBGRUP,EAEOBCODE,EAPRIORITY

FROM dbo.EOBActionCodes

GROUP BY EAEOBGRUP,EAEOBCODE,EAPRIORITY

HAVING COUNT(\*) >1

**The query results should return no records if these three columns are the unique key for a table**

**CODE TO HANDLE DIVIDE BY ZERO error**

/\*The code below will return zero if you were about to divide by zero\*/

,CASE WHEN SUM(ISNULL(Visits,0)) = 0 THEN 0 ELSE (SUM(ISNULL(RVU,0)) / SUM(ISNULL(Visits,0)))END  AS AvgRVU

**COVERED INDEX includes the columns for a specific query.**

SELECT

[TransUnique],[DimFacilityKey],[DimGuarantorKey],[DimPatientKey]\*

FROM dbo.AthenaFactTrans

WHERE TransType = 'charge'

AND DimFinancialClassKey IS NULL

CREATE NONCLUSTERED INDEX [IX\_Sample]

ON [dbo].[AthenaFactTrans] ([TransType],[DimFinancialClassKey]) --<--this covers the WHERE statement

INCLUDE ([TransUnique],[DimFacilityKey],[DimGuarantorKey],[DimPatientKey]) --<--this covers the columns returned

--WHERE 1=2 will never be true so only the structure and not the data will be --copied

**--This code copies the structure only and not the data from one table to a new table without using the CREATE statement**.

SELECT \* INTO tblNew FROM dbo.DimDiag1 WHERE 1=2

**Copy the structure with data of an existing table into new table:**  
  
SELECT \* INTO tblNew FROM tblOld  
  
This is also same like the previous query, but it copies the structure of existing table(tblOld) with data as well into the new table(tblNew).

**--This code separated the first and last names from a field.**

SELECT \*,

CASE WHEN CHARINDEX(**','**,ddrname) > 0 THEN left(ddrname,CHARINDEX(**','**,ddrname)-1)

     else ddrname END AS LName,

CASE WHEN CHARINDEX(**','**,ddrname) > 0 THEN RIGHT(ddrname,(len(ddrname)-CHARINDEX(**','**,ddrname)))

     ELSE '' end AS fname

FROM dmdoctr

/\*

This code merges data between two tables.

\*/

UPDATE MyTarget SET MyTarget.FacilityState = MySource.FacilityState

--SELECT  MyTarget.FacilityState,MySource.FacilityState, \*

FROM DW\_For\_BI360.dbo.DimFacility AS MyTarget

INNER JOIN DW\_For\_BI360\_Test.dbo.DimFacility\_ED\_01092014 AS MySource

ON MyTarget.DimFacilityKey = MySource.DimFacilityKey

WHERE MyTarget.FacilityState IS NULL

AND MySource.FacilityState IS NOT NULL

/\*

This code shows the last date and time the schedule was modified for the

New - Import CPU Packages automated job

\*/

USE MSDB

--SELECT \* FROM sysjobs WHERE Name LIKE '%CPU%'

Declare @jobid varchar(40)

SELECT @jobid =

job\_id from sysjobs where  name like '%CPU%' --New - Import CPU Packages

select date\_created,date\_modified,\* from sysschedules where schedule\_id in(select schedule\_id from msdb.dbo.sysjobschedules  where job\_id = @jobid)

INSERT INTO DimPatient( PatientNo,PatientFirstName,PatientLastName,PatientMidInt,PatientGender,PatientDOB,PatientrRelToGur,PatientSSN )

exec sp\_help DimPatient

SELECT

MaX(datalength(Staging.[Patient ID])) as pt,

MaX(datalength(Staging.[First Name])) fn,

MaX(datalength(Staging.[Last Name])) ln,

MaX(datalength(Staging.[Middle Initial])) mi,

MaX(datalength(Staging.Sex)) sx,

MaX(datalength(Staging.DOB)) db,

MaX(datalength(Staging.[Guarantor Relationship])) gr,

MaX(datalength(Staging.[Patient SSN])) ss

FROM Staging\_DimPatient Staging

LEFT OUTER JOIN DimPatient ExistingData

ON ExistingData.PatientNo = Staging.[Patient ID]

WHERE ExistingData.PatientNo IS NULL

**CREATE INDEX:**

IF OBJECT\_ID('[dbo].[Staging\_DimGuarantor]') IS NOT NULL

AND NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX\_Staging\_DimGuarantor' AND object\_name(object\_id) = 'Staging\_DimGuarantor')

BEGIN

CREATE INDEX [IX\_Staging\_DimGuarantor] ON Staging\_DimGuarantor([Patient ID]);

END

**CHECKSUM QUERIES TO COMPARE CSV FILES**

select \* FROM (SELECT [Transaction ID],[CONTEXT\_ID],[Reversal Flag], CHECKSUM(\*) As C1 from Transaction\_03\_19\_2014) T1

LEFT OUTER JOIN (select [Transaction ID],[CONTEXT\_ID],[Reversal Flag] ,CHECKSUM(\*) As C1 from Transaction\_03\_20\_2014) T2

ON T1.[Transaction ID] = T2.[Transaction ID] AND T1.[CONTEXT\_ID] = T2.[CONTEXT\_ID] AND T1.[Reversal Flag] = T2.[Reversal Flag]

WHERE T1.C1 <> T2.C1

select \* from Transaction\_03\_19\_2014 WHERE [Transaction ID] = '759'

UNION ALL

select \* from Transaction\_03\_20\_2014 WHERE [Transaction ID] = '759'

SELECT [Transaction Id] from transaction\_03\_19\_2014

intersect

SELECT [Transaction Id] from transaction\_03\_20\_2014

SELECT [Transaction Id],CHECKSUM([CONTEXT\_ID],[CONTEXT\_NAME],[CONTEXT\_PARENTCONTEXTID],[Transaction ID],[Charge ID],[Parent Charge ID],

[Custom Transaction Code],[Transaction Posted By],[Transaction Created Datetime],[Charge From Date],[Charge To Date],

[First Billed Datetime],[Last Billed Datetime],[Transaction Type],[Transaction Transfer Type],[Transaction Transfer Intent],

[Transaction Reason],[Transaction Patient Ins ID],[Procedure Code],[Other Modifier],[Closed Post Date],[Voided Date],

[Charge Void Parent ID],[Claim ID],[Transaction Method],[Place of Service],[Provider Group ID],[Payment Batch ID],

[Expected Allowed Amount],[Expected Allowable Schedule ID],[Reversal Flag],[Post Date],[Work RVU],[Total RVU],[Amount],

[Number of Charges],[Units] )As C1

FROM transaction\_03\_19\_2014

intersect

SELECT [Transaction Id],CHECKSUM([CONTEXT\_ID],[CONTEXT\_NAME],[CONTEXT\_PARENTCONTEXTID],[Transaction ID],[Charge ID],[Parent Charge ID],

[Custom Transaction Code],[Transaction Posted By],[Transaction Created Datetime],[Charge From Date],[Charge To Date],

[First Billed Datetime],[Last Billed Datetime],[Transaction Type],[Transaction Transfer Type],[Transaction Transfer Intent],

[Transaction Reason],[Transaction Patient Ins ID],[Procedure Code],[Other Modifier],[Closed Post Date],[Voided Date],

[Charge Void Parent ID],[Claim ID],[Transaction Method],[Place of Service],[Provider Group ID],[Payment Batch ID],

[Expected Allowed Amount],[Expected Allowable Schedule ID],[Reversal Flag],[Post Date],[Work RVU],[Total RVU],[Amount],

[Number of Charges],[Units] )AS C1

FROM transaction\_03\_20\_2014 T2

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

Lowells code to only count a visit for the same claim number one time per date. i.e. If there are four charges on the Same claim number on the same date.

UPDATE MyTarget

      SET MyTarget.[TransVisit] = CASE

                                      WHEN ROWDATA.RW = 1

                                      THEN 1

                                      ELSE 0

                                  END

      --SELECT \*

      FROM [AthenaFactTrans] MyTarget

      INNER JOIN (

                  SELECT

                      ROW\_NUMBER() OVER(PARTITION BY X.DimFacilityKey, X.TransClaimNo, X.TransDate

                                        ORDER BY X.DimFacilityKey, X.TransClaimNo, X.TransDate,DimCPTKey) AS RW,

                      X.[TransVisit],

                      X.DimFacilityKey,

                      X.DimPatientKey,

                      X.TransDate,

                      X.TransClaimNo,

                      X.TransType,

                      X.DimCPTKey

                      FROM [AthenaFactTrans]  X

                      WHERE X.TransType = 'CHARGE'

      ) ROWDATA

      ON  MyTarget.DimFacilityKey= ROWDATA.DimFacilityKey

      AND MyTarget.TransClaimNo = ROWDATA.TransClaimNo

      AND MyTarget.TransDate    = ROWDATA.TransDate

      AND MyTarget.DimCPTKey    = ROWDATA.DimCPTKey

    WHERE MyTarget.[TransVisit] <> CASE

                                      WHEN ROWDATA.RW = 1

                                      THEN 1

                                      ELSE 0

                                  END

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BELOW CODE IS FOR ED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--DROP TABLE #COUNT1

--CREATE TABLE #Count1

--( Diag          VARCHAR(8),

-- DiagCt        INT

--)

INSERT INTO #Count1

SELECT

--T.DTDIAG,

--T.DTDIG2,

--T.DTDIG3,

T.DTDIG4,

COUNT(\*)

FROM dbo.dmtrans AS T

LEFT JOIN dmrefno AS R

ON T.dtacro = R.drfacro

AND T.dtloc = R.drfloc

AND  T.dtfamno = R.drffamno

AND T.dtrefno = R.DRFCLMNO

LEFT JOIN dbo.dmloc AS L

ON T.dtacro=L.DLOACRO

AND T.dtloc= L.DLONUM

WHERE T.dtttyp ='C'

AND R.DRFACRO NOT LIKE 'X%'

AND L.DLOTYPE ='G'

AND R.DRFFRDT BETWEEN '1/1/2013' AND '12/31/2013'

GROUP BY

--T.DTDIAG

--T.DTDIG2

--T.DTDIG3

T.DTDIG4

--

SELECT TOP 100

Diag,

SUM(DiagCt) AS DiagCount

FROM #Count1

GROUP  BY Diag

ORDER BY SUM(DiagCt) DESC

--SELECT TOP 100 \*

--FROM #Count1

--ORDER BY DiagCt DESC

INSERT INTO DimPatient( PatientNo,PatientFirstName,PatientLastName,PatientMidInt,PatientGender,PatientDOB,PatientrRelToGur,PatientSSN,DataSource )

SELECT

LTRIM(RTRIM(Staging.[Patient ID])),

LTRIM(RTRIM(Staging.[First Name])),

LTRIM(RTRIM(Staging.[Last Name])),

LTRIM(RTRIM(Staging.[Middle Initial])),

LTRIM(RTRIM(Staging.Sex)),

CASE WHEN ISDATE(Staging.DOB) =0 OR RTRIM(Staging.DOB) = '' THEN NULL ELSE CONVERT(DATETIME,Staging.DOB) END,

LTRIM(RTRIM(Staging.[Guarantor Relationship])),

LTRIM(RTRIM(Staging.[Patient SSN]))

,'Athena'

FROM   Staging\_DimPatient Staging

LEFT OUTER JOIN DimPatient ExistingData

ON ExistingData.PatientNO = Staging.[Patient ID]

WHERE  ExistingData.PatientNo IS NULL

/\*

This only Inserts new records. Includes the Key

\*/

CREATE TABLE TransSummary

(

Date DATE,

TotalVisits INT,

TotalTrans INT,

TotalChgTrans INT,

Charges INT,

Payments INT,

Adjustments INT

)

INSERT INTO TransSummary(Date, TotalVisits, TotalTrans, TotalChgTrans, Charges, Payments, Adjustments )

SELECT GETDATE(),

SUM(TransVisit), COUNT(\*), SUM(TransChgAmt),

sum(CASE when TransType = 'charge' THEN TransVisit ELSE 0 END),

sum(CASE when TransType = 'payment' THEN TransVisit ELSE 0 END),

sum(CASE when TransType = 'adjustment' THEN TransVisit ELSE 0 END)

FROM dbo.AthenaFactTrans

TRUNCATE TABLE dbo.TransSummary

SELECT CONVERT (varchar (100), GETDATE (), 101) AS Date, (A.TotalVisits - B.TotalVisits) AS DailyVisits

FROM ( SELECT TotalVisits FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), 0)) A,

( SELECT TotalVisits FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), -1) ) B;

SELECT \* FROM dbo.TransSummary

SELECT \* FROM dbo.AthenaFactTrans

SELECT CONVERT (varchar (100), GETDATE (), 101) AS Date, (A.TotalVisits - B.TotalVisits) AS DailyVisits,

(C.TotalChgTrans - D.TotalChgTrans) AS DailyCharges, E.Charges, F.Payments, G.Adjustments

FROM ( SELECT TotalVisits FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), 0)) A,

( SELECT TotalVisits FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), -1) ) B,

( SELECT TotalChgTrans FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), 0)) C,

( SELECT TotalChgTrans FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), -1) ) D,

( SELECT Charges FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), 0)) E,

( SELECT Payments FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), 0)) F,

( SELECT Adjustments FROM dbo.TransSummary WHERE Date = DATEADD(dd, DATEDIFF(dd,0,getdate()), 0)) G;

CREATE TABLE TransSummary

(

      Date DATE,

      TotalVisits INT,

      TotalTrans INT,

      TotalChgTrans INT,

      Charges INT,

      Payments INT,

      Adjustments INT

)

INSERT INTO TransSummary(Date, TotalVisits, TotalTrans, TotalChgTrans, Charges, Payments, Adjustments )

SELECT GETDATE(),

SUM(TransVisit), COUNT(\*), SUM(TransChgAmt),

sum(CASE when TransType = 'charge' THEN TransVisit ELSE 0 END),

sum(CASE when TransType = 'payment' THEN TransVisit ELSE 0 END),

sum(CASE when TransType = 'adjustment' THEN TransVisit ELSE 0 END)

FROM dbo.AthenaFactTrans

Declare @HTMLBody varchar(max),

      @TableHead varchar(max),

      @TableTail varchar(max)

Set NoCount On;

Set @TableTail = '</table></body></html>';

Set @TableHead = '<html><head>' +

                  '<style>' +

                  'td {border: solid black 1px;padding-left:5px;padding-right:5px;padding-top:1px;padding-bottom:1px;font-size:11pt;} ' +

                  '</style>' +

                  '</head>' +

                  '<body><div> This is the list of Changes<br /><br /><br /><br /><br /></div><table cellpadding=0 cellspacing=0 border=0>' +

                  '<tr bgcolor=#FFEFD8><td align=center><b>Date</b></td>' +

                  '<td align=center><b>Daily Visit Count</b></td>' +

                  '<td align=center><b>Daily Charges Count</b></td>' +

                  '<td align=center><b>Charges Count</b></td>' +

                  '<td align=center><b>Payments Count</b></td>' +

                  '<td align=center><b>Adjustments Count</b></td></tr>';

Select @HTMLBody = (Select Row\_Number() Over(Order By A.TotalVisits) % 2 As [TRRow],

       CONVERT (VARCHAR (100), Getdate (), 101) AS [TD],

       ( A.TotalVisits - B.TotalVisits )        AS [TD],

       ( C.TotalChgTrans - D.TotalChgTrans )    AS [TD],

       E.Charges                                AS [TD],

       F.Payments                               AS [TD],

       G.Adjustments                            AS [TD]

FROM   (SELECT TotalVisits

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), 0)) A,

       (SELECT TotalVisits

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), -1)) B,

       (SELECT TotalChgTrans

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), 0)) C,

       (SELECT TotalChgTrans

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), -1)) D,

       (SELECT Charges

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), 0)) E,

       (SELECT Payments

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), 0)) F,

       (SELECT Adjustments

        FROM   dbo.TransSummary

        WHERE  Date = Dateadd(dd, Datediff(dd, 0, Getdate()), 0)) G

      For XML raw('tr'), Elements)

-- Replace the entity codes and row numbers

Set @HTMLBody = Replace(@HTMLBody, '\_x0020\_', space(1))

Set @HTMLBody = Replace(@HTMLBody, '\_x003D\_', '=')

Set @HTMLBody = Replace(@HTMLBody, '<tr><TRRow>1</TRRow>', '<tr bgcolor=#C6CFFF>')

Set @HTMLBody = Replace(@HTMLBody, '<TRRow>0</TRRow>', '')

Select @HTMLBody = @TableHead + @HTMLBody + @TableTail

IF  datename(dw,getdate()) NOT IN ('Saturday','Sunday')

BEGIN

-- return output

--Select @HTMLBody

--now the email itself:

EXEC msdb.dbo.sp\_send\_dbmail

    @profile\_name='profile\_P09',

    @recipients='lizaguirre@hppartners.com;eenglish@hppartners.com',

    @subject = 'Athena Daily counts',

    @body = @HTMLBody,

    @body\_format = 'HTML'

END

SELECT t.DimFacilityKey

,t.DimFinancialClassKey

,DimGuarantorKey

,TransClaimNo

,SUM(TransAmt)

,ClaimDate

,DATEDIFF("d",ClaimDate,GETDATE())

,CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 31 THEN 30

ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 61 THEN 60

      ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 91 THEN 90

           ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 121 THEN 120

                ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 151 THEN 150

                     ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 181 THEN 180

                          ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 211 THEN 210

                               ELSE CASE WHEN DATEDIFF("d",ClaimDate,GETDATE()) < 241 THEN 240

                                    ELSE 241

                                    END

                               END

                          END

                     END

                END

           END

      END

END

FROM dbo.FactTrans t LEFT JOIN dbo.DimFinancialClass fc ON T.DimFinancialClassKey = fc.DimFinancialClassKey

                     LEFT JOIN dbo.DimFacility f ON t.DimFacilityKey = f.DimFacilityKey

GROUP BY t.DimFacilityKey

,t.DimFinancialClassKey

,DimGuarantorKey

,TransClaimNo

,ClaimDate

HAVING SUM(TransAmt) <> 0

CREATE VIEW [dbo].[vwStageClaim]

AS

SELECT

CONVERT(VARCHAR(40),[CONTEXT\_NAME]) AS [CONTEXT\_NAME],

CONVERT(INT,[CONTEXT\_PARENTCONTEXTID]) AS [CONTEXT\_PARENTCONTEXTID],

CONVERT(MONEY,[Claim Scheduling Provider ID]) AS [Claim Scheduling Provider ID],

CONVERT(INT,[Patient Rounding List ID]) AS [Patient Rounding List ID],

CONVERT(VARCHAR(10),[Patient Claim Status]) AS [Patient Claim Status],

CASE WHEN ISDATE([Claim Created Datetime]) = 0 OR RTRIM([Claim Created Datetime]) = '' THEN NULL ELSE CONVERT(DATETIME,[Claim Created Datetime]) END AS [Claim Created Datetime],

CONVERT(VARCHAR(20),[Created By]) AS [Created By],

CASE WHEN ISDATE([Claim Service Date]) = 0 OR RTRIM([Claim Service Date]) = '' THEN NULL ELSE CONVERT(DATETIME,[Claim Service Date]) END AS [Claim Service Date],

CASE WHEN ISDATE([Hospitalization From Date]) = 0 OR RTRIM([Hospitalization From Date]) = '' THEN NULL ELSE CONVERT(DATETIME,[Hospitalization From Date]) END AS [Hospitalization From Date],

CASE WHEN ISDATE([Hospitalization To Date]) = 0 OR RTRIM([Hospitalization To Date]) = '' THEN NULL ELSE CONVERT(DATETIME,[Hospitalization To Date]) END AS [Hospitalization To Date],

CONVERT(VARCHAR(250),[Reserved19]) AS [Reserved19],

' ' AS FILLER

FROM [StageClaim];

GO

UPDATE MyTarget

SET

MyTarget.GuarantorFirstName = LTRIM(RTRIM(Staging.[Guarantor First Name])),

MyTarget.GuarantorLastNname = LTRIM(RTRIM(Staging.[Guarantor Last Name])),

MyTarget.GuarantorAdd1 = LTRIM(RTRIM(Staging.[Guarantor Address])),

MyTarget.GuarantorAdd2 = LTRIM(RTRIM(Staging.[Guarantor Address 2])),

MyTarget.GuarantorCity = LTRIM(RTRIM(Staging.[Guarantor City])),

MyTarget.GuarantorState = LTRIM(RTRIM(Staging.[Guarantor State])),

MyTarget.GuarantorZIP = LTRIM(RTRIM(Staging.[Guarantor Zip])),

MyTarget.GuarantorZIP4 = LTRIM(RTRIM(Staging.[Guarantor Zip4])),

MyTarget.GuarantorForeignZIP = LTRIM(RTRIM(Staging.[Guarantor Foreign Zip]))

FROM DimGuarantor MyTarget

INNER JOIN Staging\_DimGuarantor Staging

ON MyTarget.PatientNo = Staging.[Patient ID]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BELOW CODE IS FOR ED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--DROP TABLE #COUNT1

--CREATE TABLE #Count1

--( Diag VARCHAR(8),

-- DiagCt INT

--)

INSERT INTO #Count1

SELECT

--T.DTDIAG,

--T.DTDIG2,

--T.DTDIG3,

T.DTDIG4,

COUNT(\*)

FROM dbo.dmtrans AS T

LEFT JOIN dmrefno AS R

ON T.dtacro = R.drfacro

AND T.dtloc = R.drfloc

AND T.dtfamno = R.drffamno

AND T.dtrefno = R.DRFCLMNO

LEFT JOIN dbo.dmloc AS L

ON T.dtacro=L.DLOACRO

AND T.dtloc= L.DLONUM

WHERE T.dtttyp ='C'

AND R.DRFACRO NOT LIKE 'X%'

AND L.DLOTYPE ='G'

AND R.DRFFRDT BETWEEN '1/1/2013' AND '12/31/2013'

GROUP BY

--T.DTDIAG

--T.DTDIG2

--T.DTDIG3

T.DTDIG4

--

SELECT TOP 100

Diag,

SUM(DiagCt) AS DiagCount

FROM #Count1

GROUP BY Diag

ORDER BY SUM(DiagCt) DESC

--SELECT TOP 100 \*

--FROM #Count1

--ORDER BY DiagCt DESC

--

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BELOW CODE IS FOR HOSP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--DROP TABLE #COUNT2

--CREATE TABLE #Count2

--( Diag VARCHAR(8),

-- DiagCt INT

--)

INSERT INTO #Count2

SELECT

--T.DTDIAG,

--T.DTDIG2,

--T.DTDIG3,

T.DTDIG4,

COUNT(\*)

FROM dbo.dmtrans AS T

LEFT JOIN dmrefno AS R

ON T.dtacro = R.drfacro

AND T.dtloc = R.drfloc

AND T.dtfamno = R.drffamno

AND T.dtrefno = R.DRFCLMNO

LEFT JOIN dbo.dmloc AS L

ON T.dtacro=L.DLOACRO

AND T.dtloc= L.DLONUM

WHERE T.dtttyp ='C'

AND R.DRFACRO NOT LIKE 'X%'

AND L.DLOTYPE <>'G'

AND R.DRFFRDT BETWEEN '1/1/2013' AND '12/31/2013'

GROUP BY

--T.DTDIAG

--T.DTDIG2

--T.DTDIG3

T.DTDIG4

--

SELECT TOP 100

Diag,

SUM(DiagCt) AS DiagCount

FROM #Count2

GROUP BY Diag

ORDER BY SUM(DiagCt) DESC

--SELECT \*

--FROM #count

--remove today’s row if we are rerunning the process

DELETE

FROM TransSummary

WHERE Dateadd(dd, Datediff(dd, 0, Date), 0) = Dateadd(dd, Datediff(dd, 0, Getdate()), 0)

with MyQuery As (

select row\_number() over (order by name) as RowNumber,\*

from sysobjects )

Select \* from MyQuery

where RowNumber between 101 and 200

select @@SERVERNAME

select @@version

SELECT HOST\_NAME()

DROP TABLE BULKACT

CREATE TABLE BULKACT(RAWDATA VARCHAR (8000))

BULK INSERT BULKACT FROM '\\HOL-WKS-507\C$\Users\eenglish\Documents\BulkInsertTestFile.txt'

WITH (

DATAFILETYPE = 'char',

FIELDTERMINATOR = ',',

ROWTERMINATOR = '\n',

FIRSTROW = 1

)

select \* from bulkact

select count(\*) as count,rawdata

from bulkact

group by rawdata

order by rawdata

/\*\*\*This code is to correct column name spelling \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ALTER TABLE dbo.DataDU

ALTER COLUMN RVUPerVisirt float null

EXEC SP\_RENAME 'dbo.DataDU.RVUPerVisit', 'RVUPerVisit', 'COLUMN'

GO

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

THIS code below updates the Subscribers Database for a Data Driven Subsccription Using a SQL Task and SSIS

DECLARE @AcctPeriodDate datetime

DECLARE @ClosingDate datetime

DECLARE @FilEType bit

DECLARE @Format VARCHAR(10)

DECLARE @OutputName VARCHAR(75)

SELECT TOP 1 @AcctPeriodDate = AcctPeriodDate, @ClosingDate = ClosingDate, @FilEType = FileType, @Format = Format, @OutputName = OutputName

FROM dbo.CPU\_KBD\_AccountingPeriodByFCNoDrilldown

INSERT INTO [Subscribers].[dbo].CPU\_KBD\_AccountingPeriodByFCNoDrilldown

(

[AcctPeriodDate]

,[ClosingDate]

,[Location]

,[OutputName]

,[FileType]

,[Format]

)

SELECT @AcctPeriodDate,@ClosingDate,FacilityID\_Add, Outputname\_add + ' -' + RIGHT(@Outputname,30),@FilEType,@Format

from dbo.CPU\_FacilityAdd

/\*\*\*\*\*\*\*\*THIS SP below searches all tables and columns for a specific data\*\*\*\*\*\*\*/

USE [master]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[sp\_UGLYSEARCH] Script Date: 05/06/2014 10:56:48 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[sp\_UGLYSEARCH]

/\*

--Purpose: to search every string column in a databasefor a specific word

--returns sql statement as a string which idnetifies the matching table

-- or when the optional parameter is used, the sql statement for the specific matching column.

--usage:

-- EXEC sp\_UGLYSEARCH 'Provisional'

-- EXEC sp\_UGLYSEARCH 'TEST'

-- creates one SQL for each table that actually has a match for the searched value i.e.

-- SELECT \* FROM [ACACTSCR] WHERE [DESCRIP] LIKE '%TEST%' OR [TITLE] LIKE '%TEST%'

--optional parameter SEARCHBYCOLUMN

-- EXEC sp\_UGLYSEARCH 'TEST',1

-- creates one SQL for each Column that actually has a match for the searched value i.e.

-- SELECT \* FROM [dbo].[ACACTSCR] WHERE [DESCRIP] LIKE '%TEST%'

-- SELECT \* FROM [dbo].[ACACTSCR] WHERE [TITLE] LIKE '%TEST%'

\*/

@SEARCHSTRING VARCHAR(50),

@SEARCHBYCOLUMN INT = 0

AS

BEGIN

SET NOCOUNT ON

DECLARE @SQL VARCHAR(max),

@SCHEMANAME VARCHAR(100),

@TABLENAME VARCHAR(100),

@COLUMNNAME VARCHAR(100),

@COLZ VARCHAR(max)

CREATE TABLE #RESULTS(SCHEMANAME VARCHAR(100), TBLNAME VARCHAR(100),COLNAME VARCHAR(100),SQL VARCHAR(max))

SELECT

SCHEMA\_NAME(schema\_id) AS SCHEMANAME,

objz.name AS TBLNAME,

colz.name AS COLNAME,

TYPE\_NAME(colz.user\_type\_id) AS DATATYPE

INTO #TEMP

FROM sys.objects objz

INNER JOIN sys.columns colz ON objz.object\_id = colz.object\_id

WHERE objz.type='U'

AND TYPE\_NAME(colz.user\_type\_id) IN ('VARCHAR','NVARCHAR','CHAR','NCHAR')

AND ( colz.max\_length >= LEN('test') --smart: don't search varchar(1) columns for 'TEST' 4xmpl

OR colz.max\_length=-1 --varchar(max) needs searching as well?

)

ORDER BY TBLNAME,COLNAME

IF @SEARCHBYCOLUMN = 0

BEGIN

DECLARE C1 CURSOR FOR

SELECT SCHEMANAME,TBLNAME,COLNAME FROM #TEMP ORDER BY SCHEMANAME,TBLNAME,COLNAME

OPEN C1

FETCH NEXT FROM C1 INTO @SCHEMANAME,@TABLENAME,@COLUMNNAME

WHILE @@FETCH\_STATUS <> -1

BEGIN

SET @COLZ=''

SELECT @COLZ = @COLZ + QUOTENAME(COLNAME) + ' LIKE ''%' + @SEARCHSTRING + '%'' OR ' FROM #TEMP WHERE TBLNAME=@TABLENAME

--@COLZ has a trailing 'OR ' which must be removed

SET @COLZ = SUBSTRING(@COLZ,1,LEN(@COLZ) -3)

--PRINT @COLZ

SET @SQL = 'IF EXISTS(SELECT \* FROM ' + QUOTENAME(@SCHEMANAME) + '.' + QUOTENAME(@TABLENAME) + ' WHERE ' + @COLZ + ') INSERT INTO #RESULTS(TBLNAME,COLNAME,SQL) VALUES(''' + @TABLENAME + ''',''-'','' SELECT \* FROM ' + QUOTENAME(@TABLENAME) + ' WHERE ' + REPLACE(@COLZ,'''','''''') + ''') ;'

--PRINT @SQL

EXEC (@SQL)

FETCH NEXT FROM C1 INTO @SCHEMANAME,@TABLENAME,@COLUMNNAME

END

CLOSE C1

DEALLOCATE C1

END

ELSE --@SEARCHBYCOLUMN <> 0

BEGIN

DECLARE C2 CURSOR FOR

SELECT SCHEMANAME,TBLNAME,COLNAME FROM #TEMP ORDER BY SCHEMANAME,TBLNAME,COLNAME

OPEN C2

FETCH NEXT FROM C2 INTO @SCHEMANAME,@TABLENAME,@COLUMNNAME

WHILE @@FETCH\_STATUS <> -1

BEGIN

SET @SQL = 'IF EXISTS(SELECT \* FROM ' + QUOTENAME(@SCHEMANAME) + '.' + QUOTENAME(@TABLENAME) + ' WHERE ' + QUOTENAME(@COLUMNNAME) + ' LIKE ''%' + @SEARCHSTRING + '%'')

INSERT INTO #RESULTS(SCHEMANAME,TBLNAME,COLNAME,SQL) VALUES(''' + @SCHEMANAME + ''',''' + @TABLENAME + ''',''' + @COLUMNNAME + ''',''

SELECT \* FROM ' + QUOTENAME(@SCHEMANAME) + '.' + QUOTENAME(@TABLENAME) + ' WHERE ' + QUOTENAME(@COLUMNNAME) + ' LIKE ''''%' + @SEARCHSTRING + '%'''' '') ;'

PRINT @SQL

EXEC (@SQL)

FETCH NEXT FROM C2 INTO @SCHEMANAME,@TABLENAME,@COLUMNNAME

END

CLOSE C2

DEALLOCATE C2

END --@SEARCHBYCOLUMN <> 0

SELECT \* FROM #RESULTS ORDER BY TBLNAME,COLNAME

END --PROC

SELECT Dateadd(dd, Datediff(dd, 0, Getdate()), 0) AS TheDate,

Sum(TransVisit) AS TotalVisits ,

SUM(CASE

WHEN TransType = 'charge' THEN TransVisit

WHEN TransType = 'payment' THEN 1

WHEN TransType = 'adjustment' THEN 1

ELSE 0 END) AS TotalTrans,

Sum(TransChgAmt) AS TotalChgTrans,

Sum(CASE

WHEN TransType = 'charge' THEN TransVisit

ELSE 0

END) AS Charges,

Sum(CASE

WHEN TransType = 'payment' THEN 1

ELSE 0

END) AS Payments,

Sum(CASE

WHEN TransType = 'adjustment' THEN 1

ELSE 0

END) AS Adjustments

FROM dbo.AthenaFactTrans

**SCRIPTING IN TSQL:**

## Script No.1 - Identity Fields in Table Variables

This script is more related to table variables than identity fields:

DECLARE @T AS TABLE

([ID] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[Name] [varchar](50) NOT NULL)

INSERT INTO @T([Name]) VALUES ('Student No. 1')

INSERT INTO @T([Name]) VALUES ('Student No. 2')

INSERT INTO @T([Name]) VALUES ('Student No. 3')

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRAN

INSERT INTO @T([Name]) VALUES ('Student No. 4')

ROLLBACK TRAN

INSERT INTO @T([Name]) VALUES ('Student No. 4')

SELECT [ID], [Name] FROM @T

It first defines a table variable that has an identity field, and then inserts some data into the table variable. The insertion of "Student No. 4" is in a transaction which is rolled back immediately. After the transaction is rolled back, the "Student No. 4" is inserted again.

Run the script, we get the following result:



We can find that the "Student No. 4" shows up twice. What this means is that table variables do not participate in transactions. If we want to rely on transactions to determine the data contents in table variables, we shall be very careful.

## Script No.2 - Identity Fields in Temporary Tables

**SPLITTER function (need an example here)**

This function is used to separate multiple parameters in a SP for a SSRS report.

if exists (select \* from dbo.sysobjects where id = ob-ject\_id(N'[dbo].[fn\_Split]') and xtype in (N'FN', N'IF', N'TF'))

drop function [dbo].[fn\_Split]

GO

SET QUOTED\_IDENTIFIER OFF

GO

SET ANSI\_NULLS OFF

GO

CREATE  FUNCTION fn\_Split(@text varchar(8000), @delimiter varchar(20) = ' ')

RETURNS @Strings TABLE

(

  position int IDENTITY PRIMARY KEY,

  value varchar(8000)

)

AS

BEGIN

DECLARE @index int

SET @index = -1

WHILE (LEN(@text) > 0)

  BEGIN

    SET @index = CHARINDEX(@delimiter , @text)

    IF (@index = 0) AND (LEN(@text) > 0)

      BEGIN

        INSERT INTO @Strings VALUES (@text)

          BREAK

      END

    IF (@index > 1)

      BEGIN

        INSERT INTO @Strings VALUES (LEFT(@text, @index - 1))

        SET @text = RIGHT(@text, (LEN(@text) - @index))

      END

    ELSE

      SET @text = RIGHT(@text, (LEN(@text) - @index))

    END

  RETURN

END

GO

SET QUOTED\_IDENTIFIER OFF

GO

SET ANSI\_NULLS ON

GO

CREATE FUNCTION [dbo].[DelimitedSplit8K]

--===== Define I/O parameters

(@pString VARCHAR(8000), @pDelimiter CHAR(1))

--WARNING!!! DO NOT USE MAX DATA-TYPES HERE! IT WILL KILL PERFORMANCE!

RETURNS TABLE WITH SCHEMABINDING AS

RETURN

--===== "Inline" CTE Driven "Tally Table" produces values from 1 up to 10,000...

-- enough to cover VARCHAR(8000)

WITH E1(N) AS (

SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL

SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL

SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1

), --10E+1 or 10 rows

E2(N) AS (SELECT 1 FROM E1 a, E1 b), --10E+2 or 100 rows

E4(N) AS (SELECT 1 FROM E2 a, E2 b), --10E+4 or 10,000 rows max

cteTally(N) AS (--==== This provides the "base" CTE and limits the number of rows right up front

-- for both a performance gain and prevention of accidental "overruns"

SELECT TOP (ISNULL(DATALENGTH(@pString),0)) ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) FROM E4

),

cteStart(N1) AS (--==== This returns N+1 (starting position of each "element" just once for each delimiter)

SELECT 1 UNION ALL

SELECT t.N+1 FROM cteTally t WHERE SUBSTRING(@pString,t.N,1) = @pDelimiter

),

cteLen(N1,L1) AS(--==== Return start and length (for use in substring)

SELECT s.N1,

ISNULL(NULLIF(CHARINDEX(@pDelimiter,@pString,s.N1),0)-s.N1,8000)

FROM cteStart s

)

--===== Do the actual split. The ISNULL/NULLIF combo handles the length for the final element when no delimiter is found.

SELECT ItemNumber = ROW\_NUMBER() OVER(ORDER BY l.N1),

Item = SUBSTRING(@pString, l.N1, l.L1)

FROM cteLen l

Make no doubt about it. The current best way to split delimited strings in SQL Server is to use a CLR splitter. However, if, for whatever reason, you cannot use a CLR splitter, the new DelimitedSplit8K function provides a close second

 Create Function Split(@StringToSplit nvarchar(max), @splitOnChars nvarchar(max) )

       returns Table (

              Results nvarchar(max)

       )

       AS

              External name CLRDemo.CLRDemo.CLRFunctions.Split

       GO

9. Finally, you can test the new split function…

select \* from dbo.Split('1,2,3,4,5:6:7~8~9',',:~')

EXEC sp\_helpindex 'Sales.Customer'

--This code identifies the report run times

use reportServer

SELECT RTRIM([Name]) AS Rpt\_Name,

ltrim(rtrim (replace ([UserName], 'CALEDONIA\', ''))) AS User\_ID, cast((CAST(((datediff(ms,[TimeStart], [TimeEnd]))/1000)AS REAL)/60) as NUMERIC(10,2)) AS Run\_Mins,

[TimeStart],[Format] AS type,

parameters

FROM [ExecutionLog]

LEFT OUTER JOIN [Catalog] ON [ReportID] = [ItemID]

WHERE [TimeStart] > '11/12/2013'

ORDER BY [TimeStart] DESC

SELECT Description, LastStatus, LastRunTime

FROM dbo.Subscriptions

WHERE LEFT(Description,3) = 'cpu'

ORDER BY LastRunTime DESC

use reportServer

SELECT \*

FROM dbo.RunningJobs

/\*

This code merges data between two tables.

\*/

UPDATE MyTarget SET MyTarget.FacilityState = MySource.FacilityState

--SELECT  MyTarget.FacilityState,MySource.FacilityState, \*

FROM DW\_For\_BI360.dbo.DimFacility AS MyTarget

INNER JOIN DW\_For\_BI360\_Test.dbo.DimFacility\_ED\_01092014 AS MySource

ON MyTarget.DimFacilityKey = MySource.DimFacilityKey

WHERE MyTarget.FacilityState IS NULL

AND MySource.FacilityState IS NOT NULL

**SYSTEM TABLES**

**CODE TO SEARCH TABLES for INDEXES**

**CODE TO SEARCH for USER NAME**

SELECT \*

FROM sys.database\_principals

WHERE name = N'CONNECTIONS\connectsmart')

**CODE TO SEARCH DATABASES for COMPATIBILITY**

SELECT Compatibility\_level, Collation\_name

FROM sys.databases

WHERE name = 'cwwebapp\_connections'

sp\_helpstats 'dbo.Company';

**CODE TO SEARCH ENTIRE DATABASE by COLUMN or TABLE**

SELECT t.name AS table\_name,

SCHEMA\_NAME(schema\_id) AS schema\_name,

c.name AS column\_name

FROM sys.tables AS t

INNER JOIN sys.columns c ON t.OBJECT\_ID = c.OBJECT\_ID

WHERE c.name LIKE '%board\_name%'

ORDER BY schema\_name, table\_name;

SELECT t.name AS table\_name,

SCHEMA\_NAME(schema\_id) AS schema\_name,

c.name AS column\_name

FROM sys.tables AS t

INNER JOIN sys.columns c ON t.OBJECT\_ID = c.OBJECT\_ID

WHERE t.name LIKE '%ticket%'

ORDER BY schema\_name, table\_name;

select \* from SYS.ALL\_VIEWS

SELECT t.name AS table\_name,

SCHEMA\_NAME(schema\_id) AS schema\_name,

c.name AS column\_name

FROM SYS.ALL\_VIEWS AS t

INNER JOIN sys.columns c ON t.OBJECT\_ID = c.OBJECT\_ID

WHERE c.name LIKE '%board\_name%'

ORDER BY schema\_name, table\_name;

SELECT t.name AS table\_name,

SCHEMA\_NAME(schema\_id) AS schema\_name,

c.name AS column\_name

FROM SYS.ALL\_VIEWS AS t

INNER JOIN sys.columns c ON t.OBJECT\_ID = c.OBJECT\_ID

WHERE t.name LIKE '%All\_ticket%'

ORDER BY schema\_name, table\_name;

SELECT c.name AS column\_name

FROM SYS.ALL\_VIEWS AS V

INNER JOIN sys.columns C ON V.OBJECT\_ID = C.OBJECT\_ID

WHERE V.name = 'v\_cbi\_All\_Tickets'

**CODE TO QUERY the DMVs to VERIFY ROWCOUNTS**

**BEFORE RUNNING EXECUTION PLAN.**

SELECT SO.name, SP.rows

FROM sys.objects AS SO

INNER JOIN sys.partitions AS SP

ON SO.object\_id = SP.object\_id

WHERE SO.type IN (‘TF’, ‘U’, ‘V’)

AND sp.index\_id =1

ORDER BY SO.name

USE [Patient\_Analysis]

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[sp\_EOMARAgingTrialBalance]    Script Date: 01/16/2012 10:57:56 \*\*\*\*\*\*/

SET ANSI\_NULLS OFF

GO

SET QUOTED\_IDENTIFIER OFF

GO

ALTER PROCEDURE [dbo].[sp\_EOMARAgingTrialBalance] (@EndDate SmallDateTime, @ReportType Int)

AS

If @ReportType = 0 Begin

SELECT                              I.INSURANCECATEGORY,

                                    SUM(T.UNCREDIT) AS UNCREDIT,

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 0 AND DATEDIFF(DD,T.SHIP\_DATE, @EndDate) <= 30 THEN T.BALANCE ELSE 0 END) AS [0\_30],

                                    Null As [0\_30Invoice],

                                    Null As [0\_30Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 31 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 60 THEN T.BALANCE ELSE 0 END) AS [31\_60],

                                    Null As [31\_60Invoice],

                                    Null As [31\_60Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 61 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 90 THEN T.BALANCE ELSE 0 END) AS [61\_90],

                                    Null As [61\_90Invoice],

                                    Null As [61\_90Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 91 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 120 THEN T.BALANCE ELSE 0 END) AS [91\_120],

                                    Null As [91\_120Invoice],

                                    Null As [91\_120Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 121 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 180 THEN T.BALANCE ELSE 0 END) AS [121\_180],

                                    Null As [121\_180Invoice],

                                    Null As [121\_180Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 181 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 360 THEN T.BALANCE ELSE 0 END) AS [181\_360],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 361 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 540 THEN T.BALANCE ELSE 0 END) AS [361\_540],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 541 THEN T.BALANCE ELSE 0 END) AS [541\_OVER],

                                    Null As [541\_OVERInvoice],

                                    Null As [541\_OVERPatient],

                                    Null As TotalInvoices,

                                    Null As TotalPatients

FROM                          [HBS PATIENT INFORMATION].DBO.TRIALBALANCE\_MONTHEND T (NoLock)

LEFT OUTER JOIN[HBS PATIENT INFORMATION].DBO.UNIQUEINSURANCES U (NoLock)

ON U.PRVCOM = T.INS\_CODE

LEFT OUTER JOIN[HBS PATIENT INFORMATION].DBO.INSURANCECATEGORIES I (NoLock) ON I.INSURANCECATEGORYKEY = T.INSURANCECATEGORYKEY

WHERE T.THROUGHDATE = @EndDate

GROUP BY I.INSURANCECATEGORY

End Else If @ReportType = 1 Begin

SELECT   Case

       When I.InsuranceCategoryKey In (4,5,6,15)

Then 'Government'

         When I.InsuranceCategoryKey In (2)

Then 'Drug Card'

         When I.InsuranceCategoryKey In (13,16)

Then 'Private Pay'

         Else 'Commercial'

         End As InsuranceCategory,

                                    --I.INSURANCECATEGORY,

                                    SUM(T.UNCREDIT) AS UNCREDIT,

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 0 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 90 THEN T.BALANCE ELSE 0 END) AS [0\_30],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 0 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 90 THEN T.Order\_No ELSE 0 END) AS [0\_30Invoice],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 0 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 90 THEN T.Patno ELSE 0 END) AS [0\_30Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 91 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 180 THEN T.BALANCE ELSE 0 END) AS [31\_60],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 91 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 180 THEN T.Order\_No ELSE 0 END) AS [31\_60Invoice],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 91 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 180 THEN T.Patno ELSE 0 END) AS [31\_60Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 181 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 270 THEN T.BALANCE ELSE 0 END) AS [61\_90],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 181 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 270 THEN T.Order\_No ELSE 0 END) AS [61\_90Invoice],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 181 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 270 THEN T.Patno ELSE 0 END) AS [61\_90Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 271 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 360 THEN T.BALANCE ELSE 0 END) AS [91\_120],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 271 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 360 THEN T.Order\_No ELSE 0 END) AS [91\_120Invoice],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 271 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 360 THEN T.Patno ELSE 0 END) AS [91\_120Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 361 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 720 THEN T.BALANCE ELSE 0 END) AS [121\_180],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 361 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 720 THEN T.Order\_No ELSE 0 END) AS [121\_180Invoice],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 361 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 720 THEN T.Patno ELSE 0 END) AS [121\_180Patient],

                                    Null AS [181\_360],

                                    Null AS [361\_540],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 721 THEN T.BALANCE ELSE 0 END) AS [541\_OVER],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 721 THEN T.ORDER\_NO ELSE 0 END) AS [541\_OVERInvoice],

                                    Count(Distinct CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 721 THEN T.Patno ELSE 0 END) AS [541\_OVERPatient],

                                    COUNT(Distinct T.Order\_No) As TotalInvoices,

                                    COUNT(Distinct T.Patno) As TotalPatients

FROM                          [HBS PATIENT INFORMATION].DBO.TRIALBALANCE\_Current T (NoLock)  LEFT OUTER JOIN

                                    [HBS PATIENT INFORMATION].DBO.UNIQUEINSURANCES U (NoLock) ON U.PRVCOM = T.INS\_CODE LEFT OUTER JOIN

                                    [HBS PATIENT INFORMATION].DBO.INSURANCECATEGORIES I (NoLock) ON I.INSURANCECATEGORYKEY = T.INSURANCECATEGORYKEY

WHERE                         T.THROUGHDATE = @EndDate

GROUP BY                      Case

                                          When I.InsuranceCategoryKey In (4,5,6,15) Then 'Government'

                                          When I.InsuranceCategoryKey In (2) Then 'Drug Card'

                                          When I.InsuranceCategoryKey In (13,16) Then 'Private Pay'

                                          Else 'Commercial'

                                    End

End Else If @ReportType = 2 Begin

SELECT                              I.INSURANCECATEGORY,

                                    SUM(T.UNCREDIT) AS UNCREDIT,

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 0 AND DATEDIFF(DD,T.SHIP\_DATE, @EndDate) <= 90 THEN T.BALANCE ELSE 0 END) AS [0\_30],

                                    Null As [0\_30Invoice],

                                    Null As [0\_30Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 91 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 180 THEN T.BALANCE ELSE 0 END) AS [31\_60],

                                    Null As [31\_60Invoice],

                                    Null As [31\_60Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 181 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 360 THEN T.BALANCE ELSE 0 END) AS [61\_90],

                                    Null As [61\_90Invoice],

                                    Null As [61\_90Patient],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 361 AND DATEDIFF(DD, T.SHIP\_DATE, @EndDate) <= 540 THEN T.BALANCE ELSE 0 END) AS [91\_120],

                                    Null As [91\_120Invoice],

                                    Null As [91\_120Patient],

                                    Null AS [121\_180],

                                    Null As [121\_180Invoice],

                                    Null As [121\_180Patient],

                                    Null AS [181\_360],

                                    Null AS [361\_540],

                                    SUM(CASE WHEN DATEDIFF(DD, T.SHIP\_DATE, @EndDate) >= 541 THEN T.BALANCE ELSE 0 END) AS [541\_OVER],

                                    Null As [541\_OVERInvoice],

                                    Null As [541\_OVERPatient],

                                    Null As TotalInvoices,

                                    Null As TotalPatients

FROM                          [HBS PATIENT INFORMATION].DBO.TRIALBALANCE\_CURRENT T (NoLock)  LEFT OUTER JOIN

                                    [HBS PATIENT INFORMATION].DBO.UNIQUEINSURANCES U (NoLock) ON U.PRVCOM = T.INS\_CODE LEFT OUTER JOIN

                                    [HBS PATIENT INFORMATION].DBO.INSURANCECATEGORIES I (NoLock) ON I.INSURANCECATEGORYKEY = T.INSURANCECATEGORYKEY

WHERE                         T.THROUGHDATE = @EndDate

GROUP BY                      I.INSURANCECATEGORY

End

CREATE PROCEDURE spCaliforniaAuthors  
AS  
SELECT \* FROM authors  
WHERE state = 'CA'  
ORDER BY zip

EXECUTE spCaliforniaAuthors

Notice that I can also use a variable as the value for the SET clause of my UPDATE statement. And finally, the complete stored procedure:

CREATE PROCEDURE spNewValue @pPKValue int, @pStatus char(8) AS

SET NOCOUNT ON

INSERT MyTable

SELECT Field1,

Field2,

'Active' as Status

From MyTable

Where PKValue = @pPKValue

UPDATE MyTable

SET Status = @pStatus

WHERE PKValue = @pPKValue

go

I include a SET NOCOUNT ON so that no extra rows are returned to my client application

**MY EXAMPLE:**

USE [RCMANALYST]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ENGLISH\_Medco\_LMS\_Weekly\_Report] Script Date: 05/30/2012 09:58:36 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Edward English>

-- Create date: <04/13/2012>

-- Description: <Used for my SSRS report for MEDCO Part B LMS weekly report 8B2,8CI>

-- History

-- =============================================

CREATE PROCEDURE [dbo].[ENGLISH\_Medco\_LMS\_Weekly\_Report]

-- Add the parameters for the stored procedure here

--<@Param1, sysname, @p1> <Datatype\_For\_Param1, , int> = <Default\_Value\_For\_Param1, , 0>,

--<@Param2, sysname, @p2> <Datatype\_For\_Param2, , int> = <Default\_Value\_For\_Param2, , 0>

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

--Clean out Primary Report Table

--DROP TABLE RCMANALYST.dbo.ENGLISH\_MEDCO\_PartB\_LMS\_WeeklyReport;

--Insert New Primary data

-- Insert statements for procedure here

SELECT

DISTINCT P.INSNO,A.CUS\_NUM\_AR,A.INS\_CO\_AR,A.APPLY\_TO\_AR,

SUM(CONVERT(money, A.TOTALAMOUNT)) AS 'TOTAL\_OPEN\_AR',

P.EXP\_DATE

--INTO RCMANALYST.dbo.ENGLISH\_MEDCO\_PartB\_LMS\_WeeklyReport

FROM SQLREPL.LMSCU.dbo.PATCVG AS P

INNER JOIN SQLREPL.AR.dbo.tblAROPEN AS A

ON P.CUSNO = A.CUS\_NUM\_AR

WHERE P.INSNO IN ('8B2','8CI')

AND P.EXP\_DATE = 99991231

AND A.TOTALAMOUNT >0

AND A.INS\_CO\_AR = '0'

GROUP BY P.INSNO,A.CUS\_NUM\_AR,A.INS\_CO\_AR,A.APPLY\_TO\_AR,P.EXP\_DATE

ORDER BY P.INSNO,A.CUS\_NUM\_AR,A.APPLY\_TO\_AR

END

GO

USE [AR]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[sp\_AR\_AGING\_BY\_PATIENT\_DOS] Script Date: 06/07/2012 13:21:43 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER OFF

GO

CREATE PROCEDURE [dbo].[sp\_AR\_AGING\_BY\_PATIENT\_DOS] AS

INSERT INTO dbo.tbl\_AGING\_BY\_PATIENT\_DOS ( GroupID, ARTypeID, CUS\_NUM\_AR, NAME\_CM, INS\_CO\_AR, INS\_NAME\_IS,

CustType, DOTO30, D31TO60, D61TO90, D91TO120, D121TO150, D151TO180, D181TO210, OVER210 )

SELECT dbo.tblAR\_AGING\_BASE\_DOS.GroupID, dbo.tblAR\_AGING\_BASE\_DOS.ARTYPEID,

dbo.tblAR\_AGING\_BASE\_DOS.CUS\_NUM\_AR, dbo.tblAR\_AGING\_BASE\_DOS.NAME\_CM,

dbo.tblAR\_AGING\_BASE\_DOS.INS\_CO\_AR, dbo.tblAR\_AGING\_BASE\_DOS.INS\_NAME\_IS, dbo.tblAR\_AGING\_BASE\_DOS.CustType,

SUM(CASE WHEN [EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 31) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 30) AND

[EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 61) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 60) AND

[EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 91) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 90) AND

[EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 121) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 120) AND

[EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 151) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 150) AND

[EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 181) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 180) AND

[EffctDt] > (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 211) THEN

[TOTALAMOUNT] ELSE 0 END),

SUM(CASE WHEN [EffctDt] < (CONVERT(DATETIME, CONVERT(VARCHAR, GETDATE(), 102)) - 210) THEN

[TOTALAMOUNT] ELSE 0 END)

FROM dbo.tblAR\_AGING\_BASE\_DOS

GROUP BY dbo.tblAR\_AGING\_BASE\_DOS.GroupID, dbo.tblAR\_AGING\_BASE\_DOS.ARTYPEID,

dbo.tblAR\_AGING\_BASE\_DOS.CUS\_NUM\_AR, dbo.tblAR\_AGING\_BASE\_DOS.NAME\_CM,

dbo.tblAR\_AGING\_BASE\_DOS.INS\_CO\_AR, dbo.tblAR\_AGING\_BASE\_DOS.INS\_NAME\_IS,

dbo.tblAR\_AGING\_BASE\_DOS.CustType

**MY ARAging code**

**/\***

**This statement needs to be verified!!!**

**When using SSRS I don’t need to declare the parameters and assign a data type in the stored procedure or my code because I declare them in SSRS**

**\*/**

--IF @ReportType = LMS

--AND @AgingBuckets = 90

BEGIN

SELECT

CASE WHEN A.INS\_CO\_AR IN ('mc1','mc2','mc3','mc4','mca','mcb','mcc','mcd',

'mcf','sx4','sx5','8aa','8ab','8ac','8ad','8cc')

THEN 'MEDICARE'

WHEN A.INS\_CO\_AR IN ('15R','MAK','TG4','ST2','1A9','RB9','05Z','SO7','19X','EE1','0NQ','BC8',

'HE8',)

THEN 'MEDICAID'

WHEN A.INS\_CO\_AR IN ('0','000')

THEN 'PRIVATE PAY'

ELSE 'OTHER'

END AS 'INSURANCE',

--The group by is summing these total amounts

SUM(CONVERT(money, A.TOTALAMOUNT)) AS 'TOTAL\_OPEN\_AR',

COUNT(DISTINCT A.APPLY\_TO\_AR) AS 'TOTAL\_INVOICE',

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=0

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=90

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [DAYS\_0\_to\_90],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=0

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=90

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [INVOICE\_0\_to\_90],

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=91

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=180

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [DAYS\_91\_to\_180],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=91

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=180

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [INVOICE\_91\_to\_180],

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=181

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=270

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [DAYS\_181\_to\_270],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=181

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=270

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [INVOICE\_181\_to\_270],

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=271

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=360

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [DAYS\_271\_to\_360],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=271

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=360

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [INVOICE\_271\_to\_360],

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=361

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=450

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [DAYS\_361\_to\_450],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=361

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=450

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [INVOICE\_361\_to\_450],

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=451

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=540

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [DAYS\_451\_to\_540],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=451

AND DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate() ) <=540

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [INVOICE\_451\_to\_540],

SUM(CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=541

THEN CONVERT(money, A.TOTALAMOUNT) ELSE 0 END) AS [OVER 540 DAYS],

COUNT(DISTINCT CASE WHEN DATEDIFF(dd, C.SHIP\_DT\_CH, Getdate()) >=540

THEN A.APPLY\_TO\_AR ELSE 0 END) AS [OVER\_540 INVOICE]

FROM AR.dbo.tblAROPEN AS A WITH (NOLOCK)

INNER JOIN AR.dbo.tblCLMHDR AS C WITH (NOLOCK)

ON A.CUS\_NUM\_AR = C.CUSTNUM\_CH

AND A.APPLY\_TO\_AR = C.fINVNUM\_CH

--WHERE A.TOTALAMOUNT <>0

GROUP BY

CASE WHEN A.INS\_CO\_AR IN ('mc1','mc2','mc3','mc4','mca','mcb','mcc','mcd',

'mcf','sx4','sx5','8aa','8ab','8ac','8ad','8cc')

THEN 'MEDICARE'

WHEN A.INS\_CO\_AR IN ('15R','MAK','TG4','ST2','1A9','RB9','05Z','SO7','19X','EE1','0NQ','BC8',

'HE8')

THEN 'MEDICAID'

WHEN A.INS\_CO\_AR IN ('0','000')

THEN 'PRIVATE PAY'

ELSE 'OTHER'

END

**CODE SNIPET**

/\*

This says start after the 4th character, count the length up to the first empty space and then subtract the original 4 characters leaving me with the substring I need. This pulls out AB1 and 6780

**AB116780 115**

\*/

SELECT top 100

ft.TransUnique,

SUBSTRING( ft.TransUnique,1,3) As FacilityAcronym,

SUBSTRING( ft.TransUnique,4,CHARINDEX(' ',ft.TransUnique) -4) As CPUFamilyNumber

FROM DW\_For\_BI360.dbo.FactTrans ft

WHERE   ft.TransType = 'C'

  AND ft.DTMODF != '\*P'

  AND ft.TransDateOfService > '06/01/2014'

**ROW\_NUMBER() VS RANK() VS DENSE\_RANK()**

|  |  |
| --- | --- |
|  | CREATE TABLE t AS  SELECT 'a' v UNION ALL  SELECT 'a'   UNION ALL  SELECT 'a'   UNION ALL  SELECT 'b'   UNION ALL  SELECT 'c'   UNION ALL  SELECT 'c'   UNION ALL  SELECT 'd'   UNION ALL  SELECT 'e';  SELECT    v,    ROW\_NUMBER() OVER (ORDER BY v) row\_number,    RANK()       OVER (ORDER BY v) rank,    DENSE\_RANK() OVER (ORDER BY v) dense\_rank  FROM t  ORDER BY v;  The above will yield:  +---+------------+------+------------+  | V | ROW\_NUMBER | RANK | DENSE\_RANK |  +---+------------+------+------------+  | a |          1 |    1 |          1 |  | a |          2 |    1 |          1 |  | a |          3 |    1 |          1 |  | b |          4 |    4 |          2 |  | c |          5 |    5 |          3 |  | c |          6 |    5 |          3 |  | d |          7 |    7 |          4 |  | e |          8 |    8 |          5 |  +---+------------+------+------------+  In words   * ROW\_NUMBER() attributes a unique value to each row * RANK() attributes the same row number to the same value, leaving "holes" * DENSE\_RANK() attributes the same row number to the same value, leaving no "holes" |

MERGE FUNCTION drives the UpSert for SSIS ETLS. Use instead of the Slowly Changing Demension tool.

First table following the “merge” is the target table, table following “using” is the source table.   All insert/update/deletes actions apply to the target table.

merge  [dbo].[WorkDimAccountSource] AS Target

using  [dbo].[WorkDimAccountCustomSource] AS Source with (nolock)

on Target.[DWAccountKey] = Source.[DWAccountKey]

when matched then update

       set    Target.[Operator] = Source.[Operator]

       , Target. [CustomMembers] = Source.[CustomMembers]

       , Target. [CustomMemberOptions] = Source.[CustomMemberOptions]

       , Target. [CustomMemberEnglish] = Source.[CustomMemberEnglish]

when not matched by target

then insert

       (Target. [DWAccountKey]

       , Target. [DWAccountID]

       , Target. [DWParentAccountKey]

       , Target. [AccountName]

       , Target. [OrderValue]

       , Target. [Operator]

       , Target. [CustomMembers]

       , Target. [CustomMemberOptions]

       , Target. [ChartName]

       , Target. [ChartSection]

       , Target. [Account]

       , Target. [SubAccount]

       , Target. [AccountDesc]

       , Target. [MeasureType]

       , Target. [CustomMemberEnglish])

values

       (Source.[DWAccountKey]

       ,Source.[DWAccountID]

       ,Source.[DWParentAccountKey]

       ,Source.[AccountName]

       ,Source.[OrderValue]

       ,Source.[Operator]

       ,Source.[CustomMembers]

       ,Source.[CustomMemberOptions]

       ,Source.[ChartName]

       ,Source.[ChartSection]

       ,Source.[Account]

       ,Source.[SubAccount]

       ,Source.[AccountDesc]

       ,Source.[MeasureType]

       ,Source.[CustomMemberEnglish]) ;

**GROUP BY vs PARTITION BY**

select \* from callerlog

Output will be

Code:

CALLERID CALLER CALLEDTO CALLER\_DT OUTGOING\_SEC

3 456 123 02/21/2013 10

4 123 678 02/21/2013 20

1 123 456 02/21/2013 30

2 123 456 02/21/2013 10

5 678 456 02/21/2013 40

6 456 678 02/20/2013 20

7 789 555 02/21/2013 10

8 567 321 02/21/2013 100

We have used below in one of the article  
  
Query by using group by

Code:

select caller ,count(\*) as outgoing from callerlog group by caller

Output will be as below

Code:

CALLER OUTGOING

123 3

456 2

567 1

678 1

789 1

Group by has reduced the records from **8** to **5**.  
  
  
Query by using partition by

Code:

select caller ,count(\*) over (partition by caller ) as outgoing from callerlog

Output will be as below

Code:

CALLER OUTGOING

123 3

123 3

123 3

456 2

456 2

567 1

678 1

789 1

Partition by has not reduced the no. of records. Remains same.  
  
Below query we will not workout for group by