**DATEDIFF/DATEADD PRACTICE**

**UNIVERSAL PROPERTY**

dateadd(year, -pv.PolicyYearNum, pv.ExpirationDate) as InceptionDate,

datediff(day, dateadd(year, -pv.PolicyYearNum, pv.ExpirationDate), @AsOfDate) as DaysInForce,

**This returns prior Month End datetime**

select dateadd(day, -day(getdate()), getdate())

**This returns the last 24 hour period.**

c.CreatedDate between dateadd(day, -1, getdate()) and getdate()

**This returns the first of month,from 12 months ago**

select dateadd(month, -11, cast([dbo].[SpecialDate] (getdate(),'FOM') as date))

**This Function returns the** 'FOM', 'FOQ', 'FOY', 'EOM', 'EOQ', 'EOY'

create function [dbo].[SpecialDate] (

@Today date = null,

@DateRequested varchar(3) = null

) returns date

as

begin

declare @SpecialDate as date = null

set @Today = isnull(@Today, getdate())

if @DateRequested is not null and @DateRequested in ('FOM', 'FOQ', 'FOY', 'EOM', 'EOQ', 'EOY')

begin

set @SpecialDate = dateadd(day, -datepart(day, @Today) + 1, @Today) -- First Of Month

set @SpecialDate = case @DateRequested

when 'FOY' then dateadd(month, -datepart(month, @SpecialDate) + 1, @SpecialDate)

when 'FOQ' then dateadd(month, -(datepart(month, @SpecialDate) - 1) % 3, @SpecialDate)

when 'EOM' then dateadd(day, -1, dateadd(month, 1, @SpecialDate))

when 'EOQ' then dateadd(day, -1, dateadd(month, -(datepart(month, @SpecialDate) - 1) % 3 + 3, @SpecialDate))

when 'EOY' then dateadd(day, -1, dateadd(month, 13 - datepart(month, @SpecialDate), @SpecialDate))

else @SpecialDate end

end

else

set @SpecialDate = @Today

return @SpecialDate

end

GO

**This Function returns the first day of the quarter.**

create function [dbo].[GetFirstDayOfQuarter]

(

@date date

)

returns date

as begin

return

cast(case when month(@Date) in (1,2,3) then '1/1/'

else case when month(@Date) in (4,5,6) then '4/1/'

else case when month(@Date) in (7,8,9) then '7/1/'

else case when month(@Date) in (10,11,12) then '10/1/'

end end end end + cast(year(@date) as nchar(4)) as date)

end

GO

**This Code uses DateAdd to go back 30 days for an Aging report**

ALTER procedure [dbo].[reportPolicy30DayAging] 30

@Age int

as

begin

;with Balance

as

(

select at.PolicyID,

sum(at.Amount) as Amount

from AccountingTransaction as at

group by at.PolicyID

having sum(at.Amount) > 0

)

select distinct(p.PolicyID),

p.PolicyNumber,

dbo.GetPolicyInsureds(p.policyID) as Applicant,

pv.EffectiveDate,

a.AgencyCode as Agency,

b.Amount as Balance

from Policy as p

join Balance as b

on b.PolicyID = p.PolicyID

join Agency as a

on p.AgencyID = a.AgencyID

join PolicyVersion as pv

on pv.PolicyVersionID=P.PolicyVersionID

left join AccountingTransaction as at

on at.PolicyID = p.PolicyID

and at.AccountingTransactionTypeID IN (6, 15)

where pv.EffectiveDate < dateadd(day, -1 \* (@Age + 1), getdate())

and at.PolicyID is null

and p.Inforce = 1

end

DECLARE @startdate datetime2 = '2017-03-01 12:10:09.3312722';

DECLARE @enddate datetime2 = '2017-04-07 12:10:09.3312722';

SELECT DATEDIFF(day, @startdate, @enddate);

### Incrementing datepart by an interval of 1

drop table #PolicyDate

CREATE TABLE #PolicyDate (

ID INTEGER,

PolicyDate datetime2);

INSERT INTO #PolicyDate VALUES (1, '2017-04-07 13:10:10.1111111');

SELECT 'year', DATEADD(year,1,SYSDATETIME()) FROM #PolicyDate

UNION ALL

SELECT 'quarter',DATEADD(quarter,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'month',DATEADD(month,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'dayofyear',DATEADD(dayofyear,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'day',DATEADD(day,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'week',DATEADD(week,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'weekday',DATEADD(weekday,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'hour',DATEADD(hour,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'minute',DATEADD(minute,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'second',DATEADD(second,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'millisecond',DATEADD(millisecond,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'microsecond',DATEADD(microsecond,1,PolicyDate) FROM #PolicyDate

UNION ALL

SELECT 'nanosecond',DATEADD(nanosecond,1,PolicyDate) FROM #PolicyDate ;

**DATEADD with Window Functions**

**The following example uses a ranking function as arguments for number.**

SELECT c.FirstName, c.LastName

,DATEADD(day,ROW\_NUMBER() OVER (ORDER BY

c.LastName),SYSDATETIME()) AS 'Row Number'

FROM [dbo].[Contact] AS c

WHERE c.CreatedDate > '20170101'

and c.LastName is not null and c.LastName !='';

Get the Current System Date and Time

SELECT SYSDATETIME()

,SYSDATETIMEOFFSET()

,SYSUTCDATETIME()

,CURRENT\_TIMESTAMP

,GETDATE()

,GETUTCDATE();

### Get the Current System Date

SELECT CONVERT (date, SYSDATETIME())

,CONVERT (date, SYSDATETIMEOFFSET())

,CONVERT (date, SYSUTCDATETIME())

,CONVERT (date, CURRENT\_TIMESTAMP)

,CONVERT (date, GETDATE())

,CONVERT (date, GETUTCDATE());

### Get the Current System Time

SELECT CONVERT (time, SYSDATETIME())

,CONVERT (time, SYSDATETIMEOFFSET())

,CONVERT (time, SYSUTCDATETIME())

,CONVERT (time, CURRENT\_TIMESTAMP)

,CONVERT (time, GETDATE())

,CONVERT (time, GETUTCDATE());

**This code captures the entire day of the @AsofDate (Used to capture real time)**

if @AsOfDate is null or @AsOfDate >= cast(getdate() as date)

SET @AsofDate = DATEADD(SS,-1,DATEADD(DD,1,@AsofDate))

--set @AsofDate = cast(getdate() as date)

else

set @AsOfDate = cast(@AsOfDate as date)

if @AsOfDate is null or @AsOfDate >= cast(getdate() as date)

set @AsofDate = dateadd(day, -1, cast(getdate() as date))

else

set @AsOfDate = cast(@AsOfDate as date)

select

@FirstOfYear = dateadd(day, -datepart(dayofyear, @AsOfDate) + 1, @AsOfDate),

@FirstOfMonth = dateadd(day, -datepart(day, @AsOfDate) + 1, @AsOfDate),

@FirstOfWeek = dateadd(day, -datepart(dw, @AsOfDate) + 1, @AsOfDate),

@AsOfDate = dateadd(millisecond, -3, dateadd(day, 1, @AsOfDate))

**This code filters to prior year from beginning of prior year to todays day last year**

**i.e. 1st 9 months of the year last year**

declare @ThisYear datetime = getdate()

declare @LastYear datetime = dateadd(year, -1, @ThisYear)

where p.CreatedDate

between dateadd(day, -datepart(dayofyear, @LastYear) + 1, cast(@LastYear as date))

and @LastYear

declare @StartingDate as datetime = dateadd(month, -13, dateadd(day, -1, cast(getdate() as date))),

@EndingDate as datetime = dateadd(millisecond, -3, cast(cast(getdate() as date) as datetime))

//This code concats the year with month and formats month for sorting (2016 – 02)

CONCAT(YEAR(pa.PolicyCreationDate) , '-' , FORMAT(MONTH(pa.PolicyCreationDate),'00')) as 'Year/Month'

SELECT @ThisDate = DATEADD(dd, DATEDIFF(dd,0,GETDATE()),0);

SELECT @Yesterday = DATEADD(d, -1, @ThisDate);

SELECT @BeginCurrentMonth = DATEADD(m, DATEDIFF(m,0,@ThisDate),0);

SELECT @EndCurrentMonth = DATEADD(d, -1, DATEADD(m, 1, @BeginCurrentMonth));

SELECT @BeginPriorMonth = DATEADD(m, -1, @BeginCurrentMonth);

SELECT @EndPriorMonth = DATEADD(d, -1, @BeginCurrentMonth);

SELECT @BeginCurrentQuarter = DATEADD(q, DATEDIFF(q,0,@ThisDate),0);

SELECT @EndCurrentQuarter = DATEADD(d, -1, DATEADD(q, 1, @BeginCurrentQuarter));

SELECT @BeginPriorQuarter = DATEADD(q, -1, @BeginCurrentQuarter);

SELECT @EndPriorQuarter = DATEADD(d,-1,@BeginCurrentQuarter);

SELECT @BeginCurrentYear = DATEADD(m, 1-month(@BeginCurrentMonth), @BeginCurrentMonth);

SELECT @EndCurrentYear = DATEADD(d, -1, DATEADD(year, 1, @BeginCurrentYear));

SELECT @BeginPriorYear = DATEADD(year, -1, @BeginCurrentYear);

SELECT @EndPriorYear = DATEADD(d, -1, @BeginCurrentYear);

@StartingDate as date = dateadd(day, -datepart(day, @EndingDate) + 1, @EndingDate) -- First of Month

dateadd(month, -datepart(month, @StartingDate) + 1, @StartingDate) as StartingDate, -- First of Year

dateadd(day, -1, dateadd(month, 1, @StartingDate)) as EndingDate -- End of Month

cast(dateadd(day, -(datepart(weekday, pa.AppSubmitDate) - 1), pa.AppSubmitDate) as date) as WkStart,

DATEADD(dd, 7-(DATEPART(dw, pa.AppSubmitDate)), CAST(pa.AppSubmitDate AS date)) AS WkEnd

//Start of 3 years back

**cast(datefromparts(year(getdate())-3, 1, 1) as date) as StartingLossDate,**

//End of Last Year

**cast(datefromparts(year(getdate())-1, 12, 31) as date) as EndingLossDate**

/\*This code subtracts the day of the month then adds a day to get the first day of the month. 2016/10/15 -15 +1 = 2016/10/01 \*/

SELECT cast(dateadd(day, -day(pa.PolicyCreationDate) + 1, pa.PolicyCreationDate) as date) as FirstDayOfMonth

/\*This code subtracts the day of the month from GETDATE() to get the last day of the month. 2016/10/15 -15 = 2016/09/30 \*/

cast(dateadd(day, -day(getdate()), getdate()) as date) as LastDayOfMonth

select dateadd(day, -7, GETDATE()) as [Date],

datename(weekday, dateadd(day, -7, GETDATE())) as [DayOfWeek]

select dateadd(day, 1, GETDATE()) as [Date],

datename(weekday, dateadd(day, 1, GETDATE())) as [DayOfWeek]

//This code goes back a year

SELECT dateadd(month, -11, dateadd(day, -day(@AsOfDate) + 1, @AsOfDate))

**CODE FOR WOW REPORTS**

ALTER procedure [dbo].[reportWeekOverComparisonByMarketingRep]

@ThisWeek as date = null,

@StateIDs as varchar(4000) = null,

@CompanyIDs as varchar(4000) = null

as

begin

set nocount on

declare @LastWeek as date = null,

@LastYear as date = null

if @ThisWeek is null

begin

begin with

Today as (

select cast(getdate() as date) as Today),

SevenDays as (

select dateadd(day, -7, t.Today) as [Date],

datename(weekday, dateadd(day, -7, t.Today)) as [DayOfWeek]

from Today as t

union all

select dateadd(day, 1, sd.[Date]) as [Date],

datename(weekday, dateadd(day, 1, sd.[Date])) as [DayOfWeek]

from SevenDays as sd, Today as t

where dateadd(day, 1, sd.[Date]) < t.Today)

select @LastWeek = dateadd(day, -7, max([Date])),

@ThisWeek = max([Date])

from SevenDays

where [DayOfWeek] = 'Friday'

end

end

else

begin

set @LastWeek = dateadd(day, -7, @ThisWeek)

end

set @LastYear = dateadd(year, -1, @ThisWeek)

**CODE FOR YOY REPORTS**

ALTER procedure [dbo].[reportYearOverClaimsTrendByPerils]

@StartingDate as datetime = null,

@EndingDate as datetime = null,

@AsOfDate as datetime = null,

@CompanyID as integer = null,

@LookBackPeriod as integer = null,

@ReportTypeID as integer = null

as

begin

set nocount on

if @EndingDate is null set @EndingDate = cast(dateadd(day, -day(getdate()), getdate()) as date)

if @StartingDate is null or @StartingDate > @EndingDate set @StartingDate = dbo.SpecialDate(@EndingDate, 'FOY')

if @AsOfDate is null set @AsOfDate = cast(dateadd(day, -1, getdate()) as date)

if @CompanyID not in (1, 2) set @CompanyID = null

if @ReportTypeID is null set @ReportTypeID = 1 -- 'Reported during the Period'

declare @OccurredOnly as bit = iif(@ReportTypeID = 2, 1, 0) -- 'Reported and Occurring during the Period'

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

begin with

DateRange as (

select 1 as DateRangeID,@StartingDate as StartingDate,

dateadd(millisecond, -3, dateadd(day, 1, @EndingDate)) as EndingDate,

dateadd(millisecond, -3, dateadd(day, 1, @AsOfDate)) as AsOfDate

union all

select dr.DateRangeID + 1 as DateRangeID, dateadd(year, -1, dr.StartingDate) as StartingDate,

dateadd(year, -1, dr.EndingDate) as EndingDate,

dateadd(year, -1, dr.AsOfDate) as AsOfDate

from DateRange as dr

where dr.DateRangeID <= @LookBackPeriod

),

**CODE FOR YTD/MTD/WTD REPORTS**

ALTER procedure [dbo].[reportGetClaimsByAdjusterYTD]

@AsOfDate as datetime = null

as

begin

set nocount on

declare @FirstOfYear as datetime,

@FirstOfMonth as datetime,

@FirstOfWeek as datetime

if @AsOfDate is null or @AsOfDate >= cast(getdate() as date)

set @AsofDate = dateadd(day, -1, cast(getdate() as date))

else

set @AsOfDate = cast(@AsOfDate as date)

select @FirstOfYear = dateadd(day, -datepart(dayofyear, @AsOfDate) + 1, @AsOfDate),

@FirstOfMonth = dateadd(day, -datepart(day, @AsOfDate) + 1, @AsOfDate),

@FirstOfWeek = dateadd(day, -datepart(dw, @AsOfDate) + 1, @AsOfDate),

@AsOfDate = dateadd(millisecond, -3, dateadd(day, 1, @AsOfDate))

**NEW ABOVE**

select day(getdate())

DATEADD(dd, 7-(DATEPART(dw, pa.AppSubmitDate)), CAST(pa.AppSubmitDate AS date)) AS WkEnd

cast(dateadd(day, -(datepart(weekday, pa.AppSubmitDate) - 1), pa.AppSubmitDate) as date) as WkStart

eomonth(dateadd(month, 0, pa.AppSubmitDate)) AS MonthEnd

declare @end date = dateadd(day, -(datepart(weekday, getdate()) - 1), getdate())

declare @start date = dateadd(year, -3, @end)

select

dateadd(week, -v.Number - 1, @end) as First\_Day

,dateadd(day, -1, dateadd(week, -v.Number, @end)) as Last\_Day

insert @dates values

('Total', null, null)

,('Last30Days', dateadd(month, -1, @now), @now)

,('Prior30Days', dateadd(month, -2, @now), dateadd(month, -1, @now))

@AsOfDate as datetime = null

as

begin

set nocount on

declare @FirstOfYear as datetime,

@FirstOfMonth as datetime,

@FirstOfWeek as datetime

if @AsOfDate is null or @AsOfDate >= cast(getdate() as date)

set @AsofDate = dateadd(day, -1, cast(getdate() as date))

else

set @AsOfDate = cast(@AsOfDate as date)

select

@FirstOfYear = dateadd(day, -datepart(dayofyear, @AsOfDate) + 1, @AsOfDate),

@FirstOfMonth = dateadd(day, -datepart(day, @AsOfDate) + 1, @AsOfDate),

@FirstOfWeek = dateadd(day, -datepart(dw, @AsOfDate) + 1, @AsOfDate),

@AsOfDate = dateadd(millisecond, -3, dateadd(day, 1, @AsOfDate))

if @EndingDate is null set @EndingDate = cast(dateadd(day, -day(getdate()), getdate()) as date)

if @StartingDate is null or @StartingDate > @EndingDate set @StartingDate = dbo.SpecialDate(@EndingDate, 'FOY')

if @AsOfDate is null set @AsOfDate = cast(dateadd(day, -1, getdate()) as date)

set @AsOfDate = isnull(@AsOfDate, dateadd(day, -day(getdate()), cast(getdate() as date)))

set @AsOfDate = isnull(@AsOfDate, **eomonth**(dateadd(month, -1, getdate())))

set @CompareDate = isnull(@CompareDate, eomonth(dateadd(month, -2, getdate())))

select dateadd(day, -1, cast(getdate() as date)) as AsOfDate,

dateadd(day, -2, cast(getdate() as date)) as CompareDate,

cast(datefromparts(year(getdate())-3, 1, 1) as date) as StartingLossDate,

cast(datefromparts(year(getdate())-1, 12, 31) as date) as EndingLossDate

declare @dates table

(

Name varchar(128)

,StartDate date

,EndDate date

)

insert @dates values

(

'This Month'

,@reportDate

,@reportEnd

)

,(

'One Month Prior'

,dateadd(month,-1,@reportDate)

,dateadd(day, -1, @reportDate)

)

,(

'Two Months Prior'

,dateadd(month,-2,@reportDate)

,dateadd(day, -1, dateadd(month,-1, @reportDate))

)

,(

'YTD'

,dateadd(month,-(datepart(month, @reportdate) - 1), @reportDate)

,@reportEnd

)

,(

'Last YTD'

,dateadd(year,-1,dateadd(month,-(datepart(month, @reportdate) - 1), @reportDate))

,dateadd(year,-1, @reportEnd)

)

,(

'Last Year'

,dateadd(year,-1,dateadd(month,-(datepart(month, @reportdate) - 1), @reportDate))

,dateadd(day,-1,dateadd(month,-(datepart(month, @reportdate) - 1), @reportDate))

)

DATEDIFF(YY,g8.BirthDate,m.FirstDecisionDate) as GuarantorAge

TotalTime = (DATEDIFF(MINUTE, BeginDate, CompleteDate) /60.00)

Declare @Day int = (select day(getdate()))

Declare @Month int = (select month(getdate()))

Declare @Year int = (select year(getdate()))

IF @ReportPeriod = 1

BEGIN

SET @StartDate = CONVERT(CHAR(10),dateadd(dd,-30,getdate()),101)

SET @EndDate = getdate()

END

IF @ReportPeriod = 3

BEGIN

SET @StartDate = CONVERT(CHAR(10),dateadd(dd,-90,getdate()),101)

SET @EndDate = getdate()

END

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d5f3dc52-8272-48f8-87c2-d99d3580eaf0');" \o "Copy to clipboard.)

**DATEDIFF ( datepart , startdate , enddate )**

**DATEADD (datepart , number , date )**

**DATENAME ( datepart , date )**

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

NOTE: '1900-01-01' is represented as a 0. It is SQL Servers representation of the beginning of time.

* --get todays date

select Getdate()

* --Get the number of days from the beginning of time

select Datediff(dd, 0, Getdate())

* --Get todays date and truncate the time

select Dateadd(dd, Datediff(dd, 0, Getdate()), '1900-01-01')

select Dateadd(dd, Datediff(dd, '1900-01-01', Getdate()),'1900-01-01')

* --Get yesterdays date and truncate the time

select Dateadd(dd, Datediff(dd, 0, Getdate()),-1)

select Dateadd(dd, Datediff(dd, 0, Getdate()),'1899-12-31')

select Dateadd(dd, -1,Dateadd(dd, Datediff(dd, 0, Getdate()), 0))

* ,DATEDIFF(DD, (SELECT MAX(ChangeDate) FROM Vision.dbo.tWorkFlowHistory WHERE RecNum = a.ApplicationNum), GETDATE()) as DaysInStatus
* DATEADD(dd, -(DATEPART(dw, WeddingDate)-1), WeddingDate) [WeekStart]
* DATEADD(dd, 7-(DATEPART(dw, WeddingDate)), WeddingDate) [WeekEnd]

/\*

function that determines the last day of the month (LDOM) for a given date parameter.

\*/

CREATE FUNCTION fx\_LDOM

( @Date varchar(20) )

RETURNS datetime

AS

BEGIN

--ensure valid date

IF ISDATE(@Date) = 1

BEGIN

--determine first day of month

SET @Date = DATEADD(day,-DAY(@Date)+1,@Date)

--determine last day of month

SET @Date = DATEADD(day,-1,DATEADD(month,1,@Date))

END

ELSE

SET @Date = '1/1/80'

RETURN @Date

END

/\*

The function's parameter (@Date) is defined as varchar(20), so error-checking code can be implemented. The ISDATE function makes sure the supplied value is a valid date. When an invalid date value is supplied, the function returns '1/1/80' to the calling statement. If you do not use this type of error-checking, the call to the function will fail when an invalid date is supplied.

When a valid date value is supplied, the DATEADD function is used to:

1. Determine the first day of the month, and

2. Determine the last day of the month.

If you have never used DATEADD before this may seem a little confusing, but a quick explanation should eliminate any that might exist. You use DATEADD to add or substract a date/time unit from a given date. The first parameter (in this case 'day') indicates the portion of the date that should be incremented. You can also specify year, quarter, month...millisecond. The second parameter is the number of units to add or substract. When subtracting, you simply make the value negative as shown in the example. The third parameter is the date value on which the calculation is performed.

The first day of the month is determined by calculating the number of elapsed days in the supplied parameter with the DAY function, adding 1 and then substracting it from the parameter. For an @Date value of 1/15/01, it simply subtracts 14 (15-1) days to get 1/1/01.

The last day of the month is determined by adding 1 month to the current month value and subtracting one day. So, 1/1/01 plus 1 month is equal to 2/1/01 and when you substract one day you get: 1/31/01.

The following shows how fx\_LDOM is used in a SELECT statement to calculate the number of days remaining in a month.

CREATE TABLE fx\_Testing (DateValue datetime)

go

INSERT fx\_Testing VALUES ('1/1/01')

INSERT fx\_Testing VALUES ('2/15/01')

INSERT fx\_Testing VALUES ('2/15/02')

INSERT fx\_Testing VALUES ('2/15/03')

INSERT fx\_Testing VALUES ('2/15/04')

SELECT DateValue,

dbo.fx\_LDOM(DateValue) AS LDOM,

DATEDIFF(day,DateValue,dbo.fx\_LDOM(DateValue)) AS DaysLeftInMonth

FROM fx\_Testing

DateValue LDOM DaysLeftInMonth

------------------------ ----------------------- ---------------

2001-01-01 00:00:00.000 2001-01-31 00:00:00.000 30

2001-02-15 00:00:00.000 2001-02-28 00:00:00.000 13

2002-02-15 00:00:00.000 2002-02-28 00:00:00.000 13

2003-02-15 00:00:00.000 2003-02-28 00:00:00.000 13

2004-02-15 00:00:00.000 2004-02-29 00:00:00.000 14

The DATEDIFF function is used to determine the difference between two dates. In this case, the number of days between the value in the DateValue column and the last day of the month calculated by fx\_LDOM.

\*/

-Following script demonstrates the script to find last day of previous, current and next month.

-Last Day of Previous Month

**SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE()),0))**

LastDay\_PreviousMonth

----Last Day of Current Month

SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE())+1,0))

LastDay\_CurrentMonth

----Last Day of Next Month

SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE())+2,0))

LastDay\_NextMonth

ResultSet:

LastDay\_PreviousMonth

———————–

2007-07-31 23:59:59.000

LastDay\_CurrentMonth

———————–

2007-08-31 23:59:59.000

LastDay\_NextMonth

———————–

2007-09-30 23:59:59.000

If you want to find last day of month of any day specified use following script.

--Last Day of Any Month and Year

DECLARE @dtDate DATETIME

SET @dtDate = '8/18/2007'

SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,@dtDate)+1,0))

LastDay\_AnyMonth

ResultSet:

LastDay\_AnyMonth

———————–

2007-08-31 23:59:59.000

**SQL Server Rolling 12 Months:**

SELECT sum(TRANSACTION\_AMOUNT)

FROM TRANSACTIONS

WHERE datediff(month, TRANSACTION\_DATE, getdate()) <= 12

AND TRANSACTION\_DATE <= getdate()

SQL Server Month to Date:

SELECT sum(TRANSACTION\_AMOUNT)

FROM TRANSACTIONS

WHERE datediff(month, TRANSACTION\_DATE, getdate()) = 0

AND TRANSACTION\_DATE <= getdate()

SQL Server Year to Date:

SELECT sum(TRANSACTION\_AMOUNT)

FROM TRANSACTIONS

WHERE datediff(year, TRANSACTION\_DATE, getdate()) = 0

AND TRANSACTION\_DATE <= getdate()

SQL Server Prior Year:

SELECT sum(TRANSACTION\_AMOUNT)

FROM TRANSACTIONS

WHERE datediff(year, TRANSACTION\_DATE, getdate()) = 1

--first day of this month, and six months before that.

select DATEADD(mm, DATEDIFF(mm,0,getdate()), 0),

DATEADD(mm,-6,DATEADD(mm, DATEDIFF(mm,0,getdate()), 0))

--last day of month 6 months ago, last day of previous month

select

DATEADD(dd,-1,DATEADD(mm,-5,DATEADD(mm, DATEDIFF(mm,0,getdate()), 0))),

DATEADD(dd,-1,DATEADD(mm, DATEDIFF(mm,0,getdate()), 0))

--Usage #1: Calculate Age

DECLARE @BirthDate DATETIME = '1932/06/12'

SELECT DATEDIFF(YEAR, @BirthDate, GETDATE()) -

CASE WHEN MONTH(@BirthDate) < MONTH(GETDATE()) OR

(MONTH(@BirthDate) = MONTH(GETDATE()) AND DAY(@BirthDate) <= DAY(GETDATE()))

THEN 0 ELSE 1 END AS [Age]

--Usage #2: Get Date Portion of a DATETIME Value

SELECT DATEADD(DD, DATEDIFF(DD, 0, GETDATE()), 0) AS [Date Part Only]

--Usage #3 : Get First Day of the Month, Quarter and Year

SELECT DATEADD(MM, DATEDIFF(MM, 0, GETDATE()), 0) AS [First Day of the Month]

SELECT DATEADD(Q, DATEDIFF(Q, 0, GETDATE()), 0) AS [First Day of the Quarter]

SELECT DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()), 0) AS [First Day of the Year]

--Usage #4 : Get Last Day of the Month, Quarter and Year

SELECT DATEADD(MM, DATEDIFF(MM, 0, GETDATE()) + 1, 0) - 1 AS [Last Day of the Month]

SELECT DATEADD(Q, DATEDIFF(Q, 0, GETDATE()) + 1, 0) - 1 AS [Last Day of the Quarter]

SELECT DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) + 1, 0) - 1 AS [Last Day of the Year]

--Usage #5 : Get First Day of the Following Month, Quarter and Year

SELECT DATEADD(MM, DATEDIFF(MM, 0, GETDATE()) + 1, 0) AS [First Day of Next Month]

SELECT DATEADD(Q, DATEDIFF(Q, 0, GETDATE()) + 1, 0) AS [First Day of Next Quarter]

SELECT DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) + 1, 0) AS [First Day of Next Year]

--Usage #6 : Get Last Day of the Following Month, Quarter and Year

SELECT DATEADD(MM, DATEDIFF(MM, 0, GETDATE()) + 2, 0) - 1 AS [Last Day of Next Month]

SELECT DATEADD(Q, DATEDIFF(Q, 0, GETDATE()) + 2, 0) - 1 AS [Last Day of Next Quarter]

SELECT DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) + 2, 0) - 1 AS [Last Day of Next Year]

--Usage #7 : Get First Day of the Previous Month, Quarter and Year

SELECT DATEADD(MM, DATEDIFF(MM, 0, GETDATE()) - 1, 0) AS [First Day Of Previous Month]

SELECT DATEADD(Q, DATEDIFF(Q, 0, GETDATE()) - 1, 0) AS [First Day Of Previous Quarter]

SELECT DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) - 1, 0) AS [First Day Of Previous Year]

--Usage #8 : Get Last Day of the Previous Month, Quarter and Year

SELECT DATEADD(MM, DATEDIFF(MM, 0, GETDATE()), 0) - 1 AS [Last Day of Previous Month]

SELECT DATEADD(Q, DATEDIFF(Q, 0, GETDATE()), 0) - 1 AS [Last Day of Previous Quarter]

SELECT DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()), 0) - 1 AS [Last Day of Previous Year]

**This code allows us to capture a rolling 12 months of data and converts average minutes to avg hours.**

CREATE PROCEDURE [dbo].[sRptLeadContactTimeRolling12mth] (

@StartDate as DateTime,

@EndDate as DateTime

) AS

BEGIN

SET @StartDate = DATEADD(mm,-11,@EndDate)

SET @StartDate = DATEADD(dd,-(Day(@StartDate)-1),@StartDate)

SET @EndDate = DATEADD(SS,-1,DATEADD(DD,1,@EndDate))

SELECT

o.OpportunityID,

o.TransactionID,

[dbo].[LookupLeadTypeII](o.TransactionID) as Leadtype,

Year(o.CreatedDate)\*100+Month(o.CreatedDate) as Fiscal,

Year(o.CreatedDate) as [Year],

Month(o.CreatedDate) as [Month],

o.CreatedDate as OpportunityCreateDate,

a.ActivityID,

datediff(mi,o.CreatedDate,coalesce(a4.DateCompleted,a3.DateCompleted)) as FirstContactAttemptMin,

cast(datediff(mi,o.CreatedDate,coalesce(a4.DateCompleted,a3.DateCompleted))as decimal(14,2)) /cast(60 as decimal(14,2)) as FirstContactAttempt,

o.AssignedToUserNum

INTO #YTDLEADS

FROM Leadwarehouse.dbo.Opportunity o with(nolock)

**This code allows us to capture the beginning of the StartDate day (12am)**

SELECT EndDate = GETDATE(), StartDate = DATEADD(dd, - 4, CAST(CONVERT(CHAR(10),GETDATE(),101) as smalldatetime))

**This code allows us to capture 12 months of data from GETDATE() and convets average minutes to avg hours and puts the data into another Table for a job for faster performance.**

DECLARE @BeginFiscal int = year(dateadd(mm,-11,getdate()))\*100 + month(dateadd(mm,-11,getdate()))

--PRINT(@BeginFiscal)

INSERT INTO dbo.tOpportunityActivityData

SELECT o.OpportunityID,

o.TransactionID,

cast('' as nvarchar(50)) as LeadType,

Year(o.CreatedDate)\*100+Month(o.CreatedDate) as Fiscal,

Year(o.CreatedDate) as [Year],

Month(o.CreatedDate) as [Month],

o.CreatedDate as OpportunityCreateDate,

coalesce(a4.DateCompleted,a3.DateCompleted) as DateCompleted,

datediff(mi,o.CreatedDate,coalesce(a4.DateCompleted,a3.DateCompleted)) as FirstContactAttemptMin,

a3.ActivityID as ActivityIDa3,

a4.ActivityID as ActivityIDa4,

cast(datediff(mi,o.CreatedDate,coalesce(a4.DateCompleted,a3.DateCompleted))as decimal(14,2)) /cast(60 as decimal(14,2)) as FirstContactAttempt,

o.AssignedToUserNum,

getdate() as RunTime

--INTO dbo.tOpportunityActivityData

FROM Leadwarehouse.dbo.Opportunity o with(nolock)

--FIRST ACT with a completed date and created after opp

Left Outer Join Leadwarehouse.dbo.Activity a3 with(nolock) on a3.activityID =(SELECT TOP 1 activityID FROM Activity with(nolock)

WHERE ((PID = o.PID and POwner = o.Powner) or TransactionID = o.TransactionID)

and datecompleted is not null and YEAR(isnull(datecompleted,2000)) >= 1991

and CreatedDate >= o.CreatedDate

--is not the first act createdby sm or ra

and ActivityID != ISNULL((SELECT TOP 1 ActivityID FROM Activity with(nolock)

WHERE TransactionID = o.TransactionID and CreatorUserNum IN (834,765)

ORDER BY CreatedDate, ActivityID),0)

ORDER BY DateCompleted ASC)

Left Outer Join Leadwarehouse.dbo.Activity a4 with(nolock) on a4.activityID = o.activityID

and a4.CreatorUserNum NOT IN (834,765)

and YEAR(isnull(a4.datecompleted,2000)) >= 1991

WHERE year(o.CreatedDate)\*100 + month(o.CreatedDate) >= @BeginFiscal

ORDER BY o.OpportunityID

UPDATE dbo.tOpportunityActivityData SET LeadType = [dbo].[LookupLeadTypeII](TransactionID) WHERE RunTime =(select max(RunTime) from dbo.tOpportunityActivityData)

DELETE FROM dbo.tOpportunityActivityData WHERE RunTime !=(select max(RunTime) from dbo.tOpportunityActivityData)

SELECT \* FROM dbo.tOpportunityActivityData

**This code used DATEDIFF and COALESCE and converts minutes into hours.**

datediff(mi,o.CreatedDate,coalesce(a4.DateCompleted,a3.DateCompleted)) as FirstContactAttemptMin,

cast(datediff(mi,o.CreatedDate,coalesce(a4.DateCompleted,a3.DateCompleted))as decimal(14,2)) /cast(60 as decimal(14,2)) as FirstContactAttemptHours,

**This code allows us to capture the entire day for the end date when running a report.**

SET @EndDate = dateadd(ss,-1,dateadd(dd,1,@EndDate))

**This code returns a list of the past 6 years to use for an SSRS drop down for Years**

SELECT YEAR(GETDATE()) as [Year]

UNION

SELECT YEAR(GETDATE()) - 1 as [Year]

UNION

SELECT YEAR(GETDATE()) - 2 as [Year]

UNION

SELECT YEAR(GETDATE()) - 3 as [Year]

UNION

SELECT YEAR(GETDATE()) - 4 as [Year]

UNION

SELECT YEAR(GETDATE()) - 5 as [Year]

ORDER BY [Year] DESC

**This code returns a rolling 4 months of data**

Declare @tFiscals table(Fiscal int)

Declare @LoopNum as int = 0

WHILE (@LoopNum < 4)

BEGIN

INSERT INTO @tFiscals

SELECT (year(dateadd(m,(@LoopNum \*-1),getdate())) \*100) + month(dateadd(m,(@LoopNum \*-1),getdate()))

SET @LoopNum = @LoopNum +1

END

THIS CODDE RETREIVES **MONTH TO DATE** DATA

=dateadd("d",-datepart("d",today())+1,today()) SSRS

=dateadd("d",-datepart("d", getdate())+1,getdate()) TSQL

SET @StartDate = '04/01/2015'

SET @EndDate = '04/30/2015 23:59:59'