**Hallux Date Increment**

The Hallux Date Increment project will serve two purposes. The practical purpose will be to increment all the dates forward so as to make the sample data timelier. As of this writing the latest order date is 12/31/2014. If all dates were incremented by 8 years, the latest order information will be current at 12/31/2022.

The second purpose is to provide a learning exercise in Python for students. One potential exercise would be the following:

Write a program that will take a band’s current tour schedule and duplicate it for the next year.

This task is made easier due to no date fields being used as primary or foreign keys in the database. Challenges to the task include leap years and maintaining the data’s floating holiday and “day of week” integrity for live performances.

Leap years occur every 4 years with the addition of February 29th. If dates are incremented by a single year, then all 2/29 data would be converted to 2/28. In fact, people born on 2/29 are asked to choose either 2/28 or 3/1 to celebrate their birthday. In term of the Hallux data, the following leap year dates exist: 2/29/2000, 2/29/2004, 2/29/2008, and 2/29/2012. Luckily, the latest order information in the database is 12/31/2014. If the dates are incremented by 8 years, then the leap year information will be maintained, and the data will be current thru 12/31/2022.

Is a year a common year or a leap year?

**if** (*year* is not [divisible](https://en.wikipedia.org/wiki/Divisor) by 4) **then** (it is a common year)  
**else if** (*year* is not divisible by 100) **then** (it is a leap year)  
**else if** (*year* is not divisible by 400) **then** (it is a common year)  
**else** (it is a leap year)

Example leap years include 2000, 2004, 2008, 2012, 2016, 2020, 2024, 2028, 2032, 2036, 2040, 2044, and 2048.

Another issue of incrementing dates is any requirement to maintain the same day of the week. For example, performance dates are most likely centered on weekends. For example, if a concert date on a Saturday is incremented by one year, it would now be on a Sunday. With exception of concerts on fixed holidays like the fourth of July or New Year’s Eve, this process should maintain the integrity of weekend dates for live performance data.

Holidays can be on a Fixed date or a Floating date:

January 1 Fixed New Year's Day

January 15–21 Floating Monday Birthday of Martin Luther King, Jr.

February 15–21 Floating Monday Washington's Birthday

May 25–31 Floating Monday Memorial Day

June 19 Fixed Juneteenth National Independence Day

July 4 Fixed Independence Day

September 1–7 Floating Monday Labor Day

October 8–14 Floating Monday Columbus Day

November 11 Fixed Veterans Day

November 22–28 Floating Thursday Thanksgiving Day

December 24 Fixed Christmas Eve

December 25 Fixed Christmas Day

December 31 Fixed New Year’s Eve

In order to reduce the number of unique datetime, all datetime columns are converted to a date datatype thus removing the time element. When the original datetime field is updated to the new date, the original time is appended to the incremented date field.

The date increment process consists of three steps. First, all of the unique dates in the current data are summarized. Then each of those dates are mapped to the new incremented data. Lastly, the existing dates are updated to the new date. Following is a list of all the date fields in the Hallux database.

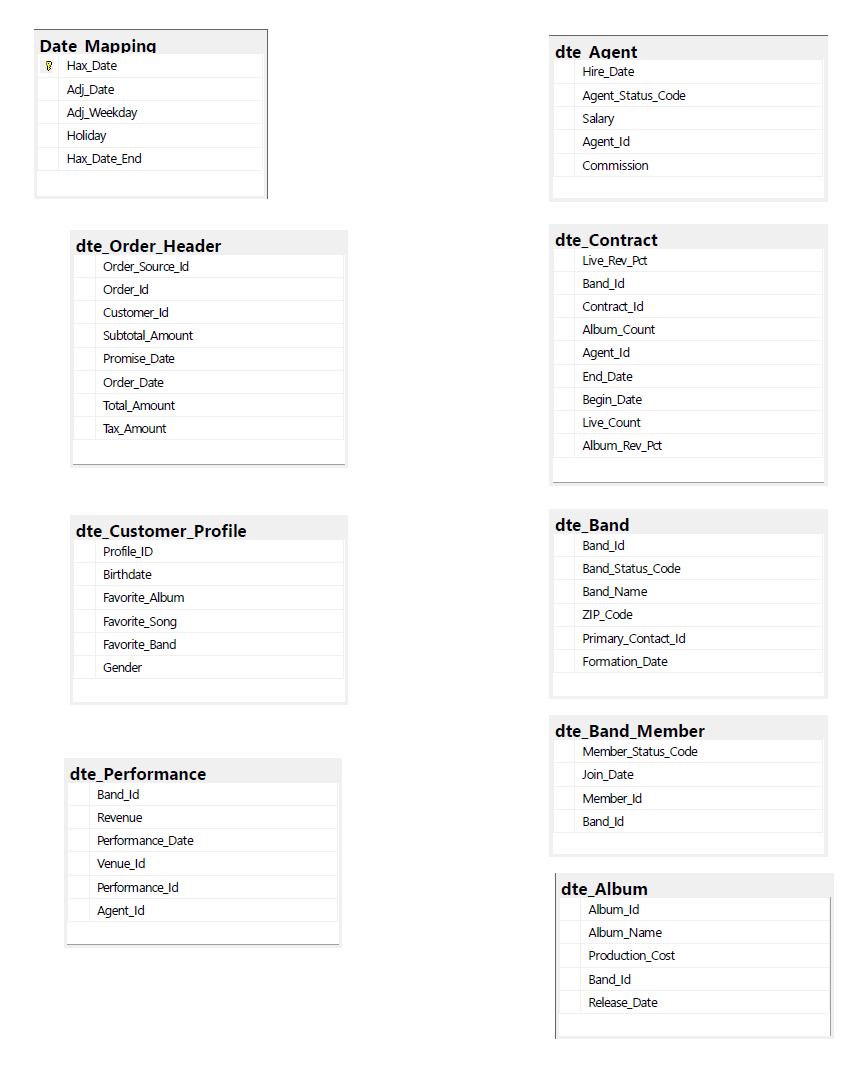
|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Column** | **Datatype** | **Comment** |
| Agent | Hire\_Date | Datetime, nullable |  |
| Album | Release\_Date | Datetime, nullable |  |
| Band | Formation\_Date | Datetime, nullable |  |
| Band\_Member | Join\_Date | Datetime, nullable |  |
| Contract | End\_Date | Datetime, nullable |  |
| Contract | Begin\_Date | Datetime, nullable |  |
| Customer | (none) | Datetime, nullable |  |
| Customer\_Profile | Birthdate | Datetime, nullable |  |
| Order\_Header | Promise\_Date | Datetime, nullable |  |
| Order\_Header | Order\_Date | Datetime, nullable |  |
| Performance | Performance\_Date | Datetime, nullable | Weekday adjustment |
| Stream | Stream\_Date | Datetime, not null | (new) |
| Band\_Follower | Follow\_Date | Datetime, not null | (new), part of PK |
| Band\_Follower | Unfollow\_Date | Datetime, nullable | (new) |

To facilitate the process, the following objects have been added to the Hallux database.

|  |  |  |
| --- | --- | --- |
| **Object** | **Type** | **Purpose** |
| Date\_Mapping | Table | Maps current date to a new incremented date |
| Prc\_Date\_Mapping\_Populate | Stored Procedure | Populates Date\_Mapping with all unique dates |
| Prc\_Date\_Mapping\_Adjust | Stored Procedure | Sets all adjustment columns in Date\_Mapping for the specified number of years. |
| Prc\_Date\_Mapping\_Update\_Hax | Stored Procedure | Updates the database with the new dates |
| Prc\_Date\_Mapping\_Update | Stored Procedure | Updates or inserts a single row in Date\_Mapping |
| Prc\_Date\_Mapping\_Retrieve | Stored Procedure | Returns date mapping rows for a specified date |
| Bak\_ | Tables | Backup of original tables |
| Dte\_ | Tables | Copy of tables with a date column used for testing |

**Appendices**

Appendix 1 – Schema



Appendix 2 – Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Column** | **Data Type** | **Description** |
| Date\_Mapping | Hax\_Date | Date | Original Date |
| Date\_Mapping | Adj\_Date | Date | Incremented Date |
| Date\_Mapping | Adj\_Weekday | Date | Adjusted date with same day of week as the original date. |
| Date\_Mapping | Holiday | Char(10) | FIXED - fixed date holiday (fourth of July)  FLOATING – based on a day of the week (Memorial Day) |
| Date\_Mapping | Hax\_Date\_End | Datetime | Hax Date +’23:59:59:990’ |

Appendix 3 – DDL

CREATE TABLE Date\_Mapping

(

Hax\_Date Date NOT NULL,

Adj\_Date Date NULL,

Adj\_Weekday Date NULL,

Holiday Char(10) NULL,

CONSTRAINT pk\_Date\_Mapping PRIMARY KEY CLUSTERED (Hax\_Date)

)

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Retrieve') IS NOT NULL

BEGIN

DROP PROCEDURE dbo.prc\_Date\_Mapping\_Retrieve

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Retrieve') IS NOT NULL

PRINT '<<< FAILED DROPPING PROCEDURE dbo.prc\_Date\_Mapping\_Retrieve >>>'

ELSE

PRINT '<<< DROPPED PROCEDURE dbo.prc\_Date\_Mapping\_Retrieve >>>'

END

go

Create Proc dbo.prc\_Date\_Mapping\_Retrieve ( @a\_From\_Date date , @a\_To\_Date date)

As

Begin

/\*-----------------------------------------------------------------------------

Procedure Name: prc\_Date\_Mapping\_Retrieve

Function: Return Date\_Mapping data for the specified date range.

Parameters: @a\_From\_Date, @a\_To\_Date

Modifications:

10/04/2022 JDS Initial Version

-----------------------------------------------------------------------------\*/

select Hax\_Date

,Adj\_Date

,Adj\_Weekday

,Holiday

,Hax\_Date\_End

from Date\_Mapping

where Hax\_Date between @a\_From\_Date and @a\_To\_Date

End /\* prc\_Date\_Mapping\_Retrieve \*/

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Retrieve') IS NOT NULL

PRINT '<<< CREATED PROCEDURE dbo.prc\_Date\_Mapping\_Retrieve >>>'

ELSE

PRINT '<<< FAILED CREATING PROCEDURE dbo.prc\_Date\_Mapping\_Retrieve >>>'

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Populate') IS NOT NULL

BEGIN

DROP PROCEDURE dbo.prc\_Date\_Mapping\_Populate

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Populate') IS NOT NULL

PRINT '<<< FAILED DROPPING PROCEDURE dbo.prc\_Date\_Mapping\_Populate >>>'

ELSE

PRINT '<<< DROPPED PROCEDURE dbo.prc\_Date\_Mapping\_Populate >>>'

END

go

Create Proc dbo.prc\_Date\_Mapping\_Populate

As

Begin

/\*-----------------------------------------------------------------------------

Procedure Name: prc\_Date\_Mapping\_Populate

Function: Populate Date\_Mapping table with all unique dates in the database

Parameters: None

Modifications:

10/01/2022 JDS Initial Version

-----------------------------------------------------------------------------\*/

DECLARE @errorMsg varchar(255)

, @errorNum int

, @Rowcount int

-- Initialization

SELECT @errorNum = 0

Delete from Date\_Mapping

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Delete of Date\_Mapping failed.'

GOTO err\_rtn

END

insert into Date\_Mapping (Hax\_Date)

select distinct convert(date,hire\_date) from agent where hire\_date is not null

union select distinct convert(date,release\_date) from album where release\_date is not null

union select distinct convert(date,formation\_date) from band where formation\_date is not null

union select distinct convert(date,join\_date) from band\_member where join\_date is not null

union select distinct convert(date,end\_date) from contract where end\_date is not null

union select distinct convert(date,begin\_date) from contract where begin\_date is not null

union select distinct convert(date,order\_date) from order\_header where order\_date is not null

union select distinct convert(date,promise\_date) from order\_header where promise\_date is not null

union select distinct convert(date,performance\_date) from performance where performance\_date is not null

union select distinct convert(date,birthdate) from customer\_profile where birthdate is not null

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Insert into Date\_Mapping failed.'

GOTO err\_rtn

END

update Date\_Mapping set Hax\_Date\_End = convert(datetime,convert(varchar(20), Hax\_Date , 101 ) + ' 23:59:59:990')

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Error updating Hax\_Date\_End.'

GOTO err\_rtn

END

good\_rtn:

Return 0

err\_rtn:

select @errorMsg = 'SQL Exception (' + object\_name(@@procid) + ') - ' + @errorMsg + case when IsNull(@errornum,0) <> 0 then ' (SQL Errno: ' + convert(varchar(30),@errorNum) + ')' else null end

--RAISERROR 50001 @errorMsg

Return -1

End /\* prc\_Date\_Mapping\_Populate \*/

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Populate') IS NOT NULL

PRINT '<<< CREATED PROCEDURE dbo.prc\_Date\_Mapping\_Populate >>>'

ELSE

PRINT '<<< FAILED CREATING PROCEDURE dbo.prc\_Date\_Mapping\_Populate >>>'

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Adjust') IS NOT NULL

BEGIN

DROP PROCEDURE dbo.prc\_Date\_Mapping\_Adjust

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Adjust') IS NOT NULL

PRINT '<<< FAILED DROPPING PROCEDURE dbo.prc\_Date\_Mapping\_Adjust >>>'

ELSE

PRINT '<<< DROPPED PROCEDURE dbo.prc\_Date\_Mapping\_Adjust >>>'

END

go

Create Proc dbo.prc\_Date\_Mapping\_Adjust ( @a\_NumYears smallint = 1 )

As

Begin

/\*-----------------------------------------------------------------------------

Procedure Name: prc\_Date\_Mapping\_Adjust

Function: Set the adjusted dates in Date\_Mapping

Parameters: @a\_NumYears

Modifications:

10/01/2022 JDS Initial Version

-----------------------------------------------------------------------------\*/

DECLARE @errorMsg varchar(255)

, @errorNum int

, @Rowcount int

, @cnt int

-- Initialization

SELECT @errorNum = 0

-- set Adj\_Date

UPDATE Date\_Mapping set Adj\_Date = dateadd( year, @a\_NumYears , Hax\_Date ), Holiday = null

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Adj\_Date failed.'

GOTO err\_rtn

END

-- identify holidays

create table #temp\_holiday (

Holiday Char(10) NULL

,Month tinyint NULL

,Weekday tinyint NULL

,Day\_Range\_Begin tinyint NULL

,Day\_Range\_End tinyint NULL

)

-- fixed holidays

insert into #temp\_holiday values ( 'FIXED' ,1 , null, 1 , 1 ) -- New Year's Day

insert into #temp\_holiday values ( 'FIXED' ,6 , null, 19 ,19) -- Juneteenth National Independence Day

insert into #temp\_holiday values ( 'FIXED' ,7 , null, 4 , 4 ) -- Independence Day

insert into #temp\_holiday values ( 'FIXED' ,10, null, 31, 31) -- Holloween

insert into #temp\_holiday values ( 'FIXED' ,11, null, 11, 11) -- Veterans Day

insert into #temp\_holiday values ( 'FIXED' ,12, null, 24, 24) -- Christmas Eve

insert into #temp\_holiday values ( 'FIXED' ,12, null, 25, 25) -- Christmas Day

insert into #temp\_holiday values ( 'FIXED' ,12, null, 31, 31) -- New Year’s Eve

-- floating holidays

insert into #temp\_holiday values ( 'FLOATING', 1 , 2, 15, 21) -- Martin Luther King , Jr. Birthday

insert into #temp\_holiday values ( 'FLOATING', 2 , 2, 15, 21) -- Washington's Birthday

insert into #temp\_holiday values ( 'FLOATING', 5 , 2, 25, 31) -- Memorial Day

insert into #temp\_holiday values ( 'FLOATING', 9 , 2, 1 , 7 ) -- Labor Day

insert into #temp\_holiday values ( 'FLOATING', 10, 2, 8 , 14) -- Columbus Day

insert into #temp\_holiday values ( 'FLOATING', 11, 5, 22, 28) -- Thanksgiving Day

UPDATE Date\_Mapping set Holiday = t.Holiday

from Date\_Mapping m

, #temp\_holiday t

where datepart(month ,m.Hax\_Date) = t.Month

and datepart(day ,m.Hax\_Date) between t.Day\_Range\_Begin and t.Day\_Range\_End

and (t.Weekday is null or datepart(weekday,m.Hax\_Date) = t.Weekday)

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Holiday failed.'

GOTO err\_rtn

END

-- set Adj\_Weekday to have the same day of the week of Hax\_Date

UPDATE Date\_Mapping set Adj\_Weekday = dateadd( day, datepart(weekday,Hax\_Date) - datepart(weekday,Adj\_Date) , Adj\_Date)

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Adj\_Weekday failed.'

GOTO err\_rtn

END

-- make sure the new floating holiday is in the appropriate date range

UPDATE Date\_Mapping set Adj\_Weekday = dateadd( day, 7, Adj\_Weekday)

from Date\_Mapping m

, #temp\_holiday t

where m.Holiday = 'FLOATING'

and datepart(month ,m.Hax\_Date) = t.Month

and datepart(day ,m.Hax\_Date) between t.Day\_Range\_Begin and t.Day\_Range\_End

and datepart(day ,m.Adj\_Holiday) < t.Day\_Range\_Begin

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Floating Adj\_Weekday failed.'

GOTO err\_rtn

END

good\_rtn:

Return 0

err\_rtn:

select @errorMsg = 'SQL Exception (' + object\_name(@@procid) + ') - ' + @errorMsg + case when IsNull(@errornum,0) <> 0 then ' (SQL Errno: ' + convert(varchar(30),@errorNum) + ')' else null end

--RAISERROR 50001 @errorMsg

Return -1

End /\* prc\_Date\_Mapping\_Adjust \*/

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Adjust') IS NOT NULL

PRINT '<<< CREATED PROCEDURE dbo.prc\_Date\_Mapping\_Adjust >>>'

ELSE

PRINT '<<< FAILED CREATING PROCEDURE dbo.prc\_Date\_Mapping\_Adjust >>>'

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Update\_Hax') IS NOT NULL

BEGIN

DROP PROCEDURE dbo.prc\_Date\_Mapping\_Update\_Hax

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Update\_Hax') IS NOT NULL

PRINT '<<< FAILED DROPPING PROCEDURE dbo.prc\_Date\_Mapping\_Update\_Hax >>>'

ELSE

PRINT '<<< DROPPED PROCEDURE dbo.prc\_Date\_Mapping\_Update\_Hax >>>'

END

go

Create Proc dbo.prc\_Date\_Mapping\_Update\_Hax

As

Begin

/\*-----------------------------------------------------------------------------

Procedure Name: prc\_Date\_Mapping\_Update\_Hax

Function: update the database with the adjusted dates in Date\_Mapping

Parameters: none

Modifications:

10/01/2022 JDS Initial Version

-----------------------------------------------------------------------------\*/

DECLARE @errorMsg varchar(255)

, @errorNum int

, @Rowcount int

-- Initialization

SELECT @errorNum = 0

update agent set Hire\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), hire\_date, 108 ) + ':' + convert(varchar(30), datepart(millisecond,hire\_date)))

from Date\_Mapping m

, agent t

where t.Hire\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Hire\_Date failed.'

GOTO err\_rtn

END

update album set Release\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), Release\_date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Release\_date)))

from Date\_Mapping m

, album t

where t.Release\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Release\_Date failed.'

GOTO err\_rtn

END

update Band set Formation\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), Formation\_date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Formation\_date)))

from Date\_Mapping m

, Band t

where t.Formation\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Formation\_Date failed.'

GOTO err\_rtn

END

update Band\_Member set Join\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), Join\_Date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Join\_Date)))

from Date\_Mapping m

, Band\_Member t

where t.Join\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Join\_Date failed.'

GOTO err\_rtn

END

update Contract set Begin\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), Begin\_Date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Begin\_Date)))

from Date\_Mapping m

, Contract t

where t.Begin\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Begin\_Date failed.'

GOTO err\_rtn

END

update Contract set End\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), End\_Date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, End\_Date)))

from Date\_Mapping m

, Contract t

where t.End\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of End\_Date failed.'

GOTO err\_rtn

END

update order\_header set Order\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), Order\_Date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Order\_Date)))

from Date\_Mapping m

, order\_header t

where t.Order\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Order\_Date failed.'

GOTO err\_rtn

END

update order\_header set Promise\_Date = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), Promise\_Date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Promise\_Date)))

from Date\_Mapping m

, order\_header t

where t.Promise\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Promise\_Date failed.'

GOTO err\_rtn

END

-- maintain the day of the week unless the date is a fixed holiday

update Performance set Performance\_Date = convert(datetime, convert(varchar(30), case when m.Holiday = 'FIXED' then m.Adj\_Date else m.Adj\_Weekday end, 101 ) + ' ' + convert(varchar(30), Performance\_Date, 108 ) + ':' + convert(varchar(30), datepart(millisecond, Performance\_Date)))

from Date\_Mapping m

, Performance t

where t.Performance\_Date between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of Performance\_Date failed.'

GOTO err\_rtn

END

-- birthday is always on the same date

update customer\_profile set birthdate = convert(datetime, convert(varchar(30), m.Adj\_Date, 101 ) + ' ' + convert(varchar(30), birthdate, 108 ) + ':' + convert(varchar(30), datepart(millisecond, birthdate)))

from Date\_Mapping m

, customer\_profile t

where t.Birthdate between m.Hax\_Date and m.Hax\_Date\_End

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Update of birthdate failed.'

GOTO err\_rtn

END

good\_rtn:

Return 0

err\_rtn:

select @errorMsg = 'SQL Exception (' + object\_name(@@procid) + ') - ' + @errorMsg + case when IsNull(@errornum,0) <> 0 then ' (SQL Errno: ' + convert(varchar(30),@errorNum) + ')' else null end

--RAISERROR 50001 @errorMsg

Return -1

End /\* prc\_Date\_Mapping\_Update\_Hax \*/

go

IF OBJECT\_ID('dbo.prc\_Date\_Mapping\_Update\_Hax') IS NOT NULL

PRINT '<<< CREATED PROCEDURE dbo.prc\_Date\_Mapping\_Update\_Hax >>>'

ELSE

PRINT '<<< FAILED CREATING PROCEDURE dbo.prc\_Date\_Mapping\_Update\_Hax >>>'

go

IF OBJECT\_ID('dbo.prc\_Increment\_Date') IS NOT NULL

BEGIN

DROP PROCEDURE dbo.prc\_Increment\_Date

IF OBJECT\_ID('dbo.prc\_Increment\_Date') IS NOT NULL

PRINT '<<< FAILED DROPPING PROCEDURE dbo.prc\_Increment\_Date >>>'

ELSE

PRINT '<<< DROPPED PROCEDURE dbo.prc\_Increment\_Date >>>'

END

go

Create Proc dbo.prc\_Increment\_Date ( @a\_date\_curr date = null , @a\_date\_new date = null, @a\_table varchar(30) = null, @a\_column varchar(30) = null)

As

Begin

/\*-----------------------------------------------------------------------------

Procedure Name: prc\_Increment\_Date

Function: Changes the specified date/column/table combination to the new date

Parameters:

a\_dt\_curr

a\_dt\_new

a\_table

a\_column

Modifications:

09/30/2022 JDS Initial Version

-----------------------------------------------------------------------------\*/

DECLARE @errorMsg varchar(255)

, @sql varchar(1000)

, @where\_clause varchar(255)

, @new\_time varchar(255)

, @new\_date varchar(30)

, @errorNum int

, @Rowcount int

, @dt\_start datetime

, @dt\_end datetime

-- Initialization

SELECT @errorNum = 0

SELECT @dt\_start = @a\_date\_curr

SELECT @dt\_end = convert(datetime, convert(varchar(30), @dt\_start, 101 ) + ' 23:59:59:999')

/\*

IF ( @a\_dt\_end < @a\_dt\_start)

BEGIN

SELECT @errorMsg = 'End date cannot be prior to Start date.'

GOTO err\_rtn

END

\*/

-- validate @a\_table , @a\_column

/\*

This is the sql prototype

update agent

set hire\_date = convert(datetime, convert(varchar(30), @a\_date\_new, 101 ) + ' ' + convert(varchar(30), hire\_date, 108 ) + ':' + convert(varchar(30), datepart(millisecond,hire\_date)))

where hire\_date between @dt\_start and @dt\_end

test run

exec prc\_Increment\_Date @a\_date\_curr = '9/1/22' , @a\_date\_new = '9/2/23', @a\_table = 'agent', @a\_column = 'hire\_date'

update agent

set hire\_date = convert(datetime, "09/02/2023" + " " + convert(varchar(30), hire\_date, 108 ) + ":" + convert(varchar(30), datepart(millisecond,hire\_date)))

where hire\_date between "09/01/2022" and "09/01/2022 23:59:59:999"

\*/

select @new\_date = convert(varchar(30), @a\_date\_new, 101 )

, @new\_time = 'convert(varchar(30), ' + @a\_column + ', 108 ) + ":" + convert(varchar(30), datepart(millisecond,'+ @a\_column +'))'

, @where\_clause = 'where ' + @a\_column + ' between "' + convert(varchar(30), @a\_date\_curr, 101 ) + '" and "' + convert(varchar(30),@a\_date\_curr, 101 ) + ' 23:59:59:999"'

select @sql = 'update ' + @a\_table + ' set ' + @a\_column + ' = convert(datetime, "' + @new\_date + '" + " " + ' + @new\_time + ') ' + @where\_clause

--BEGIN TRANSACTION

select @sql

-- execute ( @sql )

SELECT @errorNum = @@error , @Rowcount = @@Rowcount

IF @errorNum != 0

BEGIN

SELECT @errorMsg = 'Date update failed for ' + @a\_table + '.' + @a\_column

-- GOTO roll\_rtn

END

-- COMMIT TRANSACTION

good\_rtn:

Return 0

roll\_rtn:

-- ROLLBACK TRANSACTION

err\_rtn:

select @errorMsg = 'SQL Exception (' + object\_name(@@procid) + ') - ' + @errorMsg + case when IsNull(@errornum,0) <> 0 then ' (SQL Errno: ' + convert(varchar(30),@errorNum) + ')' else null end

--RAISERROR 50001 @errorMsg

Return -1

End /\* prc\_Increment\_Date \*/

go

IF OBJECT\_ID('dbo.prc\_Increment\_Date') IS NOT NULL

PRINT '<<< CREATED PROCEDURE dbo.prc\_Increment\_Date >>>'

ELSE

PRINT '<<< FAILED CREATING PROCEDURE dbo.prc\_Increment\_Date >>>'

go