SCSUG Paper 19-2017

Make it a Date! Setting up a Master Date View in SAS®

Crystal Carel, MPH[1,2]

[1]STEEEP Analytics, Baylor Scott & White Health, Dallas, TX

[2]Northern Illinois University, College of Health & Human Sciences

ABSTRACT

Dates are the basis of all pain and gain in programming. In my experience, working with beginning to seasoned programmers, dates are the number one item that make (and break) processes. Whether developing ETL or reporting, almost all data management and sourcing involves dates. Thus, dates have been a source of difficulty when programming for automation. This doesn't have to be the case though! Having a date view that is prepopulated with every variation of dates you may need while reporting helps a programmer (or team) become more effective while eliminating data manipulation that can halt processes. While this all may seem complicated, it is easier done than said!

This paper will provide a SAS® programmer instructions with the code needed to set up, create, and have their own (or shared) date view to drive data processes without having to create hard coded dates or date macros every session. We will discuss the options available to set up a view with dates and customize formats for data management and reporting. We will demonstrate setting up the date table and view with presentation of how to use the date view in code. Using a date view will help SAS® programmers become more efficient coders by using one source for dates to automate processes and by not always having to code around complex date situations such as skipping holidays.

INTRODUCTION

Regardless of the field you are in, the process you are coding, or the report you are producing – I can guarantee dates are somewhere in there. Dates are everywhere - driving ETL to being front and center on reports. However, due to the complexity of knowing how to code to get the date you need, you may revert to hard coding dates and adjusting each run cycle. This is not sustainable practice and keeps you from being hands-off your code for automation. On the other hand, you may be using date macros, which can also be problematic depending on how you set up the macro. Either case, a date view can solve these issues and make your life easier.

We will start small and work ourselves up. I've included appendixes with code you can copy/paste and make a date table quicker than you can read this paper! Stick with me though as the paper will do a high-level explanation of the steps and you can see for yourself how you can have every imaginable date you desire. The code will incorporate a fiscal year (July – June), calendar year (Jan – December), and holidays to keep it short. However, whatever crazy schedule you may follow (i.e. process is to run on the 3rd Monday after the 1st Tuesday of the month) – you can manipulate the code to serve that purpose and no longer be calendar counting!

PREPPING FOR CODE

Before we spend a lot of time coding and rewriting code – we need to consider what we want to see and how it should be displayed with regards to formatting. For this paper, I will present various date formats, however review your code and see what dates and formats you use and adjust according.

CODE STRUCTURE

The code provided in Appendix A may look repetitious and that is because it is. The code structure of creating a date table becomes easier once the components are understood and defined. We will start out small using the code shown in Figure 1 to gain an understanding. The only difference between the code shown in Figure 1 and Appendix A is the number of date variables being captured as the structure is the same.

FIGURE 1. STARTING OUT SMALL

```
DATA PERM.DATES (INDEX=(CALDATE) COMPRESS=BINARY);
      ATTRIB DATEKEY
      LENGTH=8
      FORMAT=YYMMDDN8.
      LABEL="%SYSFUNC(PROPCASE(DATE KEY))";
      ATTRIB CALDATE
      LENGTH=8
      FORMAT=DATE9.
      LABEL="%SYSFUNC(PROPCASE(CALENDAR DATE))";
      DO I = INTNX('MONTH',TODAY(),-1,'E')
         TO INTNX('MONTH',TODAY(),+1,'E')
      BY 1;
DATEKEY = INTNX('DAY',I,0,'B');
CALDATE = DATEKEY;
OUTPUT;
END;
DROP I;
RUN;
```

COMPONENTS TO CREATE A DATE TABLE

For each section of the components, we will be dissecting Figure 1 to understand what the code is doing at a high level. Some pieces may require additional reading if you are unfamiliar with the function.

1.SET TABLE NAME AND INDEX(ES)

```
      DATA PERM.DATES (INDEX=(CALDATE) COMPRESS=BINARY);
```

What's going on?

- Creating a new table called DATES.
- Making the table permanent in a library called PERM.
- Optimizing performance by creating an INDEX on CALDATES.
  - This is something you need to decide based on your variables within your own date table.
- Removing extra spaces by COMPRESS=BINARY (since mostly numeric data).
- Further INDEX= and COMPRESS= reading, see REFERENCS #1 and #2.

2. ASSIGN ATTRIBUTES

```
      ATTRIB DATEKEY
      LENGTH=8
      FORMAT=YYMMDDN8.
      LABEL="%SYSFUNC(PROPCASE(DATE KEY))";
      ATTRIB CALDATE
      LENGTH=8
      FORMAT=DATE9.
      LABEL="%SYSFUNC(PROPCASE(CALENDAR DATE))";
```

What's going on?

- Using the function ATTRIB, we are assigning the attributes to each variable (DATEKEY & CALDATE) which we will later be defining under component #4.
- LENGTH and FORMAT are used to show how the variable will be displayed for the individual variable.
- The variable name may be shorted, but we can use LABEL to better give definition to variable.

3. DEFINE DATE RANGE

```
      DO I = INTNX('MONTH',TODAY(),-1,'E')
          TO INTNX('MONTH',TODAY(),+1,'E')
      BY 1;
```

What's going on?

- Using a DO LOOP, we are taking the value of today's date associated month, looking back (-1) or forward (+1) one month and grabbing the end (E) of the month.
    - Example: Today is August 25, 2017.
        - Looking back 1 month is July and the last day in July is July 31, 2017.
        - Looking forward 1 month is September and the last day in September is September 30, 2017.
- BY 1 – we are looking at each occurrence (which in this case is a day – see next piece).

4. DEFINE VARIABLE VALUES

```
      DATEKEY = INTNX('DAY',I,0,'B');
      CALDATE = DATEKEY;
```

What's going on?

- Now we are defining the variables.
    - DATEKEY is calculated by using "I" and looking at the beginning date (moot point) for each iteration of "I"
    - CALDATE will be the same as DATEKEY
    - Previously we set to have "I" be between July 31, 2017 and September 30, 2017)
- If you remember in component #1, even though the variables are the same – we have assigned different formats.

5. CLEAN UP AND FINISH

```
      OUTPUT;
      END;
      DROP I;
      RUN;
```

What's going on?

- OUTPUT so we can write our dates to the data set.
- Every time you run a DO LOOP, it must END**.**
- We are dropping the variable "I" (it was a part of the DO LOOP).
- We RUN the process and finish.

CREATING A VIEW

Once you have the basics how to set up a date table with all the variables you may need for your ETL and reporting, we can now create a view. We create a view to free up work space and faster performance. Additionally, if we will share the table with a team or other programmers, a view is preferred as it cannot get locked down when it opens.

We will create a view using PROC SQL. Think ahead of what data you want to be shown in the view and the time period (today only, looking ahead one year, looking behind and ahead 2 years, etc.). Below is the structure of how to create a view with notes about the components. For our example, we will be creating a view that shows the past and future two fiscal years.

See Appendix B for code to copy/paste to create a view based on the table we have created in this paper. For more information about views (see REF#1 or about the INTNX function REF#2 & REF#3).

FIGURE 2. CREATE A VIEW

```
PROC SQL;
CREATE VIEW PERM.VW_NAME AS
SELECT DISTINCT
    DATEKEY
   ,CALDATE
FROM PERM.DATES
WHERE INTNX('YEAR.7',TODAY(),-2,'B')<= CALDATE <=INTNX('YEAR.7',TODAY(),+2,'E');
QUIT;
```

COMPONENTS TO CREATE A DATE VIEW

For each section of the components, we will be dissecting Figure 2 to understand what the code is doing at a high level. Some pieces may require additional reading if you are unfamiliar with the function.

1.SET VIEW NAME

```
PROC SQL;
CREATE VIEW PERM.VW_NAME AS
```

What's going on?

- Creating a new view within PROC SQL called VW_DATES.
- Making the view permanent in a library called PERM.

2.IDENTIFY VARIABLES

```
SELECT DISTINCT
 DATEKEY
,CALDATE
```

What's going on?

- Selecting only the unique rows of dates for the following variables based on the WHERE clause (component #4).
- Including only the variables we want to show in the view.

3.INDICATE SOURCE

```
FROM PERM.DATES
```

What's going on?

- Identifying where the data is located for the view we want to build which is pulled from the PERM.DATES table.

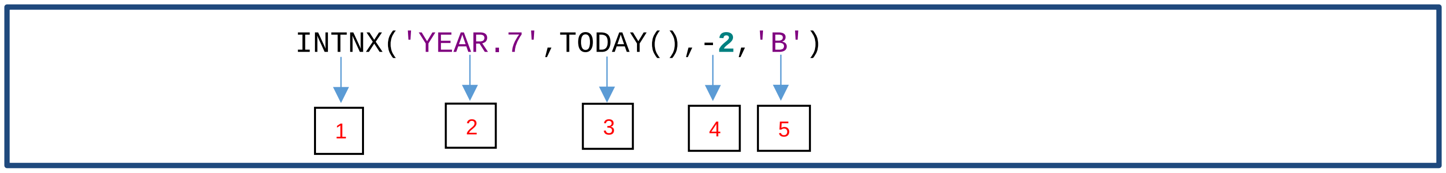
4. PARAMETERS

```
WHERE
INTNX('YEAR.7',TODAY(),-2,'B')<= CALDATE <=INTNX('YEAR.7',TODAY(),+2,'E');
```

What's going on?

- Indicate what parameters will be used to grab the dates.
- The date table may incorporate way more dates than you want to show in the view.

- To better explain the calculation of the WHERE statement, let's break it up:



```
INTNX('YEAR.7',TODAY(),-2,'B')
```

1. INTNX is the function being used which increments dates. (See REF#4 for background.)
2. An interval time period – always has to be in quotes – YEAR.7 is based on fiscal year (July – June).
3. This is our variable that the calculation will use as a 'base'.
4. This number (presented as 0 above) can be positive (+), negative (-), or 0.
5. This defines the time period interval you are wanting: E=END, B=Beginning, S=Same Day, M=Middle, etc.

Example:
Today is August 25, 2017 which is during FY2018. Looking back at 2 fiscal years at the beginning of that fiscal year would be July 1, 2016. Looking ahead at 2 fiscal years at the end of that fiscal year would be June 30, 2020. So our view would have July 1, 2016 to June 30, 2020 displayed (until the table and view are reprocessed).

5. FINISH UP

```
    QUIT;
```

- We end the PROC SQL step by QUIT and finish.

BRINGING IT ALL TOGETHER

We have created a date table and a view off that date table. While our table may be small, we can expand, calculate endless date variables and formats, and build a view off a snipit of that table. Using Appendix A, you will be able to copy/paste the code and run your own date table. Modify the formats and date variable calculations to set up a table that is customizatable but also captures a wide variety of dates in how you use them in your ETL and reporting.

CONCLUSION

Having a date view set up is advantageous for programmers to share across teams so ETL and reporting can be consistent with regards to a date structure. Additionally, it saves programmers time as code is reusable from project to project with the same date variable names and logic. Further customization based on the code provided can serve useful in creating your own date view that can be used and simply code.

REFERENCES

1. Clifford, Billy. “Frequently Asked Questions about SAS® Indexes”. SAS Institute Inc. 2004. Proceedings of the Thirtieth Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi30/008-30.pdf
2. Smith, Curtis A. “Programming Tricks For Reducing Storage And Work Space.” SAS Institute Inc. 2004. Proceedings of the Twenty-Seventh Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi27/p023-27.pdf
3. Stokes, James C. “SAS® Data Views Simply Stated” SAS Institute Inc. 2004. Proceedings of the Twenty-Ninth Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi29/067-29.pdf
4. SAS® Institute Inc., SAS® 9.2 Interval Functions INTNX and INTCK: Reference. Cary, NC: SAS® Institute Inc. Accessed August 25, 2017. http://support.sas.com/documentation/cdl/en/etsug/60372/HTML/default/viewer.htm#etsug_tsdata_sect038.htm
5. Carel, Crystal G. “Let Dates Drive Your Data: A Simple Primer Setting up Macro Dates” Proceedings of the 2015 Annual South Central SAS® Users Group Conference. http://www.scsug.org/wp-content/uploads/2015/11/Carel-Let_Dates_Drive_Your_Data.pdf
6. Yindra, Chris. “%SYSFUNC - The Brave New Macro World” SAS Institute Inc. 2004. Proceedings of the Twenty-Third Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi23/Advtutor/p44.pdf
7. Wright, Philip A. “Using the Data Step’s ATTRIB Statement to both Manage and Document Variables in a SAS® Dataset (lightly)” Proceedings of the 2009 Annual MidWest SAS® Users Group Conference. https://www.mwsug.org/proceedings/2009/stats/MWSUG-2009-D13.pdf

ACKNOWLEDGMENTS

I would like to thank current and past programmers within STEEEP Analytics who have helped me along the way with tidbits and tricks. I would especially like to thank Arthur Williams, who first introduced me to the infamous date table and explanation of how it could be used throughout automation as well as provided the basis code of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Crystal Carel
Baylor Scott & White Health
214-265-3674
Crystal.Carel@BSWHealth.org

APPENDIX A

```
/*********************************************************************************/
/**************  MACRO ISHDAY - WILL USE TO FLAG FOR HOLIDAYS ******************/
/*********************************************************************************/
%MACRO ISHDAY(DTE1=);
       &DTE1.=HOLIDAY('NEWYEAR',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('MLK',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('USPRESIDENTS',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('EASTER',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('MEMORIAL',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('USINDEPENDENCE',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('LABOR',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('COLUMBUS',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('VETERANS',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('THANKSGIVING',YEAR(&DTE1.))
    OR &DTE1.=HOLIDAY('CHRISTMAS',YEAR(&DTE1.))
%MEND ISHDAY;
/*********************************************************************************/
/***********************  CODE FOR DATA TABLE CREATION ************************/
/*********************************************************************************/
/********************  1. SET TABLE NAME AND INDEX(ES) ************************/
DATA WORK.DATES (INDEX=(FISCAL_YEAR CALYEAR)COMPRESS=BINARY);

/********************* 2. ASSIGN ATTRIBUTES ************************************/
ATTRIB DATEKEY
    LENGTH=8
    FORMAT=YYMMDDN8.
    LABEL="%SYSFUNC(PROPCASE(DATE KEY))";

ATTRIB CALDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR DATE))";

ATTRIB CAL_DOM
        LENGTH=3
    FORMAT=Z2.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR DAY OF MONTH NUMBER))";

ATTRIB CAL_DPM
        LENGTH=3
    FORMAT=Z2.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR DAYS PER MONTH NUMBER))";

ATTRIB CAL_DOQ
    LENGTH=3
    FORMAT=Z3.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR DAY OF QUARTER NUMBER))";

ATTRIB CAL_DOY
    LENGTH=3
    FORMAT=Z3.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR DAY OF YEAR NUMBER))";

ATTRIB CAL_DOW_L
    LENGTH=$9
    FORMAT=$CHAR9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR DAY OF WEEK - LONG NAME))";

ATTRIB CALWK_BDATE
        LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR WEEK BEGIN DATE))";

ATTRIB CAL_WOM
    LENGTH=3
    FORMAT=1.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR WEEK OF MONTH NUMBER))";


ATTRIB CAL_WOY
    %SYSFUNC(CATX(_,%SYSFUNC(PROPCASE(CAL)),%UPCASE(WOY)))
    LENGTH=3
    FORMAT=Z2.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR WEEK OF YEAR NUMBER))";
```

```
ATTRIB CALWK_EDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR WEEK END DATE))";

ATTRIB CALMTH_BDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR MONTH BEGIN DATE ))";

ATTRIB CAL_MOQ
    LENGTH=3
    FORMAT=1.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR MONTH OF QUARTER NUMBER))";

ATTRIB CAL_MOY
    LENGTH=3
    FORMAT=Z2.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR MONTH OF YEAR NUMBER))";

ATTRIB CAL_MOY_S
    LENGTH=$3
    FORMAT=$CHAR3.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR MONTH OF YEAR SHORT NAME))";

ATTRIB CAL_MOY_L
    LENGTH=$9
    FORMAT=$CHAR9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR MONTH OF YEAR LONG NAME))";

ATTRIB CALMTH_EDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR MONTH END DATE))";

ATTRIB CALQTR_BDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR QUARTER BEGIN DATE))";

ATTRIB CAL_QOY
    LENGTH=3
    FORMAT=1.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR QUARTER OF YEAR NUMBER))";

ATTRIB CALQTR_EDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR QUARTER END DATE))";

ATTRIB CALYEAR_BDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR YEAR BEGIN DATE))";

ATTRIB CALYEAR
    LENGTH=4
    FORMAT=4.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR YEAR NUMBER))";

ATTRIB CALYEAR_EDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(CALENDAR YEAR END DATE))";

ATTRIB FY_BDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(FISCAL YEAR BEGIN DATE))";


ATTRIB FY_MOY
    LENGTH=3
    FORMAT=Z2.
    LABEL="%SYSFUNC(PROPCASE(FISCAL YEAR MONTH OF YEAR NUMBER))";
```

```
ATTRIB FY_QOY
    LENGTH=3
    FORMAT=1.
    LABEL="%SYSFUNC(PROPCASE(FISCAL YEAR QUARTER OF YEAR NUMBER))";

ATTRIB FISCAL_YEAR
    LENGTH=4
    FORMAT=4.
    LABEL="%SYSFUNC(PROPCASE(FISCAL YEAR NUMBER))";

ATTRIB FY_EDATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(FISCAL YEAR END DATE))";

ATTRIB LFOM_DATE
    LENGTH=8
    FORMAT=DATE9.
    LABEL="%SYSFUNC(PROPCASE(LAST FRIDAY OF CALENDAR MONTH DATE))";

ATTRIB HOLIDAY_FLAG
    LENGTH=$1
    FORMAT=$CHAR1.
    LABEL="%SYSFUNC(PROPCASE(HOLIDAY FLAG))";

/****************** 3. DEFINE DATA RANGE (CURRENTLY SET AT 2 YEARS)************/
    DO I=INTNX('YEAR',TODAY(),-2,'E')
    TO   INTNX('YEAR.7',TODAY(),+2,'E')
    BY 1;

/****************** 4. DEFINE VARIABLE VALUES *********************************/
         DATEKEY        = INTNX('DAY',I,1,'B');
        CALDATE        = DATEKEY;
        CAL_DOM        = DAY(CALDATE);
        CAL_DPM        = DAY(INTNX('MONTH',CALDATE,0,'E'));
        CAL_DOQ        = INTCK('DAY',INTNX('QTR',CALDATE,0),CALDATE)+1;
        CAL_DOY        = INPUT(SUBSTR(PUT(JULDATE7(CALDATE),7.),5,3),3.);
        CAL_DOW_L      = TRIM(LEFT(PUT(CALDATE,WEEKDATE9.)));
        CALWK_BDATE    = INTNX('WEEK.1',CALDATE,0,'B');
        CAL_WOM        = INTCK('WEEK',INTNX('MONTH',CALDATE,0),CALDATE)+1;
        CAL_WOY        = INTCK('WEEK',INTNX('YEAR',CALDATE,0),CALDATE)+1;
        CALWK_EDATE    = INTNX('WEEK.1',CALDATE,0,'E');
        CALMTH_BDATE   = INTNX('MONTH',CALDATE,0,'B');
        CAL_MOQ        = INTCK('MONTH',INTNX('QTR',CALDATE,0),CALDATE)+1;
        CAL_MOY        = INTCK('MONTH',INTNX('YEAR',CALDATE,0),CALDATE)+1;
        CAL_MOY_S      = TRIM(LEFT(PUT(CALDATE,MONNAME3.)));
        CAL_MOY_L      = TRIM(LEFT(PUT(CALDATE,MONNAME.)));
        CALMTH_EDATE   = INTNX('MONTH',CALDATE,0,'E');
        CALQTR_BDATE   = INTNX('QTR',CALDATE,0,'B');
        CAL_QOY        = INTCK('QTR',INTNX('YEAR',CALDATE,0),CALDATE)+1;
        CALQTR_EDATE   = INTNX('QTR',CALDATE,0,'E');
        CALYEAR_BDATE  = INTNX('YEAR',CALDATE,0,'B');
        CALYEAR        = YEAR(CALDATE);
        CALYEAR_EDATE  = INTNX('YEAR',CALDATE,0,'E');
        FY_BDATE       = INTNX('YEAR.7',CALDATE,0,'B');
        FY_MOY         = INTCK('MONTH',FY_BDATE,CALMTH_BDATE)+1;
        FY_QOY         = INTCK('QTR',FY_BDATE,CALQTR_BDATE)+1;
        FISCAL_YEAR    = YEAR(FY_BDATE)+1;
        FY_EDATE       = INTNX('YEAR.7',CALDATE,0,'E');
        LFOM_DATE      = NWKDOM(5,6,MONTH(CALMTH_BDATE),YEAR(CALMTH_BDATE));

/*FLAGGING HOLIDAYS BASED ON MACRO*/
       IF %ISHDAY(DTE1=CALDATE) THEN HOLIDAY_FLAG = UPCASE('Y');
                                 ELSE HOLIDAY_FLAG = UPCASE('N');

/****************** 5. CLEAN UP AND FINISH *********************************/
       DROP I;
        OUTPUT;
    END;
RUN;
```

APPENDIX B

```
PROC SQL;
CREATE VIEW WORK.VW_DATES AS
SELECT DISTINCT
    DATEKEY
   ,CALDATE
   ,FISCAL_YEAR
   ,FY_BDATE
   ,FY_EDATE
   ,CAL_DOW_L
   ,CALMTH_BDATE
   ,CALMTH_EDATE
   ,CALYEAR_BDATE
   ,CALYEAR
   ,CALYEAR_EDATE
   ,HOLIDAY_FLAG
FROM WORK.DATES
WHERE INTNX('YEAR.7',TODAY(),-2,'B') <= CALDATE <= INTNX('YEAR.7',TODAY(),+2,'E');
QUIT;
```