Finding Time: SAS® and Data Warehouse Solutions for Determining Last Day of the Month

Jennifer O'Neil, RBC Financial Group, Toronto, ON, Canada
Laural Wierenga, Venturi Technology Partners, Kalamazoo, MI, USA

ABSTRACT
This paper presents the different ways a SAS programmer can automate the extraction of time series data from the warehouse where data are stored, by the last day of each month, in the column monthend date (ME_DT). The goal is to run programs and create data sets for each month with minimal program changes and no manual intervention: taking a point in time and going back months, quarters, and fiscal or calendar years for historical data, adjusting for leap years (eg, 2004, 2000, 1996) and last day of the month (28, 29, 30, 31). The two SAS solutions are a MACRO and the INTNX function. Two data warehouse solutions are given: one uses combinations of the +/- X year, +/- Y month, and +/- Z day functions in SQL, the second is a date lookup or time transformation table. We will discuss how you can apply these methods to meet different data extraction, analysis, and reporting requirements.

INTRODUCTION
This paper explores four different approaches to determining last day of the month. These dates can then be formatted and used for querying against the database, creating directories on the storage facility, forming part of a data set name, or calculating month-over-month values used in reports and other programs.

The four approaches are:
• Macro for generating global date variables
• Variations on the INTNX date interval function
• DB2/SQL functions: +/- X year, +/- Y month, +/- Z day
• Date lookup or time transformation table on the database

We will focus on monthly intervals, although the approaches can be modified to reflect weekly or daily intervals. We start with selecting a specific point in time—often, the monthend of the most recently loaded data in the warehouse—and work our way back in time. By monthend, we mean the last calendar day and not the last business or processing day of the month.

We assume that each time series table in the data warehouse, whether holding daily, weekly, or monthly data, has a monthend date (ME_DT) or snapshot date (SNAP_DT) column that is part of the index.

NOTATION
We assign today's date to the variable DT. The SAS function TODAY() returns the date on the system clock.

%global year0 month0 days;
%let dt=%sysfunc(today());
%let year0=%sysfunc(year(&dt));
%let month0=%sysfunc(month(&dt));
%let period=36;
%macro setlastday(m,y);
  %if &m=12 %then %let days=31;
  %else %if &m=11 %then %let days=30;
  %else %if &m=10 %then %let days=31;
  %else %if &m=9 %then %let days=30;
  %else %if &m=8 %then %let days=31;
  %else %if &m=7 %then %let days=31;
  %else %if &m=6 %then %let days=30;
  %else %if &m=5 %then %let days=31;
  %else %if &m=4 %then %let days=30;
  %else %if &m=3 %then %let days=31;
  %else %if &m=2 %then %do;
    %if %sysfunc(mod(&y,4))=0 %then %let days=29;
    %else %let days=28;
  %end;
  %do;
    %if %sysfunc(mod(&y,4))=0 %then %let days=29;
    %else %let days=28;
  %end;
%mend;
%macro setdate;
  data _null_;
  %do i=0 %to %period;
    %global t&i /*ds&i*/;
    %setlastday(&&month&i,&&year&i);
    %let m&i=%sysfunc(month(&dt));
    %let period=36;
    %macro setlastday(m,y);
      %if &m=12 %then %let days=31;
      %else %if &m=11 %then %let days=30;
      %else %if &m=10 %then %let days=31;
      %else %if &m=9 %then %let days=30;
      %else %if &m=8 %then %let days=31;
      %else %if &m=7 %then %let days=31;
      %else %if &m=6 %then %let days=30;
      %else %if &m=5 %then %let days=31;
      %else %if &m=4 %then %let days=30;
      %else %if &m=3 %then %let days=31;
      %else %if &m=2 %then %do;
        %if %sysfunc(mod(&y,4))=0 %then %let days=29;
        %else %let days=28;
      %end;
      %do;
        %if %sysfunc(mod(&y,4))=0 %then %let days=29;
        %else %let days=28;
      %end;
    %mend;
    %let t&i=%sysfunc(d&i.;/&days./&year&i);
%end;

We pick T1 as our starting point and assume it represents the most recent month of data available in the warehouse.

To query against the database, dates are formatted to 2/28/2003.

To create a directory or to form part of a data set name, dates can be formatted to either 20030228 or 200302 or 0302. In this paper we use the form ds200302 for data set names.

MACRO
The purpose of the macro is to generate ready-to-use date and data set macro variables for as many consecutive months as are required by your report or analysis.

The basic program has two parts. The macro SETLASTDAY is for assigning the appropriate number of days to a month (which can be 28, 29, 30, or 31) depending on the month and the year. The macro SETDATE calls the macro SETLASTDAY and creates global variables of formatted dates for your SQL code.

%global year0 month0 days;
%let dt=%sysfunc(today());
%let year0=%sysfunc(year(&dt));
%let month0=%sysfunc(month(&dt));
%let period=36;
%macro setlastday(m,y);
  %if &m=12 %then %let days=31;
  %else %if &m=11 %then %let days=30;
  %else %if &m=10 %then %let days=31;
  %else %if &m=9 %then %let days=30;
  %else %if &m=8 %then %let days=31;
  %else %if &m=7 %then %let days=31;
  %else %if &m=6 %then %let days=30;
  %else %if &m=5 %then %let days=31;
  %else %if &m=4 %then %let days=30;
  %else %if &m=3 %then %let days=31;
  %else %if &m=2 %then %do;
    %if %sysfunc(mod(&y,4))=0 %then %let days=29;
    %else %let days=28;
  %end;
  %do;
    %if %sysfunc(mod(&y,4))=0 %then %let days=29;
    %else %let days=28;
  %end;
%mend;
%macro setdate;
  data _null_;
  %do i=0 %to %period;
    %global t&i /*ds&i*/;
    %setlastday(&&month&i,&&year&i);
    %let m&i=%sysfunc(month(&dt));
    %let period=36;
    %macro setlastday(m,y);
      %if &m=12 %then %let days=31;
      %else %if &m=11 %then %let days=30;
      %else %if &m=10 %then %let days=31;
      %else %if &m=9 %then %let days=30;
      %else %if &m=8 %then %let days=31;
      %else %if &m=7 %then %let days=31;
      %else %if &m=6 %then %let days=30;
      %else %if &m=5 %then %let days=31;
      %else %if &m=4 %then %let days=30;
      %else %if &m=3 %then %let days=31;
      %else %if &m=2 %then %do;
        %if %sysfunc(mod(&y,4))=0 %then %let days=29;
        %else %let days=28;
      %end;
      %do;
        %if %sysfunc(mod(&y,4))=0 %then %let days=29;
        %else %let days=28;
      %end;
    %mend;
    %let t&i=%sysfunc(d&i.;/&days./&year&i);
%end;

If DT equals March 30, 2003, then T0 equals March 31, 2003, and the following time intervals will be set to:
• T1 = February 28, 2003
• T2 = January 31, 2003
• T3 = December 31, 2002
• T12 = March 31, 2002
• T24 = March 31, 2001
• T36 = March 31, 2000
We pick T1 as our starting point and assume it represents the most recent month of data available in the warehouse.

To query against the database, dates are formatted to 2/28/2003.

To create a directory or to form part of a data set name, dates can be formatted to either 20030228 or 200302 or 0302. In this paper we use the form ds200302 for data set names.
call symput("t&i",date&i);
/*
   USE THIS TO CREATE DATA SET NAMES;
call symput("ds&i”,
   "ds&&year&i.||put(&&month&i,z2.));
*/
%let j=%eval(&i+1);
%if %eval(&&month&i)=1 %then
  %do;
  %let year&j=%eval(&&year&i-1);
  %let month&j=12;
  %end;
%else %do;
  %let year&j=%eval(&&year&i);
  %let month&j=%eval(&&month&i-1);
  %end;
%end;
r
%mend;
%setdate;
You will create as many global time interval variables (T0,...,Tn) as needed to cover the entire period of your data extraction or reporting requirements. At the same time, the program also creates more variables than might be required. For example, if you need only previous month (T1), the same month as previous month one year ago (T13), and the same month as previous month two years ago (T25), then all the other time intervals (T2, T3, T4, and so on) are extra.

You can save this program and call it into other SAS programs with %include. Simply remove the comment marks when you want to create prefixes or suffixes for your data sets.

INTNX
The INTNX date function reverses or advances a date by the number and type of time interval specified by you and returns a SAS date. The returned date value can be aligned to the beginning (‘B’), the middle (‘M’), or the end (‘E’) of the time interval and must be formatted for querying against the database.
The basic INTNX statement is intnx('interval',start-from,increment<,'alignment'>).

data _null_;
t0=put(intnx('month',today(),0,'E'),mmddyy10.);
call symput('t0',"'"||t0||"'");
call symput('t1',"'"||put(intnx('month',input(t0,mmddyy10.),-1,'E'),mmddyy10.)||"'");
call symput('t12',"'"||put(intnx('month',input(t0,mmddyy10.),-12,'E'),mmddyy10.)||"'");
call symput('t24',"'"||put(intnx('month',input(t0,mmddyy10.),-24,'E'),mmddyy10.)||"'");
call symput('t36',"'"||put(intnx('month',input(t0,mmddyy10.),-36,'E'),mmddyy10.)||"'");
run;
/*
   USE THIS TO CREATE DATA SET NAMES;
data _null_;
call symput('ds0',
   ('"ds"||put(input(&t0,mmddyy10.),yymmnn6.).));
call symput('ds1',
   ('"ds"||put(input(&t1,mmddyy10.),yymmnn6.).));
call symput('ds12',
   ('"ds"||put(input(&t12,mmddyy10.),yymmnn6.).));
call symput('ds24',
   ('"ds"||put(input(&t24,mmddyy10.),yymmnn6.).));
call symput('ds36',
   ('"ds"||put(input(&t36,mmddyy10.),yymmnn6.).));
run;
*/
You can create as many variables as you need to cover the entire period of your requirements without creating anything extra but you will need one line of code for each type of date format per monthend date.

For yearly, weekly, or daily intervals, you can easily modify the INTNX program by changing the first argument from ‘month’ to ‘year’, ‘week’, or ‘day’ and adjust the second and third arguments accordingly.

DB2‡/SQL FUNCTIONS
This approach lets you transform ME_DT within SQL without creating date variables and offers a convenient way of determining monthend dates without having to format them in order to query against the database.
The functions work like this:
T0 - 1 year results in the same month and the same day one year ago. It also adjusts to February 29 for leap years.
  • If T0 = '3/31/2003' then T0 - 1 year = '3/31/2002'
  • If T0 = '2/28/2001' then T0 - 1 year = '2/29/2000'
  • If T0 = '2/29/2000' then T0 - 1 year = '2/28/1999'
In general, T0 - 1 month results in the same day one month ago. It does not adjust to the last day of the previous month by default; it depends on the number of days in this month and last month.
  • If T0 = '3/15/2003' then T0 - 1 month = '2/15/2003'
  • If T0 = '8/31/2003' then T0 - 1 month = '7/31/2003'
  • If T0 = '7/31/2003' then T0 - 1 month = '6/30/2003'
  • If T0 = '6/30/2003' then T0 - 1 month = '5/30/2003'
  • If T0 = '3/31/2003' then T0 - 1 month = '2/28/2003'
  • If T0 = '2/28/2003' then T0 - 1 month = '1/28/2003'

T0 - 1 day simply returns yesterday’s date.

Here is an example of how you might use these functions with the macro variable T1 (the SQL below looks for accounts past due two months ago that were charged off last month):

proc sql;
  select a2.acct_no
     from acct_table a1,
        acct_table a2
  where a1.acct_no = a2.acct_no
    and a2.me_dt = &t1
    and a1.me_dt = a2.me_dt
     + 1 day - 1 month - 1 day
    and a2.status in ('CHARGEOFF')
    and a1.status in ('PASTDUE');
We found these functions fun to use.
‡The syntax might be different or might not be supported on other databases.

DATE LOOKUP TABLE
If you want to determine the most recent month of data available in your warehouse, consider using a date lookup table for storing that information. The table can be updated as part of the load process. You can then use the stored information as a load
notification or job trigger for automating your SAS programs. This saves you time and database resources because you do not have to scan large time series tables for the most recent date.

The simplest date lookup table has one row and one column: T1.

Your SQL:

```sql
proc sql;
create table max_dt as
   select max(me_dt) as dt
   from large_time_series_table;
```

Now looks like this:

```sql
proc sql;
create table max_dt as
   select * as dt
   from date_lkup;
```

The temporary data set, MAX_DT, containing a single variable, DT, can now be used this way:

```sql
data _null_;
set max_dt;
call symput('t1','''||put(dt,mmddyy10.)||''');
run;
```

You can enhance this simple table by adding columns. Some examples:

- monthend (T2, T3, T4,...)
- quarterend (Q1, Q2, Q3,...)
- yearend (Y1, Y2, Y3,...)
- fiscal quarterend (FQ1, FQ2, FQ3,...)
- fiscal yearend (FY1, FY2, FY3,...)

You can further enhance the table by keeping records from previous months or by adding descriptions of each monthend date. If there are multiple records in the date lookup table, change your SELECT statement to:

```sql
select max(t1) as dt
```

To load such a table, use the DB2/SQL functions discussed above:

```
update date_lkup set
   (t1, t2, t3, t13, t25, t37) =
      (
         me_dt,
         me_dt + 1 day - 1 month - 1 day,
         me_dt + 1 day - 2 month - 1 day,
         me_dt + 1 day - 12 month - 1 day,
         me_dt + 1 day - 24 month - 1 day,
         me_dt + 1 day - 36 month - 1 day
      )
```

Your SQL now looks like this:

```sql
proc sql;
   select a.acct_no,
      a.status,
      s.me_dt,
      s.stmt_bal
   from acct_table a,
      stmt_table s,
      date_lkup d
   where a.acct_no = s.acct_no
      and a.me_dt = s.me_dt
      and s.me_dt in (d.t1, d.t2, d.t13, d.t25);
```

and the statement to extract selected months:

```sql
   and s.me_dt in (d.t1, d.t2, d.t13, d.t25);
```

can be written to include all months:

```sql
   and s.me_dt between d.t25 and d.t1;
```

WHEN TO USE WHICH APPROACH?

PRODUCTION REPORTS

We use production reports to monitor performance of the portfolio or response to a campaign on an ongoing basis. We also run production jobs to create data sets as input files for other systems. These reports or files are produced on a monthly basis where only the most recent month’s data need to be extracted. Previous months’ data have already been extracted, stored, distributed, or displayed.

For such reports, you can either use INTNX with the simple date lookup table or INTNX with the TODAY() function.

TREND REPORTS WITH TWO YEARS OF HISTORICAL DATA

Reports that trend data over a period of time are frequently used for setting future sales targets, developing pricing grids, or determining month-over-month activity. Generally, they do not need to be run every month once targets and prices are set or when activity levels have been determined.

Using MACRO or INTNX:

```sql
proc sql;
create table sugi28.twoyears as
   select me_dt,
      column1,
      column2
   from database_table
   where &t24 <= me_dt <= &t1;
```

Using date lookup table:

```sql
proc sql;
create table sugi28.twoyears as
   select a.me_dt,
      a.column1,
      a.column2
   from database_table a,
      date_lkup      d
   where d.t24 <= a.me_dt <= d.t1;
```

If your analysis requires separate data sets, or if your storage facility sets a limit on data set size forcing you to create many smaller data sets, write your SQL in a DO loop as shown below. It uses the global date and data set variables created by the macro SETDATE.

```sql
%macro pulldata;
%do i=1 %to &period;
   proc sql;
      create table sugi28.&ds&i as
         select me_dt,
            column1,
            column2
         from database_table
         where &t&i <= me_dt <= &t1;
   %end;
%mend;
```
AD HOC REPORTS FOR CURRENT AND SELECTED PREVIOUS MONTHS ONLY

Ad hoc reports fill the gaps when metrics are not available in the production reports or when business definitions differ. Ad hoc reports often turn into production reports and are common for determining the current health of the portfolio against some time in the past. For example, compare last quarter to previous quarter, and the same quarter a year ago.

You can use any of the approaches to accomplish this.

DATA ANALYSIS ACROSS DIFFERENT, POSSIBLY OVERLAPPING, TIME PERIODS

Your analysis may require you to compare 12-month rolling averages at every calendar quarterender. We let Q1 denote the most recent quarterender for which data exist in the warehouse, Q2 the previous quarterender, and so on. If Q1 equals March 31, 2003, then:

- Q2 = December 31, 2002
- Q3 = September 30, 2002
- Q4 = June 30, 2002
- Q5 = March 31, 2002

While T0 changes from April 30 to June 30, 2003, Q1 remains March 31, 2003, until June’s data are loaded.

Using the single-row date lookup table containing quarterender columns:

```sql
proc sql;
select a.acct_no,
    avg(case when a.me_dt > d.q5 and a.me_dt <= d.q1
            then a.debit_amt else 0
            end) as avg1,
    avg(case when a.me_dt > d.q6 and a.me_dt <= d.q2
            then a.debit_amt else 0
            end) as avg2,
    avg(case when a.me_dt > d.q7 and a.me_dt <= d.q3
            then a.debit_amt else 0
            end) as avg3,
    avg(case when a.me_dt > d.q8 and a.me_dt <= d.q4
            then a.debit_amt else 0
            end) as avg4
from activity_table a,
     date_lkup      d
group by a.acct_no;
```

CONCLUSION

Each approach is not any better or any worse than another. Answers to ‘What do you report on?’, ‘What analysis do you do?’, ‘How do you structure your data sets?’, and ‘How frequently do you run the program?’ will help determine the most elegant, efficient, or convenient approach to use. You are likely to find that a combination of these approaches is your ideal solution.

These methods help make your code less cluttered, free you from having to ‘baby-sit’ your SAS programs, and let you find time (pun intended) to do other tasks.

ACKNOWLEDGMENTS

We would like to thank our colleagues for taking the time to critically review our paper:
Robert Ellsworth
Marje Fecht

We would also like to thank Tony Anderson for his idea behind the macro structure and Neli Salapa for her assistance on the DB2/SQL functions.

CONTACT INFORMATION

We value your comments and welcome your questions. Please contact both authors at:

Jennifer O’Neil
CRM & Information Management
RBC Financial Group
200 Bay St 16th floor
Toronto, ON, Canada M5J 2J5
416.974.5885
jennifer.oneil@rbc.com

Laural Wierenga
Venturi Technology Partners
5278 Lovers Lane
Kalamazoo, MI 49002, USA
269.344.4100 ext. 552
lwierenga@venturipartners.com

TRADEMARKS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.