

Paper 134-2010

Working with SAS[®] Date and Time Functions: Foundations & Fundamentals

Andrew H. Karp

Sierra Information Services
Sonoma, CA USA

Abstract

Many new users are confused about how to work with variables (columns) representing dates and time in the SAS System. This paper provides an explanation of how to work with date and time variables, and some of the key SAS tools (e.g., functions and formats) available to simplify your work with your date and time data. Among the important concepts addressed in this paper are: 1) how the SAS System stores the values of dates and times; 2) how to create SAS date, time and datetime variables, 3) how Formats are used to control the “external representation,” or display of date, time and datetime variables, and, 4) how to use SAS programming language tools, such as functions and constants, to accomplish your programming goals.

Introduction

Most SAS Software users need to work with data sets containing one or more variables (columns) containing the date and/or time on which an event occurred. You may, for example, need to:

- **Determine the frequency** with which an event occurs in time (e.g., how many patients were admitted to the hospital in March 2010)
- **Find the time interval** which has elapsed between two events (e.g., what was the length of stay, in days, from admission to discharge for patients admitted in March 2010)
- **Select rows from a data set/table based on the values of a date or time variable** (e.g., extract observations where the patient was admitted between midnight and 8 am on all Saturdays and Sundays during March 2010).
- **Count the number of events by a date or time interval** (how many patients were admitted by month during calendar year 2009, or how many patients were admitted by hour of the day during March 2010)
- **Replace missing values in a historical time series of data** with statistically “robust” estimates of the missing values before going on to the next step in your reporting or analysis project
- **Interpolate higher time frequency interval observations from historical data collected at a lower time series frequency interval** (estimate weekly number of admissions from an historical series of monthly admission counts where the individual admission records are no longer available for analysis/aggregation)
- **Assign values to a SAS date, time or datetime variable** from “raw data” or from the values of other variables in an existing SAS data set
- **Import columns with date or time values stored in another software product** into a SAS data set using SAS/ACCESS[®] Software
- **Determine the current date and/or time on your computer’s system clock** at various points in your program’s execution.

The SAS System provides a wide range of resources you can use to address these and other programming issues that often arise when working with date and/or time values. These tools include

- **Informats:** used to assign values to SAS data set variables when a data set is created in a Data Step
- **Formats:** used to control the “external representation,” or “display” of the values of variable in your SAS-generated output
- **Functions:** applied, typically, in a SAS data step to calculate and assign values to variables/columns.
- **SAS System Options:** used to control various aspects of your SAS session
- **Procedures:** Pre-written “routines,” many of which are either written specifically for use with date and time data or which include important resources when working with dates and times.
- **Global Macro Symbol Table Variables:** A data table containing text variables assigned either automatically when you start your SAS session or whose values you can assign using SAS Macro Language commands.

We will explore some of the tools in each of these categories throughout the paper. In order to take advantage of them we first need to understand the core concepts about how the SAS System stores the values of variables representing dates and times in our data sets.

Core Concepts

A SAS date, time or datetime variable is a special case of the numeric variable.

- **Date variables:** an integer representing the number of days before or after January 1, 1960. In SAS, the number zero (0) represents January 1, 1960. If the event represented by the date variable occurred BEFORE January 1, 1960 then the value of the variable is negative. If it occurred AFTER January 1 1960 then its value is positive.
- **Time variables:** The number of seconds from midnight. Decimal values are permitted, and are often calculated by SAS programming language functions (see below) that “return” the current value of a time or datetime variable from your computer’s system clock.
- **Datetime variables:** A number representing the number of seconds the event represented by the variable occurred before or after midnight, January 1, 1960. A datetime variable stores both the date and the time in a single value. SAS programming language functions (see below) can be used to “find” or “return” the date or time “part” from a datetime variable.

Storing the values of dates and times as numbers makes it easy for SAS to perform both simple and complex operations on these data values. But, numeric date, time or datetime variables are hard, if not impossible, for users to read or interpret. That’s why we typically use a Format to display their values in SAS generated output. More on Formats for date, time and datetime variables later in this paper.

Creating SAS Date, Time or Datetime Variables

Many SAS users work with data sets where their date, time or datetime variables are already created for them. If you’re not one of these lucky users then you need to understand how to assign values to these variables when you create a SAS data set using SAS programming language statements in a SAS Data Step. Here are some examples.

Informats

An informat contains instructions SAS follows when assigning values to SAS data set variables from “raw data.” Appropriate use of informats for date, time or datetime variables can make it easy for you to convert how dates values are stored in your “raw data” to numeric SAS date, time or datetime variables in your SAS data set.

Here's an example. A text, or "raw data," file has about 42,800 rows of data, each of which contains information about the sale of a table to a customer. Columns 11 through 20 have the date the table was ordered as a text string, with the first two values as the month, the next two as the day and the last four as the year. And, in columns 21 through 30 we have the table shipment date as a text string with the first four values as year, the next two as the month and the last two as the day of shipment. The screen capture below shows the first few records from the table sales raw data file.

Order Number	Type	Material	Color	Order Date (MMDDYY10)	Ship Date (YYMMDD10)	Height	Price
000468251104	04	2005	2005-06-04	35			244
000557165605	08	2004	2004-06-07	25			169
000267234611	03	2004	2004-12-01	19			1726
000472754205	09	2005	2005-07-04	37			276
000288913603	11	2004	2004-04-19	15			379
000487634203	01	2005	2005-04-13	24			1886
000842963207	09	2005	2005-08-21	25			169
000137741104	28	2005	2005-05-07	28			200
000463262105	04	2005	2005-05-09	29			184
000908644303	26	2005	2005-03-31	31			299
000554765103	10	2005	2005-03-18	23			379
000937022605	12	2005	2005-07-13	24			817
000230955407	17	2004	2004-08-27	36			230
000702813711	28	2004	2004-11-30	17			661
000906243311	10	2004	2004-12-11	35			240
000364713305	30	2005	2005-07-31	18			641
000700325310	20	2004	2004-12-12	17			1383
000650232203	11	2004	2004-03-18	23			579

The problem is how to "tell" SAS that these text strings are dates when a Data Step is used to create a SAS data set from these raw data. The solution is to specify the appropriate informats in the INPUT Statement so that SAS "knows" how the date values are arranged and how to then convert them to numeric SAS date values.

The SAS data step on the next page shows how the appropriate SAS date informats are specified in the INPUT Statement to obtain the desired results. While there are other, more complex methods to accomplish the same result, the easiest (and most efficient) way is to use SAS date informats.

```

31 data tables;
32 infile "C:\Dates_And_Times\table_sales_raw_data.txt";
33 input
34     @1 order_number $ 7.
35     @8 type 1.
36     @9 material 1.
37     @10 color 1.
38     @11 order_date mmdyy10. /* informat for MMDDYYYY raw data values */
39     @21 ship_date yymmdd10. /* informat for YYMMDDDD raw data values */
40     @32 height 2.
41     @34 price 8.;
42 run;
43
44 options nonumber nocenter nodate;
45 proc print data=tables(obs=20);
46 title 'SAS Global Forum 2010';
47 Title2 'Foundations & Fundamentals';
48 title3 'Working With SAS Date and Time Functions';
49 title4 'Using INFORMATS: First 20 Observations from the Tables Sales Data Set';
50 run;
51
52
53 proc contents data=tables;
54 title4 'Descriptor Portion of Tables Sales Data Set';
55 run;

```

The SAS Program Editor extract on the left shows the Data Step used to create the (temporary) tables sales data set (lines 31-42) along with a PROC PRINT step that displays the first 20 rows of the data set in the Output Window and a PROC CONTENTS step that displays the data set's descriptor portion in the Output Window. Looking at the output generated by the PRINT and

CONTENTS "steps" shows what SAS "did" when it took the rows of the raw table sales file and turn them in to observations in our SAS data set: the variables ORDER_DATE and shipment_date are numeric SAS date variables showing the number of days from January 1, 1960 that the table was ordered, and subsequently, shipped.

```
SAS Global Forum 2010
Foundations & Fundamentals
Working With SAS Date and Time Functions
Using INFORMATs: First 20 Observations from the Tables Sales Data Set
```

Obs	order_number	type	material	color	order_date	ship_date	height	price
1	0004682	5	1	1	16530	16591	35	244
2	0005571	6	5	6	16199	16229	25	169
3	0002672	3	4	6	16378	16406	19	1726
4	0004727	5	4	2	16565	16621	37	276
5	0002889	1	3	6	16141	16180	15	379
6	0004876	3	4	2	16496	16539	24	1886
7	0008429	6	3	2	16626	16669	25	169
8	0001377	4	1	1	16554	16563	28	200
9	0004632	6	2	1	16560	16565	29	184
10	0009086	4	4	3	16521	16526	31	299
11	0005547	6	5	1	16505	16513	23	379
12	0009370	2	2	6	16568	16630	24	817
13	0002309	5	5	4	16269	16310	36	230
14	0007028	1	3	7	16403	16405	17	661
15	0009062	4	3	3	16385	16416	35	240
16	0003647	1	3	3	16586	16648	18	641
17	0007003	2	5	3	16364	16417	17	1383
18	0006502	3	2	2	16141	16148	23	579
19	0001394	1	4	6	16447	16504	23	1231
20	0007348	5	5	2	16552	16604	35	312

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
4	color	Num	8
7	height	Num	8
3	material	Num	8
5	order_date	Num	8
1	order_number	Char	7
8	price	Num	8
6	ship_date	Num	8
2	type	Num	8

As mentioned earlier in this paper, there are many good reasons why SAS stores the values of date (and, as we will see later, times) as numbers. But, it's obviously very hard for us to understand the calendar dates associated with these numbers. So, to get us started with the concept of using SAS Formats to control the display of data values in our SAS output, here is a revised PROC PRINT step where the SAS-supplied (or 'internal') Format MMDDYY10 has been applied, or 'associated' the variables ORDER_DATE and SHIP_DATE in the TABLES sales data set. The output generated by this revised PROC PRINT step is also shown below.

```
58 proc print data=tables(obs=20);
59 format order_date ship_date mmddyy10.;
60 title4 'First 20 Observations from the Tables Sales Data Set';
61 title5 'Using SAS Formats to Control Display of Order and Shipment Dates';
62 run;
```

The PROC PRINT step on the left includes a FORMAT Statement that 'associates' the MMDDYY10 Format to the ORDER_DATE and

```
SAS Global Forum 2010
Foundations & Fundamentals
Working With SAS Date and Time Functions
First 20 Observations from the Tables Sales Data Set
Using SAS Formats to Control Display of Order and Shipment Dates
```

Obs	order_number	type	material	color	order_date	ship_date	height	price
1	0004682	5	1	1	04/04/2005	06/04/2005	35	244
2	0005571	6	5	6	05/08/2004	06/07/2004	25	169
3	0002672	3	4	6	11/03/2004	12/01/2004	19	1726
4	0004727	5	4	2	05/09/2005	07/04/2005	37	276
5	0002889	1	3	6	03/11/2004	04/19/2004	15	379
6	0004876	3	4	2	03/01/2005	04/13/2005	24	1886
7	0008429	6	3	2	07/09/2005	08/21/2005	25	169
8	0001377	4	1	1	04/28/2005	05/07/2005	28	200
9	0004632	6	2	1	05/04/2005	05/09/2005	29	184
10	0009086	4	4	3	03/26/2005	03/31/2005	31	299
11	0005547	6	5	1	03/10/2005	03/18/2005	23	379
12	0009370	2	2	6	05/12/2005	07/13/2005	24	817
13	0002309	5	5	4	07/17/2004	08/27/2004	36	230
14	0007028	1	3	7	11/28/2004	11/30/2004	17	661
15	0009062	4	3	3	11/10/2004	12/11/2004	35	240
16	0003647	1	3	3	05/30/2005	07/31/2005	18	641
17	0007003	2	5	3	10/20/2004	12/12/2004	17	1383
18	0006502	3	2	2	03/11/2004	03/18/2004	23	579
19	0001394	1	4	6	01/11/2005	03/09/2005	23	1231
20	0007348	5	5	2	04/26/2005	06/17/2005	35	312

SHIP_DATE variables in our TABLES sales data set.

The result is shown in the Output Window screen capture which is also to the left. Now the "formatted" or "displayed" values of these two variables are easy for us "mere mortals" to comprehend, as we can now see the month, day and year that the table was first ordered and then shipped out.

We will go in to further detail about SAS date, time, and datetime variables and the SAS Format facility in the next section this paper. But, it's important at the outset of this discussion to

understand clearly the difference between the "actual" or "internal" value of a date, time or datetime variable in a SAS data set (a number) and its "displayed" or "formatted" value in our SAS-generated output. The Format we choose (and there are a lot of them for dates and times, as we will discuss below) controls how we "see" these values "outside" of the data set but they do not change their "internal" or "stored" value within the data set itself.

Assigning Values to Date, Time and Datetime Variables from Variable Variables that are "Already" in a SAS Data Set

In the previous example we used INFORMATS in an INPUT Statement to have SAS interpret values of strings in a text file as numeric SAS date variables when the rows in the raw data file were made in to observations in a SAS data set. Now, let's look some basic ways we can create SAS date, time or datetime variables from values stored in the variables/columns of an existing SAS data set.

One common situation is to have a data set with separate columns with values for the month, day and year that some event occurred. And, we might also have separate columns with the hour, minute and seconds that something happened. From these columns (variables) we want to assign the value of a SAS date variable, or a SAS time variable, or a SAS datetime variable to be used in another part of our project. Here's an example: every time a vehicle passes through a bridge's toll plaza the toll booth attendant classifies the type of vehicle by pressing a button on a console in their booth. The classification not only calculates the toll amount due for that vehicle, but also creates an electronic record of the date/time the vehicle passed through the toll plaza and the type of vehicle (passenger car, truck, bus, etc.).

The data set shown to the right, via a PROC PRINT step, has the year, month, day, hour, minute, and second that a vehicle displaying a hazardous materials (HAZMAT) placard (sign) passed through the toll plaza. These crossing are of particular interest to the bridge administration and local enforcement personnel, as extra precautions must be taken to reduce the potential for an accident which would result in release of these hazardous materials in and around the bridge.

So, our challenge is to assign to new variables in our data set values of SAS date, time and datetime variables from the information in the data shown on the right. This can be easily accomplished by the use of SAS programming language Functions designed for exactly this purpose.

In a nutshell, Functions are used to assign values to variables and for other data-related tasks. There are a wide range of Functions available in SAS Software, each of which provides a powerful data processing tool that is probably impossible for you to carry out yourself by writing your own data step code. If you are not already familiar with the concept or application of a SAS Function, see Cody (2004).

In our example, we will apply the MDY Function to assign a SAS date variable value from the values of MONTH, DAY and YEAR, the HMS Function to assign a SAS time value variable from the values of HOUR, MINUTE and SECOND, and the DHMS Function to assign a SAS datetime value variable from the previous-created date variable, and the values of the HOUR MINUTE and SECOND variables.

Here is the Data Step. On line 107 of the Program Editor screen capture the MDY Function is used to assign the value of HAZMAT_CROSS_DATE, and on line 108 the HMS Function is used to assign the value of HAZMAT_CROSS_TIME. Finally, on line 109 the DHMS Function is to assign the value of a SAS datetime variable from the value of HAZMAT_CROSS_DATE (assigned on line 107), and the variables HOUR, MINUTE and SECOND.

```

104 * assign values to new vars using functions;
105 data sgf2010.hazmat_date_and_time;
106   set sgf2010.hazmat_crossings;
107   hazmat_cross_date = mdy(month,day,year);
108   hazmat_cross_time = hms(hour,minute,second);
109   hazmat_cross_dt = dhms(hazmat_cross_date,hour,minute,second);
110 run;

```

Obs	year	month	day	hour	minute	second	hazmat_placard
1	1998	2	19	6	11	2	1
2	1998	3	12	10	20	32	1
3	1998	12	21	11	9	46	1
4	1999	1	18	6	26	30	1
5	1999	6	22	9	31	28	1
6	1999	9	3	16	5	41	1
7	1999	11	14	11	17	31	1
8	2000	3	29	1	25	20	1
9	2000	4	7	23	33	15	1
10	2000	12	28	5	24	20	1
11	2000	12	29	15	6	41	1
12	2001	1	8	21	11	21	1
13	2001	1	24	12	5	31	1
14	2001	4	28	1	28	21	1
15	2001	5	16	21	23	16	1
16	2001	5	27	9	5	8	1
17	2001	7	28	23	8	47	1
18	2001	7	29	7	14	20	1
19	2001	8	13	14	24	37	1
20	2001	8	26	19	10	31	1
21	2001	12	4	7	17	12	1
22	2001	12	15	0	20	25	1
23	2002	6	7	17	6	15	1

This example shows just how easy is it to apply SAS programming language functions to obtain the data values you need for subsequent steps in your project. By using the MDY, HMS and DHMS Functions we now have variables/columns in our data set with the SAS date, time, and datetime values for when the hazmat-placarded vehicle crossed the bridge.

Here are two displays of the data set we just created. The one on the left shows the “actual,” or “internal” values calculated for HAZMAT_CROSS_DATE, HAZMAT_CROSS_TIME and HAZMAT_CROSS_DT while the output on the right shows their displayed values after the WORDDATE, TIMEAMPM and DATETIME19 Formats are associated to them. (To save space, the PROC PRINT-generated output on the right only shows the variables HAZMAT_PLACARD, HAZMAT_CROSS_DATE, HAZMAT_CROSS_TIME and HAZMAT_CROSS_DT.

```

112- proc print data=sgf2010.hazmat_date_and_time NOOBS;
113 title4 'Hazardous Materials Placard Crossings';
114 title5 'Assigning Values to Date, Time and Datetime Variables in a Data Step';
115 run;

```

year	month	day	hour	minute	second	hazmat_placard	hazmat_cross_date	hazmat_cross_time	hazmat_cross_dt
1998	2	19	6	11	2	1	13929	22262	1203487862
1998	3	12	10	20	32	1	13950	37232	1205917232
1998	12	21	11	9	46	1	14234	40186	1229857786
1999	1	18	6	26	30	1	14262	23190	1232259990
1999	6	22	9	31	28	1	14417	34288	1245663098
1999	9	3	16	5	41	1	14490	57941	1251993941
1999	11	14	11	17	31	1	14562	40651	1258197451
2000	3	29	1	25	20	1	14698	5120	1269912320
2000	4	7	23	35	15	1	14707	84795	1270769595
2000	12	28	5	24	20	1	14972	19460	1283600260
2000	12	29	15	6	41	1	14973	54401	1293721601
2001	1	8	21	11	21	1	14983	76281	1294607481
2001	1	24	12	5	31	1	14999	43531	1295957131
2001	4	28	1	28	21	1	15093	5301	1304040501
2001	5	16	21	23	16	1	15111	76996	1305667396
2001	5	27	9	5	8	1	15122	32708	1306573508
2001	7	28	25	8	47	1	15194	35327	1311909227
2001	7	29	7	14	20	1	15185	26050	1312010060
2001	8	13	14	24	37	1	15200	51877	1313331877
2001	8	26	19	10	31	1	15213	69031	1314472231
2001	12	4	7	17	12	1	15313	26232	1323069432
2001	12	15	0	20	25	1	15324	1225	1323994825
2002	6	7	17	6	15	1	15498	61575	1339088775

```

117- proc print data=sgf2010.hazmat_date_and_time NOOBS;
118 var hazmat_placard hazmat_cross_date hazmat_cross_time hazmat_cross_dt;
119 format hazmat_cross_date worddate.
120 hazmat_cross_time timeampm.
121 hazmat_cross_dt datetime19.;
122 title5 'Displaying Created Date, Time and Datetime Variable Values with Formats';
123 run;

```

hazmat_placard	hazmat_cross_date	hazmat_cross_time	hazmat_cross_dt
1	February 19, 1998	6:11:02 AM	19FEB1998:06:11:02
1	March 12, 1998	10:20:32 AM	12MAR1998:10:20:32
1	December 21, 1998	11:09:46 AM	21DEC1998:11:09:46
1	January 18, 1999	6:26:30 AM	18JAN1999:06:26:30
1	June 22, 1999	9:31:28 AM	22JUN1999:09:31:28
1	September 3, 1999	4:05:41 PM	03SEP1999:16:05:41
1	November 14, 1999	11:17:31 AM	14NOV1999:11:17:31
1	March 29, 2000	1:25:20 AM	29MAR2000:01:25:20
1	April 7, 2000	11:33:15 PM	07APR2000:23:33:15
1	December 28, 2000	5:24:20 AM	28DEC2000:05:24:20
1	December 29, 2000	3:06:41 PM	29DEC2000:15:06:41
1	January 8, 2001	9:11:21 PM	08JAN2001:21:11:21
1	January 24, 2001	12:05:31 PM	24JAN2001:12:05:31
1	April 28, 2001	1:28:21 AM	28APR2001:01:28:21
1	May 16, 2001	9:23:16 PM	16MAY2001:21:23:16
1	May 27, 2001	9:05:08 AM	27MAY2001:09:05:08
1	July 28, 2001	11:08:47 PM	28JUL2001:23:08:47
1	July 29, 2001	7:14:20 AM	29JUL2001:07:14:20
1	August 13, 2001	2:24:37 PM	13AUG2001:14:24:37
1	August 26, 2001	7:10:31 PM	26AUG2001:19:10:31
1	December 4, 2001	7:17:12 AM	04DEC2001:07:17:12
1	December 15, 2001	12:20:25 AM	15DEC2001:00:20:25
1	June 7, 2002	5:06:15 PM	07JUN2002:17:06:15

Appendix A to this paper offers several other, “beyond the basics,” examples of assigning SAS date values to variables from date “strings” in an existing SAS data set.

Formats

In the previous examples we saw how SAS Formats are applied to control the display of date values in our output. Taking the time to understand the power of the SAS Format facility, especially with respect to dates and times, will pay off in improving your ability to complete data processing and analysis tasks in less time, and with less frustration, than if you did not know how the Format facility can work for you.

We’ve already seen several examples of using an internal, or SAS-supplied, Format that comes with your SAS Software installation to control the “external representation” or displayed values of the dates on which the table was purchased and subsequently shipped. The important thing to remember at this stage of our discussion is that the Format changes the display of the variable to which it is associated, but it does not change the “internal,” or “actual” value of the variable. Here is an example:

This paper is scheduled for presentation on April 12, 2010 at SAS Global Forum in Seattle, Washington (USA). The SAS date value for April 12, 2010 is 18364. Here is a table that shows how some of the internal SAS date Formats will display that value. Remember, the Format changes how we “see” the value, NOT the value of the variable itself. The table shows just some of the SAS-supplied date values; the SAS documentation has a complete list of all Formats that are included in your SAS installation.

PROC FORMAT can be used to create “customized” VALUE and PICTURE formats for ranges of date values. I’ve included several brief examples of how to do this in Appendix B of the paper. Please see the PROC FORMAT documentation chapter in the BASE SAS Procedures documentation manual and my downloads “My Friend the SAS Format,” “Getting in to the PICTURE Format,” and “SAS Formats: Beyond the Basics,” which are available for free download as PDF’s from the “Free Downloads” link at <http://www.sierrainformation.com> .

Format Applied	Displayed Value	Comments
<i>None</i>	18365	<i>This is the “actual,” or internal value of the a SAS date variable for April 10, 2010</i>
MMDDYY10.	04/12/2010	<i>Month, Day, Year. Slash is default separator</i>
DDMMYY10.	12/04/2010	<i>Day, Month, Year Slash is default separator</i>
WORDDATE.	April 12, 2010	<i>Text for month name, followed by day and year</i>
WEEKDATX.	Monday, 12 April 2010	<i>Text for day of week, day, text for month, year</i>
MONYY.	APR10	<i>Month and two-digit year</i>
MONYY7.	APR2010	<i>Month and four-digit year</i>
WORDDATE.	Monday	<i>Date of week as text</i>
MONTH.	4	<i>Month</i>
QTR.	2	<i>Calendar Quarter</i>
YYQ.	2010Q2	<i>Four-digit year, the letter “Q” followed by the calendar quarter</i>
YEAR.	2010	<i>Four digit year</i>

Using Formats to Group, Summarize and Analyze Date by Dates

One of the many powerful tools we get from combining SAS formats for dates with SAS analytical procedures (PROCs) is that we can group or summarize the PROCs results by the formatted value of a SAS date, time or datetime variable. For example, from our tables sales data set, we want to generate a report that shows the number of sales in each month and year. We can do this very easily using a PROC FREQ step to count the frequency of sales date and associate (for example) the above-described MONYY7 format to the sales date variable. This approach is both easy to implement and very easy to modify, if necessary. Here’s an example:

```

97 proc freq data=sgf2010.tables;
98 tables order_date;
99 format order_date monyy7.;
100 title 'SAS Global Forum 2010';
101 Title2 'Foundations & Fundamentals';
102 title3 'Working With SAS Date and Time Functions';
103 title4 'Grouping Data by the Values of a SAS Date Variable';
104 run;

```

In the PROC FREQ “task” to the left, we are asking for a frequency table of the values of the variable ORDER_DATE formatted with the MONYY7. format (see above). So, what SAS will do is count the number of observations in the data set within the formatted values of

ORDER_DATE. The results are shown on the next page. In just a few lines of code we have exactly the report we’ve been asked to create.

order_date	Frequency	Percent	Cumulative Frequency	Cumulative Percent
MAR2004	2497	5.83	2497	5.83
APR2004	2546	5.94	5043	11.77
MAY2004	2682	6.26	7725	18.03
JUN2004	2576	6.01	10301	24.04
JUL2004	2690	6.28	12991	30.32
AUG2004	2742	6.40	15733	36.72
SEP2004	2611	6.09	18344	42.82
OCT2004	2701	6.30	21045	49.12
NOV2004	2641	6.16	23686	55.29
DEC2004	2705	6.31	26391	61.60
JAN2005	2705	6.31	29096	67.91
FEB2005	2419	5.65	31515	73.56
MAR2005	2712	6.33	34227	79.89
APR2005	2601	6.07	36828	85.96
MAY2005	2647	6.18	39475	92.14
JUN2005	2604	6.08	42079	98.22
JUL2005	764	1.78	42843	100.00

The Format association on line 99 of the PROC FREQ step on the previous page shows just how easy it is to group/summarize your data by the values of a date variable just by selecting the appropriate format for the results you need.

Using the Format meant we avoided an otherwise unnecessary Data Step to create a new variable which would then be supplied to a PROC FREQ step

for analysis. Instead, SAS does all the work for us in one step, via the Format association we applied directly in the PROC FREQ step.

Of course, bosses and clients like to frequently change their minds, so when you're asked "how hard would it be to show me the number of tables ordered grouped by calendar quarter within each year, you can easily create the new report just by changing the Format association. To satisfy this request, all we need to do is associate the YYQ. Format in the PROC FREQ step and re-run it. Here's an example:

```
106 proc freq data=sgf2010.tables;
107 tables order_date;
108 format order_date yyq.;
109 title 'SAS Global Forum 2010';
110 Title2 'Foundations & Fundamentals';
111 title3 'Working With SAS Date and Time Functions';
112 title4 'Grouping Data by the Values of a SAS Date Variable';
113 title5 'Order Count by Year and Calendar Quarter';
114 run;
```

```
SAS Global Forum 2010
Foundations & Fundamentals
Working With SAS Date and Time Functions
Grouping Data by the Values of a SAS Date Variable
Order Count by Year and Calendar Quarter
```

order_date	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2004Q1	2497	5.83	2497	5.83
2004Q2	7804	18.22	10301	24.04
2004Q3	8043	18.77	18344	42.82
2004Q4	8047	18.78	26391	61.60
2005Q1	7836	18.29	34227	79.89
2005Q2	7852	18.33	42079	98.22
2005Q3	764	1.78	42843	100.00

Here's another example of using the formatted value of a date variable to group or classify the results generated by a SAS analytical procedure. Again using the Tables sales data set, let's use PROC MEANS to calculate the average of PRICE (in dollars) and HEIGHT (in inches) of tables sold, by calendar year, for all the observations in the data set. In this example PROC MEANS will display the results of the analyses we request in the Output Window, rounded to the nearest hundredth. (For more information about PROC MEANS, please download free PDFs of my presentations "Steps to Success with PROC MEANS" and "Getting the Most from PROC MEANS" from the "Free Downloads" link at <http://www.Sierrainformation.com>)

```
79 proc means data=sgf2010.tables maxdec = 2 mean;
80 var price height;
81 class order_date;
82 format order_date year.;
83 title5 'Mean Price and Height by Year of Order Date';
84 run;
```

```
SAS Global Forum 2010
Foundations & Fundamentals
Working With SAS Date and Time Functions
Hazardous Materials Placard Crossings
Mean Price and Height by Year of Order Date
```

The MEANS Procedure

order_date	N Obs	Variable	Mean
2004	26391	price	513.10
		height	24.81
2005	16452	price	511.67
		height	24.86

Using a SAS Format on the value of our CLASS statement variable made it very easy to obtain the analyses we wanted of the numeric variables PRICE and HEIGHT "grouped," or "classified" by calendar year.

Calculating Time Elapsed Between Two Events

Another common task is to determine the number of time periods between two events. With our Tables sales data set, an obvious requirements would be to figure out how many days elapsed between when the customer ordered the table and when it was shipped out. Since SAS stores the values of date variables as numbers, it's very easy to compute this value in a Data Step. All we need to do is subtract shipment_date from ORDER_DATE and we obtain a useful result. Here's an example, from the first 15 rows/observations in the tables sales SAS data set.

```

132= data example1;
133   set sgf2010.tables(obs=15);
134 * calculate the number of days between
135   order and shipment;
136 days = ship_date - order_date;
137 run;
138
139= proc print data=example1 label split = '/';
140 var order_number order_date ship_date days;
141 label days = 'Days Between/Order and/Shipment';
142 format order_date ship_date mmdyy10.;
143 title4 'Example 1:';
144 title5 'Calculating Time Periods Between Two Events';
145 run;

```

Remember, the “internal,” or “stored” value of SHIP_DATE and ORDER_DATE are numbers representing the number of days from January 1, 2010 to the date the table was ordered, and, later, shipped out to the customer. So, in the Data Step above, all we needed to do to find the number of calendar days between the two events was to subtract the value of SHIP_DATE from the value of ORDER_DATE in each row/observation of the data set (or, for our testing purposes, just the first 15 rows using the OBS= SAS Data Set Option).

Obs	order_number	order_date	ship_date	Days Between Order and Shipment
1	0004682	04/04/2005	06/04/2005	61
2	0005571	05/08/2004	06/07/2004	30
3	0002672	11/03/2004	12/01/2004	28
4	0004727	05/09/2005	07/04/2005	56
5	0002889	03/11/2004	04/19/2004	39
6	0004876	03/01/2005	04/13/2005	43
7	0008429	07/09/2005	08/21/2005	43
8	0001377	04/28/2005	05/07/2005	9
9	0004632	05/04/2005	05/09/2005	5
10	0009086	03/26/2005	03/31/2005	5
11	0005547	03/10/2005	03/18/2005	8
12	0009370	05/12/2005	07/13/2005	62
13	0002309	07/17/2004	08/27/2004	41
14	0007028	11/28/2004	11/30/2004	2
15	0009062	11/10/2004	12/11/2004	31

In some analytical situations we may want to give “credit” for the “day” on which the first of the two events being compared in the calculation. This is often the case in analyzing hospital patient discharge data. If the patient is admitted to, and then discharged from, the hospital on the same day (perhaps because they had a ‘same day’ procedure performed or only spent a short time in the emergency room), their discharge and admission dates are equal. So, if we subtracted one from the other we’d have zero (0) as the result, which is not very useful. In these situations, analysts often add the constant one (1) to the result of the subtraction when computed the number of calendar days between two days.

This approach is shown in the Data Step code and PROC PRINT and PROC PRINT output below. If the customer orders a table that is shipped out on the same day, we want our computed value of DAYS to be one (1) and not zero (0).

```

147= data example2;
148   set sgf2010.tables(obs=15);
149 * calculate the number of days between
150   order and shipment, add one day for same
151   day shipments;
152 days = (ship_date - order_date) + 1;
153 run;
154
155= proc print data=example2 label split=' /';
156 var order_number order_date ship_date days;
157 label days = 'Days Between/Order and/Shipment';
158 format order_date ship_date mmdyy10.;
159 title4 'Example 2:';
160 title5 'Calculating Time Periods Between Two Events';
161 run;

```

There is no “one right way” to perform this calculation. Choose the one that’s “right” for the specific data you are analyzing. Either approach is easy to implement in your Data Steps.

Obs	order_number	order_date	ship_date	Days Between Order and Shipment
1	0004682	04/04/2005	06/04/2005	62
2	0005571	05/08/2004	06/07/2004	31
3	0002672	11/03/2004	12/01/2004	29
4	0004727	05/09/2005	07/04/2005	57
5	0002889	03/11/2004	04/19/2004	40
6	0004876	03/01/2005	04/13/2005	44
7	0008429	07/09/2005	08/21/2005	44
8	0001377	04/28/2005	05/07/2005	10
9	0004632	05/04/2005	05/09/2005	6
10	0009086	03/26/2005	03/31/2005	6
11	0005547	03/10/2005	03/18/2005	9
12	0009370	05/12/2005	07/13/2005	63
13	0002309	07/17/2004	08/27/2004	42
14	0007028	11/28/2004	11/30/2004	3
15	0009062	11/10/2004	12/11/2004	32

Another common calculation we may need to perform is to determine the number of weekdays between two dates. For example, the manufacturing process Tables Company may be closed on Saturdays and/or Sundays (but the order and shipping departments are still open), so we want to exclude any Saturdays or Sundays that fall between ORDER_DATE and SHIP_DATE to obtain a more accurate understanding of how long it takes to complete order processing.

Fortunately, there is a very powerful SAS Programming Language Function, INTCK, (“Interval Check”) that can do this calculation for us very quickly and easily. Calculating this value without it would be very difficult!

Counting the Number of “Interval Boundaries” Between Two Dates with the INTCK Function

The INTCK Function is a very handy tool for counting the number of “interval boundaries” between two SAS date, time and datetime variables. While it is very easy to use, you must clearly understand what it does—and does not do—in order to obtain the results you need from it. This function has three “required arguments,” or three pieces of information you have to give it in order to have it give you back (or “return”) the value you want from it. The three arguments are: **interval**, **from**, and **to**, where:

- **Interval** is the time interval (e.g., WEEKDAY, MONTH, YEAR, HOUR) you want it to count
- **From** is a SAS date, time or datetime variable name, or a date, time or datetime constant (see below) that is the start of the period
- **To** is a SAS date, time or datetime variable name, or a date, time or datetime constant (see below) that is the end of the period

Here are some examples:

- *Counting Weekday Intervals (Default Exclusion of Saturday and Sundays): Number of weekdays between ORDER_DATE and SHIP_DATE for the first 15 observations in the Tables data set.*

```
163= data weekdays1;
164   set sgf2010.tables(obs=15);
165 weekdays1 = intck('weekdays',order_date,ship_date);
166 run;
167
168=proc print data=weekdays1 label split='/';
169 var order_number order_date ship_date weekdays1;
170 format order_date ship_date weekdate.;
171 label weekdays1 = 'Number of/Week Days/Between/Order & Shipment';
172 title4 'Example 1: Using the INTCK Function';
173 title5 'Number of Weekdays Between Table Order and Shipment';
174 title6 'Excluding Saturdays and Sundays (Default)';
175 run;
```

In this example, the INTCK function counted the number of WEEKDAY intervals between order and shipment, that is, the number of days between the two events EXCLUDING any day that was a Saturday or a Sunday.

- *Counting Weekday Intervals (Exclude Sundays ONLY): Number of week days between ORDER_DATE and SHIP_DATE, excluding Sundays, for the first 15 observations in the Table data set.*

In this example the “interval argument” to the INTCK Function has been modified to read “weekday1w,” which instructs SAS to only exclude Sundays when calculating the number of weekdays between the dates supplied in the FROM Days and TO arguments. Please read more about the WEEKDAY SAS Programming Language Function, below, to understand why the number one (1) in SAS refers to Sunday.

Obs	order_number	order_date	ship_date	Number of Week Days Between Order & Shipment
1	0004682	Monday, April 4, 2005	Saturday, June 4, 2005	44
2	0005571	Saturday, May 8, 2004	Monday, June 7, 2004	21
3	0002672	Wednesday, November 3, 2004	Wednesday, December 1, 2004	20
4	0004727	Monday, May 9, 2005	Monday, July 4, 2005	40
5	0002899	Thursday, March 11, 2004	Monday, April 19, 2004	27
6	0004876	Tuesday, March 1, 2005	Wednesday, April 13, 2005	31
7	0008429	Saturday, July 9, 2005	Sunday, August 21, 2005	30
8	0001377	Thursday, April 28, 2005	Saturday, May 7, 2005	6
9	0004632	Wednesday, May 4, 2005	Monday, May 9, 2005	3
10	0009086	Saturday, March 26, 2005	Thursday, March 31, 2005	4
11	0005547	Thursday, March 10, 2005	Friday, March 18, 2005	6
12	0009370	Thursday, May 12, 2005	Wednesday, July 13, 2005	44
13	0002309	Saturday, July 17, 2004	Friday, August 27, 2004	30
14	0007028	Sunday, November 28, 2004	Tuesday, November 30, 2004	2
15	0009062	Wednesday, November 10, 2004	Saturday, December 11, 2004	22

```

177 data weekdays2;
178   set sgf2010.tables(obs=15);
179 weekdays2 = intck('weekdays1w',order_date,ship_date);
180 run;
181
182 proc print data=weekdays2 label split=' /';
183 label weekdays2 = 'Number of /Days Between/Order & Shipment/Excluding Sundays';
184 format order_date ship_date weekdate.;
185 var order_number order_date ship_date weekdays2;
186 title4 'Example 1: Using the INTCK Function';
187 title5 'Number of Weekdays Between Table and Order Shipment';
188 title6 'Excluding Sundays ONLY';
189 run;

```

```

SAS Global Forum 2010
Foundations & Fundamentals
Working With SAS Date and Time Functions
Example 2: Using the INTCK Function
Number of Weekdays Between Table and Order Shipment
Excluding Sundays ONLY

```

Obs	order_number	order_date	ship_date	Number of Days Between Order & Shipment Excluding Sundays
1	0004682	Monday, April 4, 2005	Saturday, June 4, 2005	53
2	0005971	Saturday, May 8, 2004	Monday, June 7, 2004	25
3	0002672	Wednesday, November 3, 2004	Wednesday, December 1, 2004	24
4	0004727	Monday, May 9, 2005	Monday, July 4, 2005	48
5	0002889	Thursday, March 11, 2004	Monday, April 19, 2004	33
6	0004876	Tuesday, March 1, 2005	Wednesday, April 13, 2005	37
7	0008429	Saturday, July 9, 2005	Sunday, August 21, 2005	36
8	0001377	Thursday, April 28, 2005	Saturday, May 7, 2005	8
9	0004852	Wednesday, May 4, 2005	Monday, May 9, 2005	4
10	0009086	Saturday, March 26, 2005	Thursday, March 31, 2005	4
11	0005547	Thursday, March 10, 2005	Friday, March 18, 2005	7
12	0009370	Thursday, May 12, 2005	Wednesday, July 13, 2005	53
13	0002309	Saturday, July 17, 2004	Friday, August 27, 2004	35
14	0007028	Sunday, November 28, 2004	Tuesday, November 30, 2004	2
15	0009062	Wednesday, November 10, 2004	Saturday, December 11, 2004	27

Understanding the Difference Between What INTCK Calculates and What Is Obtained Via Subtraction of Date, Time or Datetime Variable Values

One common “gotcha moment” experienced by newer users of SAS’ date and time processing tools occurs when they fail to understand the important difference between what the INTCK function “returns” and the results they receive when subtracting the value of one date, time or datetime variable from another. Let’s look at a couple of examples so that you can hopefully avoid misunderstanding these important differences when you apply these tools in your own projects and programs.

INTCK counts the number of specified “interval boundaries” (that is, what you supply in the first argument) between the date, time or datetime variables provided in the second and third arguments to the function. A very obvious example is when you are trying to calculate the age of an infant when it is discharged from the hospital. If the baby was born (and therefore ‘admitted’ on December 30, 2009) and discharged on January 2, 2010), it’s easy to agree that the little bundle of joy is four days old, assuming we agree that we are counting the number of “whole days” starting on the birth day and ending on the discharge day.

If, however, we tried to calculate age using an INTCK Function call with the YEAR interval boundary as the first argument, the value returned by the function would be “1” (one). The baby is really only four days old, not one year “old,” as suggested by what the INTCK Function is telling us. Is the INTCK Function “wrong”? No. What the INTCK Function is telling us is that one instance of the YEAR “interval boundary” has been “crossed” between the baby’s day of birth and its subsequent day of discharge.

This is one reason why we need to be very careful in using SAS programming language and other tools to accurately calculate ages and other differences between two dates. An excellent resource for more details on various approaches to this problem is Billy Kreuter’s 1998 *SAS Communications* article “*Accurately Calculating Age in One Line*,” which is available online as a PDF at http://staff.washington.edu/billyk/TechTips_SC4Q98.pdf.

- *Using the INTCK Function with Datetime Variables*

The following examples expand on the hazardous materials vehicle crossing example data set used above, and show how to accurately use the INTCK Function to calculate time intervals between two datetime variables. Let’s assume that a truck with a hazardous materials placard, after paying the computed toll, is then diverted to an inspection area where bridge police check the driver’s paperwork, the rigging of the load, and for other safety factors. Then, two police cars are detailed to provide a “front and back escort” of the truck as it crosses the bridge. We have another data set that has three datetime variables in it: 1) date and time the truck crossed the toll booth; 2) date and time the truck inspection started; and 3) date and time the truck inspection ended and the escort has been arranged. The bridge district’s directors have been receiving complains from hazardous materials shippers alleging excessive inspection time delays, so we need to analyze the data in SFG2010.HAZMAT_INSPECTION_TIMES to get understand the problem. Here is a PROC-PRINT generated display of the data set’s observations and variables:

```

Foundations & Fundamentals
Working with SAS Date and Time Functions
More on the INTCK Function
Hazmat Inspection Start/End Times

```

Crossing ID	hazmat_placard	Date & Time Toll Booth Crossed	Date & Time Inspection START	Date & Time Inspection END
1	1	19FEB1998:06:11:02	19FEB1998:06:24:00	19FEB1998:06:29:35
2	1	12MAR1998:10:20:32	12MAR1998:10:27:36	12MAR1998:10:33:38
3	1	21DEC1998:11:09:46	21DEC1998:11:21:07	21DEC1998:11:30:49
4	1	18JAN1999:06:26:30	18JAN1999:06:33:46	18JAN1999:06:40:15
5	1	22JUN1999:09:31:28	22JUN1999:09:43:52	22JUN1999:09:57:44
6	1	03SEP1999:16:05:41	03SEP1999:16:14:00	03SEP1999:16:24:32
7	1	14NOV1999:11:17:31	14NOV1999:11:26:32	14NOV1999:11:32:05
8	1	29MAR2000:01:25:20	29MAR2000:01:26:27	29MAR2000:01:48:56
9	1	07APR2000:23:33:15	07APR2000:23:38:54	08APR2000:00:28:07
10	1	28DEC2000:05:24:20	28DEC2000:05:37:28	28DEC2000:05:46:34
11	1	29DEC2000:15:06:41	29DEC2000:15:15:49	29DEC2000:15:26:27
12	1	08JAN2001:21:11:21	08JAN2001:21:27:57	08JAN2001:21:51:21
13	1	24JAN2001:12:05:31	24JAN2001:12:10:45	24JAN2001:12:16:50
14	1	28APR2001:01:28:21	28APR2001:01:43:11	28APR2001:02:03:01
15	1	16MAY2001:21:23:16	16MAY2001:21:39:55	16MAY2001:21:56:45
16	1	27MAY2001:09:05:08	27MAY2001:09:15:41	27MAY2001:09:20:47
17	1	28JUL2001:23:08:47	28JUL2001:23:13:29	29JUL2001:00:01:38
18	1	29JUL2001:07:14:20	29JUL2001:07:15:31	29JUL2001:07:38:38
19	1	13AUG2001:14:24:37	13AUG2001:14:35:28	13AUG2001:14:59:16
20	1	26AUG2001:19:10:31	26AUG2001:19:24:37	26AUG2001:19:47:37
21	1	04DEC2001:07:17:12	04DEC2001:07:23:12	04DEC2001:07:31:17
22	1	15DEC2001:00:20:25	15DEC2001:00:28:36	15DEC2001:00:46:58
23	1	07JUN2002:17:06:15	07JUN2002:17:15:45	07JUN2002:17:23:25

We might be tempted to use the INTCK FUNCTION with the DAY or HOUR “interval boundary arguments to analyze these data. If we did, SAS would gladly calculate values for us and execute our Data Steps without errors (assuming, of course, we didn’t make any syntax or other programming mistakes.

Here’s an example. Suppose we want to find out the number of HOURS and DAYS that elapsed between toll booth crossing and end of the inspection. Let’s execute the data step below and see what it it does for us.

```

211 data days_and_hours;
212 set sgf2010.hazmat_inspection_times;
213 hours = intck('hours', hazmat_cross_dt, hazmat_inspect_end_dt);
214 days = intck('day', datepart(hazmat_cross_dt), datepart(hazmat_inspect_end_dt));
215 run;
216
217 proc print data=days_and_hours (firstobs=7 obs=18) label split = '/';
218 title5 'Days and Hours Between Toll Booth Crossing and Hazmat Inspection End';
219 title6 'Via the INTCK Function';
220 run;

```

Take a look at line 214 of the Data Step shown on the left. The DATEPART function has been used to extract the SAS date value “part” from the SAS datetime variable’s value *before* having the INTCK function determine the number of DAY interval boundaries between

when the truck crossed the toll plaza and when inspection ends. We’ll take a deeper look at using SAS Functions to “find” or “extract” the “parts” of a SAS date, time or datetime variable in the next section of this paper. In the meantime, here is the PROC PRINT output generated by the step on lines 217-220 in the above SAS Enhanced Program Editor screen capture:

Using the FIRSTOBS and OBS options in the PROC PRINT step (see line 217 of the Program Editor screen capture) we can focus in on the problematic results generated by trying to use the INTCK function to solve our problem. It looks as if Observations 9 and 17 “waiting a day” until inspection completed and Observation 14 “waited an hour” for its inspection to conclude. While these are accurate values as computed by the INTCK Function, the do not tell the “whole story.”

For example, for Observation 9, the truck crossed the toll plaza at 11:33:15 pm and its inspection started less than seven minutes later, at 11:38:54 pm. Inspection completed at 12:28:07 AM (or “zero-hundred hours, 28 minutes, seven seconds”). Two INTCK Function “boundaries” were “crossed” between these two events: HOUR and DAY. So, in the first INTCK Function call on line 213, the Function accurately returned the value one (“1”) as the number of interval boundaries “crossed” between the two events. And, since the DAY interval boundary was also “crossed” between the two events, the second INTCK Function call, on line 218, correctly returned the value one (“1”) as the number of day interval boundaries between the two events. But, less than an hour of time elapsed between toll booth crossing and end of inspection.

In my opinion, a more accurate representation of the delay time between toll booth crossing and inspection start, between inspection start and inspection end, and between toll booth crossing and

inspection end is obtained by: 1) remembering that SAS datetime variables are store the number of second before/after midnight January 1, 1960 as integers; and 2) subtracting the value of one datetime variable from another gives to obtain the number of seconds between the two events. With that value “in hand,” (or more accurately, “in the SAS Program Data Vector,”) we can then add some more code to our data step to calculate the number of minutes between these events. Here’s how I did it:

```

239 data days_and_hours3;
240   set sgf2010.hazmat_inspection_times;
241   between_cross_and_start = round(((hazmat_inspect_start_dt - hazmat_cross_dt)/60),.01);
242   between_start_and_end = round(((hazmat_inspect_end_dt - hazmat_inspect_start_dt)/60),.01);
243   between_cross_and_end = round(((hazmat_inspect_end_dt - hazmat_cross_dt)/60),.01);
244   label   between_cross_and_start = 'Mins Betwn/Toll Booth/Cross and/Insp. Start'
245           between_start_and_end = 'Mins Betwn/Insp. Start/and/Insp. End'
246           between_cross_and_end = 'Mins Btwn/Toll Booth/Cross and/Insp. End';
247   run;
248
249 options orientation = landscape;
250 proc print data=days_and_hours3(firstobs=7 obs=18 drop=hazmat_placard) label split = '/';
251 title5 'Days and Hours Between Toll Booth Crossing and Hazmat Inspection End';
252 title6 'Via Direct Calculation';
253 run;

```

In the Data Step to the left, I subtracted one datetime value from the next and then divided the result by 60 to go from seconds of elapsed time to minutes. Then, the ROUND Function was used to round the results to the nearest hundredth. The resulting data set looks is shown below:

Obs	Date & Time Toll Booth Crossed	Date & Time Inspection START	Date & Time Inspection END	Mins Betwn Toll Booth Cross and Insp. Start	Mins Betwn Insp. Start and Insp. End	Mins Btwn Toll Booth Cross and Insp. End
7	14NOV1999:11:17:31	14NOV1999:11:26:32	14NOV1999:11:32:05	9.02	5.55	14.57
8	29MAR2000:01:25:20	29MAR2000:01:26:27	29MAR2000:01:48:56	1.12	22.48	23.60
9	07APR2000:23:33:15	07APR2000:23:38:54	08APR2000:00:28:07	5.65	49.22	54.87
10	28DEC2000:05:24:20	28DEC2000:05:37:28	28DEC2000:05:46:34	13.13	9.10	22.23
11	29DEC2000:15:06:41	29DEC2000:15:15:49	29DEC2000:15:26:27	9.13	10.63	19.77
12	08JAN2001:21:11:21	08JAN2001:21:27:57	08JAN2001:21:51:21	16.60	23.40	40.00
13	24JAN2001:12:05:31	24JAN2001:12:10:45	24JAN2001:12:16:50	5.23	6.08	11.32
14	28APR2001:01:28:21	28APR2001:01:43:11	28APR2001:02:03:01	14.83	19.83	34.67
15	16MAY2001:21:23:16	16MAY2001:21:39:55	16MAY2001:21:56:45	16.65	16.83	33.48
16	27MAY2001:09:05:08	27MAY2001:09:15:41	27MAY2001:09:20:47	10.55	5.10	15.65
17	28JUL2001:23:08:47	28JUL2001:23:13:29	29JUL2001:00:01:38	4.70	48.15	52.85
18	29JUL2001:07:14:20	29JUL2001:07:15:31	29JUL2001:07:38:38	1.18	23.12	24.30

The results shown on the left offer a more accurate and informative analysis of delay times for the hazardous materials crossing our bridge than we obtained from using the INTCK Function

Observation #9, which originally “seemed” to have a delay time of one “day,” in fact had a

total elapsed delay time of 54.87 minutes. And, Observation #17’s delay time was “only” 52.85 minutes, not one “day.” By carefully understanding the benefits, and limitations of the INTCK Function we can make appropriate use of it under the right circumstances and apply other methods, such as direct calculation of the differences between the values of two SAS date, time or datetime variables, as in the previous example, when doing so gives us both the “right” and “correct” answer.

Working with SAS Date Functions

In the previous example the DATEPART Function was used to extract the “date part”, or the number of days before or after January 1, 1960, from a datetime variable. Now, let’s take a deeper look at this, and other programming language Functions we can use to “return the parts” of a SAS date, time or datetime variable. Here’s a list of some of the most common functions, and a brief description of what they can do for us:

- Functions that “extract the parts” from a SAS date, time or datetime variable

Function Name	Applied to....	What It Returns....
DATEPART	Datetime Variables	SAS date value from a SAS datetime variable
TIMEPART	Datetime Variables	SAS time value from a SAS datetime variable
MONTH	Date Variables	Month of year (1=Jan.) from a SAS date variable
DAY	Date Variables	Day of month from a SAS date variable
YEAR	Date Variables	Four-digit year from a SAS date variable
QTR	Date Variables	Calendar quarter from a SAS date variable
HOURL	Time Variables	Hour (24-hour clock) from a SAS time variable
MINUTE	Time Variables	Minute from a SAS time variable
SECOND	Time Variables	Second from a SAS time variable
WEEKDAY	Date Variables	Day of the Week (1=Monday) from a SAS date variable
WEEK (new to SAS 9.1)	Date Variables	Returns the week number. See BASE SAS Language: Reference for default and optional calculations for this Function

Selecting Observations/Rows from an Existing Data Set Using SAS Functions Extracting the “parts” from a SAS Date, Time or Datetime Variable

The Functions listed above are often used to select, or “pull,” rows in a data set in to a new data set based on the values of date, time or datetime variables. Of these, extra should be taken with the WEEKDAY Function to ensure that you obtain the desired results. Most SAS users apply these functions in either a Data Step “Subsetting IF” Statement, in a WHERE Statement or in a WHERE Clause Data Set Option, which can be applied in both Data and Procedure Steps that read rows/observations from a SAS data set. Here are some examples:

- From the TABLES Data Set, Select Observations Where the ORDER_DATE Occurred in the Fourth Quarter of 2004 and Place Them in a Temporary SAS Data Set

```
260=Data Q1_2004;
261 set sgf2010.tables;
262 if year(order_date) = 2004 and qtr(order_date) = 4;
263 run;
```

```
2178 Data Q1_2004;
2179 set sgf2010.tables;
2180 if year(order_date) = 2004 and qtr(order_date) = 4;
2181 run;
NOTE: There were 42843 observations read from the data set SGF2010.TABLES.
NOTE: The data set WORK.Q1_2004 has 8047 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time      0.70 seconds
      cpu time       0.07 seconds
```

- Using PROC FREQ on the TABLES Data Set, determine the frequency of the values of COLOR for tables ordered in Fourth Quarter of 2004. Select rows/observations to be analyzed using a WHERE Clause Data Set Option in the PROC FREQ “step.”

```
264
265=proc freq data=sgf2010.tables;
266 where year(order_date) = 2004 and qtr(order_date) = 4;
267 tables color;
268 title3 'Selecting Rows for Analysis Using the WHERE Statement and';
269 title4 'Date Functions';
270 title5 '*****';
271 title6 'Frequency of Color for Tables Ordered in the 4th Quarter of 2004';
272 run;
```

```
SAS Global Forum 2010
Foundations & Fundamentals
Selecting Rows for Analysis Using the WHERE Statement and
Date Functions
*****
Frequency of Color for Tables Ordered in the 4th Quarter of 2004

The FREQ Procedure

color    Frequency    Percent    Cumulative    Cumulative
         Frequency    Frequency    Frequency    Percent
-----
1         1189         14.70         1189         14.70
2         1121         13.93         2304         28.63
3         1146         14.24         3450         42.87
4         1164         14.47         4614         57.34
5         1141         14.18         5755         71.52
6         1116         13.87         6871         85.39
7         1176         14.61         8047         100.00
```

- From the BYHOUR2 Data Set (bridge crossings by hour starting midnight January 1, 1998 and ending at 11 pm on December 31, 2002), put in a temporary SAS data set hourly observations that meet the following conditions: January 2001, between 6 am and 10 am (i.e., the “rush hour”) on a Monday, Tuesday, Wednesday, Thursday or Friday.

```

274 data rush_hour_Jan_2001;
275   set sgf2010.byhour2;
276   * note: date is a sas datetime variable;
277   if year(datepart(date)) = 2001
278       and month(datepart(date)) = 1
279       and (2 <= weekday(datepart(date)) <= 6)
280       and (6 <= hour(timepart(date)) <= 10);
281   run;

```

```

2303 data rush_hour_Jan_2001;
2304   set sgf2010.byhour2;
2305   * note: date is a sas datetime variable;
2306   if year(datepart(date)) = 2001
2307       and month(datepart(date)) = 1
2308       and (2 <= weekday(datepart(date)) <= 6)
2309       and (6 <= hour(timepart(date)) <= 10);
2310   run;

NOTE: There were 43824 observations read from the data set $0F2010.BYHOUR2.
NOTE: The data set WORK.RUSH_HOUR_JAN_2001 has 115 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time        0.10 seconds
      cpu time         0.06 seconds

```

- Repeat the analysis above, but use a WHERE Clause Data Set Option in the PROC MEANS Step rather than first creating a subset SAS data set with the observations needed to conduct the analysis.

```

284 proc means data=sgf2010.byhour2(where=(year(datepart(date)) = 2001
285       and month(datepart(date)) = 1
286       and (2 <= weekday(datepart(date)) <= 6)
287       and (6 <= hour(timepart(date)) <= 10)));
288   n sum max mean min maxdec=0;
289 var car_count;
290 title3 'Selecting Observations for Analysis by PROC MEANS Using a';
291 title4 'WHERE Clause SAS Data Set Option';
292 title5 'Analysis of Vehicle Crossings During All Weekday Rush Hours';
293 title6 'Monday Through Friday in January 2001';
294 run;

```

```

SAS Global Forum 2010
Foundations & Fundamentals
Selecting Observations for Analysis by PROC MEANS Using a
WHERE Clause SAS Data Set Option
Analysis of Vehicle Crossings During All Weekday Rush Hours
Monday Through Friday in January 2001

```

The MEANS Procedure

Analysis Variable : car_count				
N	Sum	Maximum	Mean	Minimum
115	11753	128	102	74

Selecting Observations/Rows with a SAS Date Constant

The examples above give you a idea of the power and flexibility of SAS Functions that return “parts” of values to date, time and datetime variables. But, what do we do when we need to give SAS a specific date, or just a few dates, to select the observations/rows we want to analyze. As we’ve seen in previous examples, SAS stores the values of date, time and datetime variables as numeric variables. So, if we have a specific date we need to have SAS test for us, we need to know the numeric value of the date (or time, or datetime) value we want it to test...or, do we?

For example, suppose several customers at the Table Company whose orders were shipped on July 28, 2005 have called to complain about the quality of their table. Management is concerned that all tables that were shipped on that date might need to be recalled, at company expense, and replaced with new tables. How many tables were shipped on July 28, 2005 and what was the sum (total) of prices for these tables? With this information in hand management can start to assess the costs of implementing a return and replacement of all tables shipped to customers on that date, if need be.

In order to do this we have to select rows from the SGF2010.TABLES SAS data set for which the value of the SAS date variable SHIP_DATE is July 28, 2005. Can we do this WITHOUT somehow trying to figure out ourselves what the actual SAS date value (that is, the number of days from January 1, 1960 to July 29, 2006) is for that variable? As it turns out, we can specify a SAS Date Constant in a Data Step, WHERE Statement or WHERE Clause SAS Data Set Option and avoid the otherwise tedious task of calculating a SAS date, time or datetime value ourselves.

Here’s an example of how a SAS Date Constant would be used in 1) in a Data Step “Subsetting IF Statement;” 2) in a WHERE Statement in a PROC MEANS Step; and, 3) in a WHERE Clause SAS Data

Set Option in PROC MEANS Step. Note that when specifying our Date Constant we have to use a two digit year, three-character month abbreviation and then the year as either a two- or four-digit value. Using a four-digit year value is recommended (see our discussion of the YEARCUTOFF SAS System Option below for more details).

- *Supplying a SAS Date Constant in a Data Step “Subsetting IF” Statement*

```
302 * using a SAS Date Constant in a Data Step 'Subsetting If';
303 data july_28_2005;
304 set sgf2010.tables;
305 if ship_date = '28jul2005'D;
306 run;
```

```
2383 * using a SAS Date Constant in a Data Step 'Subsetting If';
2384 data july_28_2005;
2385 set sgf2010.tables;
2386 if ship_date = '28jul2005'D;
2387 run;

NOTE: There were 42843 observations read from the data set SGF2010.TABLES.
NOTE: The data set WORK.JULY_28_2005 has 77 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time           0.09 seconds
      cpu time            0.06 seconds
```

- *Using a SAS Date Constant in a WHERE Statement*

```
308 proc means data=sgf2010.tables
309           n mean sum max min maxdec=0;
310 where ship_date = '28jul2005'D;
311 var price;
312 title4 'Using a SAS Date Constant';
313 title5 'Number of Tables Sold and Total Sale Price';
314 title6 'For Tables Shipped on July 28, 2005';
315 run;
```

```
SAS Global Forum 2010
Foundations & Fundamentals
Freq of Ship Date
Using a SAS Date Constant
Number of Tables Sold and Total Sale Price
For Tables Shipped on July 28, 2005

The MEANS Procedure

Analysis Variable : price

N          Mean          Sum          Maximum          Minimum
-----
77          563          43343          1764          169
```

- *Using a SAS Date Constant in a WHERE Clause SAS Data Set Option*

```
317 proc means data=sgf2010.tables(where=(ship_date = '28jul2005'D))
318           n mean sum max min maxdec=0;
319 var price;
320 title4 'Using a SAS Date Constant';
321 title5 'Number of Tables Sold and Total Sale Price';
322 title6 'For Tables Shipped on July 28, 2005';
323 run;
```

```
SAS Global Forum 2010
Foundations & Fundamentals
Freq of Ship Date
Using a SAS Date Constant
Number of Tables Sold and Total Sale Price
For Tables Shipped on July 28, 2005

The MEANS Procedure

Analysis Variable : price

N          Mean          Sum          Maximum          Minimum
-----
77          563          43343          1764          169
```

Notice that we obtained the same results from both PROC MEANS steps. The only difference was how we chose the rows we wanted the PROC to analyze for us. You can use either a WHERE Statement or WHERE Clause SAS Data Set option. With extremely large files, using it as a Data Set Option tends to be more resource-efficient and reduce processing time that applying it in a WHERE Statement

SAS System Options for Date Processing

System Options control various aspects of how the SAS System runs your programs and displays your output. For newer users of SAS, understanding three of these Options can help address common issues and problems with dates. See the chapter in the SAS Language: Reference documentation manual (bibliographic citation below) for more information on all System Options and BASE SAS Procedures documentation manual (bibliographic citation below) for details on PROC OPTIONS.

The three options we will discuss in this paper are: DATE/NODATE, NODTRESET/DTRESET and YEARCUTOFF. Here's a bit about each of them:

- **DATE:** By default, the value of this option is DATE, which means SAS will display the date and time of you started your SAS session in the upper right-hand corner of your output. Using an OPTIONS statement to change this Option value to NODATE suppresses display of the date and time of session initialization in your output. All the SAS output examples in this paper were generated with the NODATE option in effect.
- **NODTRESET:** The default value of this Option is NODTRESET, which means that the date and time value displayed in the upper right-hand corner of your output will *a/ways* be the date and time of SAS System initialization. Changing this to DTRESET (new in SAS 9.0 and above) instructs SAS to re-set the date and time displayed in your output immediately before a page is displayed. This option is very useful if you have very jobs that run for very long times, especially those which run overnight.
- **YEARCUTOFF:** This option assigns to a century the value of two-digit years. The default value, starting in SAS Version 8, is 1920. This means that any two-digit year values from “00” to “19” will be interpreted as 2000 to 2019 and all two-digit year values from “20” to “99” will be interpreted as 1920 to 1999. You can change the default value of the Option via an OPTIONS Statement, but extreme caution is suggested in doing so. One easy way to avoid any potential problems with two-digit year values and SAS Software is to take the time to code all four digits of your year values. Otherwise, you should become very good friends with the YEARCUTOFF SAS System Option.

Summary and Conclusions

The SAS System offers a wide range of tools to work effectively with the values of date and time variables. Since SAS stores these values as numeric variables in your data sets, the use of Formats with your date, time and datetime variables is usually required. By mastering a few key concepts of how SAS works with dates and times you can successfully use these tools in your own projects and programs. This paper has identified some of the core concepts and tools for the working with dates and times in SAS; as you become more proficient with what has been discussed in this paper you will want to learn about other SAS System tools for dates and times by reading other papers on this topic and by consulting Morgan (2006) and the SAS documentation manuals.

Acknowledgments and Thanks:

Attendees at previous presentations I have given on this topic at SGF and other events for SAS users have made excellent suggestions and posed helpful questions that have hopefully helped me improve it over time. Also, I'd like to thank David Dickey, Ph.D., Professor of Statistics at NC State University, for the insights he has shared with me over the years on time series analysis and SAS' implementation of time series analytic tools. And, Philip W. Wirtz, Ph.D., Professor of Decision Sciences and of Psychology at The George Washington University deserves special thanks for exposing me to the SAS System while I was a graduate student at GWU more years ago than either of us care to admit!

Copyright Acknowledgement:

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 or product names are registered trademarks or trademarks of their respective companies.

Author contact:

Andrew H. Karp
Principal Consultant
Sierra Information Services
19229 Sonoma Hwy #264
Sonoma, CA 95476 USA
andrew@sierrainformation.com
www.Sierrainformation.com

APPENDIX A: Creating SAS Date Values from Text or Number Strings in a SAS Data Set

At the outset of this paper I discussed using SAS Informats in an INPUT Statement to convert text strings in a raw data file in to the values of SAS date variables during a Data Step. Here is a “beyond the basics” set of examples of how to convert text or number “strings” that are “already” in a SAS data set in to SAS date and time variables. We’ll use a small example from the TABLES data set.

First, let’s create an example data set with two observations. The table order and ship dates are CHARACTER variables in temporary SAS data set DATES2.

```

324
325 data dates;
326 input order_date $ ship_date $;
327 datalines;
328 4/28/2005 2005/05/10
329 05-06-2006 2006.08.09
330 ;
331 run;

```

Working with SAS Date and Time Functions Dates as Character Strings in a SAS Data Set

orderdate	shipdate
04/28/2005	2005/05/10
05-06-2006	2006.08.29

In our example data set ORDER_DATE is a string with month, day and year separated by the slash (or stroke) mark in the first observation and by the hyphen (or dash) mark in the second. SHIP_DATE is a string with year, month and day separated by the slash in the first observation and a period in the second observation. Since these are character variables in our SAS data set, we need to convert them to numeric SAS date values in a Data Step.

The solution is to use an INPUT Function call to convert the text strings to numeric date values. The second required argument to the INPUT Function is an Informat that “tells” SAS how the month, day and year values are ordered in the text string. In the Data Step shown below, the MMDDYY10. Informat is used for ORDER_DATE and the YYMMDD10. Informat is used for SHIP_DATE. Notice that both Informats accept a wide range of separator characters (slash, hyphen, period, and a few others documented in the BASE SAS Language: Reference text). Combining the INPUT Function with the right Informat makes this task easy to accomplish!

```

42 data dates3;
43 set dates2;
44 orderdate_new = input(orderdate,mmddy10.);
45 shipdate_new = input(shipdate,yyymmdd10.);
46 run;
47
48 proc print data=dates3;
49 title2 'Using the INPUT Function to Convert Text Strings';
50 title3 'to Numeric SAS Data Values';
51 run;

```

Working with SAS Date and Time Functions Using the INPUT Function to Convert Text Strings to Numeric SAS Data Values

orderdate	shipdate	orderdate_new	shipdate_new
04/28/2005	2005/05/10	16554	16566
05-06-2006	2006.08.29	16927	17042

Now that we have SAS date values in our data set we can easily apply many of the tools and features discussed elsewhere on them!

Next, let’s look at a situation where the dates in our SAS data set are a string of numbers. How can we convert those to values of numeric SAS date variables? Now things are a bit trickier. In the data set shown below we have numbers, not text strings with a separator character between the month, day and year, as we had in the previous example. And, if the month value is between January and September, (and assuming a four-digit year) the length of the string of numbers will be seven, and eight for October, November and December.

First, let's look at the made-up Data Set we're going to work with:

```

55 data dates4;
56 input orderdate shipdate;
57 datalines;
58 04282005 20050510
59 05062006 20060829
60 ;
61 run;
62
63 proc print data=dates4;
64 title2 'Numeric Date Strings';
65 run;

```

Working with SAS Date and Time Functions Numeric Date Strings

orderdate	shipdate
4282005	20050510
5062006	20060829

Even though we had “leading zeros” for the values of ORDERDATE in the “raw data,” we “lost” them when the numeric strings were created in SAS Data Set DATES4. If the number “string” for ORDERDATE is the month, day and year the table was ordered, the length of that string will be seven (7) if the order occurred in January through September and eight (8) if the order was taken in October, November or December. So, whatever solution we come up with has to accommodate number “strings” of both lengths.

The Data Step below shows how a combination of the INPUT and PUT Functions are used, along with the “Z” Informat to accomplish the desired outcome.

```

70 data dates5;
71 set dates4;
72 * z informat pads string w/leading zeros;
73 orderdate_new = input(put(orderdate,z8.),mmdyy10.);
74 shipdate_new = input(put(shipdate,$10.),yymmdd10.);
75 run;
76
77 proc print data=dates5;
78 title2 'Converting Numeric Date Strings to SAS Date Values';
79 run;

```

Working with SAS Date and Time Functions Converting Numeric Date Strings to SAS Date Values

orderdate	shipdate	orderdate_new	shipdate_new
4282005	20050510	16554	16566
5062006	20060829	16927	17042

In this example the PUT Function calls on lines 73 and 74 of the Enhanced Program Editor screen capture convert the existing numeric values of ORDERDATE and SHIPDATE to character strings. For ORDERDATE, the Z Informat with a length of 8 is used. That Informat will “pad” the string it assigns with leading zeros. So, if the date of order occurred in January through September, we “automatically” get a text string with the leading zero we need to “pad out” its length. Then, the INPUT Function is used to convert the text string value assigned by the PUT Function to a numeric SAS date value. Combining both Function calls in a single assignment statement, plus use of the appropriate Informats makes this an easy problem to solve!

Appendix B Creating SAS Value Formats for “Non-Standard” Date Ranges

Here are two examples of creating customized Value Formats for “non-standard” date ranges. Notice how SAS Date Constants, discussed above, are used to supply to PROC FORMAT the start and end of the date ranges to which we want the Format labels assigned. For more information on the SAS Format facility see Bilenas (2005) or my presentations “My Friend the SAS Format,” and “SAS Formats: Beyond the Basics” available as PDFs from the “Free Downloads” link at www.SierralInformation.com. The PROC FORMAT step on the left aggregates dates by the United States government’s fiscal year and the

The PROC FORMAT step on the left creates labels corresponding the start and end dates of the US federal government's fiscal year while the one on the right carries out the same task for the State of California fiscal year

```

12 * create format for us federal fiscal year;
13 * starts oct 1, ends sept 30;
14 proc format;
15 value fed_fy_fmt
16 '01oct2003'd - '30sep2004'd = 'Fed FY 2004'
17 '01oct2004'd - '30sep2005'd = 'Fed FY 2005'
18 '01oct2005'd - '30sep2006'd = 'Fed FY 2006'
19 '01oct2006'd - '30sep2007'd = 'Fed FY 2007';
20 run;
21
22 proc freq data=sasclass.tables_sales_final;
23 tables ship_date/nocum;
24 format ship_date fed_fy_fmt.;
25 title3 'Using Customized Date Format';
26

```

```

1
2 * create customized format for California State Fiscal Year;
3 proc format library = calworks;
4 value ca_fy_fmt
5 '01ju12002'd - '30jun2003'd = 'CA FY 2003'
6 '01ju12003'd - '30jun2004'd = 'CA FY 2004'
7 '01ju12004'd - '30jun2005'd = 'CA FY 2005'
8 '01ju12005'd - '30jun2006'd = 'CA FY 2006'
9 '01ju12006'd - '30jun2007'd = 'CA FY 2007'
10 '01ju12007'd - '30jun2008'd = 'CA FY 2008';
11 run;

```

Finally, a MULTILABEL Format is created that aggregates data by the quarters in the State of California fiscal year, as well as by Fiscal Year, and then across all data values to which the Format is associated.

```

4 proc format library = calworks;
5 value ca_fy_fq_fmt (multilabel notsorted)
6 '01ju12002'd - '30sep2002'd = ' FY 2003 Q 1'
7 '01oct2002'd - '31dec2002'd = ' FY 2003 Q 2'
8 '01jan2003'd - '31mar2003'd = ' FY 2003 Q 3'
9 '01apr2003'd - '30jun2003'd = ' FY 2003 Q 4'
10 '01ju12002'd - '30jun2003'd = 'CA FY 2003'
11
12 '01ju12003'd - '30sep2003'd = ' FY 2004 Q 1'
13 '01oct2003'd - '31dec2003'd = ' FY 2004 Q 2'
14 '01jan2004'd - '31mar2004'd = ' FY 2004 Q 3'
15 '01apr2004'd - '30jun2004'd = ' FY 2004 Q 4'
16 '01ju12003'd - '30jun2004'd = 'CA FY 2004'
17
18 '01ju12004'd - '30sep2004'd = ' FY 2005 Q 1'
19 '01oct2004'd - '31dec2004'd = ' FY 2005 Q 2'
20 '01jan2005'd - '31mar2005'd = ' FY 2005 Q 3'
21 '01apr2005'd - '30jun2005'd = ' FY 2005 Q 4'
22 '01ju12004'd - '30jun2005'd = 'CA FY 2005'
23

```

```

24 '01ju12005'd - '30sep2005'd = ' FY 2006 Q 1'
25 '01oct2005'd - '31dec2005'd = ' FY 2006 Q 2'
26 '01jan2006'd - '31mar2006'd = ' FY 2006 Q 3'
27 '01apr2006'd - '30jun2006'd = ' FY 2006 Q 4'
28 '01ju12005'd - '30jun2006'd = 'CA FY 2006'
29
30 '01ju12006'd - '30sep2006'd = ' FY 2007 Q 1'
31 '01oct2006'd - '31dec2006'd = ' FY 2007 Q 2'
32 '01jan2007'd - '31mar2007'd = ' FY 2007 Q 3'
33 '01apr2007'd - '30jun2007'd = ' FY 2007 Q 4'
34 '01ju12006'd - '30jun2007'd = 'CA FY 2007'
35
36 '01ju12007'd - '30sep2007'd = ' FY 2008 Q 1'
37 '01oct2007'd - '31dec2007'd = ' FY 2008 Q 2'
38 '01jan2008'd - '31mar2008'd = ' FY 2008 Q 3'
39 '01apr2008'd - '30jun2008'd = ' FY 2008 Q 4'
40 '01ju12007'd - '30jun2008'd = 'CA FY 2008'
41 '01ju12002'd - '30jun2008'd = '** CA FY 2002 - CA FY 2008 **';
42 run;

```

Bibliography

- Bilenas, Jonas V. 2006. *The Power of PROC Format*. Cary, NC: SAS Institute, Inc.
- Cody, Ron. 2004. *SAS® Functions by Example*. Cary, NC: SAS Institute, Inc.
- Morgan, Derek. 2006. *The Essential Guide to SAS® Dates and Times*. Cary, NC: SAS Institute, Inc.
- SAS Institute Inc. 2009 *SAS 9.2 Language Reference: Dictionary*. Cary, NC: SAS Institute, Inc.