SAS® Date and Time Functions 101
Nikki Carroll, Kaiser Permanente, Denver, CO

ABSTRACT
Since the beginning of time, people have had the desire to document the passage of hours, days, weeks and years. After documenting the passage of time, calculating differences in time came next. Calculating this passage of time using dates, times and datetimes in SAS® can seem intimidating at first, however, after learning a few tips and tricks, this task can be invigorating and, some might say, fun! An introduction to SAS dates will be discussed as well as techniques for 1) formatting dates, 2) reading dates into datasets, 3) how to manipulate data using dates, and 4) using functions with dates.

INTRODUCTION – WHAT ARE SAS DATES AND TIMES?
SAS dates are numeric values that measure time from an arbitrary starting point by counting the number of elapsed time units. The arbitrary starting point used by SAS is January 1, 1960 (date zero). Dates before January 1, 1960 are negative numbers; dates after are positive numbers. For example, The SAS date value for July 17, 2006 is 16999 and the SAS date value of July 17, 1950 is -3455. SAS date values account for all leap year days, including the leap year day in the year 2000.

Similarly, SAS datetime values are numeric variables the represent the number of seconds between midnight, January 1, 1960 and an hour/minute/second within a specified date. For example, the SAS datetime value for August 8, 1997 at 30 seconds after 1:46 in the afternoon is 1186667190.

In addition, SAS also supports time values. SAS time values are just like datetime values, except that the date part is not given. Time values are numeric variables that represent the number of seconds elapsed since midnight. SAS time values are between 0 and 86,400. For example, 30 seconds after 1:46 in the afternoon is 45590.

SAS can perform calculations on dates represented this way ranging from A.D. 1582 to A.D. 19,900. By referencing the number of days from a reference date the computer can store data, determine time intervals, use dates as constants, and perform calendar calculations much more efficiently, but these numbers are not meaningful to users. However, you never have to use SAS date values directly, since SAS automatically converts between this internal representation and ordinary ways of expressing dates, provided that you indicate a “format” that describes how you want the date values to be displayed.

DISPLAYING DATES WITH FORMATS
HOW SAS DISPLAYS NUMERIC VALUES
A format is a set of directions that tell SAS how to display date values. By default, SAS uses a standard numeric format with no commas, letters, or other special characters to display the value of numeric variables. In order to display your numeric variable in the style that you want, you need to give SAS these directions, or formats, on how to display your date variable.

SAS date formats are available for the most common ways of writing calendar dates. The DATE9. format represents dates in the form ddMMMyyyy (for example, 12JUL2006). If you want the more standard representation of a date, use MMDDYY8., which displays the date in the form of mm/dd/yy (for example, 07/17/06), or you can use the MMDDYY10., which displays the date in the form of mm/dd/yyyy (for example, 07/17/2006). SAS has many more formats which can all be found in the Online Documentation.

FORMATTING A DATE VALUE
The FORMAT statement, which uses a variable name and the format name, is how you tell SAS which format to use. The following format statement will give the variable My_Birthday a format of MMDDYY10.:

```sas
format my_birthday mmddyy10.;
```

You can also format your date variables when displaying the dates in a PROC PRINT statement. Notice that for all of the formats, each name ends with a period and contains the width specification that tells SAS how many columns to use when displaying the date value.

```sas
proc print data=mydir.birthday_list;
  title 'Listing of Acem Widget Employee Birthdays';
  format my_birthday mmddyy10.;
run;
```
ASSIGNING PERMANENT DATE FORMATS TO VARIABLES

When using the format statement in a PROC PRINT, you are only assigning that format to that variable during the time of the print. In order to permanently attach the format to the variable, the FORMAT statement needs to be assigned in a data step. The following data step will assign the variable My_Birthday a format of DATE9. This will be confirmed by doing a PROC CONTENTS.

```sas
data mydir.new_bday_list;
set mydir.old_bday_list;
    format my_birthday date9.;
run;

proc contents data=mydir.new_bday_list nodetails;
run;
```

The OUTPUT shows that the DATE9. format is permanently associated with My_Birthday.

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>MYDIR.NEW_BDAY_LIST</th>
<th>Observations</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>2</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>Tuesday, July 18, 2006 08:59:14 PM</td>
<td>Observation Length</td>
<td>16</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Tuesday, July 18, 2006 08:59:14 PM</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td>Compressed NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Set Type</td>
<td>Sorted NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>WINDOWS_32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>wlatin1 Western (Windows)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CHANGING FORMATS TEMPORARILY

If you need a different format temporarily for a special report on your data, you can override the permanent format by using a FORMAT statement in a PROC step. For example, to change the format of My_Birthday to a word representation, the WORDDATE FORMAT statement can be used in the PROC PRINT:

```sas
proc print data=mydir.new_bday_list;
    title 'Acme Widget Employee Birthday List';
    var my_birthday;
    format my_birthday worddate18.;
run;
```
The OUTPUT shows:

<table>
<thead>
<tr>
<th>Obs</th>
<th>my_birthday</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>July 17, 2006</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>August 8, 1997</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>August 26, 2001</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>February 14, 1971</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>October 22, 1963</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>April 14, 1941</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>November 19, 1955</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>December 16, 1965</td>
<td>38</td>
</tr>
<tr>
<td>9</td>
<td>May 22, 1966</td>
<td>41</td>
</tr>
</tbody>
</table>

READING IN DATES WITH INFORMATS

If you are reading in a SAS data set that contains dates, it is in your best interest to read the variables in as SAS dates instead of as characters or standard numeric variables. This will prevent trouble down the road.

UNDERSTANDING INFORMATS FOR DATE VALUES

In order for SAS to read a value into a new data set, it must be given a set of directions called an informat. Informats are similar to formats in that they give special instructions to SAS on how to display numeric variables. The only difference is that an INFORMAT is used when you are reading in data. A FORMAT statement is used on data that is already in a SAS data set.

SAS provides many INFORMATS. Four commonly used INFORMATS are

- **MMDDYY8.** reads dates written as mm/dd/yy
- **MMDDYY10.** reads dates written as mmd//d yyyy
- **DATE7.** reads dates in the form ddMMMyy
- **DATE9.** reads dates in the form ddMMMyyyy

READING A DATE VALUE

Our data set that we’re reading in contains birthdates that are already in the format of a DATE9. Therefore, it will be easy to read the My_Birthday field in using the DATE9. format. The code presented below reads in My_Birthday as SAS date values and by adding the format to the INPUT statement, the dates will be permanently attached to My_Birthday and will display with the DATE9. format.

```sas
data mydir.new_bday_list;
  infile 'input file';
  input my_birthday date9.
    employee_num;
  run;

proc print data=mydir.new_bday_list;
  title 'Acme Widget Employee Birthday List';
run;
```

<table>
<thead>
<tr>
<th>Obs</th>
<th>my_birthday</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17JUL1968</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>08AUG1997</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>26AUG2001</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>14FEB1971</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>22OCT1963</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>14APR1941</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>19NOV1955</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>16DEC1965</td>
<td>38</td>
</tr>
<tr>
<td>9</td>
<td>22MAY1966</td>
<td>41</td>
</tr>
</tbody>
</table>
“ANYDATE” INFORMATS
Sometimes you are very lucky in that the raw data you receive contains dates that are the same format. Sometimes you will encounter a messy data file where the dates are all different types of formats. The “anydate” informats are designed to allow you to read in a variety of date forms including:

- DATE, DATETIME, and TIME
- DDMMYY, MMDDYY, and YYMMDD
- JULIAN, MONYY, and YYQ

Using the anydate informats can be particularly useful when you are reading in data that contains a mixture of date forms and you want certain parts of the dates you are reading in. Anydate informats include:

- ANYDTDTE. Extracts the date portion
- ANYDTDTM. Extracts the datetime portion
- ANYDTTME. Extracts the time portion

```options datestyle = mdy;
data new_list;
input date anydtdte10.;
put date;
format date date9.;
datalines;
17JUL1968
08/08/1997
26/08/2001
14FEB71
10/22/1963
14APR1941
11/19/55
16/12/65
5/22/1966;
run;
```

The LOG shows:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17JUL1968</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>26AUG2001</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>22OCT1963</td>
<td></td>
</tr>
<tr>
<td>14APR1941</td>
<td></td>
</tr>
<tr>
<td>19NOV1955</td>
<td></td>
</tr>
<tr>
<td>16DEC1965</td>
<td></td>
</tr>
<tr>
<td>22MAY1966</td>
<td></td>
</tr>
</tbody>
</table>

Notice that the second and fourth dates were set to missing because the date was too ambiguous for SAS to resolve using the anydtdte10. informat. The DATESTYLE= system option can be used to resolve these ambiguities and can take on values such as MDY (default), DMY, YMD, etc.

DATE CONSTANTS
A SAS date constant is a date that is coded by the programmer, usually during the data step in order to introduce a date value into SAS. When date values are entered through the data step, it’s usually for comparisons or corrections to data. The value ‘17JUL2006’D would be a SAS date constant. Notice the quotes, the date style of ddMMMyyyy and then the letter D. The D suffix tells SAS to convert the calendar date to a SAS date value. The following program uses a SAS date constant to correct an employee’s birthday:

```data new_bday_list2; set mydir.new_bday_list;
  if employee_num = 11 then my_birthday='17JUL1968’d;
run;
```

```proc print data=mydir.new_bday_list ;
title 'Corrected Birthday List';
run;
```
The OUTPUT shows:

<table>
<thead>
<tr>
<th>Obs</th>
<th>my_birthday</th>
<th>num</th>
<th>Todaysdate</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>07/17/1968</td>
<td>11</td>
<td>09/29/2006</td>
<td>13953</td>
</tr>
<tr>
<td>2</td>
<td>08/08/1997</td>
<td>14</td>
<td>09/29/2006</td>
<td>3339</td>
</tr>
<tr>
<td>3</td>
<td>08/26/2001</td>
<td>22</td>
<td>09/29/2006</td>
<td>1860</td>
</tr>
<tr>
<td>4</td>
<td>02/14/1971</td>
<td>25</td>
<td>09/29/2006</td>
<td>13011</td>
</tr>
<tr>
<td>5</td>
<td>10/22/1963</td>
<td>29</td>
<td>09/29/2006</td>
<td>15683</td>
</tr>
<tr>
<td>6</td>
<td>04/14/1941</td>
<td>30</td>
<td>09/29/2006</td>
<td>23909</td>
</tr>
<tr>
<td>7</td>
<td>11/19/1955</td>
<td>33</td>
<td>09/29/2006</td>
<td>18577</td>
</tr>
<tr>
<td>8</td>
<td>12/16/1965</td>
<td>38</td>
<td>09/29/2006</td>
<td>14897</td>
</tr>
<tr>
<td>9</td>
<td>05/22/1966</td>
<td>41</td>
<td>09/29/2006</td>
<td>14740</td>
</tr>
</tbody>
</table>

DATETIME CONSTANTS
Similarly, a datetime constant is written with the date and time in single quotes followed by DT. To write the date and time in a SAS datetime constant, write the date part using the same syntax as for date constants, and follow the date part with the hours, the minutes, and the seconds, separating the parts with colons. The seconds are optional.

For example, in a SAS program you would write August 8, 1997 at 1:46 in the afternoon as ’8AUG97:13:46’DT. SAS reads this as 1186667190 and this datetime value can be formatted in the format of your choice.

TIME CONSTANTS
The SAS System also supports time values. SAS time values are similar to datetime values, except that the date part is not given. To write a time value in a SAS program, write the time the same as for a datetime constant but use T instead of DT. For example, 1:46:30 p.m. is written ’13:46:30’T. Time values are represented by the number of seconds since midnight, so SAS reads ’13:46:30’T as 45590.

SAS time values are not very useful on their own, since usually both the date and the time of day are needed.

COMPARING DURATIONS AND SAS DATE VALUES
Perhaps the most common thing you will do with dates in your data sets will be to find units of time or intervals between dates. Because SAS date values are numeric values, they can be sorted and used easily in calculations. For example, it is September 29th and you want to find out how old all of your employees are and you write the following program:

```sas
data age_calc; set mydir.new_bday_list2;
todaysdate='29SEP2006'd;
age = todaysdate - my_birthday;
format todaysdate my_birthday mmddyy10.;run;
proc print data=age_calc;
title 'Employee Ages';run;
```

The OUTPUT shows that the age values for the employees look like unformatted SAS date values, however, they are actually the number of days between TodaysDate (September 29th) and the employee’s birthday and not a SAS date value.
CALCULATING A DURATION IN YEARS
To make the value of Age more understandable, divide the number of days by 365.25 to calculate the age of the employee in years:

```sas
data age_calc; set mydir.new_bday_list2;
todaysdate='29SEP2006'd;
ageInDays = todaysdate - my_birthday;
ageInYears = AgeInDays/365.25;
format todaysdate my_birthday mmddyy10.;
run;
```

```sas
proc print data=age_calc;
  title 'Employee Ages In Years';
run;
```

The OUTPUT shows:

<table>
<thead>
<tr>
<th>Obs</th>
<th>my_birthday</th>
<th>num</th>
<th>Todaysdate</th>
<th>AgeInDays</th>
<th>AgeInYears</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>08/08/1997</td>
<td>14</td>
<td>09/29/2006</td>
<td>3339</td>
<td>9.1417</td>
</tr>
<tr>
<td>3</td>
<td>08/26/2001</td>
<td>22</td>
<td>09/29/2006</td>
<td>1860</td>
<td>5.0924</td>
</tr>
<tr>
<td>4</td>
<td>02/14/1971</td>
<td>25</td>
<td>09/29/2006</td>
<td>13011</td>
<td>35.6222</td>
</tr>
<tr>
<td>5</td>
<td>10/22/1963</td>
<td>29</td>
<td>09/29/2006</td>
<td>15683</td>
<td>42.9377</td>
</tr>
<tr>
<td>6</td>
<td>04/14/1941</td>
<td>30</td>
<td>09/29/2006</td>
<td>23909</td>
<td>65.4593</td>
</tr>
<tr>
<td>7</td>
<td>11/19/1955</td>
<td>33</td>
<td>09/29/2006</td>
<td>18577</td>
<td>50.8611</td>
</tr>
<tr>
<td>8</td>
<td>12/16/1965</td>
<td>38</td>
<td>09/29/2006</td>
<td>14897</td>
<td>40.7858</td>
</tr>
<tr>
<td>9</td>
<td>05/22/1966</td>
<td>41</td>
<td>09/29/2006</td>
<td>14740</td>
<td>40.3559</td>
</tr>
</tbody>
</table>

SORTING SAS DATES
Let's say you now want the list sorted in ascending years of age. It's done easily with the SAS date values:

```sas
proc sort data = age_calc out=age_sorted;
  by AgeInYears;
run;
```

```sas
proc print data=age_sorted;
  title 'Employee Ages In Ascending Order'; run;
```

The OUTPUT shows the employee list is now sorted in chronological order:

<table>
<thead>
<tr>
<th>Obs</th>
<th>my_birthday</th>
<th>num</th>
<th>Todaysdate</th>
<th>AgeInDays</th>
<th>AgeInYears</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>08/26/2001</td>
<td>22</td>
<td>09/29/2006</td>
<td>1860</td>
<td>5.0924</td>
</tr>
<tr>
<td>2</td>
<td>08/08/1997</td>
<td>14</td>
<td>09/29/2006</td>
<td>3339</td>
<td>9.1417</td>
</tr>
<tr>
<td>3</td>
<td>02/14/1971</td>
<td>25</td>
<td>09/29/2006</td>
<td>13011</td>
<td>35.6222</td>
</tr>
<tr>
<td>5</td>
<td>10/22/1966</td>
<td>41</td>
<td>09/29/2006</td>
<td>14740</td>
<td>40.3559</td>
</tr>
<tr>
<td>6</td>
<td>12/16/1965</td>
<td>38</td>
<td>09/29/2006</td>
<td>14897</td>
<td>40.7858</td>
</tr>
<tr>
<td>7</td>
<td>11/19/1955</td>
<td>33</td>
<td>09/29/2006</td>
<td>18577</td>
<td>50.8611</td>
</tr>
<tr>
<td>8</td>
<td>10/22/1963</td>
<td>29</td>
<td>09/29/2006</td>
<td>15683</td>
<td>42.9377</td>
</tr>
<tr>
<td>9</td>
<td>04/14/1941</td>
<td>30</td>
<td>09/29/2006</td>
<td>23909</td>
<td>65.4593</td>
</tr>
</tbody>
</table>
CHANGING VARIABLES TO SAS DATE VALUES
The above code works well if you are starting with raw data and have the choice of how to read it in. A lot of times you will be given data that is already read in and formatted as numeric or character values. If you need to work with these variables as SAS date values, then they will need to be converted. By using the INPUT and PUT statements in a DATA step, these can easily be converted.

CHANGING CHARACTER VALUES TO SAS DATE VALUES
You can change a character value to a SAS date value by using the INPUT function with the appropriate format when creating a new variable. In this example, say you have 2 character variables that need to be converted into dates. The dates are as follows:

```plaintext
wrong_date1 = '17/JUL/2006'
wrong_date2 = '07-17-06'
```

The INPUT function will convert the value from a character to a numeric value. This code shows how SAS converts the previous dates into SAS date values:

```plaintext
correct_date1 = input (wrong_date1, date11.);
correct_date2 = input (wrong_date2, mmddyy8.);
```

The OUTPUT shows the incorrect and corrected SAS date values:

<table>
<thead>
<tr>
<th>Corrected Date Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

CHANGING NUMERIC VALUES TO SAS DATE VALUES
You can change a numeric value to a SAS date value by using both the INPUT and PUT functions. For example, let’s say these are the two dates in your dataset that need to be converted.

```plaintext
wrong_date3 = 170706
wrong_date4 = 20060717
```

The PUT function will convert the value from numeric to character and the INPUT will convert the character value to a numeric SAS date. Using this code will convert these numbers into SAS date values:

```plaintext
correct_date3 = input (put (wrong_date3, z6.), ddmmyy6.);
correct_date4 = input (put (wrong_date4, z8.), yymmdd8.);
```

The OUTPUT shows the incorrect and corrected SAS date values:

<table>
<thead>
<tr>
<th>Corrected Date Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

SAS DATE FUNCTIONS
SAS9 contains numerous functions that help us work with DATE, TIME and DATETIME values. These functions are used to modify, convert and otherwise manipulate these values from one form to another.

MDY FUNCTION
Some data will contain the month, day and year for a date, however, they might all be separate variables. In order to put them together and make it a SAS date, the MDY function can be used. The MDY function consists of a numeric argument that contains the month (must be between 1 and 12), a numeric argument that contains the day (must be between 1 and 31), and a numeric year argument (must be between 1589 and 19900). The following is an example of how the MDY function returns a SAS date value from separate month, day and years variables:
data;
month_field = 7;
day_field = 17;
year_field = 1968

my_birthday=mdy(month_field,day_field,year_field);
put my_birthday worddate.;
run;

The OUTPUT shows:

July 17, 1968

OBTAINING THE CURRENT SYSTEM DATE/TIME
SAS also contains functions that will return the current date and time as shown with the following code:

data today;
a = date();
b = today();
c = time();
d = datetime();
put a= date. B=mmddyy8. c=time10.2 d=datetime16.;
run;

The OUTPUT shows:

A=19JUL06 B=07/19/06 C=18:22:37.3 D=19JUL06:18:22:37

EXTRACTING ‘PARTS’ FROM SAS DATE VALUES
Sometimes you might only want a part of the date value. Some of the popular parts that can be extracted from a SAS date value are:

- **MONTH** returns the month
- **DAY** returns the day
- **YEAR** returns the year
- **QTR** returns the quarter of the year
- **WEEKDAY** returns the day of the week numerically (Sunday = 1)

Assume a SAS data set contains a variable called Birthdate. Coding the date part functions for Birthdate would look like the following:

data parts;
birthdate = '17jul1968'd;
bmonth = month(Birthdate);
bday = day(birthdate);
byear = year(birthdate);
bqtr = qtr(birthdate);
bweekday = weekday(birthdate);
format birthdate mmddyy10.;
put _all_;
run;

The OUTPUT shows:

Birthdate=07/17/1968 Bmonth=7 Bday=17 Byear=1968 Bqtr=3 Bweekday=4

EXTRACTING ‘PARTS’ FROM A SAS TIME VALUE
SAS also has time part functions that include HOUR, MINUTE, and SECOND functions. These are obtained in the same way as the date functions discussed above.

EXTRACTING THE DATE/TIME PARTS OF A DATETIME VALUE
Using the DATEPART and TIMEPART functions you can easily get the date or the time value of a DATETIME variable. The following example demonstrates these functions:
data datetimeparts;
now = datetime();
now_date = datepart(now);
now_time = timepart(now);

format now datetime18.
    now_date mmdyy10.
    now_time time10.2;
put _all_; run;

The LOG shows:
| Now=19JUL06:20:06:46 | Now_date=07/19/2006 | Now_time=20:06:46.5 |

THE INTCK FUNCTION
The INTCK function counts the number of intervals between two dates.

data intervals;
birthdate = '17JUL1968'd;
yrs = intck('year', Birthdate,'29SEP2006'd);
months = intck('month', Birthdate,'29SEP2006'd);
weeks = intck('week', Birthdate,'29SEP2006'd);
qtrs = intck('qtr', Birthdate,'29SEP2006'd);

put _all_; run;

The LOG shows:
| Birthdate=3120 | Yrs=38 | Months=458 | Weeks=1993 | Qtrs=152 |

THE YRDIF FUNCTION
The YRDIF function is similar to the INTCK function in that it calculates the number of intervals between two dates. The difference between the two functions is that YRDIF only calculates the number of years between two dates. This function comes with different options that will calculate the difference based on the number of days in a month and/or in a year.

- ‘30/360’ specifies each month to have 30 days and each year to have 360 days regardless of the actual number of days in each month or year.
- ‘ACT/ACT’ uses the actual number of days between the dates.

The YRDIF statement consists of the two variables between which you are trying to calculate a difference and one of the above calculation methods:

data agecalc;
todaysdate = today();
birthdate = '17JUL1968'd;
age_in_yrs = yrdif(birthdate, todaysdate, 'ACT/ACT');
run;

proc print data=agecalc;
format todaysdate birthdate mmdyy10.;
title 'Calculating Age using the YRDIF Function';
run;

The LOG shows:
<table>
<thead>
<tr>
<th>Calculating Age using the YRDIF Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
TWO-DIGIT AND FOUR-DIGIT YEARS
SAS software can read two-digit or four-digit year values. If SAS encounters a two-digit year, the YEARCUTOFF= option can be used to specify which century within a 100 year span the two-digit year should be attributed to. For example, YEARCUTOFF=1950 means that two-digit years 50 through 99 correspond the 1950 through 1999, while two-digit years 00 through 49 correspond to 2000 through 2049. To correctly handle 2-digit years representing date between 2000 and 2099, you should specify an appropriate YEARCUTOFF= value between 1901 and 2000.

Before using the YEARCUTOFF option, examine the dates in your data:
- If the dates in your data fall within a 100-year span, then you can use the YEARCUTOFF= system option
- If the dates in your data do not fall within a 100-year span, then you must either convert the two-digit years to four-digit years or use a DATA step with conditional logic to assign the proper century prefix.

After you’ve determined that the YEARCUTOFF= system option is appropriate for your range of data you can determine the setting to use. The best setting for YEARCUTOFF= is the year before the lowest year in your data. For example, if you have data in a range from 1921 to 2001, then set YEARCUTOFF= to 1920, if that is not already your system default. The result of setting YEARCUTOFF= to 1920 is that
- SAS interprets all two-digit dates in the range of 20 through 99 as 1920 through 1999.
- SAS interprets all two-digit dates in the range of 00 through 19 as 2000 through 2019.

With YEARCUTOFF= set to 1920, a two-digit year of 10 would be interpreted as 2010 and a two-digit year of 22 would be interpreted as 1922.

CONCLUSIONS
This paper just begins to scratch the surface of all you could know or learn regarding SAS DATES, TIMES and DATETIMES. Once dates are read in as SAS date values, SAS provides a wide variety of tools and functions that help you manipulate and convert dates and time.

REFERENCES


CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Nikki Carroll
Kaiser Permanente
P.O. Box 378066
Denver, CO 80237-8066
(303) 614-1251 (work)
(303) 614-1225 (fax)
nikki.m.carroll@kp.org

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 trademarks of their respective companies.