SESUG Paper 084-2025

An Overview Comparing SAS® Dates and Times

Patrick J. Mercer, Rho Inc.

ABSTRACT

Working with date fields can present frustration for programmers of all levels, but what if date comparisons could be made simpler? Dates can be stored and formatted in a variety of ways, so it is crucial to understand how to compare them. Untangling the mystery of SAS® dates and datetimes adds a powerful tool to the hands of the aspiring programmer. This paper will compare numeric versus character dates, date versus datetime fields, and defining lengths and formats.

INTRODUCTION

SAS recognizes pieces of data as either numeric or character. This is clear for most data: Text data will certainly be character, and numbered data will often be numeric. Date and time values are not so clear because of the varied ways we can present them.

SAS understands dates and times in their numeric form: the system stores numeric date values as the number of days relative to January 1, 1960, and stores times/datetimes as the number of seconds relative to midnight of January 1, 1960. Neither you nor I would find these representations useful: If I told the doctor's office my date of birth was "13413" rather than "September 21, 1996", I would receive some confused looks.

DATETIME BASICS

If you apply an appropriate **format** to a date value, SAS will display the value as a legible date, as shown below:

```
FORMAT fdate date7. ftime time5. fdatetime datetime21.;
```

Formats

Obs	date	fdate	time	ftime	datetime	fdatetime
1	21730	30JUN19	36600	10:10	2579840280	01OCT2041:06:18:00

Table 1. Formats

As shown above in Table 1, when you apply a format of date7. to 21730, SAS will display the value as 30JUN19, understandable as June 30, 2019. Similarly, when you apply the format datetime21. to 2579840280, SAS will display the as 01OCT2041:06:18:00, or October 1, 2041, at exactly 6:18 in the morning.

Similarly, you may apply an **informat**, or an instruction to tell SAS how to read the data you input:

```
INFORMAT indate date9. intime time5. indatetime datetime21.;
```

Informats

Obs	date	indate	time	intime	datetime	indatetime
1	31MAR2000	14700	10:10	36600	01OCT2041:06:18:00	2579840280

Table 2. Informats

As shown above in Table 2, SAS understands date = "31MAR2000" with informat date9. as the numeric value of indate = 14700.

We may apply formats and informats to a variable implicitly with the **PUT()** and **INPUT()** functions respectively:

```
input_date = INPUT(date,date9.);
put_date = PUT(input_date,yymmdd10.);
cutoff_date = PUT('07JUL2025'd,mmddyy10.);
```

Put & Input Functions

Obs	date	input_date	put_date	cutoff_date
1	31MAR2000	14700	2000-03-31	07/07/2025

Table 3. Put & Input Functions

We use the PUT() and INPUT() functions above to assign three additional dates and output to Table 3.

The PUT() and INPUT() functions are useful in cases where your final data must remain unformatted. You may also opt to declare the options `FORMAT _all_;` and `INFORMAT _all_;` which will clear formats and informats from all columns in your dataset.

COMPARING DATES, TIMES, AND DATETIMES

Issues often arise when comparing date, time and datetime values. We examine these issues in the remainder of this paper. The first two issues pertain to numeric values, while the following three issues relate to character values.

NUMERIC VALUES

You might find that numeric date, time, and datetimes values are more “anchored” than character values. We can simplify any numeric date value to its number of relative days, and we can simplify any numeric time or datetime value to its number of relative seconds. Despite this foundation, you may encounter a couple of key issues when comparing numeric values:

Issue 1: Comparing dates to datetimes

Recall that numeric values can either be dates (numbers of days) or datetimes (numbers of seconds). Let us see the example below where we conflate date and datetime values with each other:

```
DATA n_mismatch1;
   LENGTH date1 datetime1 date2 datetime2 8.;
   FORMAT date1 datetime2 date9. datetime1 date2 datetime21.;
   INPUT date1 datetime1 date2 datetime2;
   DATALINES;
      19996 19996      1999999996 1999999996
      20001 20002      20003      20000
      32123 2000000000 12345      54321
   ;
RUN;

DATA n_mismatch2;
   SET n_mismatch1;
   IF date2 > date1 THEN dateflag = "Y"; ELSE dateflag = "N";
   if datetime2 > datetime1 THEN datetimeflag = "Y"; ELSE datetimeflag
= "N";
RUN;
```

Incorrect comparisons - Numeric

Obs	date1	datetime1	date2	datetime2	dateflag	datetimeflag
1	30SEP2014	01JAN1960:05:33:16	18MAY2023:03:33:16	*********	Y	Y
2	05OCT2014	01JAN1960:05:33:22	01JAN1960:05:33:23	04OCT2014	Y	N
3	13DEC2047	18MAY2023:03:33:20	01JAN1960:03:25:45	22SEP2108	N	N

Table 4. Incorrectly Comparing Numeric Dates to Datetimes

Notice in Table 4 that, despite their names, date2 is formatted as a datetime, and datetime2 is formatted as a date. Due to this disorganized data, we see issues in the output:

- In observations 2-3, date2 has values of 20003 and 12345 (seconds) respectively, which are displayed as two separate times on the morning of January 1, 1960.
- In observation 1, datetime2 has a value of almost 2 billion (days), which is too far in the future for SAS to display- years are limited to four digits in most formats. We see nine asterisks because datetime2 is formatted as date9.: the number of asterisks displayed will correspond to the format.

We also see issues in the value comparison flags:

- In observation 1, datetimeflag = 'Y', suggesting that a value of nine asterisks is greater than 01JAN1960:05:33:16.
- In observation 2, dateflag = 'Y', suggesting that a value with a date of 01JAN1960 is greater than a value with a date of 05OCT2014.
- In observation 3, datetimeflag = 'N', suggesting that a value with a date of 22SEP2108 is not greater than a value with a date of 18MAY2023.

Assuming the dates are compared to dates and datetimes are compared to datetimes, SAS will compare numeric values accurately even when presented with different formats:

Comparing numeric dates of differing formats

Obs	date1	date2	date3	flag12	flag13	flag23	flag123
1	2019-06-30	2019-06-30	30JUN2019	Y	Y	Y	Y
2	1999-12-31	2000-01-01	01JAN2000			Y	
3	2016-04-08	2016-08-04	08APR2016		Y		

Table 5. Comparing Numeric Dates of Differing Formats

Each flag outputs "Y" when the dates compared are equal, otherwise null. We see in Table 5 above that each flag is accurate based on the dates provided. Comparing differently formatted dates and times only works with numeric dates/datetimes.

Issue 2: Levels of precision

We consider date and datetime values to have different **levels of precision** because the former counts days and the latter counts seconds. In cases where we need to compare numeric dates/datetimes, the **DATEPART()** function is convenient to store the date portion, as is the **TIMEPART()** function to store the time portion. We highlight them in the example below:

```
datepart = DATEPART(datetime1);
timepart = TIMEPART(datetime1);
```

Comparing numeric dates of differing precision

Obs	datepart	timepart	date1	datetime1	flag1	flag2
1	2019-06-30	12:00	2019-06-30	2019-06-30T12:00:00	Y	N

Table 6. Comparing Numeric Dates of Differing Levels of Precision

In Table 6 above, flag1 = “Y” if datetime1 > date1, otherwise “N”; flag2 = “Y” if datepart > date1, otherwise “N”. Flag1 represents the mistake of comparing a datetime to a date. Using the DATEPART() function to store the date portion of datetime1 enables an accurate comparison of the two dates, which is represented by flag2 = “N”.

CHARACTER VALUES

Character representations of date, time, and datetime values have the advantage over numeric values of being easier for the human reader to understand. However, these character values often involve greater effort than numeric values when you need SAS to understand them. We shall investigate three key issues that you may run into when comparing character date, time, and datetime values:

Issue 1: Formatting

When evaluating character representations of dates using the typical less-than, greater-than, and equal-to comparators, the same format must be used between all dates. In the example below, we discover the consequences of failing to adhere to this rule:

```
DATA c_mismatch1;
   LENGTH cdate1 $10 cdate2 $9 ctime1 $5 ctime2 $8;
   INPUT cdate1 cdate2 ctime1 ctime2;
   DATALINES;
      1987-05-19 19MAY1987 0:20  00:20:34
      2028-06-12 12JUN2028 1:23  01:23:20
      2042-02-19 19FEB2042 2:30  02:30:00
      2042-02-19 19FEB2042 11:06 11:06:40
      2069-07-07 07JUL2069 22:13 22:13:20
   ;
RUN;

DATA c_mismatch2;
   SET c_mismatch1;
   cdatetime1 = cdate1||"T"||ctime1;
   cdatetime2 = cdate1||"T"||ctime2;
   IF cdate1 > cdate2 THEN dateflag = "Y";
   IF ctime1 > ctime2 THEN timeflag = "Y";
RUN;
```

Incorrect comparisons - Character

Obs	cdate1	cdate2	ctime1	ctime2	cdatetime1	cdatetime2	dateflag	timeflag
1	1987-05-19	19MAY1987	0:20	00:20:34	1987-05-19T0:20	1987-05-19T00:20:34		Y
2	2028-06-12	12JUN2028	1:23	01:23:20	2028-06-12T1:23	2028-06-12T01:23:20	Y	Y
3	2042-02-19	19FEB2042	2:30	02:30:00	2042-02-19T2:30	2042-02-19T02:30:00	Y	Y
4	2042-02-19	19FEB2042	11:06	11:06:40	2042-02-19T11:06	2042-02-19T11:06:40	Y	
5	2069-07-07	07JUL2069	22:13	22:13:20	2069-07-07T22:13	2069-07-07T22:13:20	Y	

Table 7. Incorrectly Comparing Differently Formatted Character Values

We see in the code above that dateflag = “Y” when cdate1 > cdate2, and timeflag = “Y” when ctime1 > ctime2. However, because these operations are performed on character values of differing formats, we run into unanticipated results.

For example, in Table 7, Observation 2, what does it mean to say that “2028-06-12” > “12JUN2028”? These clearly refer to the same date, yet one is flagged as greater than the other. What is happening here? SAS is only noticing that the first digit of cdate1 is larger than the first digit of cdate2 to make this conclusion. Thus, dateflag is a junk variable.

Similarly, timeflag is only comparing digits: In Table 7, Observations 1-3, it sees a non-zero digit in ctime1 sooner than ctime2 and considers ctime1 “greater than” ctime2, and in Observations 4-5, ctime1 lacks the precision (number of digits) of ctime2, so ctime1 is considered “less than” ctime2. Once again: junk!

Issue 2: Sorting

Issues can also arise when sorting by formatted character dates, as demonstrated below by sorting Table 7 from the prior example:

Inaccurate sorting - by cdate1, ctime1

Obs	cdate1	cdate2	ctime1	ctime2	cdatetime1	cdatetime2	dateflag	timeflag
1	1987-05-19	19MAY1987	0:20	00:20:34	1987-05-19T0:20	1987-05-19T00:20:34		Y
2	2028-06-12	12JUN2028	1:23	01:23:20	2028-06-12T1:23	2028-06-12T01:23:20	Y	Y
3	2042-02-19	19FEB2042	11:06	11:06:40	2042-02-19T11:06	2042-02-19T11:06:40	Y	
4	2042-02-19	19FEB2042	2:30	02:30:00	2042-02-19T2:30	2042-02-19T02:30:00	Y	Y
5	2069-07-07	07JUL2069	22:13	22:13:20	2069-07-07T22:13	2069-07-07T22:13:20	Y	

Table 8. Previous Table Sorted by cdate1, ctime1

Table 8 is sorted by cdate1, then ctime1. cdate1 appears to be in chronological order. However, looking at Observations 3-4, the record where ctime1 = 11:06 is placed above the record where ctime1 = 2:30. On a typical workday, 11:06am may come before 2:30pm, but the time5. format writes hours and minutes in 24-hour time, so this is out of order. SAS sorts these character values such that “1” comes before “2”.

Inaccurate sorting - by cdate2, ctime2

Obs	cdate1	cdate2	ctime1	ctime2	cdatetime1	cdatetime2	dateflag	timeflag
1	2069-07-07	07JUL2069	22:13	22:13:20	2069-07-07T22:13	2069-07-07T22:13:20	Y	
2	2028-06-12	12JUN2028	1:23	01:23:20	2028-06-12T1:23	2028-06-12T01:23:20	Y	Y
3	2042-02-19	19FEB2042	2:30	02:30:00	2042-02-19T2:30	2042-02-19T02:30:00	Y	Y
4	2042-02-19	19FEB2042	11:06	11:06:40	2042-02-19T11:06	2042-02-19T11:06:40	Y	
5	1987-05-19	19MAY1987	0:20	00:20:34	1987-05-19T0:20	1987-05-19T00:20:34		Y

Table 9. Previous Table Sorted by cdate2, ctime2

Table 9 is sorted by cdate2, then ctime2. When we sort by cdate2, we find that the dates are out of order because SAS prioritizes the value of the numbered day and breaks ties on the first letter of the three-letter month. However, the times are placed in correct order for Observations 3-4 since the initial zero in “02:30:00” allows for correct sorting among character time values.

Accurate sorting - by cdatetime2

Obs	cdate1	cdate2	ctime1	ctime2	cdatetime1	cdatetime2	dateflag	timeflag
1	1987-05-19	19MAY1987	0:20	00:20:34	1987-05-19T0:20	1987-05-19T00:20:34		Y
2	2028-06-12	12JUN2028	1:23	01:23:20	2028-06-12T1:23	2028-06-12T01:23:20	Y	Y
3	2042-02-19	19FEB2042	2:30	02:30:00	2042-02-19T2:30	2042-02-19T02:30:00	Y	Y
4	2042-02-19	19FEB2042	11:06	11:06:40	2042-02-19T11:06	2042-02-19T11:06:40	Y	
5	2069-07-07	07JUL2069	22:13	22:13:20	2069-07-07T22:13	2069-07-07T22:13:20	Y	

Table 10. Previous Table Sorted by cdatetime2

Lastly, Table 10 sorts by cdatetime2, equivalent to a concatenation of cdate1 and ctime2- the two variables in the previous tables that sorted their respective values chronologically. We see that Table 10 is sorted in chronological order because the order of timing elements in cdatetime2 for each observation

is organized from largest to smallest. Thus, we may sort this data chronologically by year, month, day, hour, minute, and second, using cdatetime2 alone.

The format cdatetime2 uses is e8601dt., an ISO8601-compliant datetime format. It is strongly recommended to use ISO8601 formats when performing character-based date or datetime comparisons and sorts, as these formats ensure dates are evaluated in chronological order.

Issue 3: Levels of precision

Comparing levels of precision between character dates, times, and datetimes involves careful judgement. Recall that numeric values must consider days versus seconds. Character values, however, must consider any level of time measurement you can think of, from years to decimals of seconds.

As we saw in Tables 8-10, cdate1 and cdate2 represent dates of the same level of precision (year, month, day). If these two variables used the same ISO8601-compliant format, comparisons between them would be trustworthy.

Ctime1 and ctime2, however, compare times of different levels of precision (ctime1 = hour, minute. ctime2 = hour, minute, second). Even if ctime1 used a format with a two-digit hour, comparing ctime1 to ctime2 would prove inconclusive because ctime1 does not have a seconds value. We must only compare the hour and minute portion of ctime2 to ctime1. Similarly, we must only compare the date portion of cdatetime1 to the entirety of cdate1.

If the dates or times are of similar formats, but have different lengths or levels of precision, we follow the best practice of truncating all values to the length of the shortest value. We cannot definitively conclude that 22:13:20 > 22:13 if we do not know the seconds of the latter value. We may only compare the hour and minute portions of each field, and the result would be equal at the hour & minute level of precision. We observe this in Table 11 below:

Comparing character dates of differing precision

Obs	c_date1	c_datetime1	c_datepart	c_flag1	c_flag2
1	2019-06-30	2019-06-30T12:00:00	2019-06-30	Y	N

Table 11. Comparing Character Values of Differing Levels of Precision

C_flag1 = “Y” if c_datetime1 > c_date1, otherwise “N”. SAS considers c_datetime1 > c_date1 solely because c_datetime1 contains a time portion and c_date1 does not have a time portion. This flag does not accurately calculate what we need to know at the lowest level of precision.

Using the SUBSTR() function, we define c_datepart, equivalent to the date portion of c_datetime1. We also define c_flag2 = “Y” if c_datepart > c_date1, otherwise “N”. When we compare c_date1 to c_datepart, the values are equal at the year/month/day level of precision, thus c_flag2 = “N”.

The following example demonstrates a common method that you can use to compare values at the appropriate level of precision, using the MIN() and LENGTH() functions:

```
*mls = minimum lengths between pairs of values;
ml12 = MIN(LENGTH(cdate1),LENGTH(cdate2));
ml13 = MIN(LENGTH(cdate1),LENGTH(cdate3));
ml23 = MIN(LENGTH(cdate2),LENGTH(cdate3));

*lflags = flags for length-shortened values;
IF substr(cdate1,1,ml12) = substr(cdate2,1,ml12) THEN lflag12 = "Y";
ELSE lflag12 = "N";
IF substr(cdate1,1,ml13) = substr(cdate3,1,ml13) THEN lflag13 = "Y";
ELSE lflag13 = "N";
IF substr(cdate2,1,ml23) = substr(cdate3,1,ml23) THEN lflag23 = "Y";
ELSE lflag23 = "N";
```

```
*oflags = flags for original values;
IF cdate1 = cdate2 THEN oflag12 = "Y"; ELSE oflag12 = "N";
IF cdate1 = cdate3 THEN oflag13 = "Y"; ELSE oflag13 = "N";
IF cdate2 = cdate3 THEN oflag23 = "Y"; ELSE oflag23 = "N";
```

Using min() and length() functions to compare at correct precision

Obs	cdate1	cdate2	cdate3	ml12	ml13	ml23	lflag12	lflag13	lflag23	oflag12	oflag13	oflag23
1	2019-06-30	2019-06-30T12:30	2019-06-30T12:30:15	10	10	16	Y	Y	Y	N	N	N
2	2019-02-20	2019-02-20T12:30	2019-02-20T08:30:00	10	10	16	Y	Y	N	N	N	N

Table 12. Using Functions to Compare Character Dates at Lowest Level of Precision

Each row in Table 12 represents the same date values, cdate1 (date only), cdate2 (date with hours and minutes), and cdate3 (date with hours, minutes, and seconds). The minimum length of cdate1 is 10 when compared with cdate2 or cdate3, and the minimum length of cdate2 is 16 when compared with cdate3. Here, OFLAG represents direct comparison of these date values, while LFLAG shows a comparison of the values up to the length defined by the respective ML values. Thus, all OFLAGS are equal to “N”, while in Observation 1, all LFLAGs = “Y”, and Observation 2 has LFLAG23 = “N” because the time portions are different.

CONCLUSION

You may find these tips helpful when comparing dates, times, and datetimes:

- For numeric fields, only compare dates to dates, and datetimes to datetimes
- For character fields, ISO8601-compliant formatting is recommended for correct sorting
- Where possible, use date formats with a four-digit year
- Make sure your time values are formatted to use a two-digit hour
- Compare values at the lowest level of precision between and within all fields
- Retain the original data and store partial dates and times in new fields

RECOMMENDED READING

The easiest place to find most SAS date, time, and datetime formats are the SAS 9.4 Support pages titled “About SAS Date, Time, and Datetime Values” and “Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations”. These are both linked below or can be found by searching for the titles on your preferred online search engine.

SAS Help Center: About SAS Date, Time, and Datetime Values

SAS Help Center: Working with Dates and Times by Using the ISO 8601 Basic and Extended Notations

CONTACT INFORMATION

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

Patrick J. Mercer
Statistical Programmer II, Rho, Inc.
Patrick_mercer@rhoworld.com
LinkedIn https://linkedin.com/in/Patrick-mercer-675058156/