Working With SAS® System Date and Time Functions

Andrew H. Karp

Sierra Information Services, Inc.

19229 Sonoma Highway PMB 264

Sonoma, California 95476

Andrew@SierraInformation.com

www.SierraInformation.com

sas® Certified Professional Version 8

Thanks!

- Thanks for attending WUSS 12 and this presentation
- Please hold questions until the end of the presentation
- Paper in the Conference Proceedings
- Copy of these slides as PDF at
  - www.SierraInformation.com

Working with SAS System Date and Time Functions

- Many applications require that operations be performed on data collected in the time domain, such as:
  - determining the frequency with which a phenomenon of interest occurs in time
    - How many babies were born each day in January 2004?
  - determining the time interval which has elapsed between two phenomena
    - How many days elapsed between birth and discharge?

Working with
SAS System Date and Time Functions

– operating conditionally on observations in a SAS data set based on values of date and/or time variables

- create a SAS data set containing records for births in January 2004 from a larger data set containing birth records for all of 2004

– *aggregation* of observations from one time frequency to another

- from daily records, create a SAS data set containing monthly number of births

Working with
SAS System Date and Time Functions

– *interpolation* of higher frequency observations from data collected at a lower frequency

- · Estimate weekly number of births from a data set containing monthly counts
  - · Performed by **PROC EXPAND**, in the SAS/ETS module

SAS System Tools for Working With Data Collected in the Time Domain

- Functions
  - **create** SAS date, time or datetime variables from either raw data or from variables in an existing SAS data set
  - **determine** the interval between two periods
  - **declare** a SAS date or time variable as a constant
  - **extract** ‘parts’ from a SAS date variable, such as the month, day, or year

SAS System Tools for Working With Data Collected in the Time Domain

- *Formats*
  - **modify** the ***external representation*** of the **values** of SAS date, time or datetime **variables**
    - over 30 formats are available in Version 8, and users can create customized formats using PROC FORMAT
- *Informats*
  - **convert raw data** into SAS date, time or datetime variables

SAS System Tools for Working With Data Collected in the Time Domain

- Procedures
  - BASE
    - ***PLOT***
    - ***TIMEPLOT***
  - Econometrics and Time Series (ETS) Module
    - ***FORECAST***
    - ***ARIMA***
    - ***AUTOREG***
    - ***EXPAND***

Key Concept

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

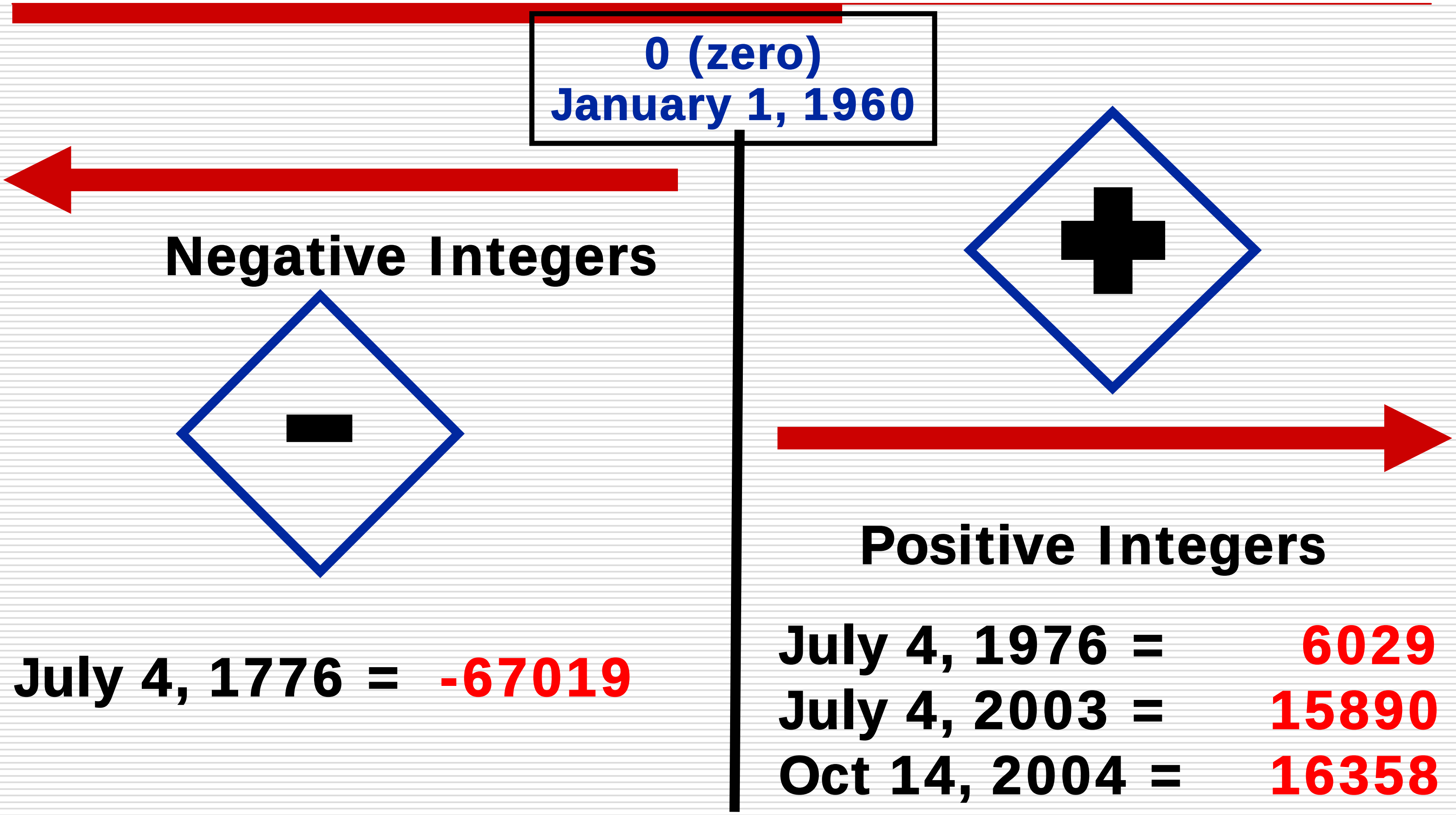
- *Date* variable:

  number of days from January 1, 1960
- *Time* variable:

  number of seconds from midnight
  - *A time variable is independent of a date variable*
- *Datetime* variable:

  number of seconds from midnight 1/1/1960

How The SAS System Stores the Values of Date Variables



0 (zero)
January 1, 1960

Negative Integers

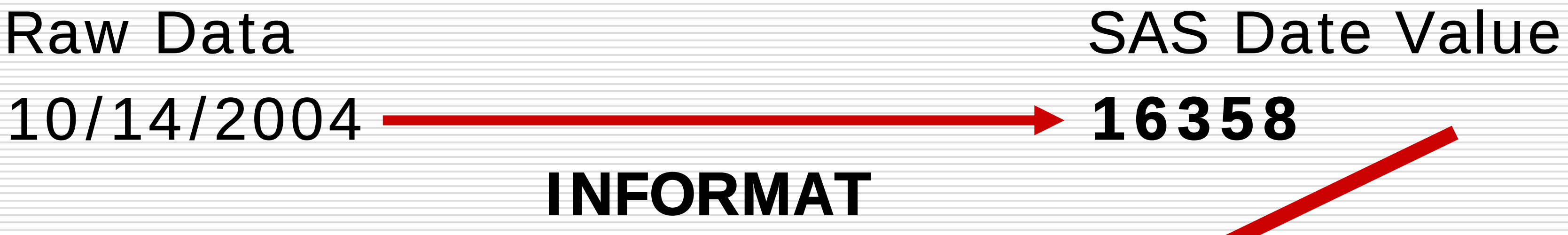
July 4, 1776 = -67019

Positive Integers

July 4, 1976 = 6029
July 4, 2003 = 15890
Oct 14, 2004 = 16358

SAS System Date Variables: Using an INFORMAT

Example: A raw data file contains a date variable in MMDDYY representation. A SAS date variable needs to be created in a data step which creates a SAS System data set



Raw Data

SAS Date Value

10/14/2004

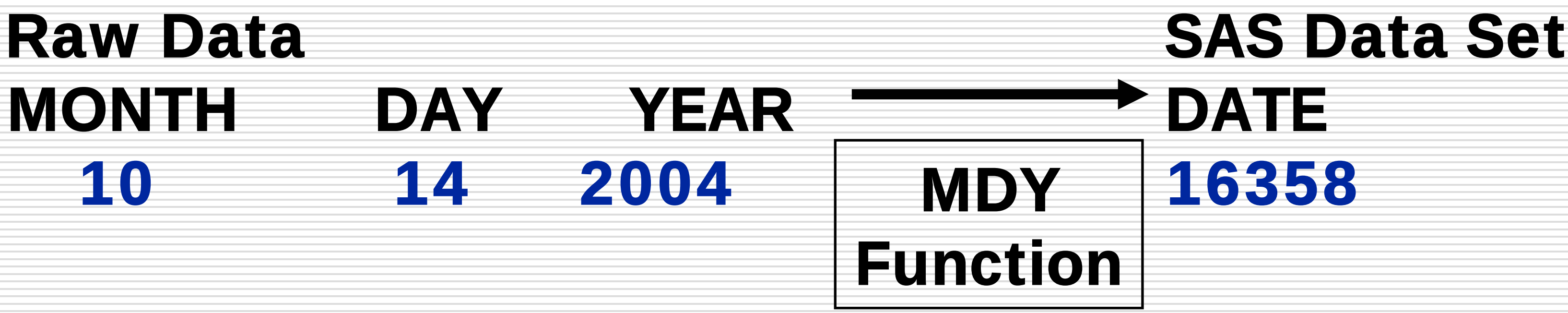
16358

INFORMAT

```
DATA MYDATA;
  INFILE  more SAS statements ;
INPUT @1 DATE MMDDYY10. ;
 more SAS statements
```

SAS System Date Variables: Using the MDY Function

- **Example: A data set contains separate variables for month, day and year. Create a SAS date variable using these values**



Raw Data

MONTH	DAY	YEAR
10	14	2004

MDY Function

SAS Data Set

DATE
16358

```
DATE = MDY(MONTH,DAY,YEAR);
```

SAS System Date Variables

Extracting the **"parts"** of a SAS System date variable using the:

- MONTH
- DAY
- YEAR
- QTR
- WEEKDAY

functions

Extracting the "parts" of a SAS Date, Time or Datetime Variable

Date: 16358 (October 24, 2004)

SAS Programming Statements:	**Results:**
A = MONTH(TODAY);	**10**
B = DAY(TODAY);	**14**
C = YEAR(TODAY);	**2004**
D = QTR(TODAY);	**4**
E = WEEKDAY(TODAY);	**5**

The WEEKDAY Function

- Returns a numeric value for the day of the week.
  - **1=Sunday**
  - 2=Monday
  - 3=Tuesday
  - 4=Wednesday
  - 5=Thursday
  - 6=Friday
  - **7=Saturday**

Extracting the “parts” of a SAS Date, Time or Datetime Variable

Extracting the ‘parts’ from a SAS System time variable using the

- – HOUR
- – MINUTE
- – SECOND

functions

Extracting the “parts” of a SAS Date, Time or Datetime Variable

Time: 1:30 pm (1300 hours, 0 seconds)

48600

SAS Programming Statements	Results
A = HOUR(TIME);	13
B = MINUTE(TIME);	30
C = SECOND(TIME);	00

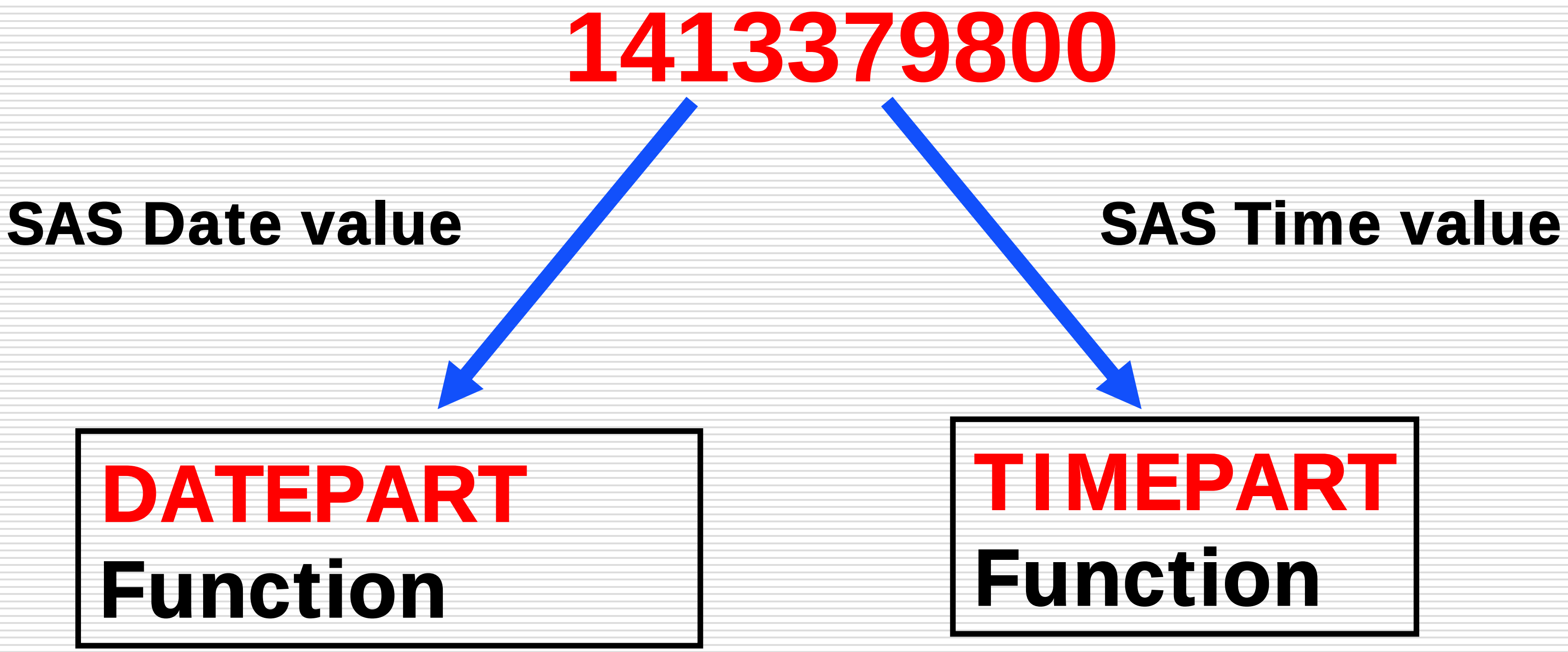
SAS System Datetime Variables

- □ A SAS System Datetime Variable contains information on both the date and time
  - the number of seconds before/after January 1, 1960

  Extracting the DATE and/or TIME ‘parts’ of a SAS datetime variable
  - **DATEPART** function
  - **TIMEPART** function

SAS System Datetime Variables

Thursday, Oct. 21, 2004 at 1300 hrs



1413379800

SAS Date value

SAS Time value

DATEPART Function

TIMEPART Function

SAS System Datetime Variables

Tues., Sept 21, 2004 at 0930 hrs

1413379800

SAS Programming Statement:	**Result:**
DATE = DATEPART(TODAY);	**16358**
TIME = TIMEPART(TODAY);	**48600**

SAS System Date Constants

- Declaring a SAS Date, Time or Datetime Constant

```
X = '14OCT2004'D;                    16358
Y = '13:30:00'T;                     46800
Z = '14OCT2004:13:30:00'DT;     1413379800
```

SAS Programming Language Functions

- There are several functions that return the values of dates and/or times from the system clock
  - **TODAY(), DATE()** Returns current date from system clock as a SAS date value
  - **DATETIME()** Returns the current date & time from system clock as a SAS datetime value
  - **TIME()** Returns the current time from system clock as a SAS time value

SAS System Date Constants

Question: How old am I, as of today?

SAS Programming Statements:

Constant

Result:

```
DAYS= TODAY() - ‘18DEC1956’D;            17467
```

As of 1330 hrs on Oct. 14, 2004

```
SECONDS =
DATETIME() - ‘18DEC1956:09:00:00’DT;
```

1509165000

External Representations of SAS System Date, Time and Datetime Variables

- Formats alter the *external representation* of SAS date, time or datetime variables.Over 30 such formats are available in the BASE SAS product
  - Examples: October 24, 2004 = **16335**

SAS Statement:	Result:
FORMAT TODAY MMDDYY8. ;	**09/21/04**
FORMAT TODAY YYMMDD8. ;	**04/09/21**
FORMAT TODAY DATE7. ;	**21SEP04**
FORMAT TODAY WEEKDATE29. ;	**Tuesday, September 21, 2004**

External Representations of SAS System Date, Time and Datetime Variables

Format Applied	Formatted_Value
No Format	16358
MMDDYY8. Format	10/14/04
MMDDYY10. Format	10/14/2004
DDMMYY10. Format	14/10/2004
DATE. Format	14OCT04
DATE7. Format	14OCT04
YYQ. Format	2004Q4
YEAR. Format	2004
WORDDATE. Format	October 14, 2004
WEEKDATE. Format	Thursday, October 14, 2004

Calculating Time Intervals

- Accomplished by one of two methods
  - **arithmetic operation** on SAS date, time or datetime variables, or between a variable and a constant

    YEARS = (date2-date1)/365.25;

    MONTHS = (date2-date1)/30.4;
  - use of the **INTCK function**

INTCK Function

- Determines the number of ***interval boundaries*** which have been ***crossed*** between two SAS date, time or date time variables

  INTCK(‘*interval*’ , *from* , *to)*

‘interval’ = character constant or variable name enclosed in single quotes representing the time period of interest

from = SAS date, time or datetime variable identifying the **start** of a time interval

to = SAS date, time or datetime variable identifying the **end** of a time interval

Arithmetic Operation vs. INTCK Function

Example: a child is born (and therefore ‘admitted’ to the hospital) on **December 28, 2003** and discharged on **January 2, 2004**. The child is therefore five days old at discharge.

```
AGE = ‘02JAN2004’D - ‘28DEC2003’D;
```

yields 5, which is the desired result

BUT, using the **INTCK** function

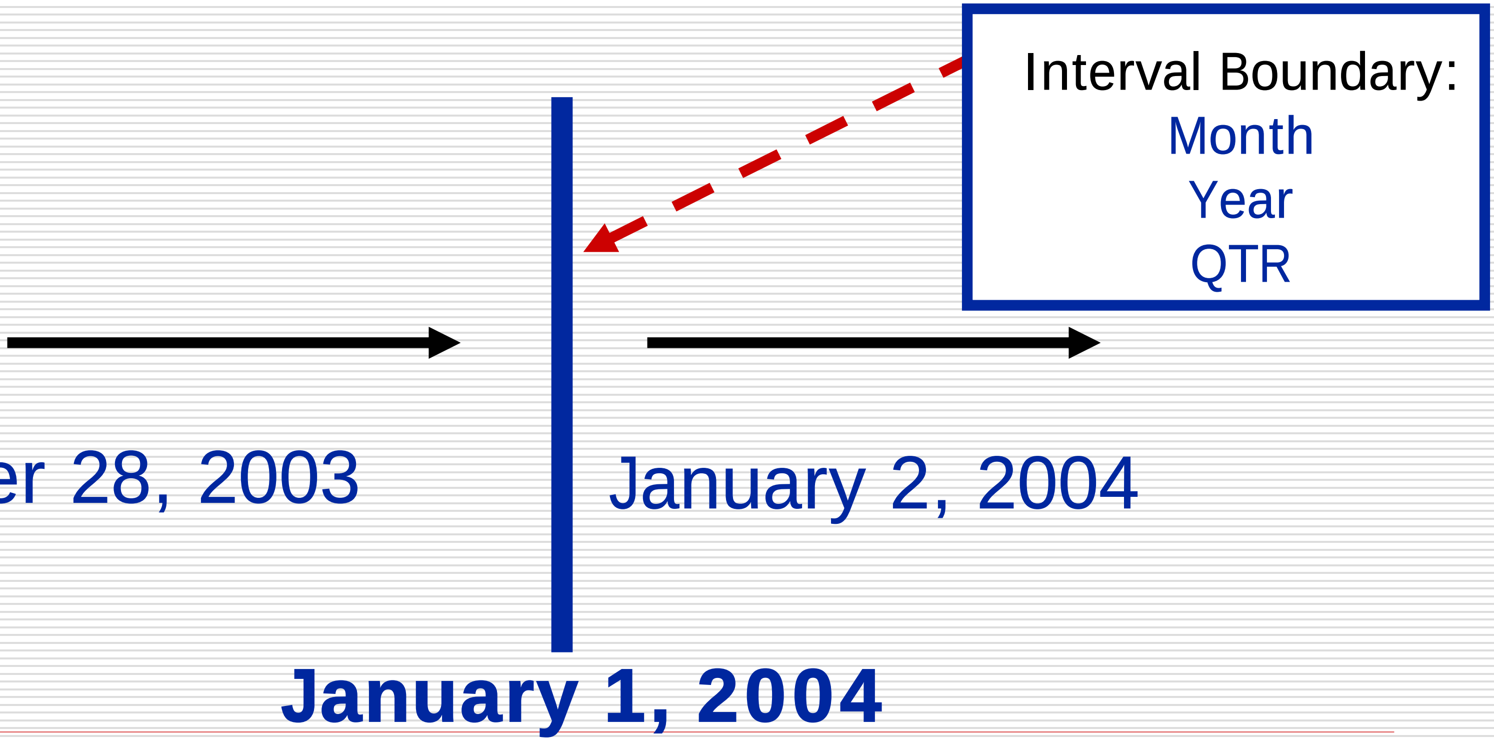
```
AGE =INTCK(‘YEAR’,’28DEC2003’D,’02JAN2004’D);
```

yields **1** as the result. Why?

How old is this child?

The INTCK Function vs. Direct Operation on SAS Date Variables

Child is born on December 28, 2003 and is discharged on January 2, 2004 How old is the baby?



Interval Boundary:
Month
Year
QTR

December 28, 2003

January 2, 2004

January 1, 2004

Enhancements to the INTCK Function in Release 6.07

- The INTCK Function counts the number of fixed time intervals. Several new intervals were added in Release 6.07 of SAS System software.
  - WEEKDAY
  - TENDAY
  - SEMIMONTH
  - SEMIYEAR

Using the WEEKDAY Argument to the INTCK Function

- Example One:
  - How many weekdays have elapsed between January 1, 2004 and October 14, 2004?

  -
    ```
    weekdays =
    intck('weekday','01jan2004'd,today());
    ```
  - Result: **205** weekdays have elapsed

Using the WEEKDAY Argument to the INTCK Function

- Example Two:
  - How many days have elapsed between Jan 1, 2004 and October 14, 2004, ***excluding Sundays?***

  -
    ```
    except_sun =
    intck('weekday1w','01jan2004'd,today());
    ```

Result: 246

WEEKDAY1W as the alignment argument instructs SAS to consider Sunday (weekday value 1) as the only day in the weekday period. WEEKDAY17W is the same as specifying the WEEKDAY argument.

The INTNX Function

- Creates a SAS date, time or datetime value that is a given number of time intervals from a starting value

  INTNX('interval',*from*,*to*)

Example: on what date should a postcard be sent to the parents of a newborn asking them to schedule a three-month well-baby exam, three months after the child's birthday (BDATE)?

MAILDATE = **INTNX**('month',BDATE,3)

the result is a SAS date variable representing the first day of the month which is three months past the child's birthday

The INTNX Function: Enhancements in Release 6.11

- By default, the INTNX Function will return a SAS date value representing the beginning date of the interval supplied in the first argument.
  - Starting in Release 6.11, the INTNX Function contains options which allow you to specify the *alignment* of the interval.
    - The alignment arguments are:

END	Aligns date to the *end*
MIDDLE	Aligns date to the *midpoint*
BEGINNING	Default alignment

Why are the New Alignment Operators So Useful?

- □ Suppose we want to create a SAS date variable representing the last day of the month in which a transaction occurred.
- □ Since there are some months with 30 days, others with 31 days and one month with either 28 or 29 days, a data step approach is tedious and time consuming.

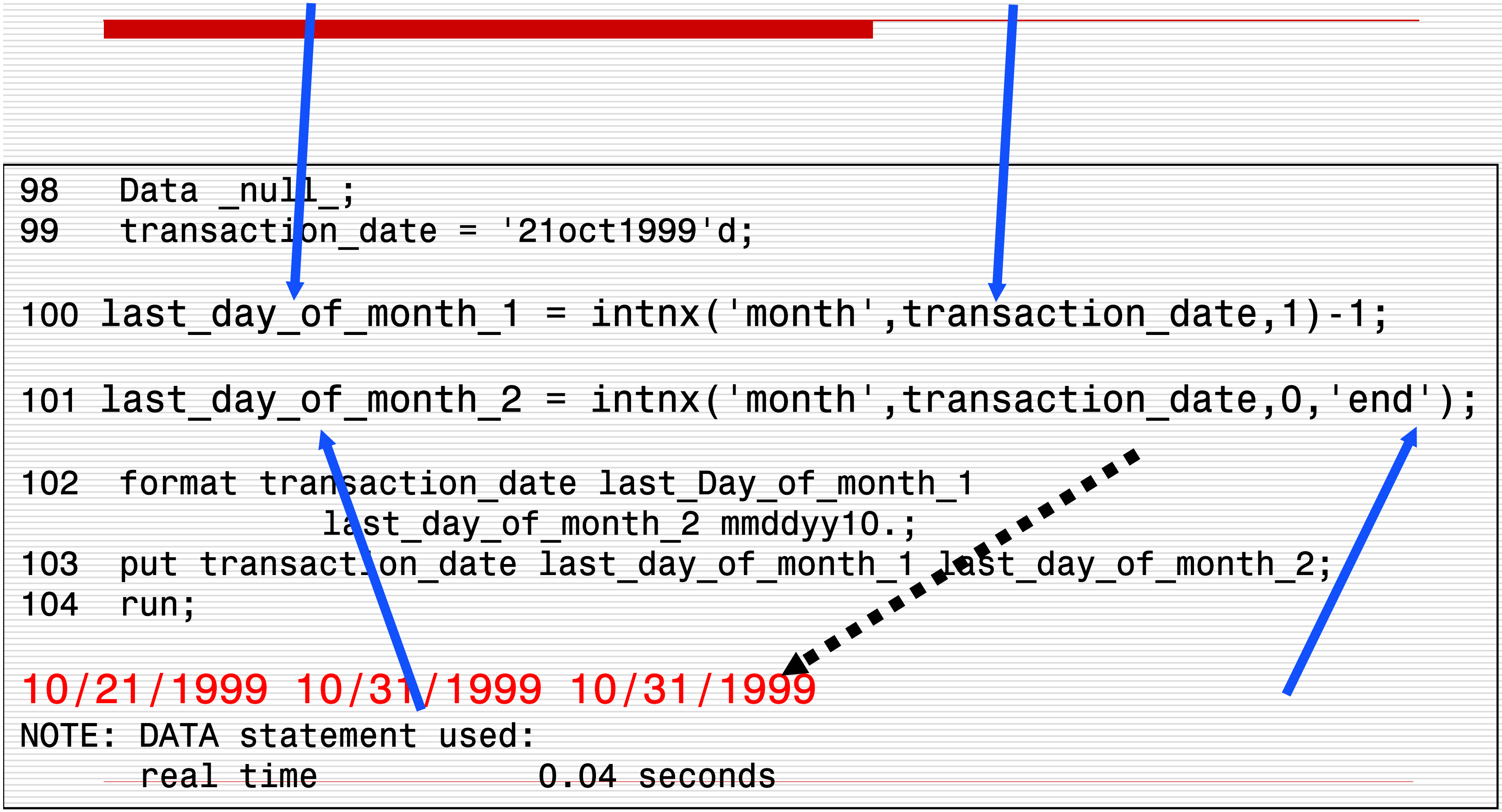
Determining the Last Day of the Month via the Data Step

```
112  data _null_;
113    trans_date = '15oct2002'd;
114
115      if month(trans_date) in (9,4,6,11) then
116    trans_month = mdy( month(trans_date),30,year(trans_date) );
117
118      else if month(trans_date) = 2 then
119    trans_month = mdy( month(trans_date),28,year(trans_date) );
120
121 else trans_month =
       mdy( month(trans_date),31,year(trans_date) );
122
123      format trans_month mmddyy10.;
124
125      put 'TRANSACTION MONTH:' trans_month;
126
127      run;
TRANSACTION MONTH:10/31/2002
```

Using the New Alignment Intervals in the INTNX Function

- Example: Create a SAS date value representing the last day of the month in which a transaction occurred
  - We want to then count the total number of transactions that occurred in that month
  - Two approaches:
    - Using the INTNX Function and subtraction
    - Using the INTNX Function and the *END alignment argument*

Using the New Alignment Intervals in the INTNX Function



```
98   Data _null_;
99   transaction_date = '21oct1999'd;

100 last_day_of_month_1 = intnx('month',transaction_date,1)-1;

101 last_day_of_month_2 = intnx('month',transaction_date,0,'end');

102  format transaction_date last_Day_of_month_1
                last_day_of_month_2 mmddyy10.;
103  put transaction_date last_day_of_month_1 last_day_of_month_2;
104  run;

10/21/1999 10/31/1999 10/31/1999
NOTE: DATA statement used:
      real time                 0.04 seconds
```

Using PROC EXPAND with SAS Date Variables

- Very useful procedure for working with data collected in the time domain
- In the **SAS/ETS** module
  - Substantial enhancements in Releases 6.06 and 6.12 of **SAS/ETS** software
- creates SAS data sets

Using PROC EXPAND with Data Forming a Time Series

- □ **Replaces** missing values in a time series
  - □ **applies a cubic spline function is apply to obtain the continuous time approximation of the missing values to be replaced**
- □ **Aggregates** data from a higher to a lower sampling frequency (e.g., daily to monthly)
  - □ **By default, missing value replacement is carried out on observations before aggregation**
- □ **Interpolates** data from a lower to a higher sampling frequency (e.g., monthly to daily)
  - □ **By default, missing value replacement is carried out on observations before interpolation**

Using PROC EXPAND with Data Forming a Time Series

- □ Transformation operators
  - □ Can be applied to a series before it is operated upon by PROC EXPAND, or to the series AFTER it has been processed by the procedure
    - ■ Often eliminates the need for intensive data step manipulation of time series
  - ■ Backward, Centered and Cumulative Moving
    - ■ Averages, sums
    - ■ moving sum
    - ■ maximum, minimum
    - ■ median, range

Using PROC EXPAND with Data Forming a Time Series

- Benefits to using **PROC EXPAND**
  - **Clean up your series** before applying other procedures to it
  - **Avoid data step** manipulation of a series prior to aggregation or interpolation
  - **Eliminate tedious data step** coding for complex inter-observation handling of time series values

Two Problems

- How can we
  - Round the values interpolated by PROC EXPAND?
    - Solution: **TRANSFORMOUT** Option
  - Change the weekly series start day from Sunday to Monday?
    - Solution: Use **SAS Date Alignment Operators**

Solutions

```
proc expand data=sasclass.carmiss
             observed = total out=weekly
                   from = month
                   to = week.2;
id date;
convert cars = weekly/transformout=(ceil);
run;
```

Solutions

Effect of applying the Week.2 alignment operator. "Week Starting on Monday"

```
PROC EXPAND
Interpolation from Monthly to Weekly
Default Missing Value Replacement
Weekly Series Starts on Monday
CEIL Transformation Operator Used to Round Results

Obs                 DATE    weekly
276       Mon, 7 Oct 2002     8721
277      Mon, 14 Oct 2002     8582
278      Mon, 21 Oct 2002     8623
279      Mon, 28 Oct 2002     8821
280       Mon, 4 Nov 2002     9134
281      Mon, 11 Nov 2002     9517
282      Mon, 18 Nov 2002     9927
283      Mon, 25 Nov 2002    10320
284       Mon, 2 Dec 2002    10651
285       Mon, 9 Dec 2002    10878
286      Mon, 16 Dec 2002    10957
287      Mon, 23 Dec 2002    10843
288      Mon, 30 Dec 2002    10509
```

Effect of Using the CEIL [ceiling] operator in the TRANSFORMOUT Option

Another Example of SAS Date Alignment Operators

- SAS Date Alignment Operators are Documented in the SAS/ETS Software Manual
  - Example: Align a monthly series to a fiscal year starting in June

Another Example of SAS Date Alignment Operators

```
ods date number;
ods listing close;
ods rtf file = 'c:\reporting classes 2003\expand.rtf'
                  style = analysis bodytitle;
* calendar vs. fiscal year aggregation;

proc expand data=sasclass.carmiss
               from = month to = year out = cal_year;
id date;
convert cars = cal_year/transformout=(ceil);
run;
proc print data=cal_year;
format cal_year comma12.;
title1 'PROC EXPAND';
title2 'Aggregation from Month to Calendar Year';
title3 'Default Missing Value Replacement';
title4 'CEIL Transformation Operator Used to Round Results';
run;
```

Another Example of SAS Date Alignment Operators

PROC EXPAND
Aggregation from Month to Calendar Year
Default Missing Value Replacement
CEIL Transformation Operator Used to Round Results

Obs	*DATE*	*cal_year*
1	1998	48,676
2	1999	53,400
3	2000	58,600
4	2001	56,919
5	2002	59,097

Another Example of SAS Date Alignment Operators

```
proc expand data=sasclass.carmiss
                                            from = month to = year.6
                              out = fiscal_year;
id date;
convert cars = fiscal_year/transformout=(ceil);
run;

proc print data=fiscal_year label split = '/';
label fiscal_year = 'Fiscal/Year/Starting in June';
format fiscal_year comma12.;
title1 'PROC EXPAND';
title2 'Aggregation from Month to Fiscal Year Starting in June';
title3 'Default Missing Value Replacement';
title4 'CEIL Transformation Operator Used to Round Results';
run;
ods rtf close;
ods listing;
```

Another Example of SAS Date Alignment Operators

PROC EXPAND
Aggregation from Month to Fiscal Year Starting in June
Default Missing Value Replacement
CEIL Transformation Operator Used to Round Results

Obs	*DATE*	*Fiscal Year Starting in June*
1	JUN1997	55,833
2	JUN1998	35,619
3	JUN1999	39,144
4	JUN2000	38,333
5	JUN2001	42,941
6	JUN2002	38,515

Understanding the YEARCUTOFF SAS System Option

- Specifies the first year of the 100-year span that is used to determine the century of a two-digit year
  - Valid values are from 1582 to 19990
    - Not supported in Version 5 or Release 6.03/6.04
  - Default value is 1900 in Version 6
  - In Version 8 and SAS 9 the “shipped default” value is 1920

Understanding the YEARCUTOFF SAS System Option

- If you change the **YEARCUTOFF** System Option Value to 1900, all two digit year values are assumed to have occurred in the 1900’s
- What happens if you change the YEARCUTOFF value to 1950?
  - Two digit year values from 00 to 49 will be treated as if they occurred from 2000 to 2049
  - Two digit year values from 50 to 99 will be treated as if they occurred from 1950 to 1999

Version 8 Enhancements

- **DATDIF** and **YRDIF** Functions
  - Returns the number of days (DATDIF) or years (YRDIF) between two dates.
  - Similar to the INTCK Function
    - But, behavior is different
    - More options

Version 8 Enhancements

- DATDIF(start_date,end_date,basis)
- YRDIF(start_date,end_date,basis)
  - Start_date: period start date
  - End_date: period end date (not included in the calculations)

BASIS	*Result*
Act/Act	*Uses actual number of days*
Actual	*Same as Act/Act*
30/360	*Assumes 30 day months and 360 day years*
Act/360 (YRDIF Only)	*Years=actual # of days/360 day year*
Act/365 (YRDIF Only)	*Years=actual # of days/365 day year*

Version 8 Enhancements

```
data years;
  start = '18dec1956'd;
  stop  = '31oct2001'd;

intyears = intck('year',start,stop);
approxyears = (stop-start)/365.25;
difyears_act = yrdif(start,stop,'actual');
difyears_360 = yrdif(start,stop,'act/360');
difyears_365 = yrdif(start,stop,'act/365');
* put it in the sas log;
put start = stop= ;
put intyears= approxyears= ;
put difyears_act = difyears_360 = difyears_365 = ;
run;
```

Version 8 Enhancements

The YEARDIF Function

```
start=-1109
stop=15628
intyears=46
approxyears=45.823408624
difyears_act=45.824552736
difyears_360=46.491666667
difyears_365=45.854794521
```

Version 8 Enhancements

- The separator character can now be specified with the
  - DDMMYY
  - MMDDYY
  - YYMMDD Formats

Version 8 Enhancements

```
data x;
nov5 = '05nov2001'd;
put " " nov5 mmddyy10.;
put "B " nov5 mmddyyb10.;
put "C " nov5 mmddyyc10.;
put "P " nov5 mmddyyp10.;
run;
```

Version 8 Enhancements

```
969  nov5 = '05nov2001'd;
970  put " " nov5 mmddyy10.;
971  put "B " nov5 mmddyyb10.;
972  put "C " nov5 mmddyyc10.;
973  put "P " nov5 mmddyyp10.;
974  run;
```

```
 11/05/2001
B 11 05 2001
C 11:05:2001
P 11.05.2001
```

Version 8 Enhancements

- New Date Formats in Release 8.2
  - DTWKDATX
    - Displays the Weekday, Month, Day and Year of a Datetime Variable
  - DTDATE
    - Displays the Day, Month and Year of a Datetime Variable
- Eliminate the need to use the DATEPART function in the Data Step to “extract” the date “part” from a SAS datetime variable.

Version 8 Enhancements

```
data _null_;
datetime1 = '15sep2003:08:00:00'dt;
datetime2 = datetime1;
format   datetime1 dtwkdatx.
         datetime2 dtdate.;
put datetime1= datetime2=;
run;
```

```
datetime1=Friday, 21 November 2003
datetime2=21NOV03
```

Version 8 Enhancements

- The %SYSFUNC (system function) capability was added in V6 and can be used to execute SAS Functions within the SAS Macro Facility
  - Example: Format today's date in the TITLE
    - Use the %SYSFUNC facility and the DATE() SAS Programming Language Function
    - Apply the WORDDATE. Format

Version 8 Enhancements

```
options nocenter ls=80;
proc print data=mwsug.cardtrans(obs=10);
title1 'MWSUG 2003 Conference: Minneapolis';
title2 "Report Prepared on %SYSFUNC(date(),worddate.)";
title3 "Report Run Time:  %SYSFUNC(time(),time.)";
run;
```

Version 8 Enhancements

```
MWSUG 2003 Conference: Minneapolis       22:53 Sunday, September 14, 2003   4
Report Prepared on September 15, 2003
Report Run Time:   6:50:39

       trans_ card_                       charge_   transaction_
   Obs  type  type      cardnumber         amount      datetime      year LOB01

     1    2    12   9630-0420-1039-0391   452.34 08APR99:23:59:59 1999    0
     2    1    12   9630-0420-1039-0391   955.20 24MAY99:23:59:59 1999    0
     3    1    12   9630-0420-1039-0391   413.37 03AUG99:23:59:59 1999    0
     4    1    12   9630-0420-1039-0391   795.78 12AUG99:23:59:59 1999    0
     5    1    12   9630-0420-1039-0391   898.87 07SEP99:23:59:59 1999    0
     6    1    12   9630-0420-1039-0391   361.14 11SEP99:23:59:59 1999    0
     7    1    12   9630-0420-1039-0391    93.42 14OCT99:23:59:59 1999    0
     8    1    12   9630-0420-1039-0391   386.35 28DEC99:23:59:59 1999    0
     9    1    12   9630-0420-1039-0391   409.61 05JAN00:23:59:59 2000    0
    10    1    12   9630-0420-1039-0391   190.32 11JAN00:23:59:59 2000    0
```

New in SAS 9.1

- **DTRESET** SAS System Option
  - Useful with long SAS jobs. Re-sets the date and time information displayed in the SASLOG.
    - Smallest increment is minutes
    - Reset occurs when page is being written
  - Default is **NODTRESET**
    - SASLOG displays date/time of session initialzation
- ANYDATE, ANYTIME Informats

New in SAS 9.1

- ANYDTDTEw
  - Reads and extract date values from “messy” raw data.
  - DATE, DATETIME, DDMMYY, JULIAN, MMDDYY, MONYY, TIME, YYMMDD, YYQ informat values
- ANYDTDTw
- ANYDTMEw

New in SAS 9.1

```
data messy;
input date anydtdte21.;
date2 = date;
datalines;
14JAN2004
14JAN2004 12:24:32.8
14012004
2004014
01142004
20040114
04Q1
;
run;
```

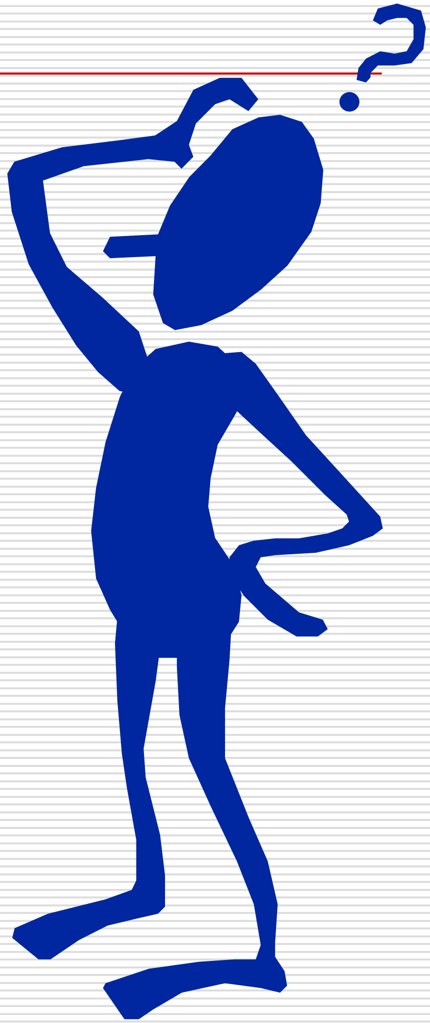
New in SAS 9.1

```
Working with SAS Date and Time Functions
The ANYDTDTE Informat, New in SAS 9.1

Obs      date             date2

 1      16084     January 14, 2004
 2      16084     January 14, 2004
 3      16084     January 14, 2004
 4      16084     January 14, 2004
 5      16084     January 14, 2004
 6      16084     January 14, 2004
 7      16071      January 1, 2004
```

New in SAS 9.1

- The WEEK Function
  - Returns the week number from a SAS date
    - Default: First Sunday of the week is WEEK 1
      - For dates prior to the first Sunday, the value returned by the WEEK Function is 0 (zero)
  - Optional Descriptors
    - Change the basis for determining the first week of the year
      - See the SAS 9.1 Documentation for the WEEK Function

Thanks for Attending !



Questions ?

Comments ?

Copies of this presentation:

www.SierraInformation.com