# Working with DATEs in SAS

## How does SAS store the dates?

- When once SAS knows it is reading a date, it converts the date into the **Number of Days from January 1, 1960**.
  - ✓ Dates after January 1, 1960 are stored as positive numbers
    **For example:** **01/01/2001 is stored as 14976**
  - ✓ Dates before January 1, 1960 are stored as negative numbers.
    **For example:** **01/01/1959 is stored as -365**

- SAS uses ***INFORMATS*** for reading dates,
  ***FUNCTIONS*** for manipulating dates
  ***FORMATS*** for printing dates.

- **CONSTANT DATE**:

  We can use a date as a constant in a SAS expression by adding ***QUOTES*** & a letter D.
  **For example:** **semesterstartdate = ‘13JAN2020’D;**

## YEARCUTOFF option in SAS

- **YEARCUTOFF is used to select a range of 100 years.**
- **The default value for the YEARCUTOFF is 1920. That is, any 2-digit year will be treated as a 4-digit year between 1920 and 2019.**
  - **Suppose the date value is 03/10/57. SAS takes this year as 1957 since the default value for YEARCUTOFF is 1920**
- **Suppose we are using a two-digit year. Then, we can select the range of 100 years.**
  - **For example:**
    **If the two-digit year falls within the range of 1905 to 2004, then the following SAS statement is needed in the program:**
    **options yearcutoff = 1905;**
- **Note that any year entered as a 4-digit year will not be affected by the YEARCUTOFF option.**

# Working with DATEs in SAS

## **The following examples illustrate the INFORMATS and FORMATS necessary to read as well as print the dates**

Following are various **informats** that can be used to **read** the date values:

| **SAS informat to read dates** | **Example of input data** |
|---|---|
| mmddyy6. | 112195 |
| ddmmyy6. | 211195 |
| yymmdd6. | 951121 |
| mmddyy8. | 11211995 |
| mmddyy8. | 11/21/95 |
| ddmmyy8. | 21111995 |
| yymmdd8. | 19951121 |
| mmddyy10. | 11/21/1995 |
| date7. | 21NOV95 |
| date9. | 21NOV1995 |
| monyy5. | NOV95 |
| monyy7. | NOV1995 |

Suppose we want to **print** all these dates in the form, **mm/dd/yyyy**, then the **FORMAT** required is **mmddyy10.**

Hence, the SAS statement will be

**format DateVariableName mmddyy10.;**

where "**DateVariableName**" is the name of the actual SAS date variable.

If all DATE variables have to be printed in the same format as mm/dd/yyyy, then list all variable names between the words "**format**" and "**mmddyy10.**".

# Working with DATEs in SAS

## How to read data entered on multiple lines for each person

**Sometimes we get the data set in which each person's data may have been entered on multiple lines.**

**The following program illustrates how to read such data sets using SAS without having to reenter the data**

**Suppose we add a *second line of data* with two variables *GENDER* and *WEIGHT*, *third line of data* with *SMOKE* and *DRINK*, and the *fourth line of data* with the variable, *HRATE*.**

**NOTE: The method of data entered as described in the above two lines should be the same for every person.**

**That is, every person must have the data for the variables:**
**NAME, AGE, BIRTH, SPORTS, ADMITDAY, DXDATE and RELDATE on the first line;**
**GENDER and WEIGHT on the second line;**
**SMOKE and DRINK on the third line;**
**HRATE on the fourth line.**

**EXAMPLE11.SAS**

```
options nodate nonumber;

data example11;
input name $ age birth $char12. sports $
        admitday mmddyy8. dxdate mmddyy6. reldate date7.
/ gender  $ weight /  smoke $  drink $ / hrate ;

format admitday mmddyy10. dxdate mmddyy8. reldate date9.;
datalines;
Victor 47 Mar 10 1955 Baseball 3101955 031055 10mar55
Male 180
yes no
70
Julie  34 Jan 20 1968 Tennis 1201968 012068 20jan68
female 120
no no
65
;
proc print data=example11;
      title 'Data Tabulation from EXAMPLE11 data set';
run;
```

## REMARKS:

**The symbol, / (back-slash) tells SAS to go to the next line of the raw data before reading the next variable.**

**If there are 3 lines of data per observation, then there should be two /'s in the INPUT statement, and three /'s if 4 lines of data, and so on.**

# Working with DATEs in SAS

## Example of Reading Dates in a variety of forms

**The purpose of EXAMPLE12.SAS is to illustrate the use of various built-in functions of SAS.**

**In this program, we use the following SAS built-in functions:**

- ✓ **DAY**
- ✓ **MONTH**
- ✓ **YEAR**
- ✓ **MDY**
- ✓ **INT**
- ✓ **ROUND**

**As an example: Suppose 05/20/2017 is entered into the variable, DATE1. Then,**

- **DAY will extract only the "day" part of a full date.**
- **MONTH will extract only the "month" part of a full date.**
- **YEAR will extract only the "year" part of a full date.**
- **MDY will combine three variables, MONTH, DAY & YEAR to form one variable with complete date.**
- **INT function retains only the INTEGER part of the result**
- **ROUND function rounds off the value to one decimal digit or two decimal digit depending upon how many decimal digits you want to keep**
- **CONSTANT DATE: How to define a constant date.**
- **YRDIF function computes the difference between two dates.**

**For example:**

```
age = YRDIF(todaysdate - dateofbirth);
```

**This computes the age as of today.**

# Working with DATEs in SAS

**EXAMPLE12.SAS**

```
data example;
      input @1 semesterstartdate mmddyy6.
                  @8 date2 mmddyy8.
                  @17 date3 date7.
                  @25 date4 julian5.
                  @31 date5 ddmmyy6.
                  @38 date6 mmddyy8.
                  @47 day 2.
                  @50 month 2.
                  @53 year 4.;
format SEMESTERSTARTDATE DATE2 - DATE5 mmddyy8. date6 mmddyy10.;
                  ** This FORMAT statement is used to PRINT the DATES using a
                  specified formats such as mmddyy8, mmddyy10, or WORDDATE
                  with "slashes";
datalines;
102101 10/21/46 21oct46 46294 211046 10211946 17 09 1980
122502 12/25/96 25dec96 96360 251296 12251996 30 11 1982
122503 12/25/98 25dec98 96360 251296 12251899 25 12 1984
;
run;
proc print data=example;
      title 'Tabulation of data from EXAMPLE12 data set';
      title2 'Dates were read in a variety of formats';
run;
*****************************************************;
data example12;
      set example;
*************** ASSIGNMENT statements to create new variables**************;
day1 = day(semesterstartdate);
                  ** The newly created variable, DAY1, will contain only the
                  DAY part of the variable SEMESTERSTARTDATE;

month1 = month(semesterstartdate);
                  ** The newly created variable, MONTH1, will contain only
                  the MONTH part of the variable SEMESTERSTARTDATE;

year1 = year(semesterstartdate);
                  * The newly created variable, YEAR1, will contain only the
                  YEAR part of the variable SEMESTERSTARTDATE;

Date_of_birth = mdy(month,day,year);
                  ** This MDY function combines the three different
                  variables, MONTH, DAY, and YEAR to form one single
                  variable, DATE_OF_BIRTH;

age = (semesterstartdate - date_of_birth)/365.25;
                  ** This computes the age in years between DATE_OF_BIRTH &
                  SEMESTERSTARTDATE, stores the age (with 4 decimal places)
                  in the newly created variable, AGE1.
                  (For example: AGE = 21.0924  for the first person in the
                  data set);
```

# Working with DATEs in SAS

```
age1 = yrdif(date_of_birth, semesterstartdate);
                  ** This computes the age in years between DATE_OF_BIRTH &
                  SEMESTERSTARTDATE, stores the age (with 4 decimal places)
                  in the newly created variable, AGE1_A.
                  (For example: AGE1 = 21.0932 for the first person in the
                  data set);

integer_part_of_age1 = int(age1);
                  ** The INT function applied on the variable, AGE1, gives
                  only the INTEGER part of the data and ignores the decimal
                  part. This INTEGER part of the data will be stored in a
                  newly created variable, INTEGER_PART_OF_AGE1.
                  (For example: INTEGER_PART_OF_AGE1 = 21  for the first
                  person in the data set);

age1_nearest_tenth = round(age1, 0.1 );
                  ** The ROUND function applied on the variable, AGE1, rounds
                  off AGE1 to the nearest tenth and stores it in a newly
                  created variable, AGE1_NEAREST_TENTH.
                  (For example: AGE1_NEAREST_TENTH = 21.1 for the first
                  person in the data set);

Age1_nearest_year = round(age1, 1 );

                  ** The ROUND function applied on the variable, AGE1, rounds
                  off AGE1 to the nearest year and stores it in a newly
                  created variable, AGE1_NEAREST_YEAR.
                  (For example: AGE1_NEAREST_YEAR = 21  for the first person
                  in the data set);

* To compute age as of a specific date, say January 13, 2020, use the
following SAS statement;

num_years = int (('13JAN20'D - semesterstartdate) / 365.25);

                  ** NUM_YEARS is the number years between a FIXED Date,
                  January 13, 2020, and SEMESTERSTARTDATE, with only INTEGER
                  part being stored in the newly created variable, NUM_YEARS.
                  (For example: NUM_YEARS = 15 for the first person in the
                  data set);

format SEMESTERSTARTDATE DATE2 - DATE5 mmddyy8. date6 mmddyy10. Date_of_birth
worddate.;
                  ** This FORMAT statement is used to PRINT the DATES using a
                  specified formats such as mmddyy8, mmddyy10, or WORDDATE
                  with "slashes";

proc print data=example12;
      title 'Tabulation of data from EXAMPLE12 data set';
      title2 'Dates were read in a variety of formats';
run;
```

# Working with DATEs in SAS

**EXAMPLE12A.SAS**

## Suppose the data set contains a mix of 2-digit and 4-digit years. Reading such dates using proper INFORMAT

**In the following SAS program, note that some DATES have been entered with a 2-digit year and other with a 4-digit year.**

**In order to READ all dates correctly, use the informat, DATE9.**

**To PRINT all dates in the format mm/dd/yyyy, use the following format statement,**
**FORMAT DEPARTURE_DATE MMDDYY10.;**

```
options yearcutoff=1920;
```

**Note the use of YEARCUTOFF = 1920.**

```
data example12a;
      input country $ 1 - 12 @13 departure_date date9. nights;
      format departure_date mmddyy10.;
datalines;
Japan       13may2000  8
Greece      17oct99   12
New Zealand 03feb2014 16
Brazil      28feb2015  8
Venezuela   10nov00    9
Italy       25apr2001  8
USSR        03jun1997 14
Switzerland 14jan2001  9
Australia   24oct98   12
Ireland     27mar2015  7
;
run;
**************************************************;
proc print data=example12a;
   title 'Dates Using the DATE9. Informat';
   title2 'Printed as 4-Digit Calendar Dates';
   format Departure_Date mmddyy10.;
run;
*************************************************;
```

| Obs | country | departure_date | nights |
|---|---|---|---|
| 1 | Japan | 5/13/2000 | 8 |
| 2 | Greece | 10/17/1999 | 12 |
| 3 | New Zealand | 2/3/2014 | 16 |
| 4 | Brazil | 2/28/2015 | 8 |
| 5 | Venezuela | 11/10/2000 | 9 |
| 6 | Italy | 4/25/2001 | 8 |
| 7 | USSR | 6/3/1997 | 14 |
| 8 | Switzerland | 1/14/2001 | 9 |
| 9 | Australia | 10/24/1998 | 12 |
| 10 | Ireland | 3/27/2015 | 7 |

# Working with DATEs in SAS

EXAMPLE12B.SAS;

## Suppose the data set contains DATES entered in both American and European (or Non-American) formats.

**Multi-center clinical trials involving USA and other countries are common and hence, the dates collected are in different formats (American and non-American)**

**The following program converts Non-American format into American format**

```
**********************************************************************
* Suppose we have dates entered in American and Non-American format,
  for example, in a multi-center clinical trials.

* First, we READ the two datasets separately, and then use SET statement to
  combine them into a single file which will have dates in the AMERICAN
  format.
**********************************************************************;

data one;
            * DATES are entered in AMERICAN Format;
input id age dob mmddyy6.;
format dob mmddyy10.;
datalines;
1 23 123106
2 45 011068
3 52 031055
4 12 073096
;
run;
data two;
            * DATES are entered in EUROPEAN Format;
input id age dob ddmmyy6.;
format dob mmddyy10.;
datalines;
5 32 100355
6 29 200165
7 42 301082
8 33 120652
;
run;

data final;
            * FINAL date set is created by adding the two files ONE and TWO;
      set one two;
            * Note the use of SET statement here to add two files into FINAL;
run;

proc print data=final;
run;
*******************************************************************;
```