SAS Programming (BIOL-4V190)

Chapter 9
Working with Dates

9.1 Introduction

Dates are stored as the elapsed number of days between 01 January 1960 and the date itself.
Dates prior to 01 January 1960 are negative values, dates after 01 January 1960 are positive values.
Times are stored as the elapsed number of seconds between midnight and that time of day.

9.3 Reading Date Values from Raw Data

Date **informats** are used to properly **read dates into** a SAS data set.
Date **formats** are used to properly **display** dates.
Only **ONE** date informat will be the correct one to use for a given date type.
But **ANY** date format can be used to display the date.

YEARCUTOFF=yyyy is a system option that determines the start of a 100-year interval that SAS uses when it encounters a two digit year. In v8 and v9, YEARCUTOFF=1920.

In some of the examples you will encounter the following new concepts.

On a format statement if several variables are specified before the format name, the format is applied to all of those variables.

The TRUNCOVER option on the infile statement prevents SAS from moving down to the next line of data when there are missing values at the end of a row of data

As mentioned in Chapter 3, to determine the correct informat for a date variable, you need to visually inspect the data and then determine which informat will read it properly.

You could refer to the informats section on the online documentation and use the informats by category section to browse all the Date and Time informats.

The details of each informat provide the specification of what the data must look like.

In Program 9-1, we are reading in a data file that contains 4 date variables. The data are shown on page 143.

The four dates are of the form: mm/dd/yyyy, mmddyyyy, mm/dd/yy, and ddmmmyyyy

Looking through the date informats on the online documentation, the mmddyy informat is needed for the the first three dates while the date informat is needed for the fourth date.

The first date column has dates of the form: mm/dd/yyyy, so the informat is mmddyy10.

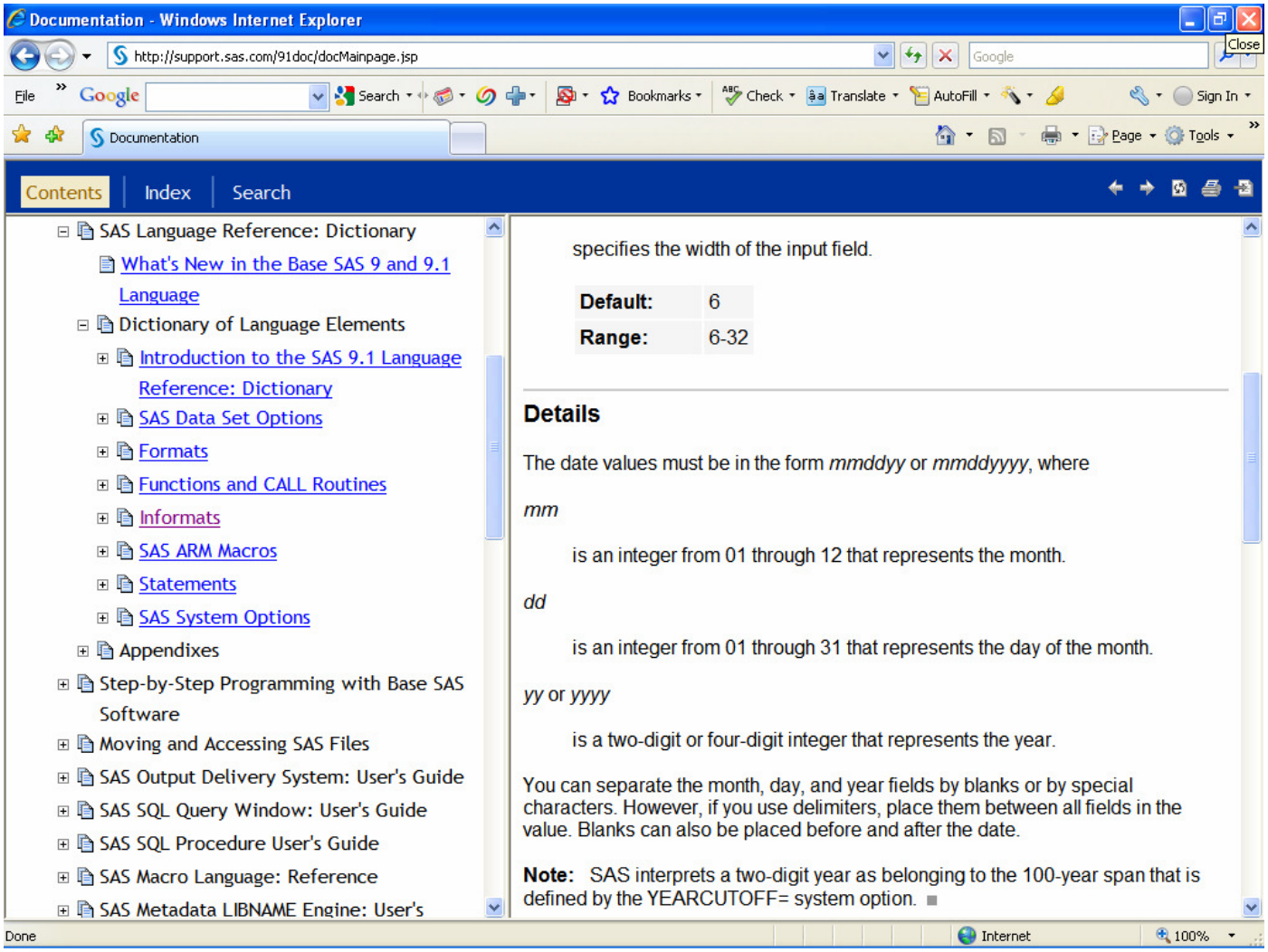
The second date column has dates of the form: mmddyyyy, so the informat is mmddyy8.

The third date column has dates of the form: mm/dd/yy, so the informat is mmddyy8.

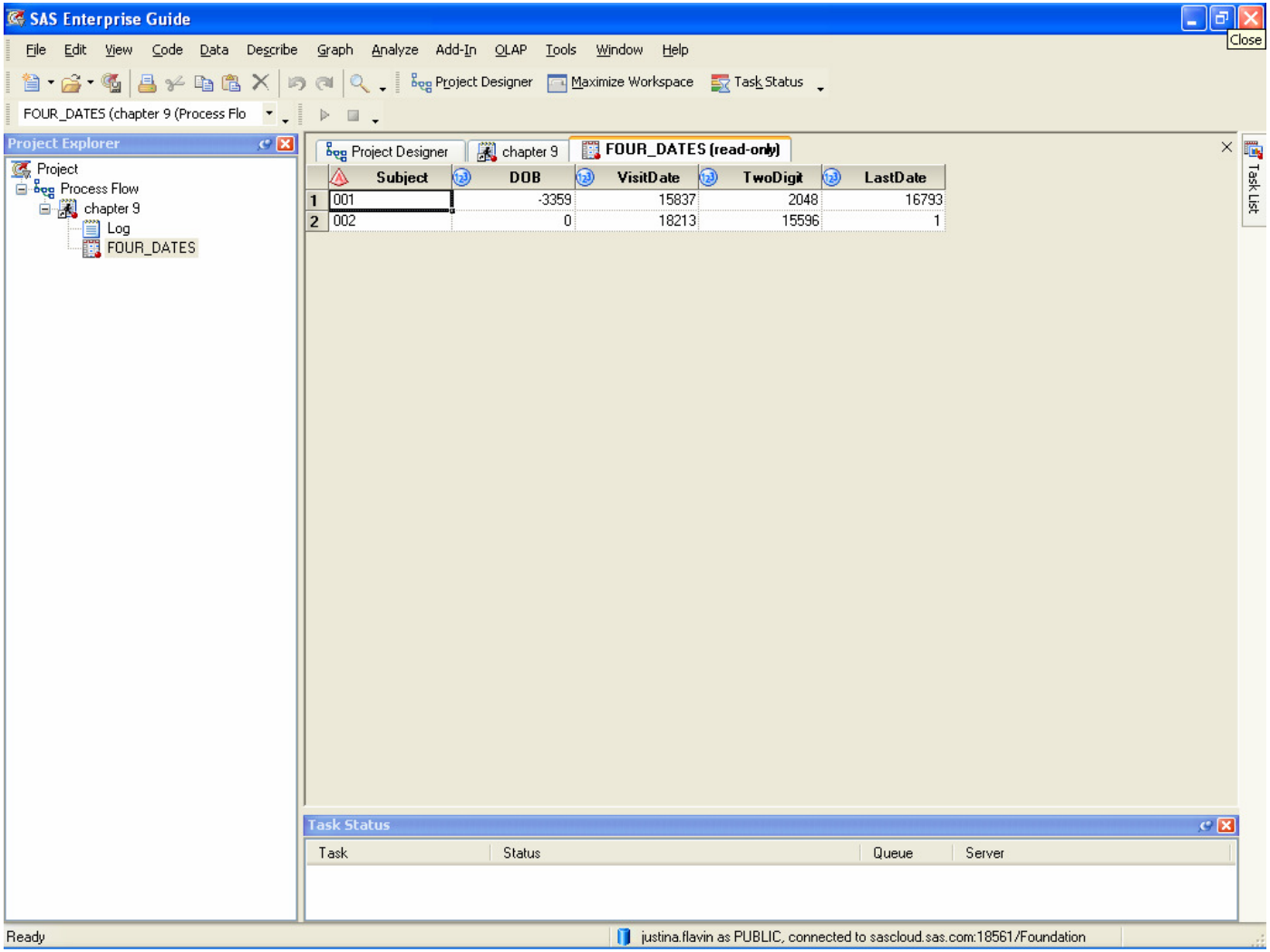
The number at the end of the informat tells SAS the number of characters in the date field.

```
input @1  Subject   $3.
      @5 DOB       mmddyy10.
      @16 VisitDate mmddyy8.
      @26 TwoDigit  mmddyy8.
      @34 LastDate  date9.;
```

Here is the online documentation page for the mmddyy informat.



Running Program 9-1 generates the data set FOUR_DATES.



In FOUR_DATES, since a format was not applied to the variables, the values are the number of days between 01JAN1960 and the date value.

Unless you are good at calculating such differences, it would be hard to know if these values are correct.

But you could perform a sanity check on a couple of the values.

First, we know that dates before 01JAN1960 will appear as negative numbers.

Looking at the raw data, there is one date that is before 01JAN1960.

And in the data set, there is one negative value which does correspond to the position of that date in the raw data. The value below it in the data set is 0, which would correspond to a date of 01JAN1960.

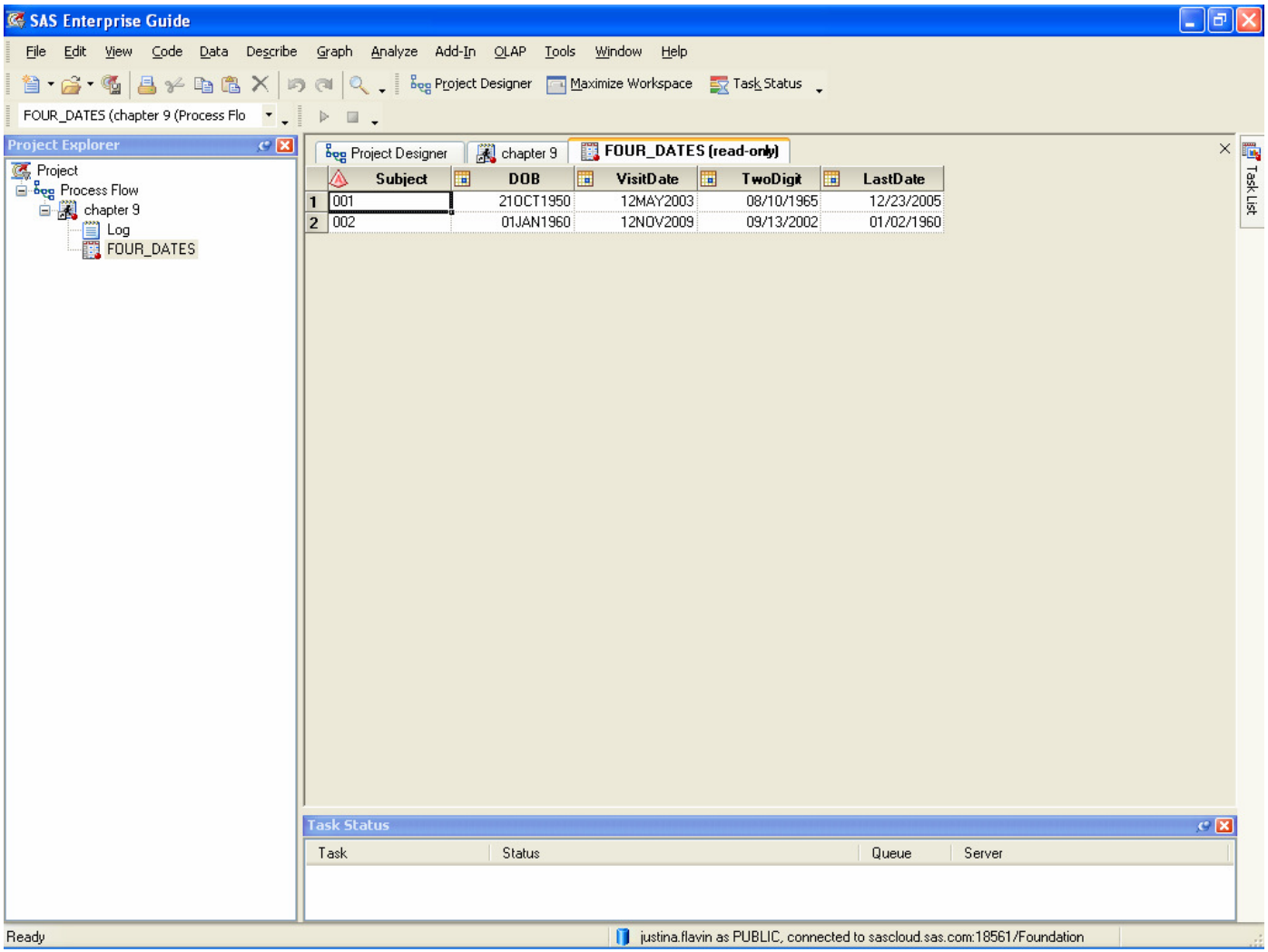
Again looking at the raw data, we do have that value.

Finally, in the raw data, there is also a date of 02JAN1960, which would correspond to the value 1 and yes, we do have a value of 1 in the data set.

Most of the time you will probably not be so lucky and have dates that are so easily verified.

A better way to ensure that your dates are being read in properly is to apply a format, so that the dates appear as recognizable dates.

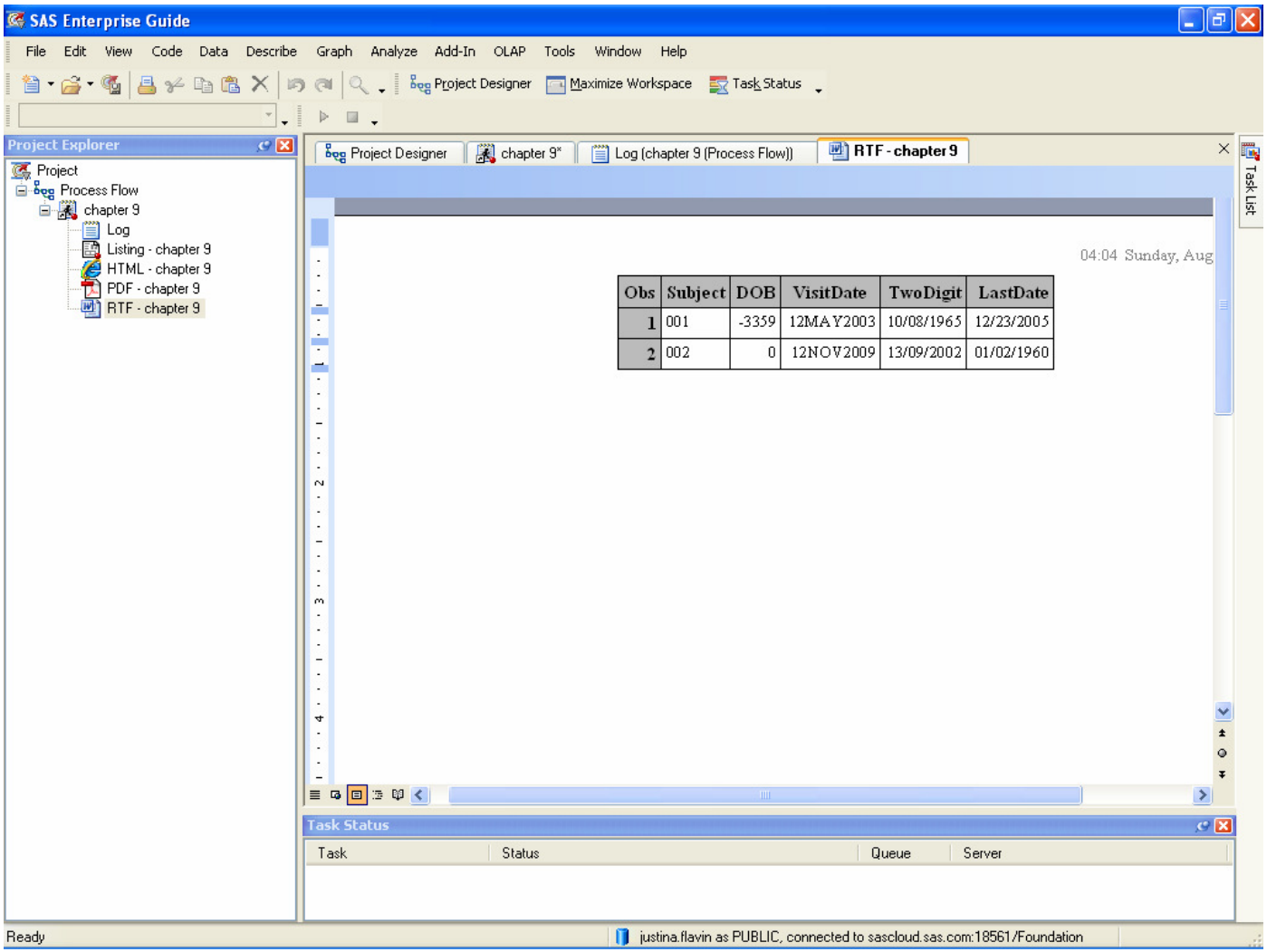
The output data set from Program 9-2 is shown below. Its much easier to check these formatted dates against the raw data. Note that the first two variables are formatted using the DATE format, while the last two use the MMDDYY format. Remember that there is only 1 correct informat for reading in the data, but ANY date format can then be applied. Again to see a list of the possible formats, refer to the SAS online documentation.



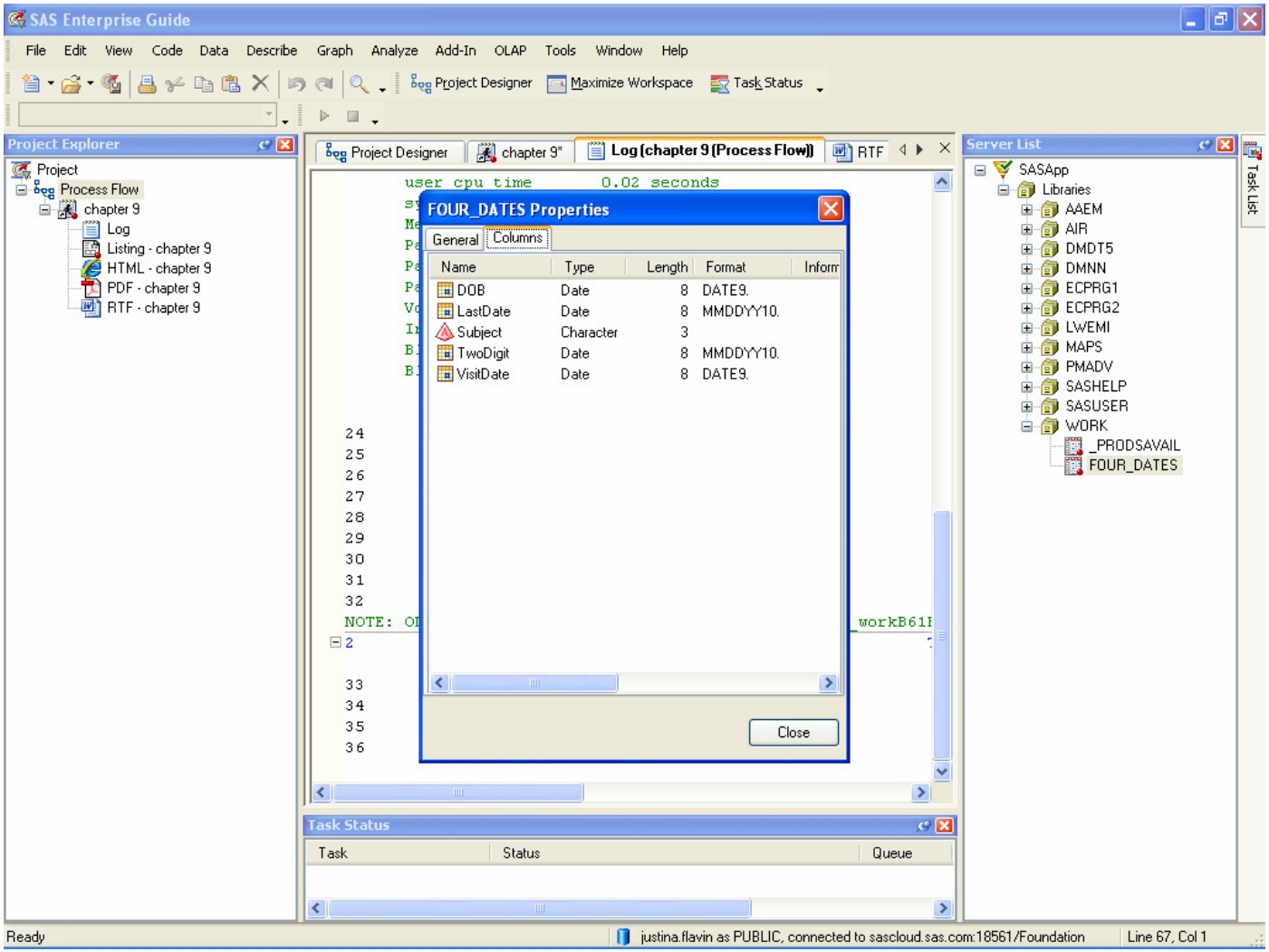
Remember that adding a format to a date variable in your data step code only affects the display of the data, it has no impact on how SAS stores the date (as the elapsed days).

To illustrate this, look at the results of the PROC PRINT output. The format statement changes the format on TwoDigit and removes the format from DOB.

```
format VisitDate date9. TwoDigit ddmmyy10. LastDate mmddyy10. DOB;
```



The format statement applied to the date variables in the PROC PRINT is a temporary format statement.
It only affects the data display in the output for that procedure.
The formats that were added in the data step have not been affected as illustrated below.



9.4 Computing the Number of Years between Two Dates

Functions are built-in routines that enable you to manipulate data.

Like variables, functions fall into two classes, character functions (which are used with character variables) and numeric functions (which are used with numeric variables).

Since dates are stored as numerics, date functions are a subcategory of numeric functions.

Information about functions in the online documentation can be found at:

http://support.sas.com/documentation/onlinedoc/91pdf/index.html
SAS OnlineDoc 9.1 for the Web -> Base SAS -> SAS Language Reference: Dictionary -> Dictionary of Language Elements -> Functions and CALL Routines

The Functions and CALL Routines by Category link is especially helpful.

Look up the following functions in the online documentation:

yrdif - returns the difference in years between two dates

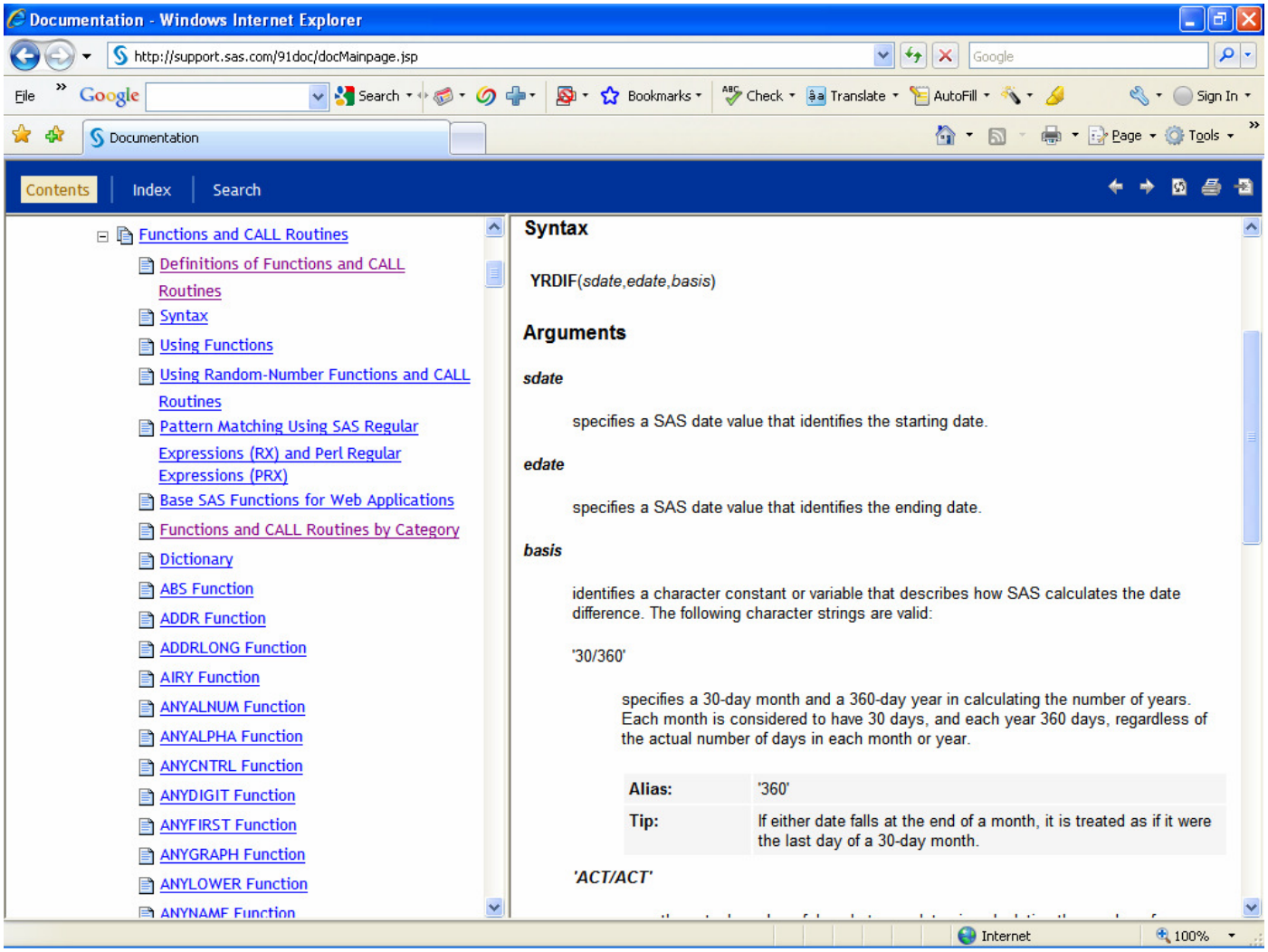
round - rounds the first argument to the nearest multiple of the second argument, or to the nearest integer when the second argument is omitted

int - returns the integer value

We will look at an example that uses the yrdif function. Use the Syntax section to understand how to set up the function.

YRDIF(*sdate*,*edate*,*basis*) tells us that there are 3 arguments we need to provide: sdate, edate, and basis followed by an explanation of these arguments. Sdate and edate are SAS date variables, while basis is one of four specific character strings. Which value of basis is chosen is dependent upon how you wish to calculate the difference. For our example we wish to calculate the actual difference, so either ACT/ACT or ACTUAL can be used.

.



Functions can also be embedded within another function.

The inner function is evaluated first, and then the result is passed back to the outer function.

In the example on page 147,

```
Age = int(yrdif(DOB,VisitDate,'Actual'))
```

The inner function **yrdif( )** is evaluated first.

For the first observation, this function passes back a value of 52.5562.

So now the **int( )** function becomes: int(52.5562) which is 52.

Thus age=52.

In Program 9-3 (modified), we wish to calculate the difference between the VisitDate and the Date of Birth to determine the person's age.

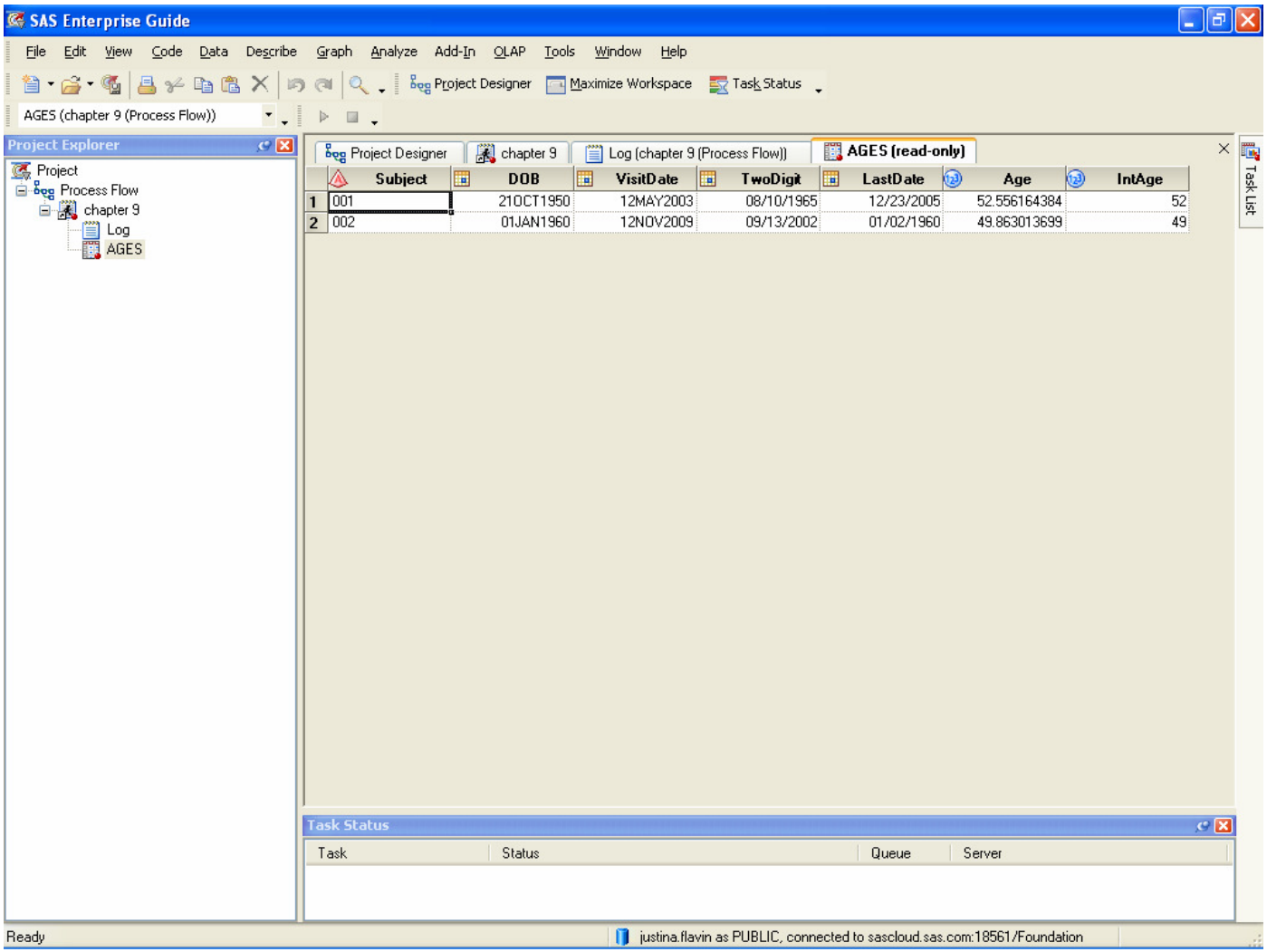
The code is written as:

```
Age = yrdif(DOB,VisitDate,'Actual');
```

The resulting value for Age will not be an integer (unless VisitDate happens to also be the person's birthday). So we can then apply the int function to the result to obtain an integer value for the person's age.

```
IntAge = int(yrdif(DOB,VisitDate,'Actual'))
```

Here is the data set AGES.



9.5 Demonstrating a Date Constant

Sometimes it is necessary to specify a particular date within your code. This is accomplished by using a date constant (or date literal).

The format of a date constant is: 1 or 2 digits for day, 3 characters for month, 2 or 4 digit year enclosed in single or double quotes and followed by the letter ‘d’ (upper or lowercase).

Example: `‘16JUL07’d`

Since a date (or a date constant) represents a number, dates and date constants can be used like numbers for making comparisons and calculations.

Program 9-4 (modified) illustrates two examples using date constants. The first example illustrates using the yrdif function with a date constant in place of one of the date variables. By replacing VisiDate with a date constant, every row will use the same date in performing the calculation.

The second example illustrates using a date constant as a comparitor. SAS compares the underlying numeric values of DOB and 05MAY1976 and assigns the values of the variable agecat depending upon the results of the comparison.

```
*Program 9-4 Demonstrating a date constant - page 148;
data ages;
   set four_dates;
   Age = yrdif(DOB,'01Jan2006'd,'Actual');
   if DOB <= '5MAY1976'd then agecat=1;
   else agecat=2;
run;
```

9.6 Computing the Current Date

The function **today( )** returns the current date.

The current date is the date that the current SAS session started.

The function does not require an argument.

Program 9-5 (modified) illustrates creating a variable a_week_ago by subtracting 7 from the current date.

The calculation of Age is modified to use the today() function.

```
*Program 9-5 Using the TODAY function to return the current date - page 148;
data ages;
   set four_dates;
   a_week_ago=today()-7;
   Age = yrdif(DOB,today(),'Actual');
   format a_week_ago date.;
run;
```

9.7 Extracting the Day of the Week, Day of the Month, Month, and Year from a SAS Date

There are several date functions that can be used to parse dates.

All of these return numeric values, so you cannot obtain the name of month from the month function, just the number of the month.

Likewise for the weekday, you cannot get the name of the day, just the day of the week. Sunday=1, Monday=2, Tuesday=3, etc.

weekday( ) returns the day of the week as a number

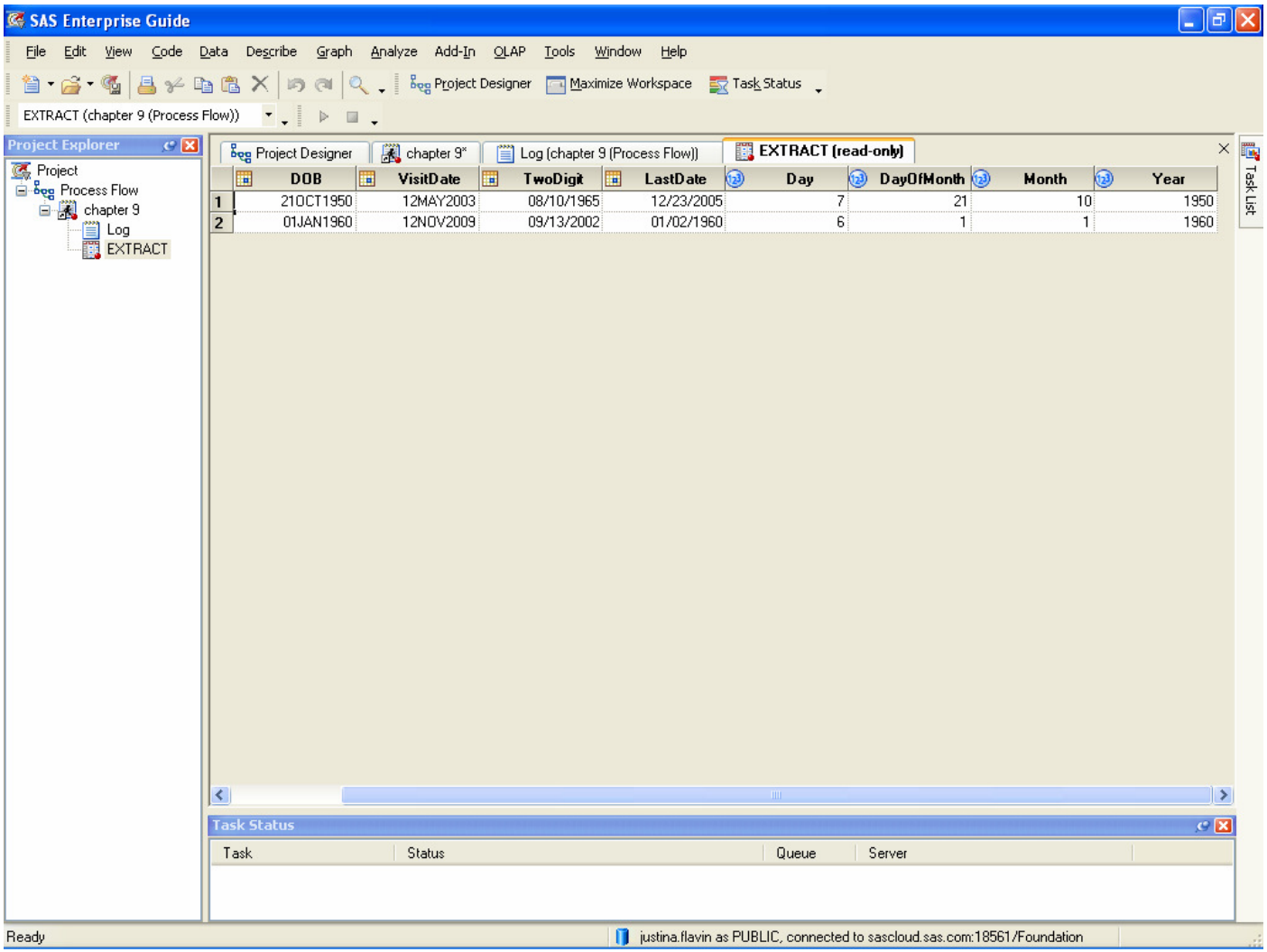
day ( ) returns the day of the month

month ( ) returns the month as a numeric

year ( ) returns the year as 4 digits

```
*Program 9-6 Extracting the day of the week, day of the month, month and year from a SAS date -
    page 149;
data extract;
   set four_dates;
   Day = weekday(DOB);
   DayOfMonth = day(DOB);
   Month = Month(DOB);
   Year = year(DOB);
run;
```

Here is the data set EXTRACT.



9.8 Creating a SAS Date from Month, Day, and Year Values

9.9 Substituting the 15th of the Month when the Day Value Is Missing

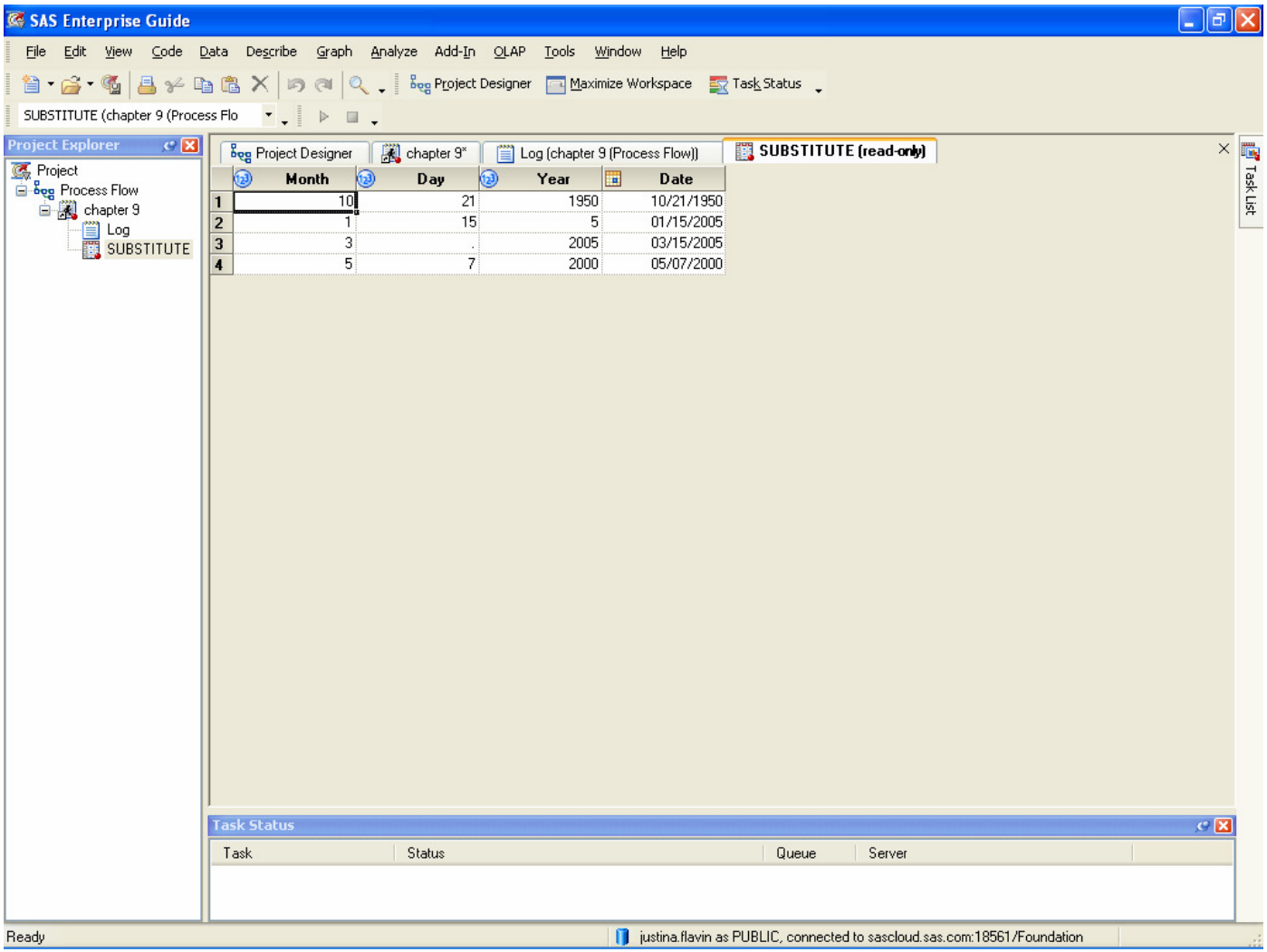
As you may expect, since there are functions that parse dates into integers, there is also a function that creates dates from intergers.

mdy (*month, day, year*) creates a date value from three integer values for the month, day, year

```
In Program 9-8, the data set contains numeric variables that can be used to create a date. Additionally, when
    the value for day is missing, the value 15 is used.

*Program 9-8 Substituting the 15th of the month when a day value is missing - page 151;
data substitute;
   set learn.month_day_year;
   if missing(day) then Date = mdy(Month,15,Year);
   else Date = mdy(Month,Day,Year);
   format Date mmddyy10.;
run;
```

Here is the data set SUBSTITUTE



9.10 Using Date Interval Functions

This section is omitted.