SAS Programming (BIOL-4V190)

Chapter 7

Performing Conditional Processing

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

7.1 Introduction

Conditional Processing entails the use of logical statements that, when applied to the data values, enable you to reshape the data set.

BE CAREFUL – it is very easy to write code of this type that is syntactically correct and displays no errors in the log – but the logic may be flawed, so that the code is not actually doing what you want it to do.

It is also important to remember that there are often many ways to write code, all of which do the same thing and give the same results. Many times there is not one "correct" way and how you choose to write your code is a matter of personal preference.

7.2 The IF and ELSE IF Statements

IF-THEN-ELSE and ELSE IF statements can be used to recode data or create new variables based on the statement logic and the data values.

The table on page 103 lists the logical comparison operators that can be used in these statements.

In the examples that follow, there will be missing values in the data. Missing Values in SAS are always treated as the smallest possible value – this is important to keep in mind when writing your code.

Additionally, you should always write your code to account for the possibility of missing values in the data.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

SAS has many built-in functions and we will look at many of them in later lectures.

The general syntax for a function is: *functionname(optional parameters)*

Some of the examples in this chapter include a function called MISSING which can be used to check for the presence of missing values in the data.

In this example, we have a variable Age and we wish to create a new variable that categorizes the data into 4 20 year intervals based on the value of Age. So Group 1 is ages under 20, Group 2 is Ages between 20 and 40, etc.

These statements would seem to be correct:

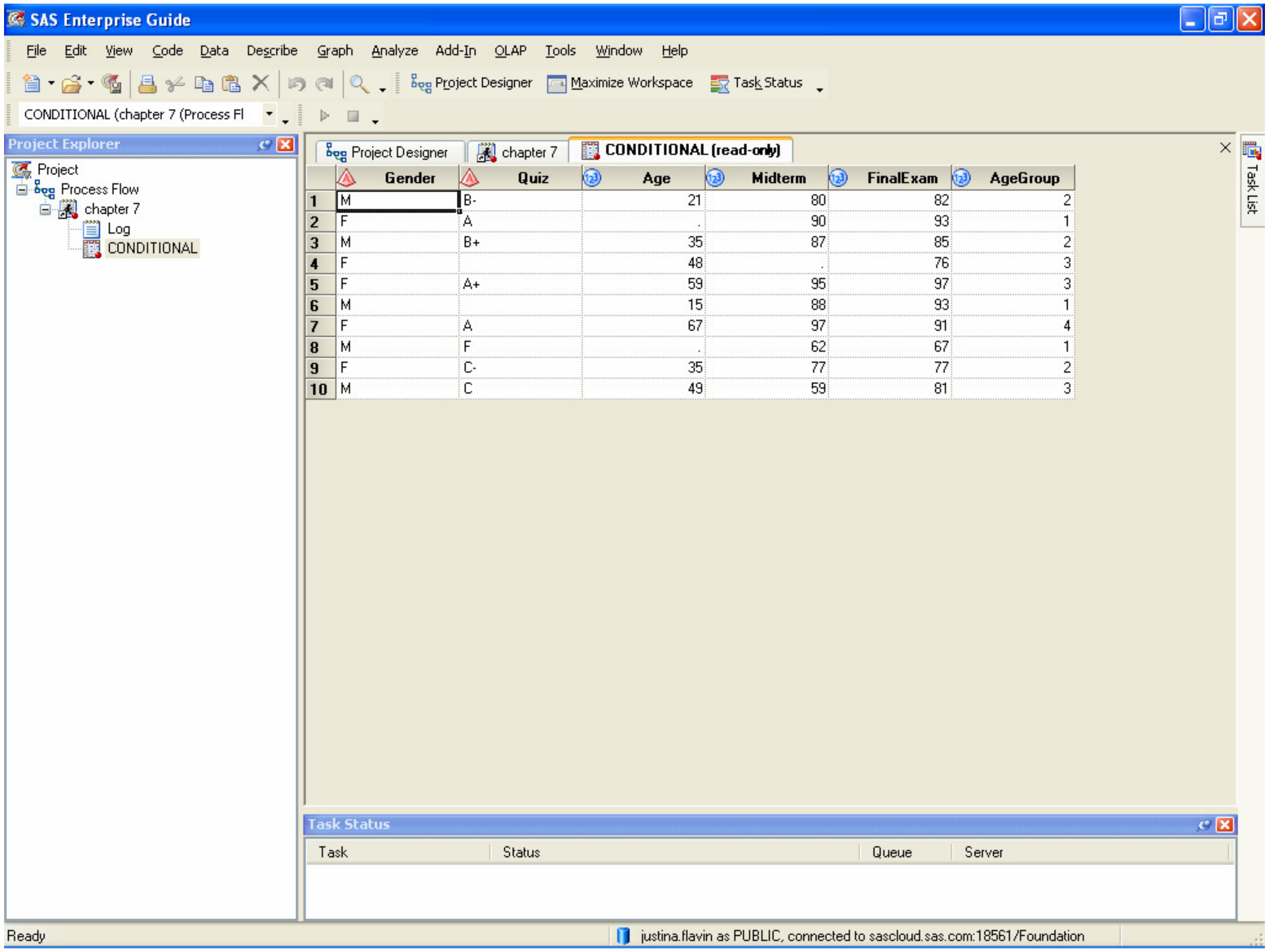
```
if Age lt 20 then AgeGroup = 1;
if Age ge 20 and Age lt 40 then AgeGroup = 2;
if Age ge 40 and Age lt 60 then AgeGroup = 3;
if Age ge 60 then AgeGroup = 4;
```

An equivalent way to write this code would be:

```
if Age < 20 then AgeGroup = 1;
if 20 <= Age < 40 then AgeGroup = 2;
if 40 <= Age < 60 then AgeGroup = 3;
if Age >= 60 then AgeGroup = 4;
```

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Here is the data set created in Program 7-1. Look at the value for AgeGroup for rows 2 & 8. These rows have missing values for Age. In both cases, AgeGroup=1, which is not correct. A missing value for Age should not be included in any of the categories. But because SAS treats missing values as a small negative value less than the smallest value in your data, missing values are included in the first if statement:
`if Age lt 20 then AgeGroup = 1;` Thus the missing value are assigned the value of 1 for AgeGroup.



	Gender	Quiz	Age	Midterm	FinalExam	AgeGroup
1	M	B-	21	80	82	2
2	F	A	.	90	93	1
3	M	B+	35	87	85	2
4	F		48	.	76	3
5	F	A+	59	95	97	3
6	M		15	88	93	1
7	F	A	67	97	91	4
8	M	F	.	62	67	1
9	F	C-	35	77	77	2
10	M	C	49	59	81	3

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

One way to fix this problem is to rewrite the first if statement as:

```
if . < Age < 20 then AgeGroup = 1;
```

"." is the notation used to denote a missing value. Since a missing value is smaller than the smallest value in your data, this will exclude all missing values.

Since we would not anticipate negative values for Age, we could also rewrite the statement as:

```
if  0 <=  Age < 20 then AgeGroup = 1;
```

Program 7-2 provides another option, illustrating the use of the missing function:

```
if Age lt 20 and not missing(age) then AgeGroup = 1;
```

The missing function checks if a variable value is missing (or not).

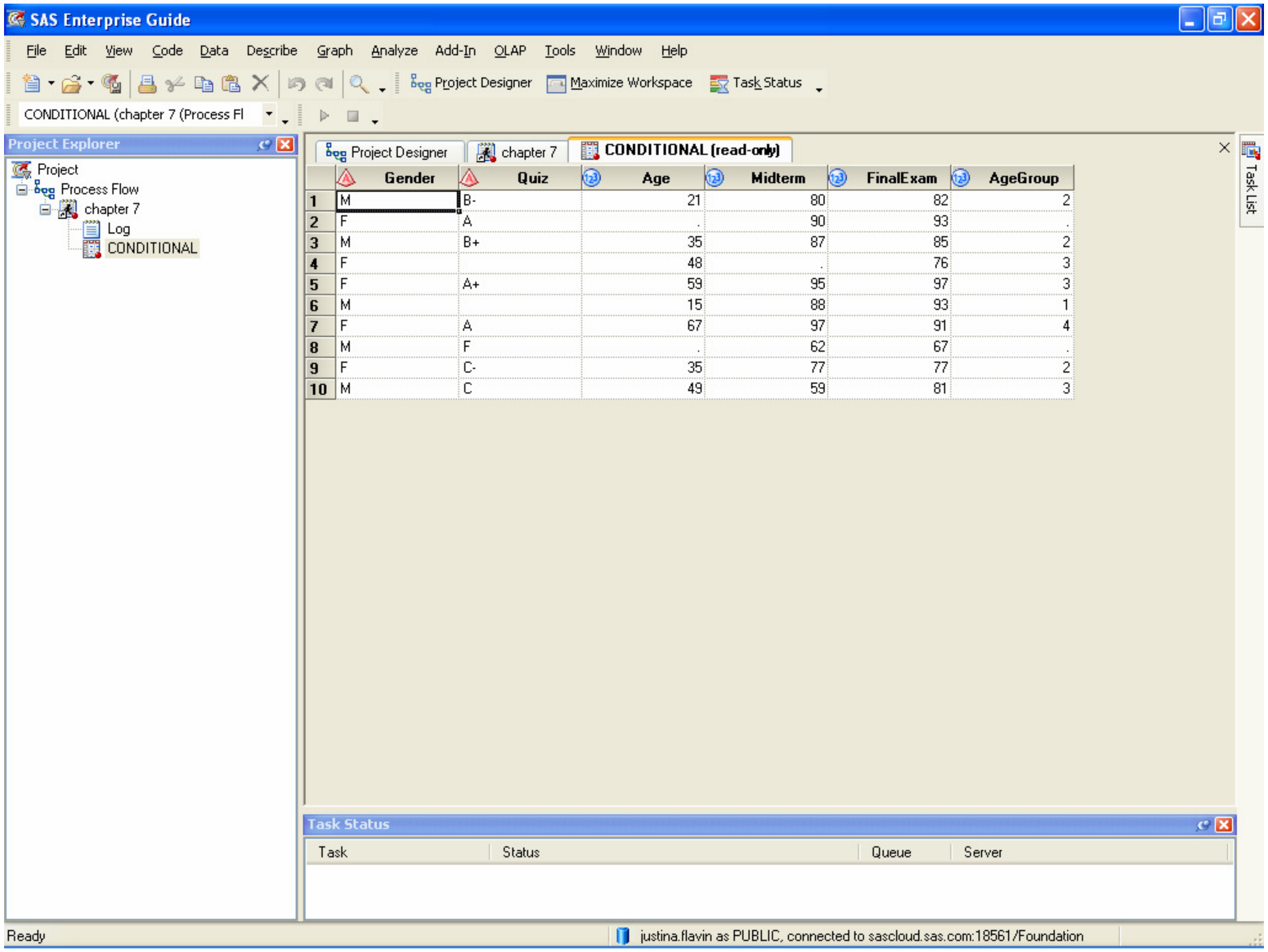
SAS processes this statement by evaluating each value of age. If Age is less than 20 and age is not missing, then AgeGroup=1

Yet another equivalent way to write the line is:

```
if Age lt 20 and age ne . then AgeGroup = 1;
```

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Now running the code for Program 7-2, we see that the value of AgeGroup is a missing value when Age is a missing value.


SAS Enterprise Guide
File Edit View Code Data Describe Graph Analyze Add-In OLAP Tools Window Help
Project Designer Maximize Workspace Task Status
CONDITIONAL (chapter 7 (Process Fl
Project Explorer
Project
Process Flow
chapter 7
Log
CONDITIONAL
Project Designer
chapter 7
CONDITIONAL (read-only)
Task List
Task Status
Task Status Queue Server
Ready
justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation

	Gender	Quiz	Age	Midterm	FinalExam	AgeGroup
1	M	B-	21	80	82	2
2	F	A	.	90	93	.
3	M	B+	35	87	85	2
4	F		48	.	76	3
5	F	A+	59	95	97	3
6	M		15	88	93	1
7	F	A	67	97	91	4
8	M	F	.	62	67	.
9	F	C-	35	77	77	2
10	M	C	49	59	81	3

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Program 7-3 illustrates yet another way to write the code using if-then-else logic:

```
if missing(Age) then AgeGroup = .;
else if Age lt 20 then AgeGroup = 1;
else if Age lt 40 then AgeGroup = 2;
else if Age lt 60 then AgeGroup = 3;
else if Age ge 60 then AgeGroup = 4;
```

It is important to understand that once a value of AgeGroup is assigned, SAS skips over the remaining else-if statements. So depending upon the value of Age, SAS may or may not evalute all of these lines of code.

For example, if age=43:

if missing(Age) is false
else if Age lt **20** is false
else if Age lt **40** is false
else if Age lt **60** is true

At this point SAS assigns AgeGroup=3 and thus does not evaluate the last else-if statement.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

7.3 The Subsetting IF Statement

The subsetting IF statement is used to select a subset of data based on the criteria specified in the statement.

For example this statement would keep all observations where the value of age is less than 43.

This would include missing values.

```
if age < 43;
```

`if . < age < 43;` would keep all values of age that are less than 43 and exclude missing values.

Compound statements can also be written:

```
if age < 43 and Gender='M';
```

This would keep all observations where values of age are less than 43 and Gender is equal to M.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

```
*Program 7-4 Demonstrating a subsetting IF statement - page 106;
data females;
  length Gender $ 1
         Quiz   $ 2;
  input Age Gender Midterm Quiz FinalExam;
  if Gender eq 'F';
datalines;
21 M 80 B- 82
.  F 90 A  93
35 M 87 B+ 85
48 F  . .  76
59 F 95 A+ 97
15 M 88 .  93
67 F 97 A  91
.  M 62 F  67
35 F 77 C- 77
49 M 59 C  81
;
run;
```

This code creates a data set containing only the data for Gender='F'.

To understand the processing, when the INPUT statement is reached, SAS will read every line of data into the program data vector or PDV (refer to page 22).

Then each data line in the PDV is evaluated against the `if` statement and if Gender='F' then the observation is written to the females data set.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

7.4 The IN Operator

The IN operator can be used in place of multiple OR statements.

Usually it is easier to write code using IN rather than OR. This is especially true when there are many conditions.

Additionally the code is more concise and easier to edit and check for errors.

Some examples of statements using the IN operator are shown on page 107.

7.8 The WHERE Statement

The WHERE statement can be used to select a subset of data when a SET statement is also used in the data step.

While the same result is achieved, WHERE is usually more efficient than IF.

This is because WHERE subsets the data PRIOR to bringing it into the data set, whereas IF subsets the data after bringing in all the data and then deleting the unneeded observations.

Unlike the subsetting IF statement which can only be used in a DATA step, the WHERE statement can be used in both DATA steps and PROCs.

In summary:

Use	When you have
IF	Data Step with an INPUT statement, Data Step with a SET statement
WHERE	Data Step with a SET statement, PROC

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

A WHERE clause can contain more than one expression.

Syntax:

```
where expression1 and expression2 and expression3;
```

Multiple WHERE statements can also be appended through the addition of the “WHERE SAME AND” on the subsequent WHERE statements.

Syntax:

```
where expression1;
where same and expression2;
where same and expression3;
```

7.9 Some Useful WHERE Operators

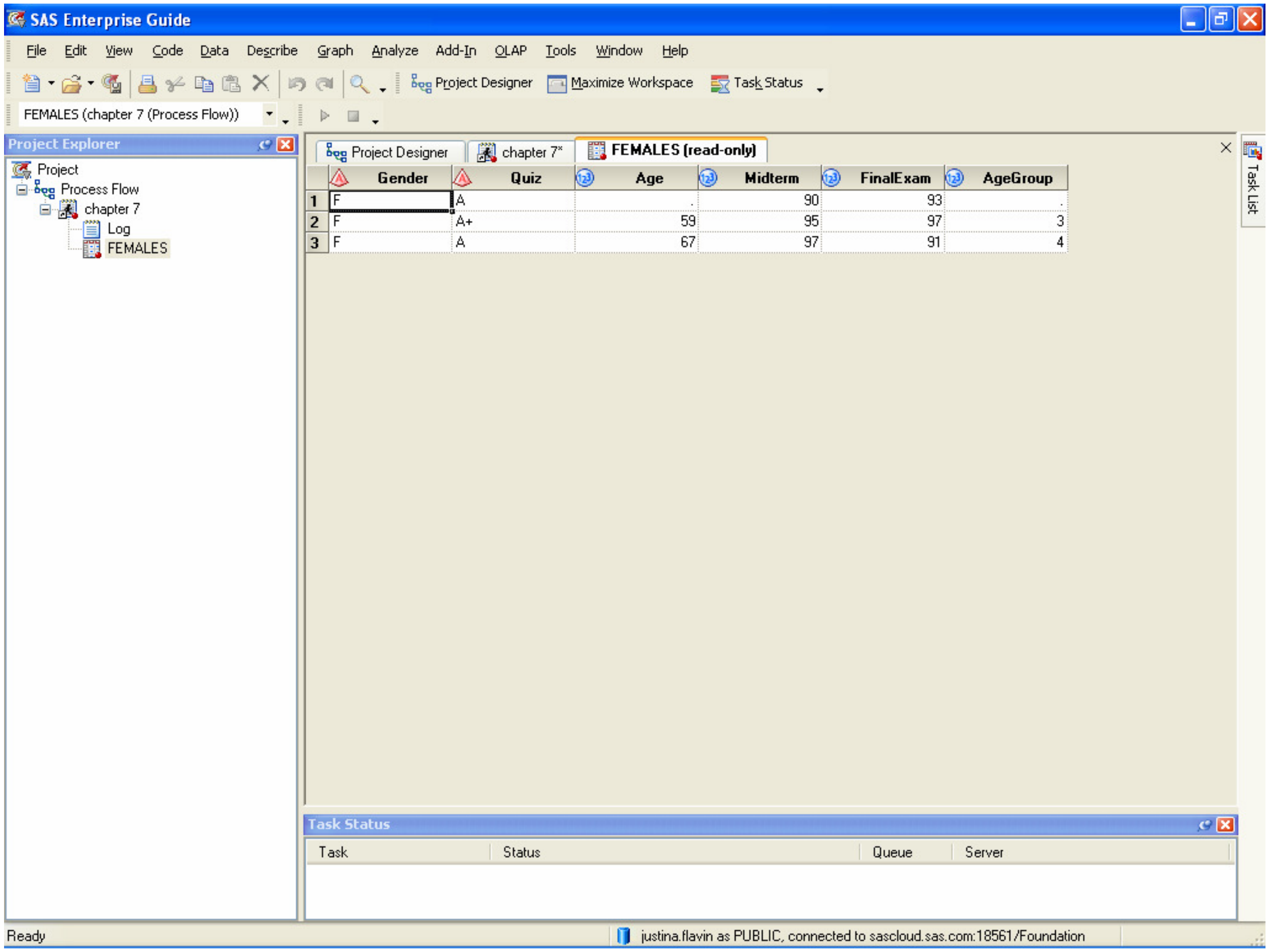
The table on page 113 provides a list of WHERE operators.

Program 7-8 illustrates the use of multiple where statements.

```
*Program 7-8 Using a WHERE statement to subset a SAS data set - page 112;
data females;
  set conditional;
  * where Gender eq 'F';
  where midterm >= 50 and finalexam > 60;
  where same and quiz in ('A' 'A+');
run;
```

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

In this example, the observations selected are those that meet the criteria of midterm >= 50 and finalexam > 60 and a quiz value of either A or A+


SAS Enterprise Guide
File Edit View Code Data Describe Graph Analyze Add-In OLAP Tools Window Help
Project Designer Maximize Workspace Task Status
FEMALES (chapter 7 (Process Flow))
Project Explorer
Project
Process Flow
chapter 7
Log
FEMALES
Project Designer
chapter 7*
FEMALES (read-only)
Task List
Task Status
Task Status Queue Server
Ready
justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation

	Gender	Quiz	Age	Midterm	FinalExam	AgeGroup
1	F	A	.	90	93	.
2	F	A+	59	95	97	3
3	F	A	67	97	91	4

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

7.5 Using a SELECT Statement for Logical Tests

The SELECT statement is usually a better alternative to a series of many IF-THEN-ELSE statements.

Although the OTHERWISE clause is optional, it should ALWAYS be included.

If no condition in the SELECT statement is met and the OTHERWISE statement is omitted, an error may be generated in the log.

Syntax:

Version 1:

```
select (variablename1);
  when (condition1) variablename2 = value1;
  when (condition2) variablename2 = value2;
  when (condition3) variablename2 = value3;
  otherwise;
end;
```

Version 2:

```
select;
  when (variablename1=condition1) variablename2 = value1;
  when (variablename1=condition2) variablename2 = value2;
  when (variablename1=condition3) variablename2 = value3;
  otherwise;
end;
```

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Program 7-5 illustrates the use of one version of the select statement.

This is yet another way to code the if-then-else statement that were shown in previous examples:

```
select;
      when (missing(Age)) AgeGroup = .;
      when (Age lt 20) AgeGroup = 1;
      when (Age lt 40) AgeGroup = 2;
      when (Age lt 60) AgeGroup = 3;
      when (Age ge 60) Agegroup = 4;
      otherwise;
end;
```

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

7.6 Using Boolean Logic (AND, OR, and NOT Operators)

The logical operators AND, OR and NOT are also called Boolean Operators.
These can be used to create logical statements using several variables.
Sometimes parentheses around various parts of the statement are necessary to get the desired result.

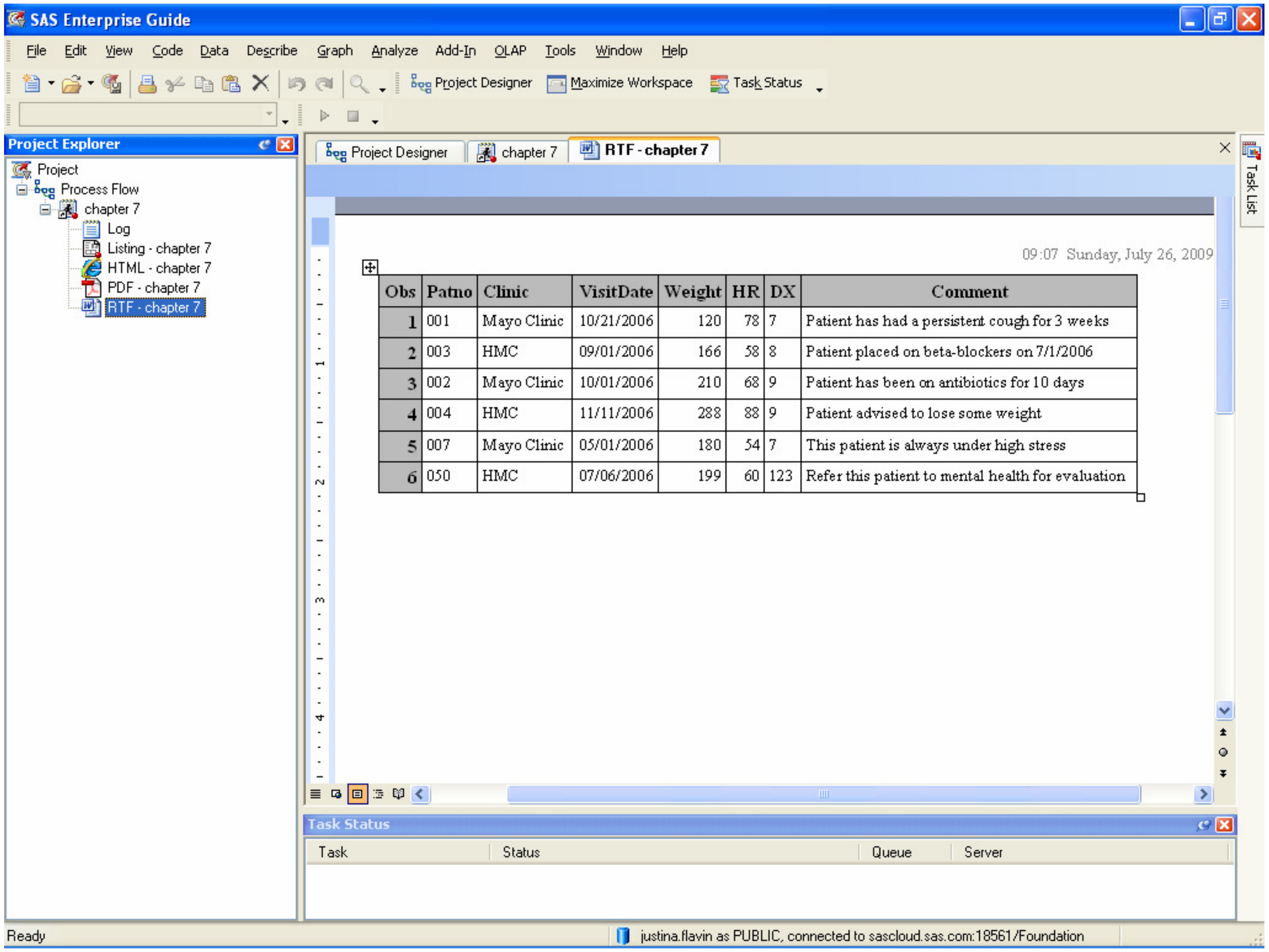
Program 7-6

```
where Clinic eq 'HMC' and  (DX in ('7' '9') or  Weight gt 180);
```

In this example, we are selecting observations where Clinic=HMC and either one of DX or Weight meet a second criteria.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Here is the data set LEARN.MEDICAL

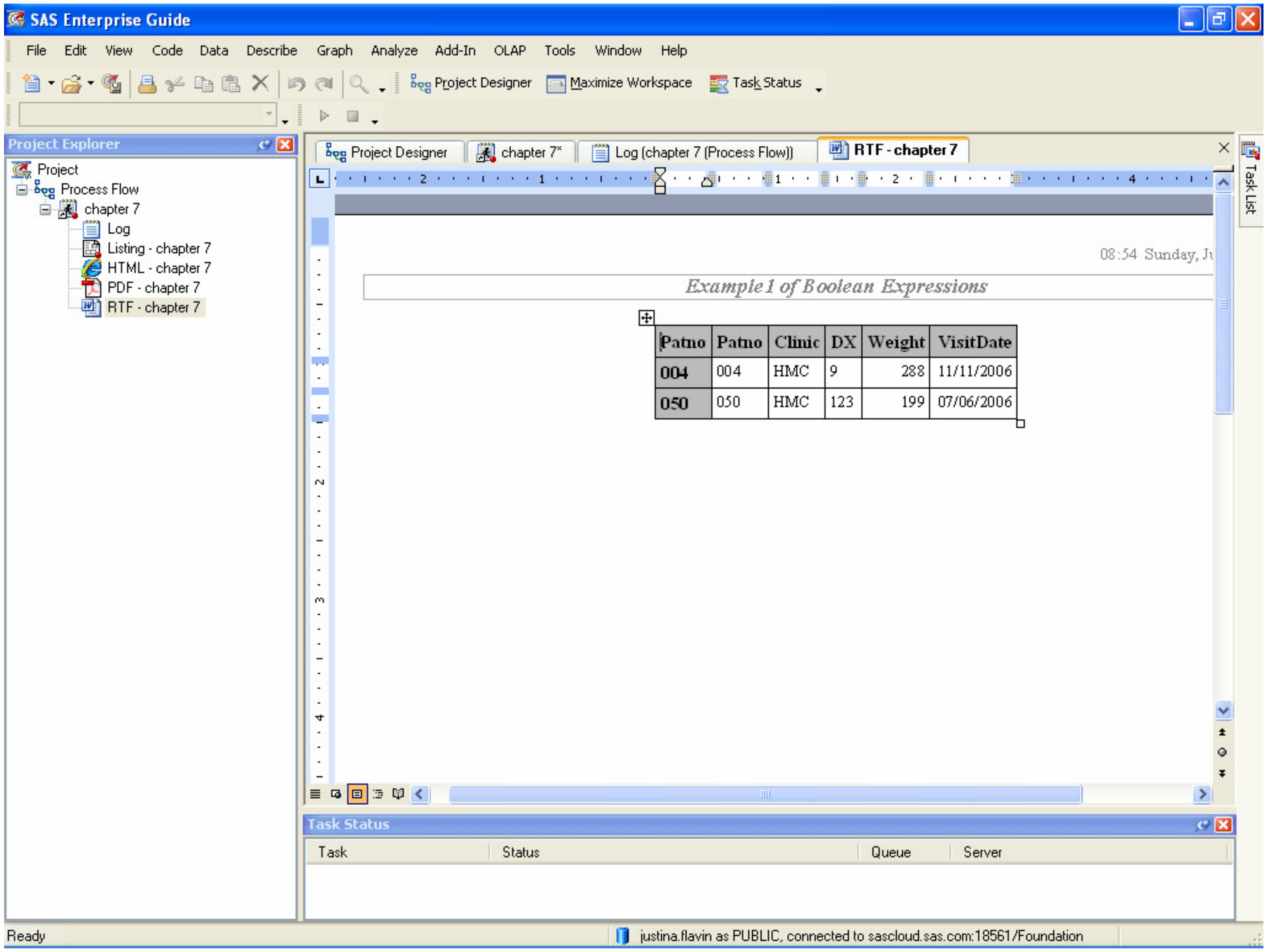


09:07 Sunday, July 26, 2009

Obs	Patno	Clinic	VisitDate	Weight	HR	DX	Comment
1	001	Mayo Clinic	10/21/2006	120	78	7	Patient has had a persistent cough for 3 weeks
2	003	HMC	09/01/2006	166	58	8	Patient placed on beta-blockers on 7/1/2006
3	002	Mayo Clinic	10/01/2006	210	68	9	Patient has been on antibiotics for 10 days
4	004	HMC	11/11/2006	288	88	9	Patient advised to lose some weight
5	007	Mayo Clinic	05/01/2006	180	54	7	This patient is always under high stress
6	050	HMC	07/06/2006	199	60	123	Refer this patient to mental health for evaluation

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

The WHERE clause results in the selection of the following two observations:



Example 1 of Boolean Expressions

Patno	Patno	Clinic	DX	Weight	VisitDate
004	004	HMC	9	288	11/11/2006
050	050	HMC	123	199	07/06/2006

justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

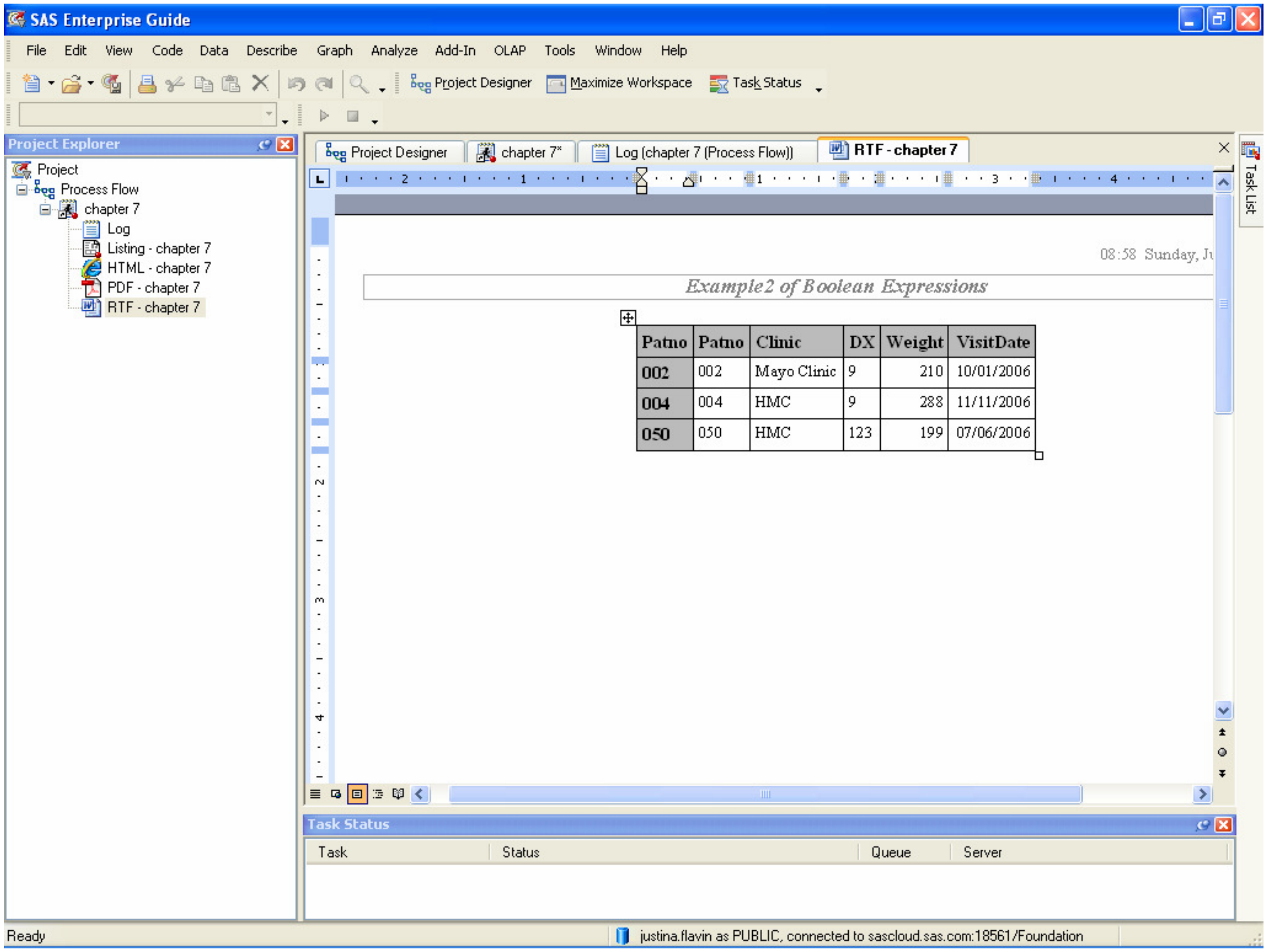
Now let’s see what happens if the parentheses are removed:

```
where Clinic eq 'HMC' and  DX in ('7' '9') or  Weight gt 180;
```

Now we are selecting observations where Clinic=HMC and DX is 7 or 9 or a second criteria, observations where Weight > 180, is met.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

This gives a slightly different subset of the data:


SAS Enterprise Guide
File Edit View Code Data Describe Graph Analyze Add-In OLAP Tools Window Help
Project Designer Maximize Workspace Task Status
Project Explorer
Project
Process Flow
chapter 7
Log
Listing - chapter 7
HTML - chapter 7
PDF - chapter 7
RTF - chapter 7
Project Designer
chapter 7*
Log (chapter 7 (Process Flow))
RTF - chapter 7
Task List
Task Status
Task
Status
Queue
Server
Ready
justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation

08:58 Sunday, Ju

Example2 of Boolean Expressions

Patno	Patno	Clinic	DX	Weight	VisitDate
002	002	Mayo Clinic	9	210	10/01/2006
004	004	HMC	9	288	11/11/2006
050	050	HMC	123	199	07/06/2006

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

7.7 A Caution When Using Multiple OR Operators

This last example reiterates the importance of carefully checking your code for logic errors:

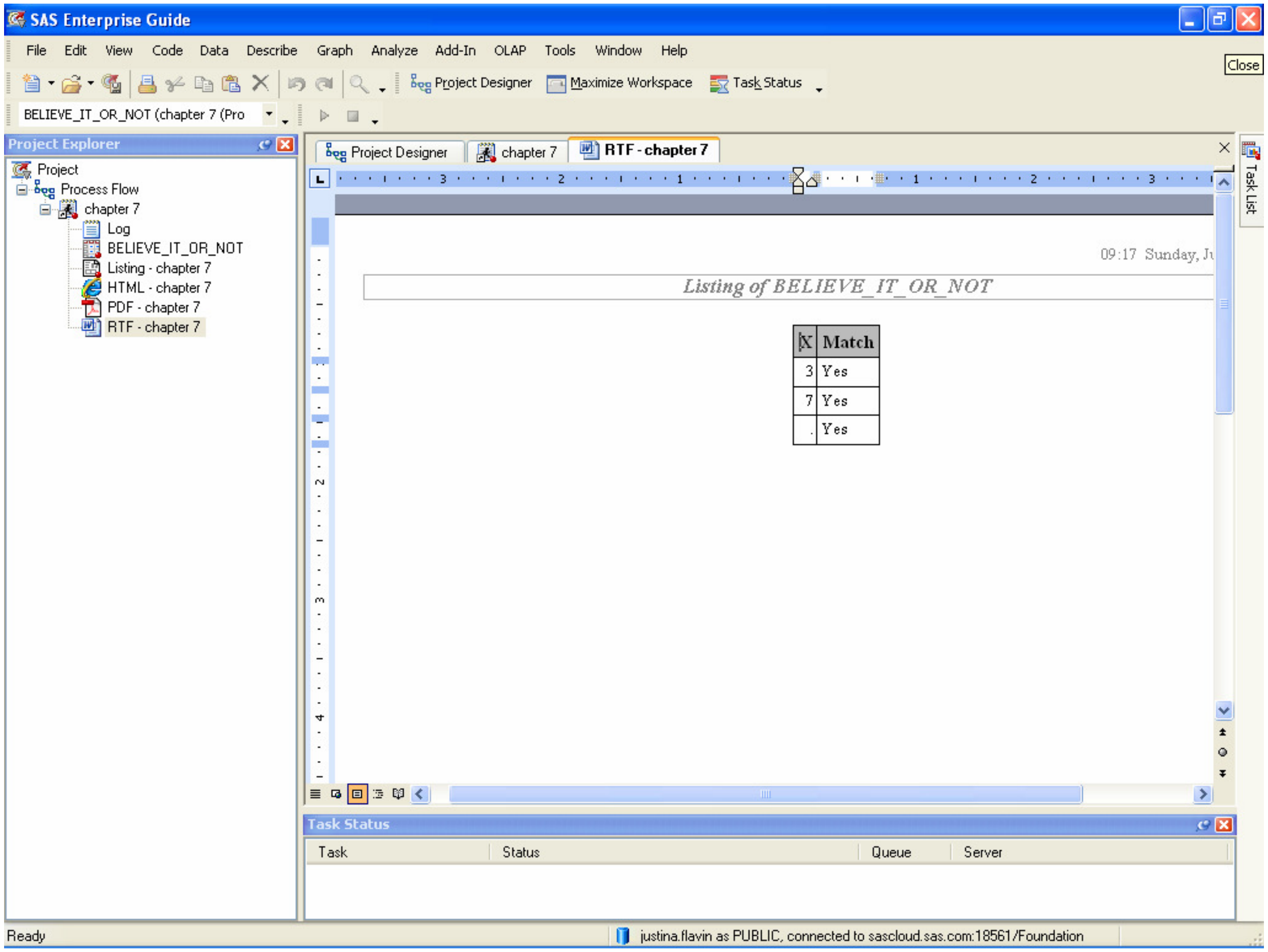
```
data believe_it_or_not;
  input X;
  if X = 3 or 4 then Match = 'Yes';
  else Match = 'No';
datalines;
3
7
.
;
run;
```

In the IF statement, SAS evaluates the first part of the statement as either X=3 or the value 4.

The value 4 is always a true statement, so regardless of the value of X, Match will always be set to Yes.

Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Here is the resulting data set.



09:17 Sunday, Ju

Listing of BELIEVE_IT_OR_NOT

X	Match
3	Yes
7	Yes
.	Yes