SAS Programming (BIOL-4V190)

Chapter 10

Subsetting and Combining SAS Data Sets

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.

10.1 Introduction

This chapter covers the SET and MERGE statements which are used to combine SAS data sets.

10.2 Subsetting a SAS Data Set

A subset of a data set can be created using a WHERE statement or a subsetting IF statement.

```
*Program 10-1 Subsetting a SAS data set using a WHERE statement - page 162;
data females;
   set learn.survey;
   where gender = 'F';
run;
```

The variable GENDER has two values, F & M. To create a new data set containing only the data for the females, a subsetting where statement can be used.

The same result can be obtained using a subsetting IF statement.

```
*Subsetting a SAS data set using a subsetting IF statement;
data females;
   set learn.survey;
   if gender = 'F';
run;
```

Using a WHERE statement is more efficient than using a subsetting IF statement.
When a WHERE statement is used, SAS makes the comparison on the incoming variables and then determines if the observation should be written out to the new data set.
When using a subsetting IF statement, all of the observations are first written to the new data set and then observations not meeting the IF criteria are deleted.

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.

Similar to the comparison of the WHERE and subsetting IF, there are efficiencies to consider in the placement of the DROP statement.

```
*Program 10-2 Demonstrating a DROP= data set option - page 163;
data females;
   set learn.survey(drop=Salary);
   where gender = 'F';
run;

*Using a DROP statement;
data females;
   set learn.survey;
   where gender = 'F';
   drop salary;
run;
```

Both of these examples create the same data set.

The first example is more efficient since the variable Salary is dropped from the incoming data set and thus never becomes part of the new data set.

The second example is less efficient since Salary is first included in the new data set first (so SAS much allocate resources for the variable in the new data set) and then dropped from the 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.

10.3 Creating More Than One Subset Data Set in One DATA Step

Multiple output data sets can be created from one data step.
Each output data set must be listed on the DATA statement. If output criteria are not specified within the data step code and an OUTPUT statement is not used, all observations are written to all data sets specified on the DATA statement.

Syntax:

```
data dataset1 dataset2 dataset3...dataset'n';
```

```
*Program 10-3 Creating two data sets in one DATA step - page 164;
data males females;
   set learn.survey;
   if gender = 'F' then output females;
   else if gender = 'M' then output males;
run;
* what data set will be printed? ;
proc print;
run;
```

This example illustrates the use of IF-THEN logic to subset the data and to create two new data sets by outputting the observations to the appropriate data sets.

Also of interest is the PRINT procedure. No data set has been specified, so by default, SAS uses the last data set created which is the last data set listed on the **data** line.

Since females is the last data set listed, that is the data set which will be printed.

This illustrates why it is good programming practice to use a `data=datasetname` option on your procedures.

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.

10.4 Adding Observations to a SAS Data Set

When more than one data set is specified on the SET statement, the observations from each data set are added to the output data set.

This is called concatenation.

Other statements can be used to control which observations are added and the order in which these observations are added.

Even if the data sets do not all contain the same variables, the output data set will have all variables on all observations.

The variables that are in each input data set determine which variables will have missing values on each observation in the output data set.

When working with data sets that have variables of the same name, make sure that the length and the type are the same in all of the data sets.

If there are varying lengths, SAS sets the length to that of the variable's length in the first data set where it encounters the variable.

If the variable type is not the same in all data sets, SAS stops processing the code and an ERROR will be generated.

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 the example code for this section, three data sets are created. ONE and TWO have exactly the same data set structure, both data sets have three variables of the same name, type, and length. When these two data sets are combined into a new data set, the new data set also has exactly the same structure. ONE_TWO contains all of the observations in ONE followed by all of the observations in TWO.

```
*Program 10-4 Using a SET statement to combine observations from two data sets - page 165;
data one_two;
   set one two;
run;
```

In this next example, we will combine data sets that do not have the same structure. Both data sets have the variables ID and NAME, but data set ONE has the variable WEIGHT while data set THREE has GENDER.

```
*Program 10-5 Setting two data sets containing different variables - page 166;
data one_three;
   set one three;
run;
```

The resulting data set is shown on page 166. The new data set ONE_THREE has four variables - all of the variables from ONE (ID, NAME, WEIGHT) plus all of the variables from THREE (ID NAME GENDER).

The first four observations come from ONE. All of the values of GENDER are missing since ONE did not have the variable GENDER.

The last three observations come from THREE. All of the values of WEIGHT are missing since THREE did not have the variable WEIGHT.

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.

These next examples will illustrate the concepts of combining data sets where the same variables have different attributes.

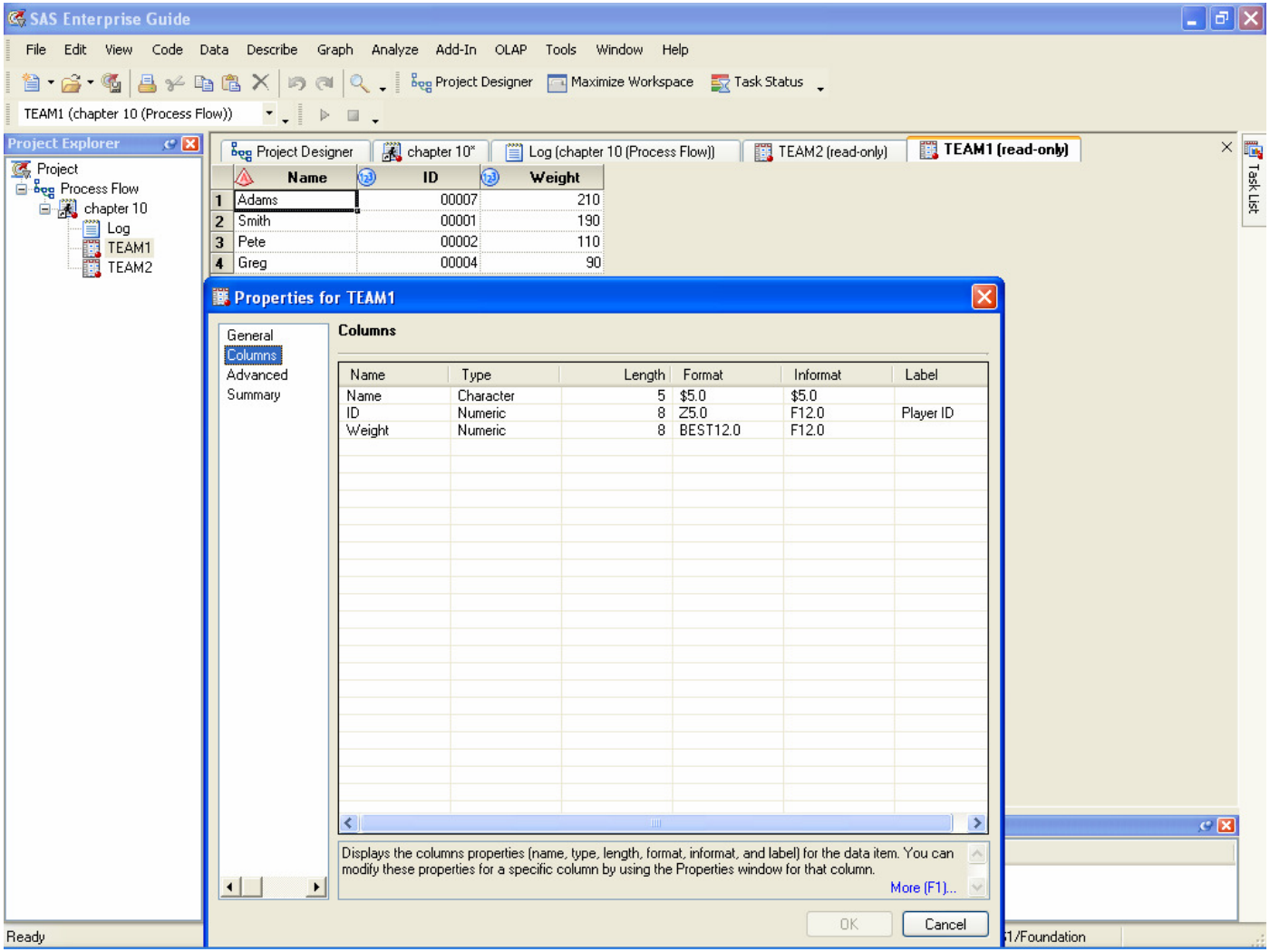
NAME has different lengths. It has no label in one data set and a label in the other data set.

ID has different labels in each data set. It is formatted in one data set and unformatted in the other.

```
data team1;
   length Name $ 5;
   input ID Name Weight;
   label id='Player ID';
   format id z5.;
datalines;
7 Adams 210
1 Smith 190
2 Pete  110
4 Greg 90
;
run;

data team2;
   length Name $ 12;
   input ID Name Weight salary;
   label name="Last Name"
         id='unique id';
   format salary dollar8.;
datalines;
9 Shea 120 1000
3 O'Brien 180 2000
5 Bessler 207 10000
;
run;
```

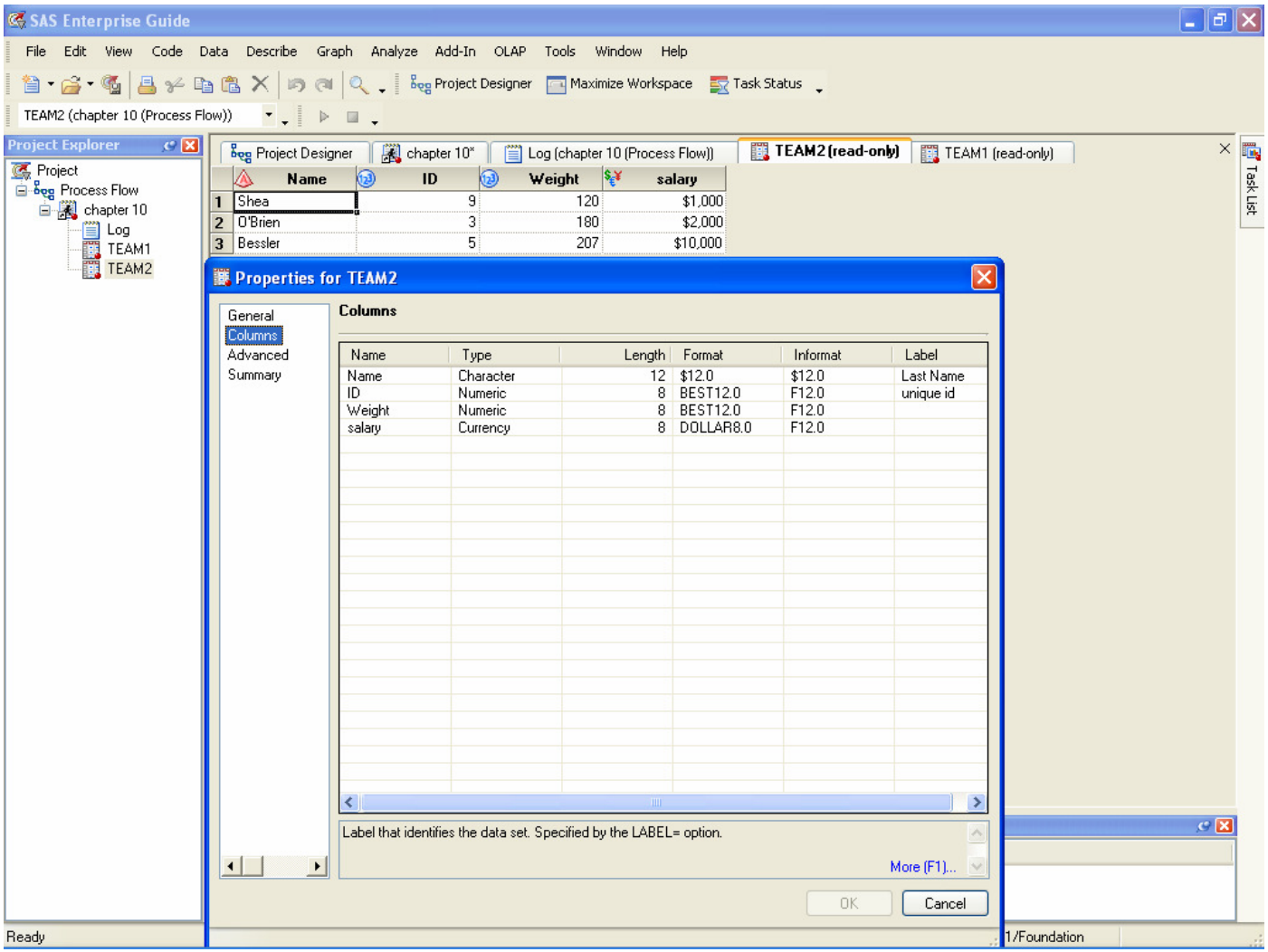
Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Here is the TEAM1 data set. The z5. format pads the value with leading zeroes to create a numeric with 5 digits.



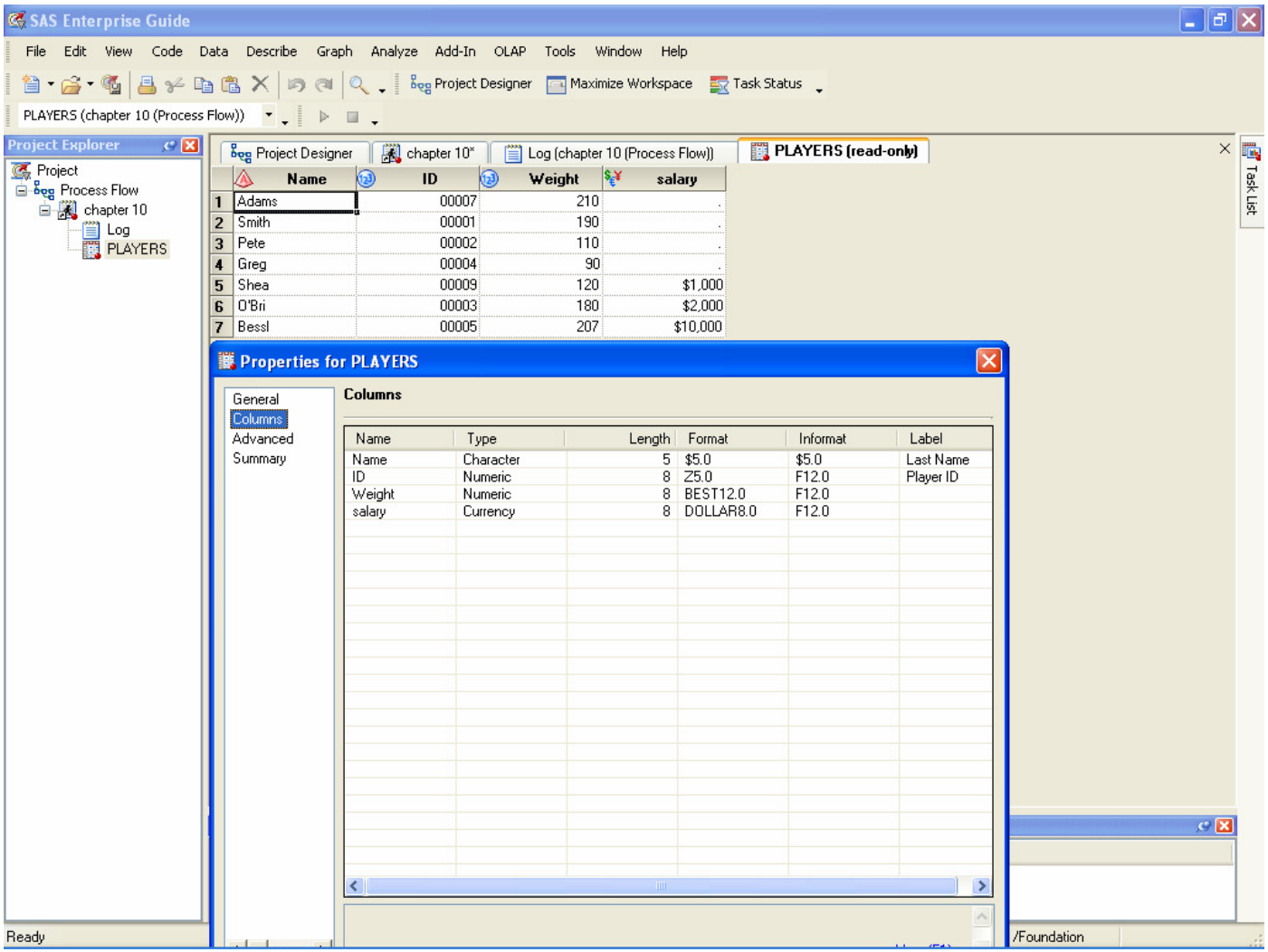
Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Here is the TEAM2 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.

Here is the PLAYERS 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.

When variables have different attributes in two data sets that are subsequently combined, SAS uses the first nonmissing attribute that it encounters as the data are combined.

Let's look at the variable attributes in PLAYERS.

NAME has length 5.
SAS first encountered NAME in TEAM1 and the length was 5. Thus NAME was set to length 5 in PLAYERS.
Notice that this causes truncation in the last two values from TEAM2.

The label for NAME is 'Last Name'.
The first label that SAS found for NAME was 'Last Name' in TEAM1.

ID is formatted with the .z5 format.
This format was applied in TEAM1.
The variable was not formatted in TEAM2.

The label ion ID s 'Player ID' which was the label found in TEAM1.
So although the variable is labelled "unique id" in TEAM2, that label is ignored since SAS has already assigned the label from TEAM1.

Modify the sample code to set in team2 followed by team1 and then compare the two data set attributes.

You may also find it helpful to modify the attributes by adding or removing labels or adding or removing formats.

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.

10.5 Interleaving Data Sets

In the previous section we saw that when data sets were combined using a SET statement, all of the observations from the first data set were included first, followed by all of the observations from the second data set.

There are times when you may wish to include some observations from one data set followed by some observations from the second data set followed by more observations from the first data set, etc.

This is know as interleaving data sets.

To create a new data set in this way, a BY statement must be used.

Further, when a BY statement is used in SAS, the data in the data set must first be rearranged or sorted using PROC SORT.

Once the data are sorted, SAS creates the new data set, by setting in the observations based on the values of the BY variables.

If there are observations with the same BY values in two or more of the data sets, SAS sets in the observations from the first data set, followed by the observations in the subsequent data sets in the order in which the data sets are listed on the SET 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.

The order of the observations in a data set can be changed by using PROC SORT.

The data are reordered in ascending order using the variable(s) specified on the BY statement.

PROC SORT must be used with care, since sorting the data replaces the data set.

As a general rule, it is best to use the OUT= option, especially when working with permanent data sets that may contain original source data.

PROC SORT is an inefficient procedure.

In processing the sort, SAS requires twice as much disk space as the size of the data set.

This is because during processing, the original data set is not changed and SAS writes the sorted observations to a temporary data set.

The original data set is not replaced until the sorting is complete and then the temporary data set is deleted.

In the data sets being used for this class, the space considerations are negligible.

But the space allocation issues can cause problems with extremely large data sets – and the procedure can run for a very long time.

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.

PROC SORT is required whenever you use a BY statement on a subsequent DATA step or PROC.

Syntax:

```
proc sort data=libref.datasetname1 out=datasetname2;
    by variablename1 variablename2...variablename'n';
run;
```

Setting multiple data sets together based on the values in certain variables is known as interleaving.

To interleave data sets, each data set first must be sorted by the values of these variables.

Syntax:

```
proc sort data=datasetname1;
    by variablename1 variablename2...variablename'n';
run;

proc sort data=datasetname2;
    by variablename1 variablename2...variablename'n';
run;

data datasetname;
  set datasetname1 datasetname2;
    by variablename1 variablename2...variablename'n';
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.

The data sets ONE_3 and TWO contain the variable ID.

ID=3 occurs in both data sets.

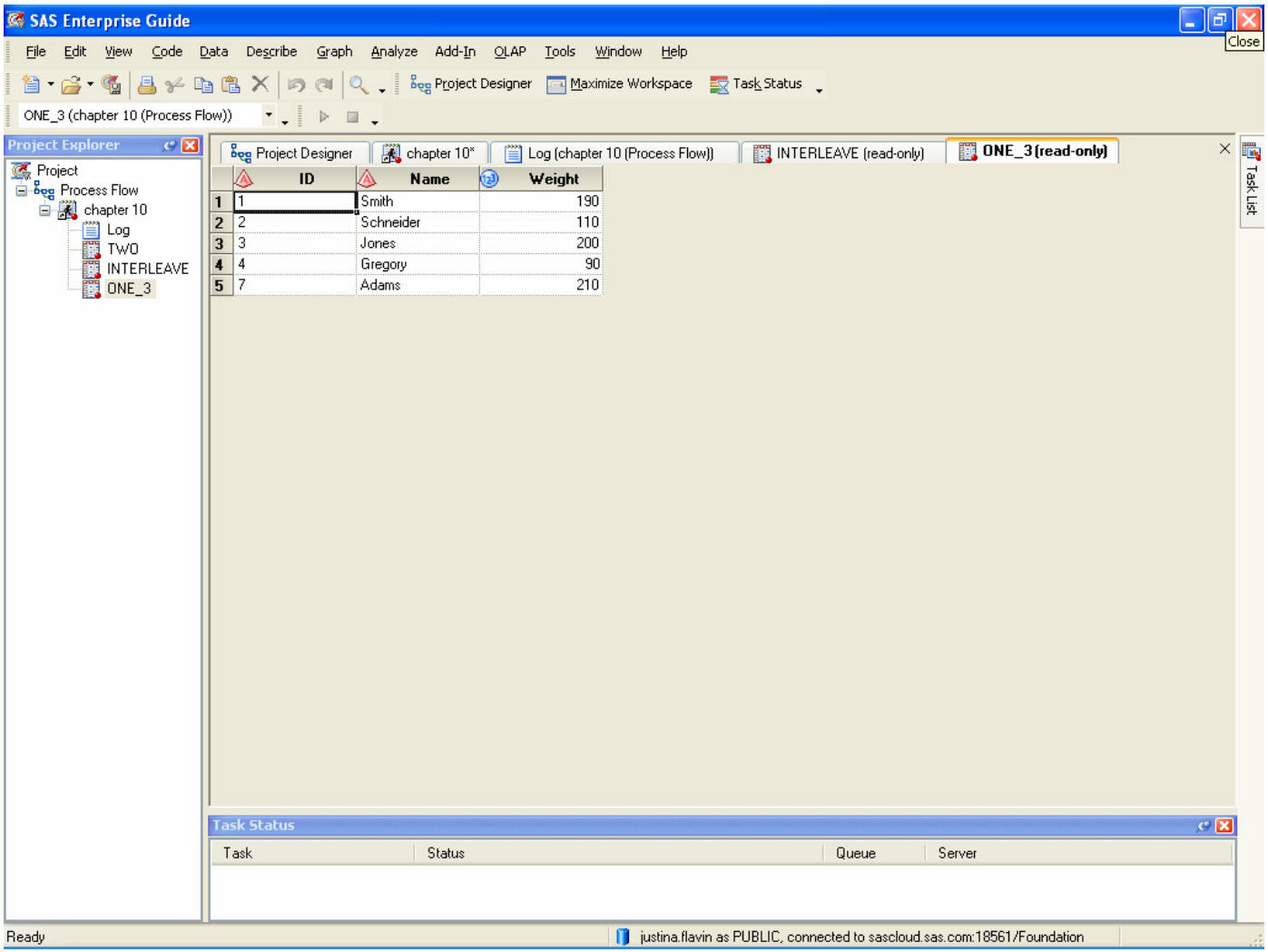
We wish to create a new data set by interleaving the two data sets using the variable ID.

Using PROC SORT, the observations will be reordered in ascending order of ID, then the data will be combined to create the data set INTERLEAVE.

```
*Program 10-6 Interleaving data sets - page 167;
proc sort data=one_3;
   by ID;
run;

proc sort data=two;
   by ID;
run;

data interleave;
   set one_3 two;
   by ID;
run;
```

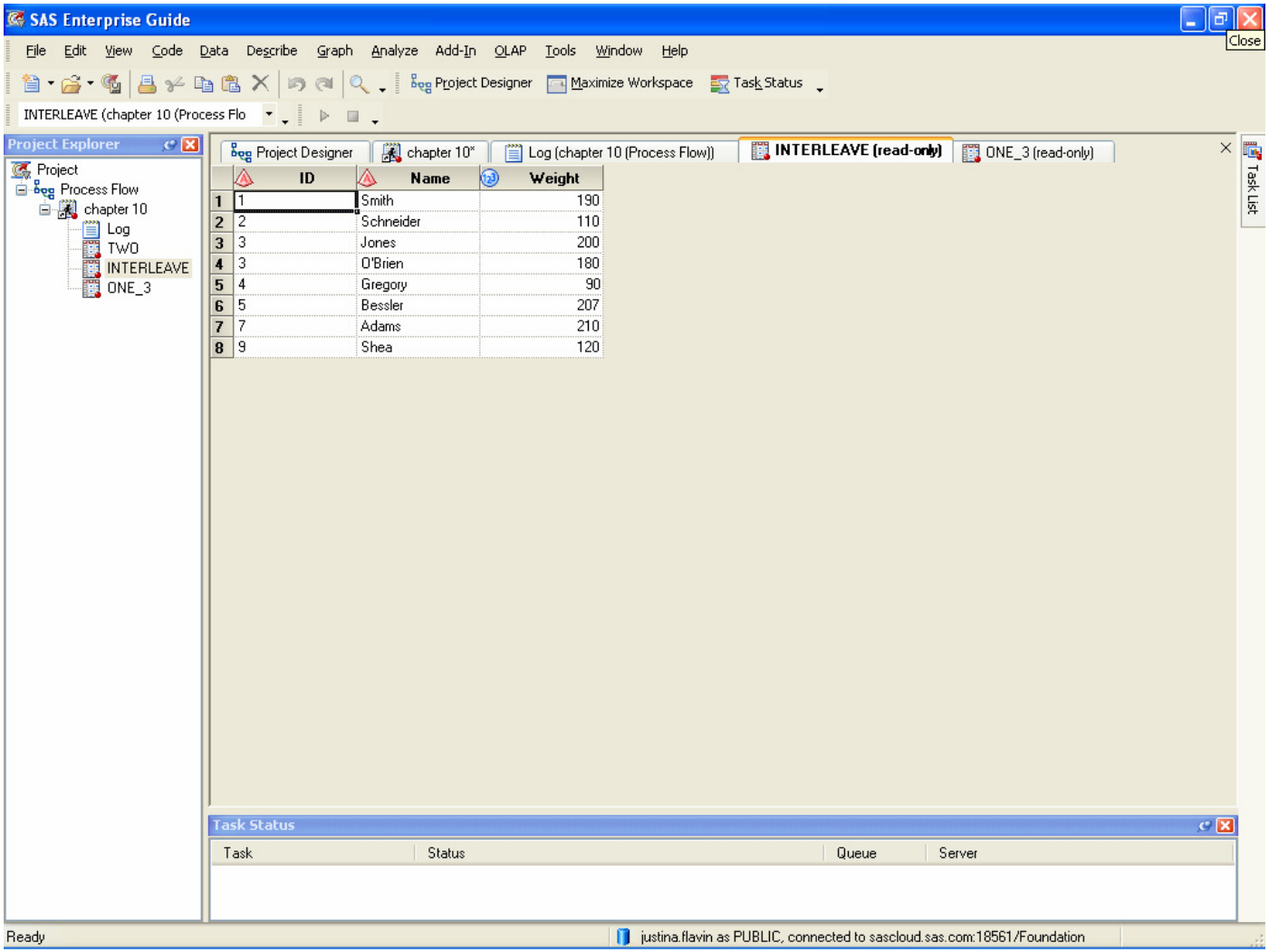
Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

The SORT procedure rearranges the observations in ascending order of ID.



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 observations are set into the new data set based upon the values of ID.
Each data set has one observation having ID=3.
The observation from ONE_3 is set in first because ONE_3 is the first data set specified on the SET 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.

10.6 Combining Detail and Summary Data

This section will be covered in a future lecture.

10.7 Merging Two Data Sets

The MERGE statement is used to combine data sets by joining observations from the different input data sets.

Usually this is done by combining observations based on the values of the variables on the BY statement. When a BY statement is used, the data must be sorted before the data sets can be merged.

Syntax:

```
proc sort data=datasetname1;
    by variablename1 variablename2...variablename'n';
run;

proc sort data=datasetname2;
    by variablename1 variablename2...variablename'n';
run;

data datasetname;
  merge datasetname1 datasetname2;
    by variablename1 variablename2...variablename'n';
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, there are two data sets EMPLOYEE and HOURS. EMPLOYEE contains the employee ID and employee name. HOURS contains the employee's job classification and hours worked.

We wish to create a single record for each employee that contains all of the information for that employee.

Each data set is sorted on the variable ID and then the data are merged.

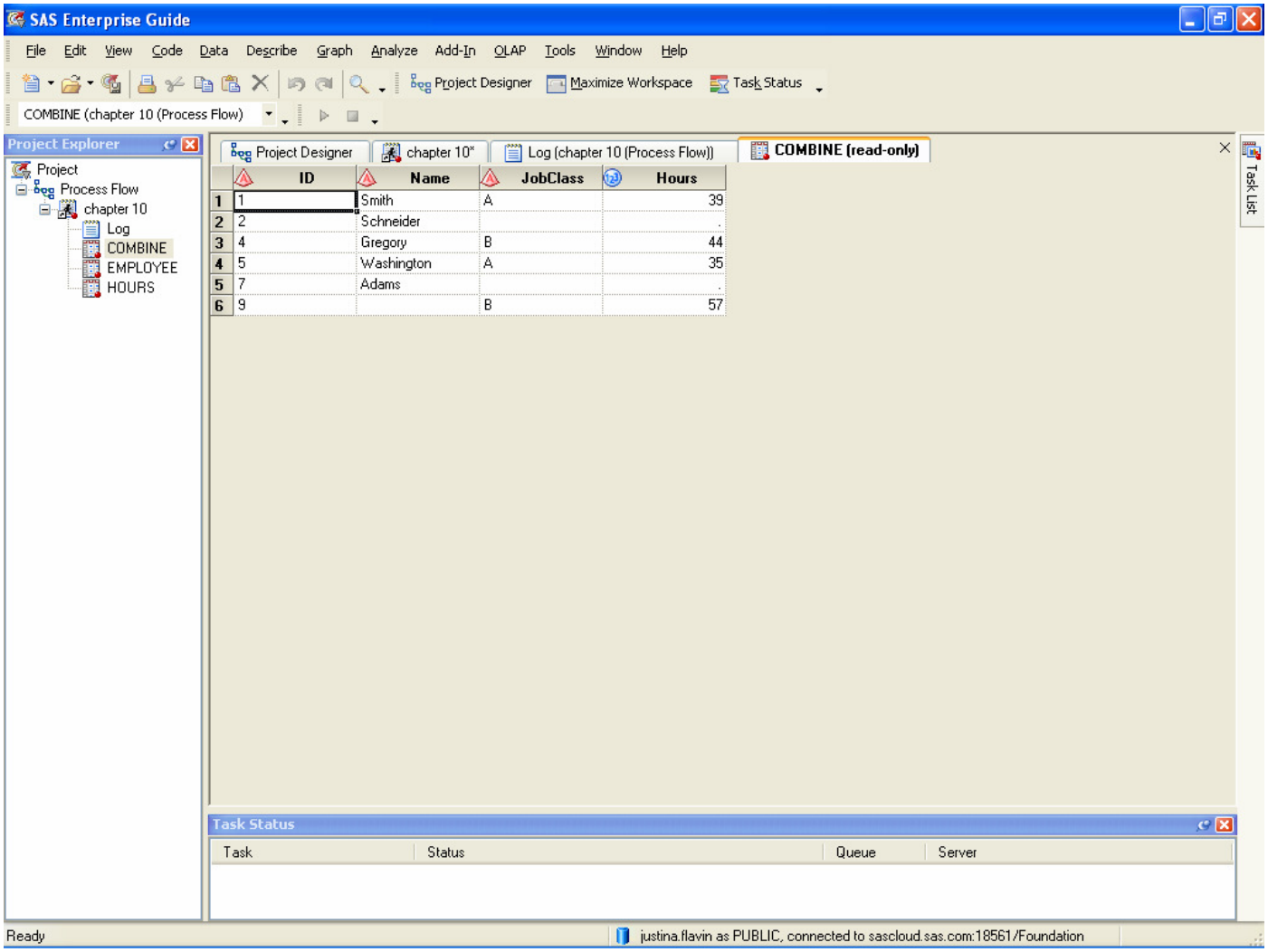
```
*Program 10-8 Merging two SAS data sets - page 171;
proc sort data=employee;
   by ID;
run;

proc sort data=hours;
   by ID;
run;

data combine;
   merge employee hours;
   by ID;
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.

Notice that the new data set contains a number of missing values because not all values of ID were in both data sets.

ID values 2 & 7 were in EMPLOYEE but not in HOURS, so JobClass and Hours are missing.

Likewise, ID 9 was only in HOURS, so Name is missing.



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.

You may find it helpful to compare the two data sets that are created using MERGE and SET.

The number of variables is the same, but the number of observations and the data values on each observation is different.

The number of observations when using a SET statement is (almost) always equal to the sum of the observations in each of the input data sets.

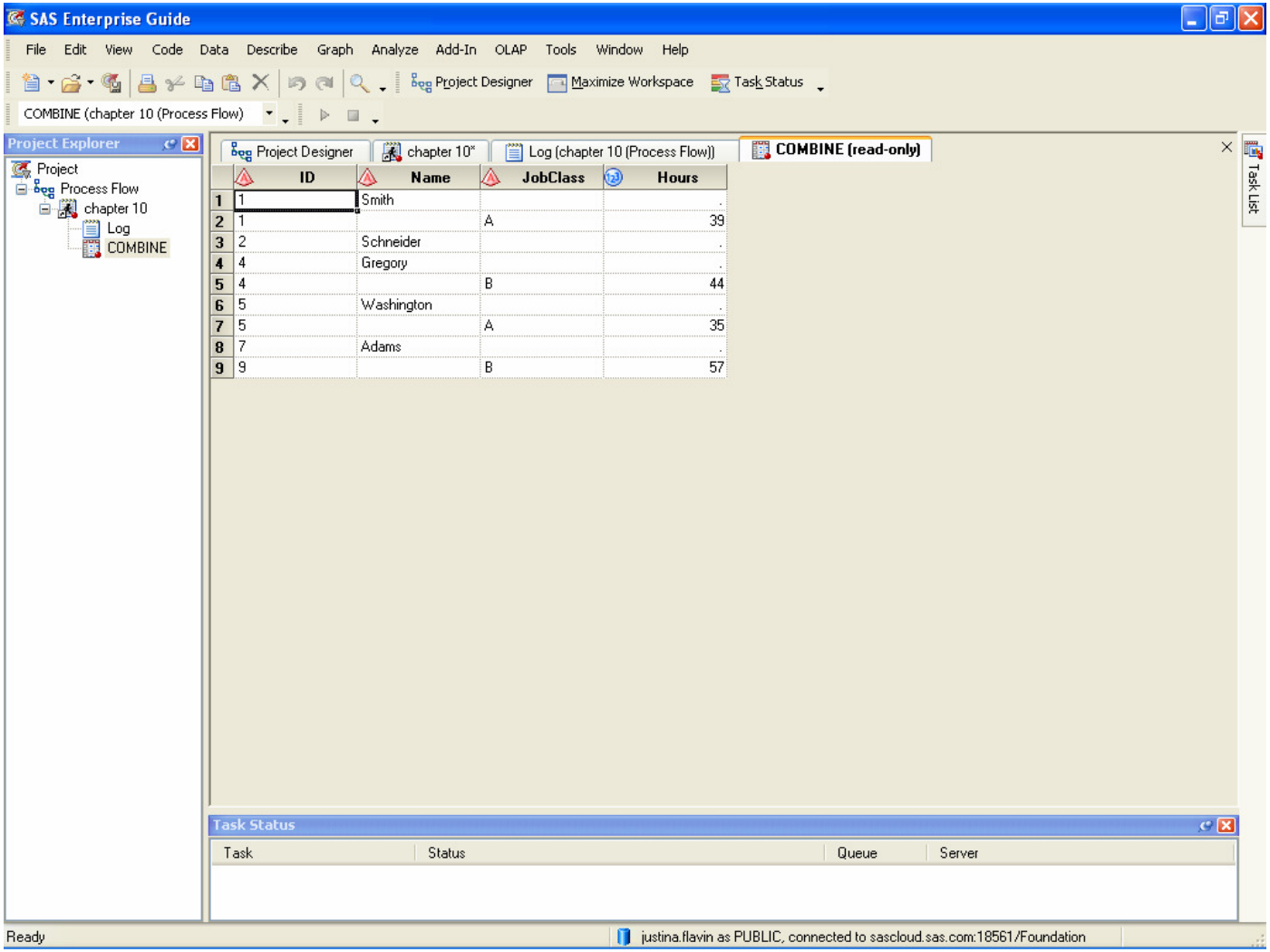
In this example, there are 5 observations in EMPLOYEE and 4 observations in HOURS, so COMBINE has 9 observations.

The number of observations when using a MERGE is usually less than the sum of the observations in each of the input data sets

```
data combine;
  set employee hours;
  by ID;
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.

Here is the data set COMBINE that was created using a SET statement.



	ID	Name	JobClass	Hours
1	1	Smith		.
2	1		A	39
3	2	Schneider		.
4	4	Gregory		.
5	4		B	44
6	5	Washington		.
7	5		A	35
8	7	Adams		.
9	9		B	57

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.

10.8 Omitting the BY Statement in a Merge

Usually a BY statement is needed when performing a merge.

A system option (MERGENOBY) can be used to control error and warning messages when a BY statement is omitted.

Syntax:

```
options mergenoby=value;
```

The *value* can be:

NOWARN – no message written to the log (default)

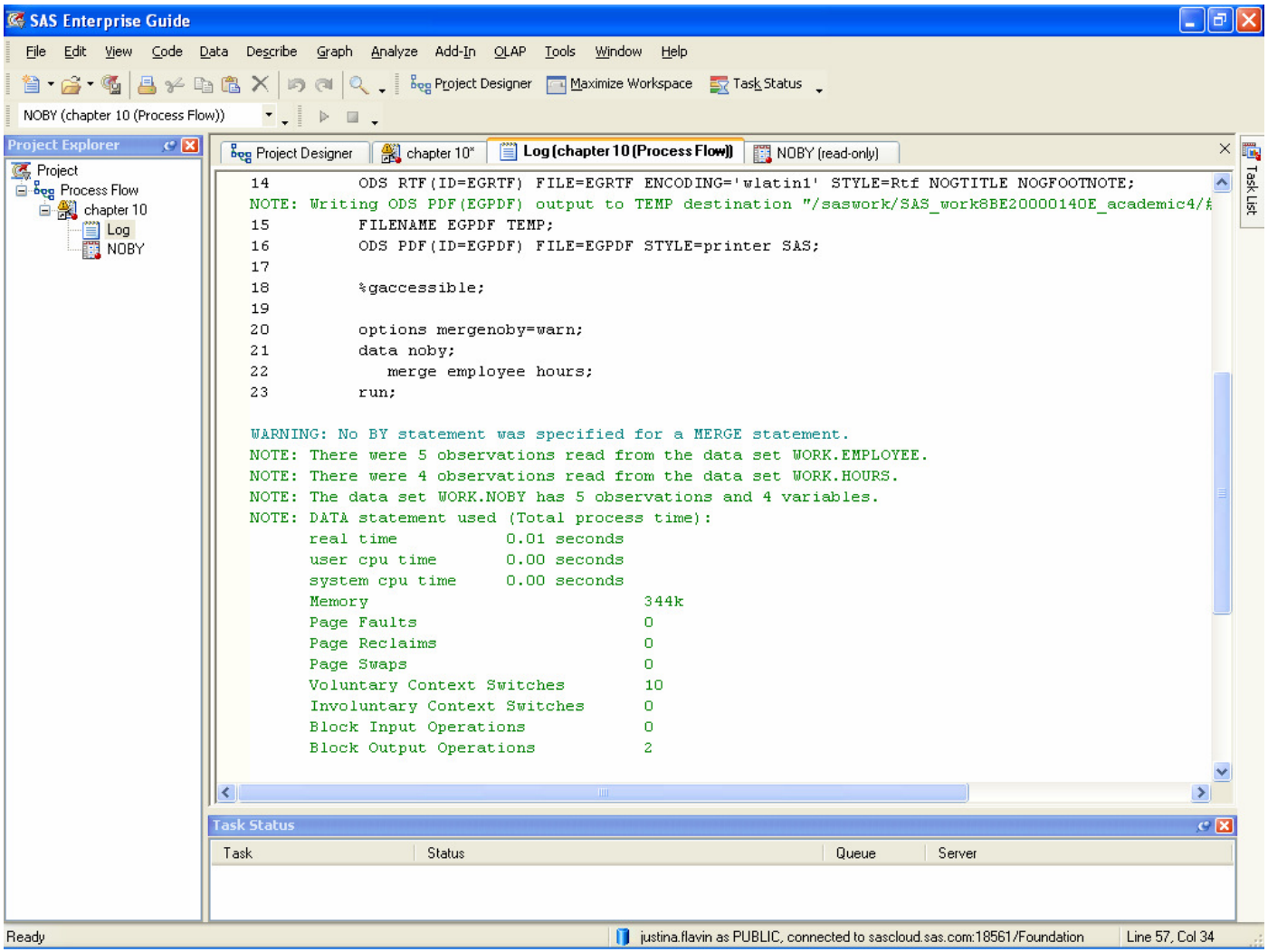
WARN – a warning message is written to the log

ERROR – SAS stops processing the code and issues an ERROR message

We will look at an example of merging two data sets without a BY statement.

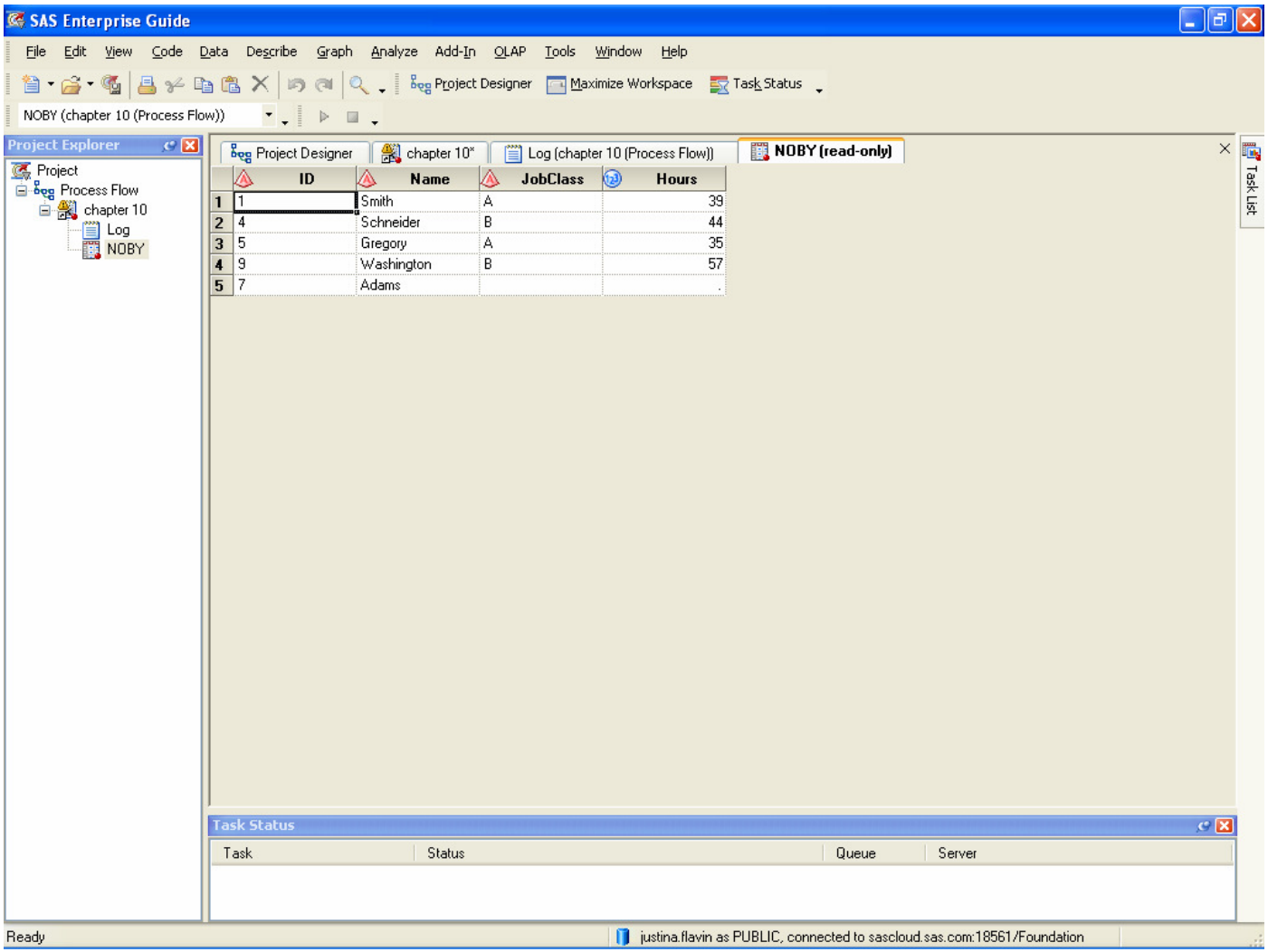
Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Notice the WARNING message in the log



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 NOBY.



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.

Without the use a BY statement, SAS joins the first observation in the first data set to the first observation in the second data set, the second observation in the first data set to the second observation in the second data set, etc.

Look at observation 2.

In the second observation in EMPLOYEE, ID=2, while in the second observation in HOURS, ID=4.

When SAS merges the two data sets, for any common variables that are not specified on a BY statement, the last value encountered is the value that is written out to the new data set.

Since HOURS is the second data set specified on the MERGE statement, its value of ID is written to COMBINE.

The same problem occurs with the next three observations, and the ID value from HOURS is written out to COMBINE.

For the last observation in COMBINE, ID=7. This is the value from EMPLOYEE because HOURS does not have a 5th observation.

This example illustrates the importance of including common variables on a BY statement when joining data sets using a MERGE statement.

If variables common to both data sets are not included on the BY statement, then these variables should be dropped from one or both data sets before the merge takes place.

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.

10.9 Controlling Observations in a Merged Data Set

Many times it's not necessary or desirable to create an output data set that has all observations from all of the input data sets.

The IN= option along with a subsetting IF statement can be used to control which observations are included from each input data set.

Syntax:

```
data datasetname;
  merge datasetname1(in=younameit1) datasetname2(in=younameit2);
  by variablename1 variablename2...variablename'n';
  if younameit1 and younameit2;
run;
```

In this example, we want to create a data set that only includes observations where the ID value is in both data sets. ID values that are only in one data set will not be included in COMBINE.

```
*Program 10-10 Using the IN= variables to select ID's that are in both data sets - page 174;
data combine;
   merge employee(in=InEmploy)
         hours(in=InHours);
   by ID;
   if InEmploy and InHours;
run;
```

The ID values that are common to both data sets are 1,4, 5.

These are the only ID values that will be included in COMBINE as shown on page 175.

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.

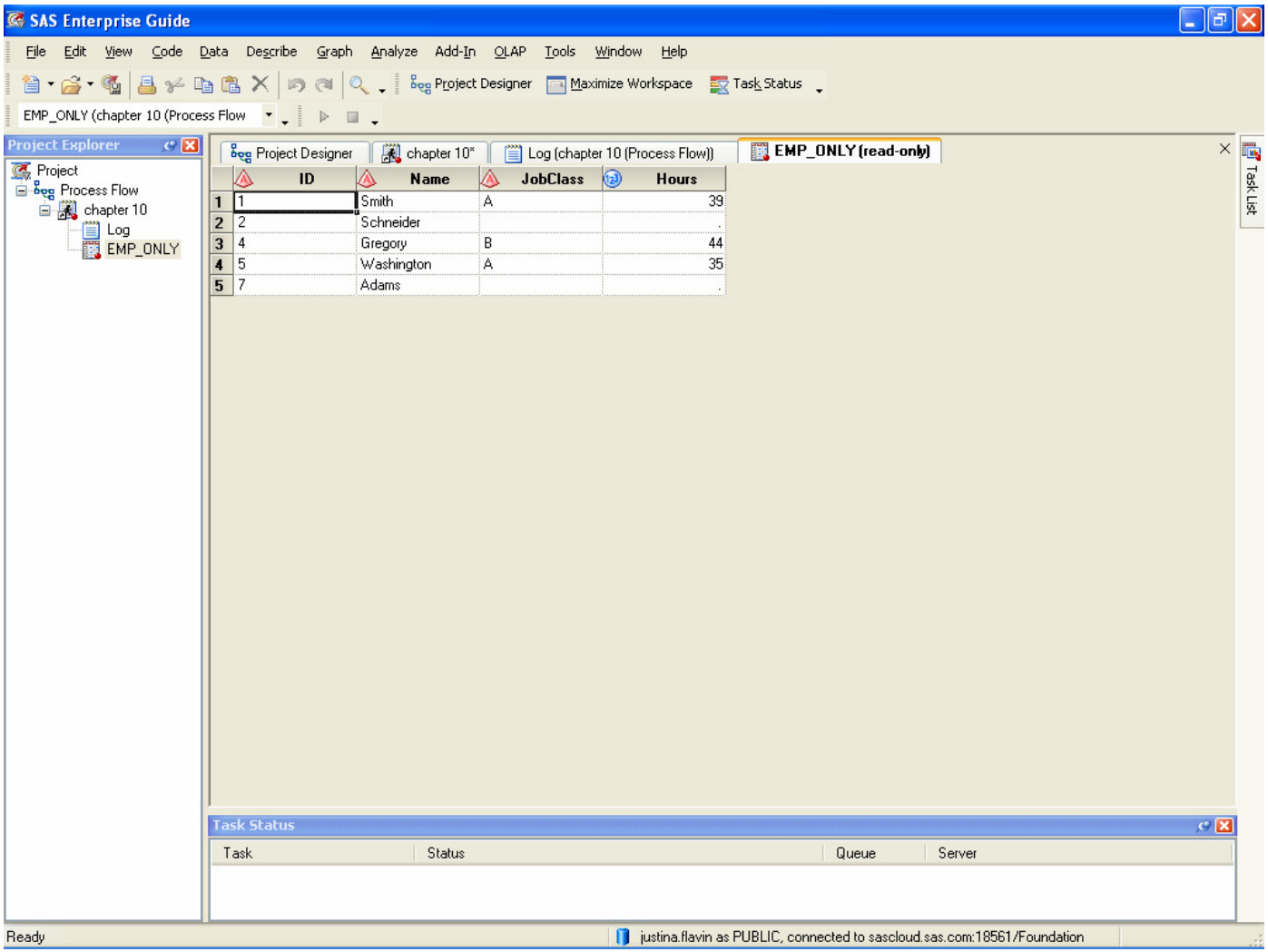
The IN= option can also be used to select all observations in one of the data sets.

In this example, EMP_ONLY will contain all of the observations from EMPLOYEE regardless of whether or not there is an observation with the same ID value in hours.

```
data emp_only;
   merge employee(in=InEmploy)
         hours;
   by ID;
   if InEmploy;
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.

Here is the data set EMP_ONLY



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.

10.10 More Uses for IN= Variables

Many times it is not necessary or desirable to create an output data set that has all of the variables from all of the input data sets.
Controlling which variables to include can be accomplished by the use of the DROP and KEEP statements.
When merging data sets that have common variables which are not used on the BY statement, the attributes of the variable (such as the length, label, and format) are the attributes of the variable in the first data set, but the variable values are those of the last data set.
For this reason it is usually a good idea to drop duplicate variables from one (or more) of the input data sets.

When multiple data sets are created in a data step, the IN= option along with a subsetting IF statement can also be used to control which observations are written out to each output data set.

```
*Program 10-11 More examples of using the IN= variables - page 175;
data in_both
     missing_name(drop = Name);
     merge employee(in=InEmploy)
           hours(in=InHours);
     by ID;
     if InEmploy and InHours then output in_both;
     else if InHours and not InEmploy then
     output missing_name;
run;
```

In this example, ID values that are in both EMPLOYEE and HOURS are included in IN_BOTH (ID values 1,4,5)

Observations that are in HOURS but not in EMPLOYEE are included in MISSING_NAME (ID values 9). Additionally the variable NAME is not included in the MISSING_NAME data set.

Observations that are in EMPLOYEE but not in HOURS are deleted (ID values 2,7)

The contents of the output data sets are shown on page 176.

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 example illustrates what happens when merging data sets that have common variables with different attributes.

This is similar to the earlier example with the SET statement.

Notice that ID=7 has Name=Adams in CHARLIE and NAME=Shea in LUNA.

```
data charlie;
   length Name $ 5;
   input ID Name Weight;
   label id='Player ID';
   format id z5.;
datalines;
7 Adams 210
1 Smith 190
2 Pete  110
4 Greg 90
9 Shea 120
3 O'Brien 180
5 Bessler 207
;
run;

data luna;
   length Name $ 12;
   input ID Name Weight salary;
   label name="Last Name"
         id='unique id';
   format salary dollar8. id 1.;
datalines;
7 Shea 120 1000
3 O'Brien 180 2000
5 Bessler 207 10000
;
run;
```

We will merge these two data sets to create two new data sets.

CHARLIE_LUNA is created by using the following statement: MERGE CHARLIE LUNA;

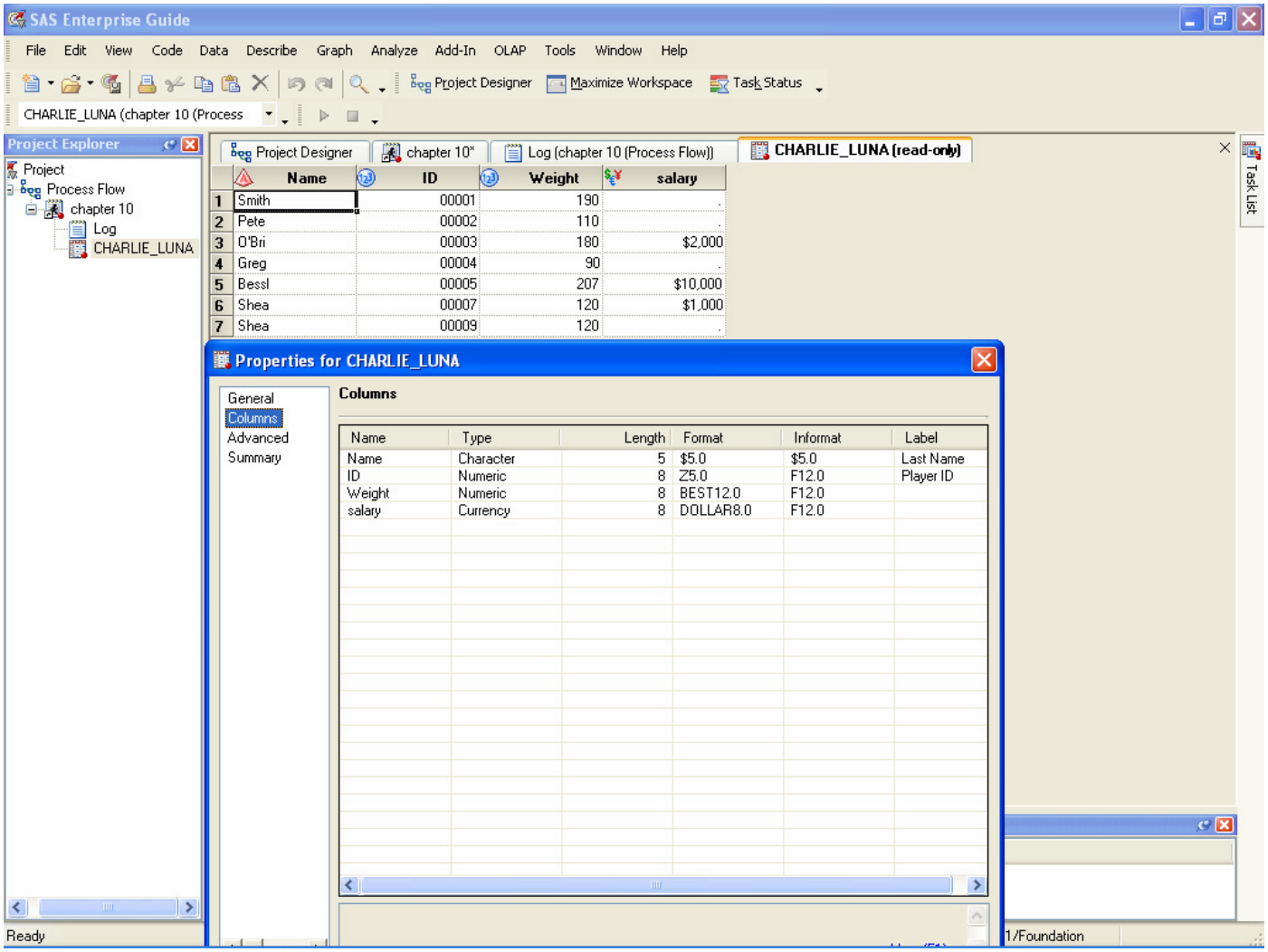
Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

For ID=7, NAME=Shea.

This is because ID=7 is in both data sets but the value of NAME is different in each of the data sets.

The last value of NAME that SAS encounters is the value that is written out to CHARLIE_LUNA.

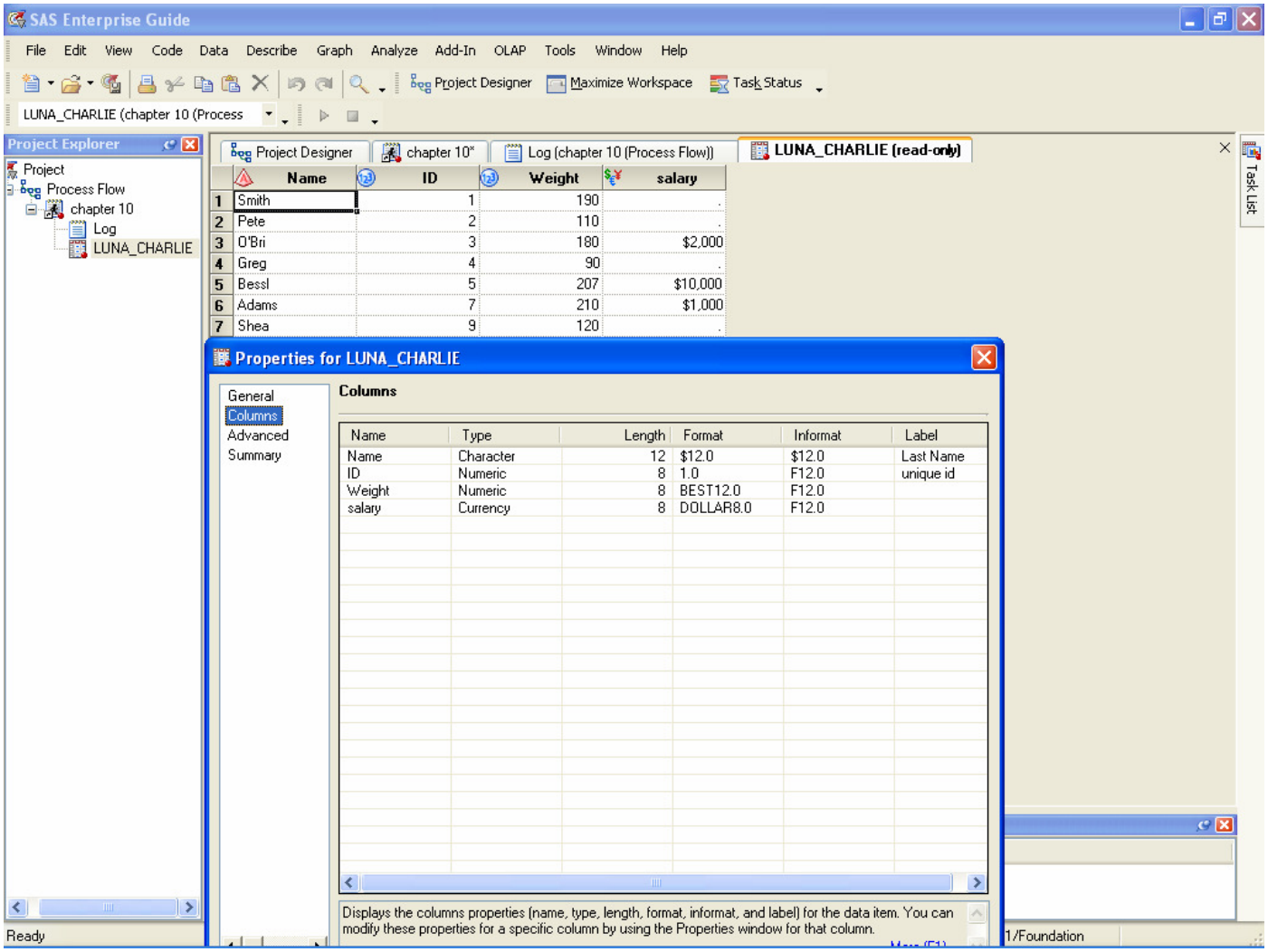
Since LUNA is the second data set listed on the MERGE statement, for ID=7 the last value of NAME is Shea.



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.

LUNA_CHARLIE is created by using the following statement: MERGE LUNA CHARLIE;

This time, for ID=7, NAME=Adams, because Adams is the value of NAME in the second data set (CHARLIE).



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 LUNA_CHARLIE, notice that the value of NAME in the 3rd observation is truncated to O'Bri.

The length of 12 is seemingly long enough to accommodate "O'Brien".

The problem lies in the fact that NAME was set to a length of 5 in CHARLIE, so O'Brien was already truncated to "O'Bri" in that data set.

Then when SAS performed the merge on ID, the second value that it encountered for ID=3 was "O'Bri" in CHARLIE, so this is the value written to LUNA_CHARLIE.

In CHARLIE_LUNA, NAME is also O'Bri, but for a different reason.

SAS first encounters the variable NAME in CHARLIE and thus sets the length in CHARLIE_LUNA to the length in CHARLIE (5).

Then when the merge is performed, the last value of NAME is encountered in LUNA.

This value is O'Brien.

However, because the length of NAME is 5 in CHARLIE_LUNA, O'Brien is truncated to 5 characters and hence the value becomes O"Bri.

Hopefully these examples have illustrated the importance of making sure your common variables have the same attributes when joining data using a SET or a MERGE.

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.

10.11 When Does a DATA Step End?

This section is omitted.

10.12 Merging Two Data Sets with Different BY Variable Names

Sometimes it is necessary to merge two data sets together using variables that have different names in the input data sets.

To accomplish the merge, the variable(s) must be renamed in one or more of the data sets.

Syntax:

```
proc sort data=datasetname1;
    by variablename1;
run;

proc sort data=datasetname2;
    by variablename2;
run;

data datasetname;
  merge datasetname1(rename=(variablename1=variablename2)) datasetname2;
    by variablename2;
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.

```
*Program 10-13 Merging two data sets, rename a variable in one data set - page 178;
data sesame;
   merge bert
         ernie(rename=(EmpNo = ID));
   by ID;
run;
```

In this example we have a variable called ID in BERT and a variable called EmpNo in ERNIE.

To merge these data sets together, one of the variables needs to be renamed.

EmpNo is renamed to ID as shown in this example and then the data sets can be merged on ID.

The same result could be obtained by using a rename statement on BERT to rename ID to EmpNo: `Bert(rename=(ID=EmpNo))` and changing the by statement to `by EmpNo.`

The resulting data set is shown on page 178.

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.

10.13 Merging Two Data Sets with Different BY Variable Data Types

To merge data sets using a variable that is of different types in the various input data sets, the variable must be converted so that it is the same type in all of the data sets.

This can be accomplished by renaming the variable and using a put or input function in one or more of the input data sets.

There are two ways to accomplish this, by converting a numeric variable to a character variable, or by converting a character variable to a numeric variable.

Method 1: Convert a numeric variable to a character variable

Syntax:

```
data datasetname1;
  set datasetname1(rename=(varname=temp));  *Renames numeric variable to a temporary name;
  varname=put(temp,formatname.);  *create a character variable with the (old) numeric name;
  drop temp;  *Delete the temporary numeric variable;
run;

data datasetname;
  merge datasetname1 datasetname2;
  by varname;
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.

```
/* Method 1: numeric to character */

*Program 10-14 Merging two data sets where the BY variables are different data types - page 179;
data division1c;
   set division1(rename=(SS = NumSS));
   SS = put(NumSS,ssn11.);
   drop NumSS;
run;

data both_divisions;
   ***Note: Both data sets already in order
      of BY variable;
   merge division1c division2;
   by SS;
run;
```

In DIVISION1, SS is a numeric variable while in DIVISION2, it is a character variable.

So one of these variables must be converted to the other type in order to merge the two data sets together.

In DIVISION1C, SS is first renamed to NumSS, so the variable SS no longer exists in DIVISION1.

Next a new character variable SS is created. The (old) numeric variable NumSS is no longer needed, so it is dropped from DIVISION1C. Now the two data sets may be merged on the common character variable SS.

The resulting data set is shown on page 180.

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.

Method 2: Convert a character variable to a numeric variable

Syntax:

```
data datasetname1;
  set datasetname1(rename=(varname=temp));  *Renames character variable to a temporary name;
  varname=input(temp,formatname.);  *create a numeric variable with the (old) character name;
  drop temp;  *Delete the temporary character variable;
run;

data datasetname;
  merge datasetname1 datasetname2;
  by varname;
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.

```
/* Method 2: character to numeric */

*Program 10-15 An alternative to Program 10-14 - page 180;
data division2n;
   set division2(rename=(SS = CharSS));
   SS = input(compress(CharSS,'-'),9.);
   ***Alternative:
   SS = input(CharSS,comma11.);
   drop CharSS;
run;

data both_divisions2;
   merge division1 division2n;
   by SS;
run;
```

In DIVISION2N, SS is first renamed to CharSS, so the variable SS no longer exists in DIVISION2.

Next a new numeric variable SS is created.

Note: the COMPRESS function removes the nonnumeric '-' characters from CharSS.

The (old) character variable CharSS is no longer needed, so it is dropped from DIVISION2N.

Now the two data sets may be merged on the common numeric variable SS.

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.

10.14 One-to-One, One-to-Many, and Many-to-Many Merges

There are three types of merges:

one-to-one merge – has exactly one observation for each value of the BY variables in each input data set

one-to-many merge – has exactly one observation for each value of the BY variables in one (or more) of the input data set(s) and more than one observation for each value of the BY variables in one of the input data set(s)

many-to-many merge – has more than one observation for each value of the BY variables in more than one of the input data set(s)

Usually a many-to-many merge does not create a desirable result.

All of the examples previously shown illustrated the one-to-one merge.

There was at most 1 observation in each data set for each value of the BY variable.

The contents of data sets BERT and OSCAR are shown on page 181.

BERT has one observation per unique ID value. OSCAR has several ID values with multiple observations for a particular value of ID.

Merging BERT and OSCAR is an example of a One-to-Many merge, and the result is shown at the bottom of page 181.

For ID=123, the value of X=90 is on all the observations in COMBINE.
Likewise for ID=333, the value of X=100 appears on all of the observations.

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.

As an example of a many-to-many merge, we have data sets ONE and TWO.

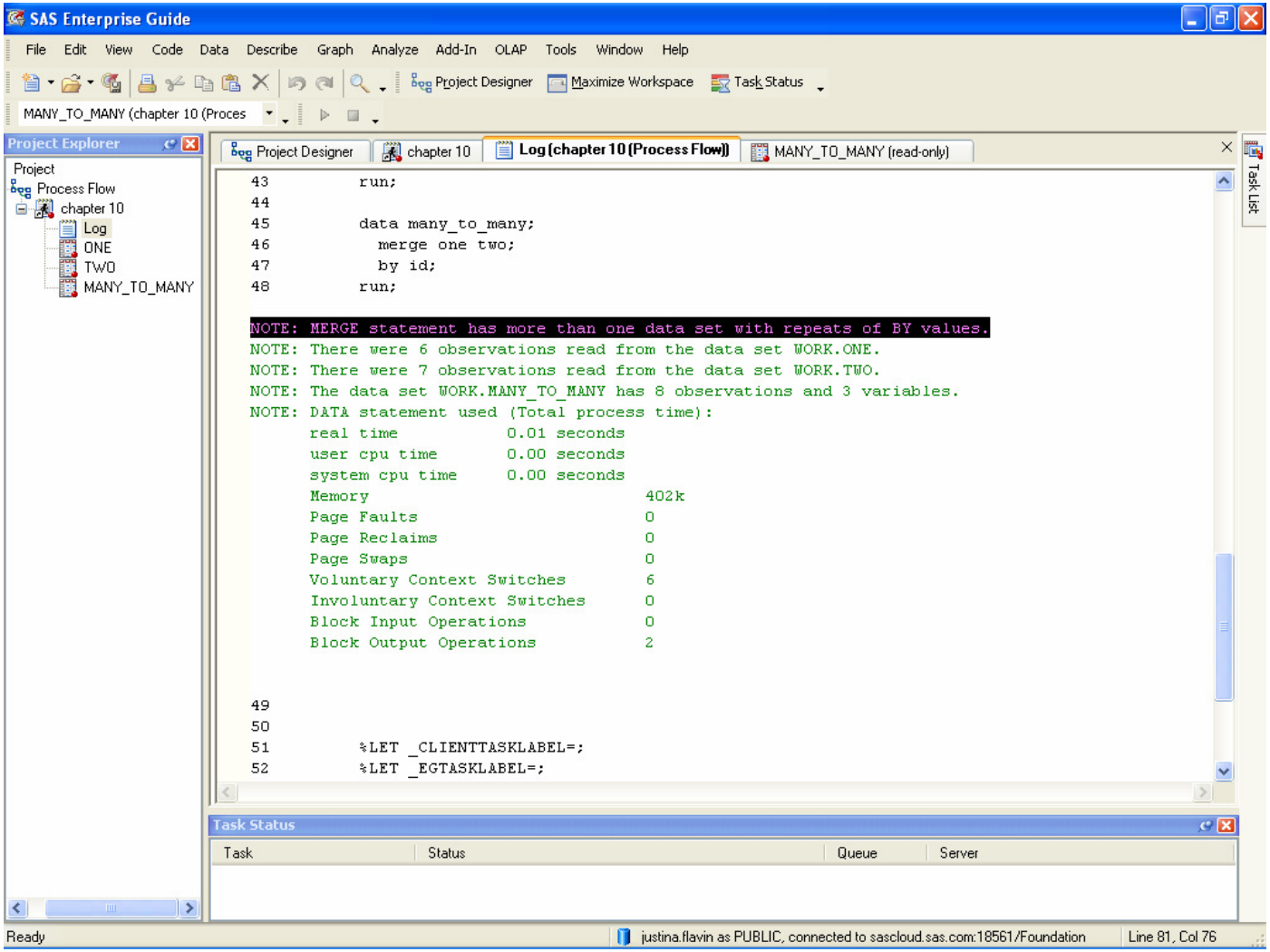
There are multiple values of ID=123 and 333 in both data sets.

```
data one;
   input ID X;
datalines;
123 90
123 80
222 95
333 100
333 150
333 200
;
run;

data two;
   input ID Y;
datalines;
123 3
123 4
123 5
222 6
333 7
333 8
400 9
;
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.

The data sets are merged together.

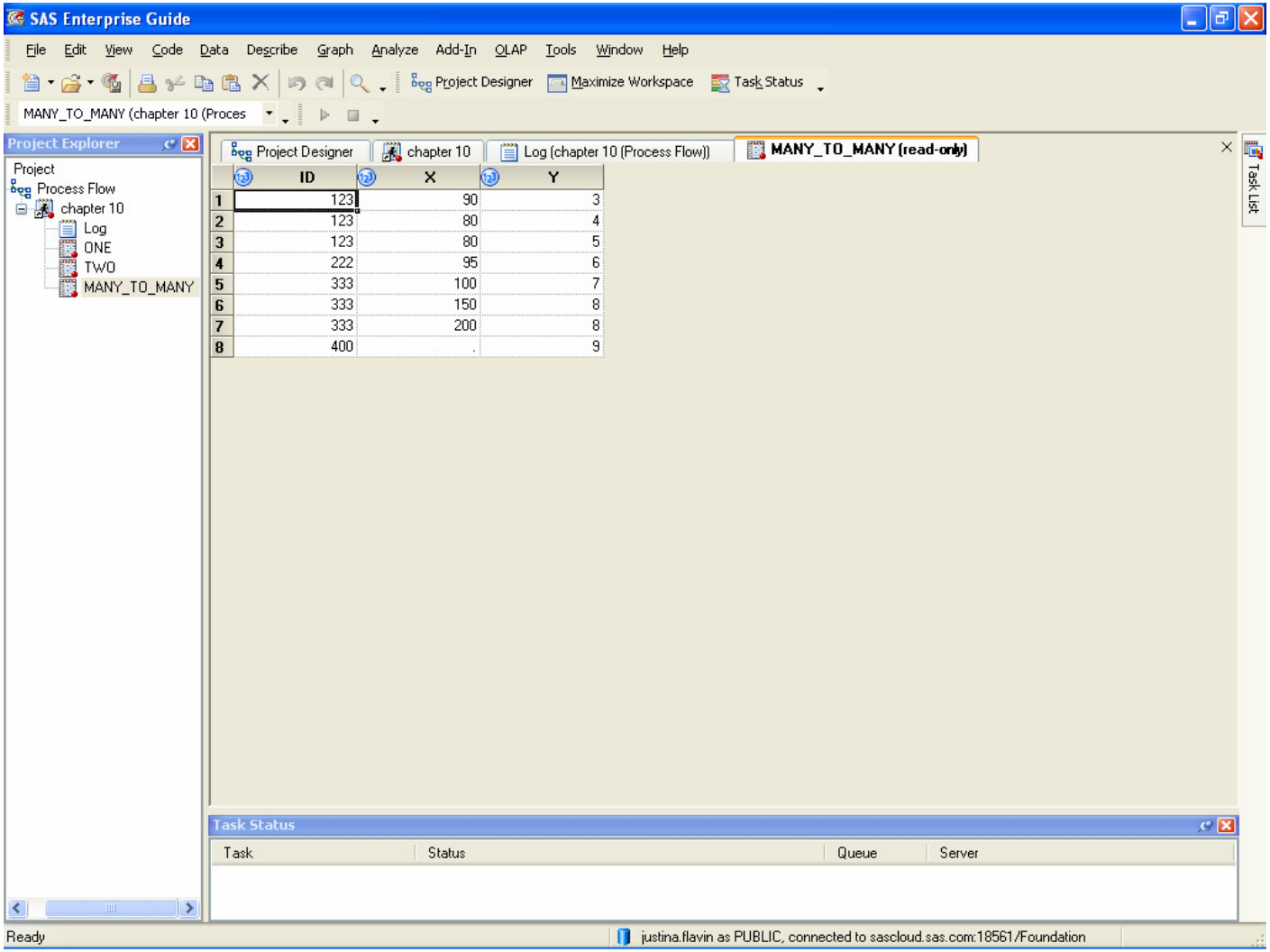
The NOTE in the log highlights that there may be a problem.



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 the output data set, the first two observations for ID=123 are joined from ONE and TWO.

A third observation of ID=123 is in data set TWO, but not in ONE, so SAS retains the value of X (80) from the previous observation. The same thing happens with ID=333. There are 3 observations in ONE and 2 observations in TWO for this ID value, so SAS retains the value of Y (8) on the 3rd observation. Finally, the value of 400 only exists in TWO, so the value of X is missing.



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.

10.15 Updating a Master File from a Transaction File

This section is omitted.

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.