SAS Programming (BIOL-4V190)

Chapter 4

Creating Permanent 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.

4.1 Introduction

Once you have your data in a SAS data set, there may be times when you wish to make this data set permanent.

In Chapter 3, all of the SAS data sets that we created were temporary data sets.
Temporary data sets exist while a SAS session is open, but once you exit the session, they disappear.

A permanent SAS data set is one which is saved to a file and can then be accessed in a future SAS session without having to recreate it.

Before we get into the concepts of permanent and temporary SAS data sets, we will go over some topics that will be introduced in the chapter.

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 LENGTH statement:

The **length** statement is used in the data step. Its purpose is to specify the length of a variable that will be created in the data step. When used, it is typically the first statement in the data step.

Syntax:

```
length variablename variablelength;
```

Example:

```
length lastname $20;
```

To find information on the LENGTH statement in the online documentation, refer to:

Base SAS -> SAS Language Reference: Dictionary -> Dictionary of Language Elements -> Statements -> LENGTH Statement

Variable Lists:

When you have a series of variables with the same prefix, you can refer to these in your code using variable list notation.

Syntax:

```
variablename1-variablename'n'
```

Example:

```
exam1-exam12;
```

SAS understands this to mean exam1 exam2 exam3…exam12

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.

4.2 SAS Libraries—The LIBNAME Statement
4.3 Why Create Permanent SAS Data Sets?

All SAS data sets have a two part name. The first part of the name is indicative of the physical location where the data set is stored. The second part of the name is the name of the data set. The two parts are separated by a dot.

For temporary data sets, the first part of the name is ‘WORK’ and is implied. It can be specified in your code, but usually is not. So in your SAS code DATA HORSE is the same as DATA WORK.HORSE

For permanent SAS data sets, WORK is replaced by a user supplied character string. The character string is defined on a global statement called the libref statement which also provides the physical location for the file. The libref statement is similar to the fileref statement that we saw in Chapter 3.

Syntax:

```
libname libref ‘directory name and path’;
data libref.datasetname;
```

In this class, there is only one physical location where we can store the data on the server, in the directory:
`/courses/u_ucsd.edu1/i_536036/c_629/saslib`

A libname statement could be:

```
libname class ‘/courses/u_ucsd.edu1/i_536036/c_629/saslib’;
```

Class is the libref and if a data set called HORSE existed in that directory, we could reference it in the program as DATA CLASS.HORSE

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.

Technical Note:

In this course, you will not be creating permanent SAS data sets.

Every student in the course is accessing exactly the same directory and data sets, so I cannot have students accidentally overwriting the data sets that are used in the examples and the homework assignments.

Therefore it is important that your libref statements also include the **access=readonly** parameter to prevent you from modifying the data:

```
libname mydata "/courses/u_ucsd.edu1/i_536036/c_629/saslib" access=readonly;
```

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 an example modified from the example on page 55. This example illustrates how you would create a permanent SAS data set on the server. I have run this code to create the permanent data set TEST_SCORES in the MOZART library and we will be using this data set to illustrate ways of examining SAS data sets.

```
*Program 4-1 Creating a permanent SAS data set;
libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib";

/* this is modified from the example in the book */

data mozart.test_scores;
  length Z_ID $3 Name $15;
  input Z_ID Score1-Score3 Name;
datalines;
1 90 95 98 larry
2 78 77 75 moe
3 88 91 92 curly
;
run;
```

Notice the use of a variable list on the input line. Instead of Score1 Score2 Score3, the variable list notation has allowed us to shorten this to Score1-Score3. This can be very helpful when you have many variables with the same prefix.

This example also illustrates the use of the length statement. The variables Z_ID and Name are thus initialized (or set up) as character variables of the specified width before the data are read into them.

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.

To illustate why the length statement is important and why it may be necessary in your code, let's look at an example.

If the length statement is removed (and the input line modified slightly) as shown below, SAS will create Z_ID as a numeric variable of length 8 (the default length) and Name as a character variable of length 8.

```
* the length statement;
data test_scores;
  * length Z_ID $3 Name $15;
  input Z_ID Score1-Score3 Name $;
datalines;
1 90 95 98 larry
2 78 77 75 moe
3 88 91 92 curly
;
run;
```

So you can see that the length statement can be used to control the variable type (character or numeric) and width of a variable, but with this example, its not completely obvious why you may want to do that.

So let's modify the data slightly as follows:

```
datalines;
1 90 95 98 larry
2 78 77 75 moe
3 88 91 92 curly
3 88 91 92 Supercalifragilisticexpialidocious
;
```

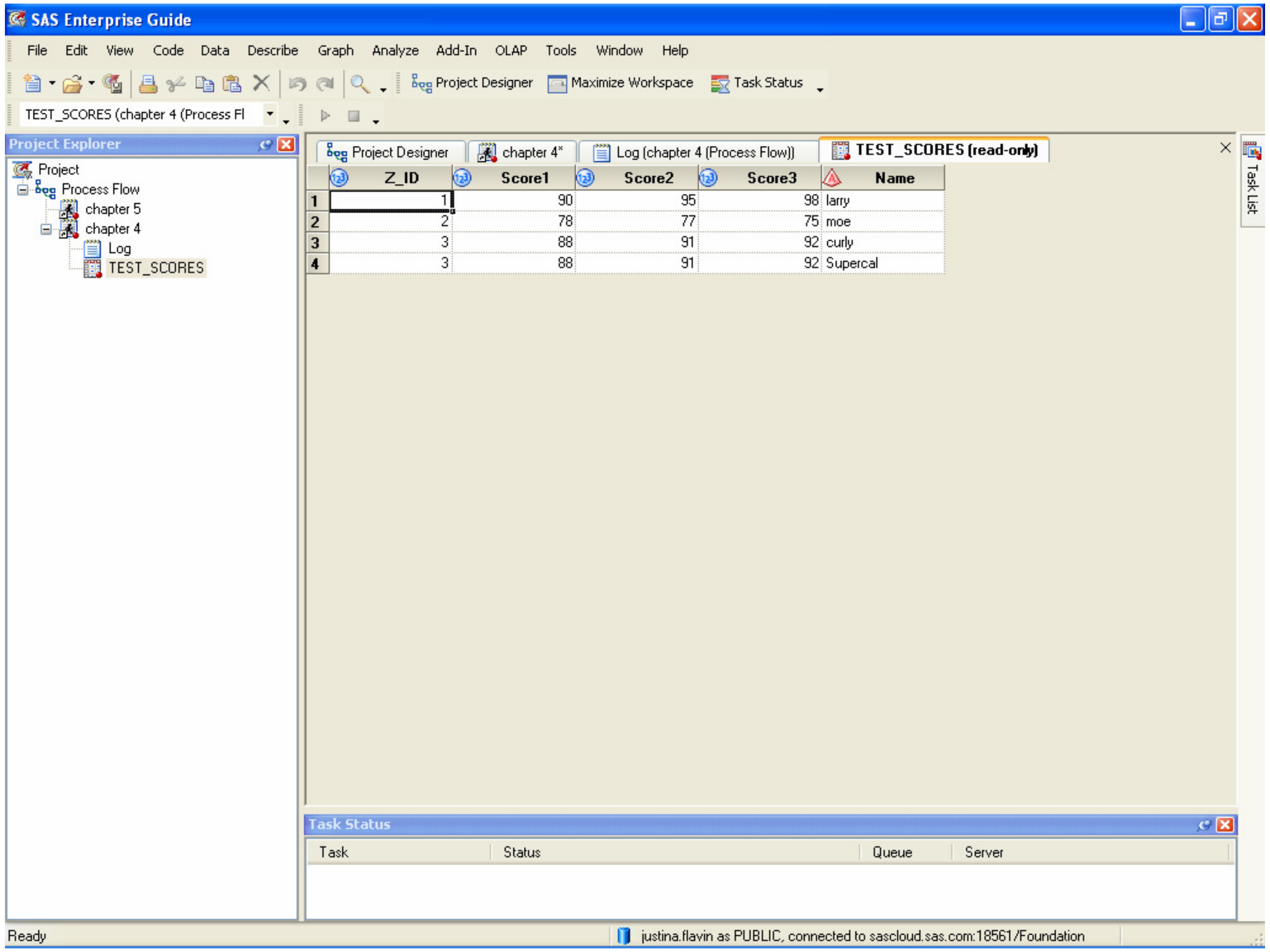
Copyright 2009, Justina Flavin. All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the copyright holder.

Notice what happened to the last data value for Name. It was truncated to 8 characters.

So to properly read in the variable Name, we need to add a length statement to this data step to specify a width that is at least long enough to hold the longest value in our data.

To fix the problem, add the following length statement to the code:

```
length Name $40;
```



	Z_ID	Score1	Score2	Score3	Name
1	1	90	95	98	larry
2	2	78	77	75	moe
3	3	88	91	92	curly
4	3	88	91	92	Supercal

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.

On a previous slide, I provided the code used to create the permanent TEST_SCORES data set, so you know how the data set is structured and what the variable values are.

But suppose I had not provided you with that information?

How would you go about examining the data set and its contents?

Three possible ways are:

PROC CONTENTS

PROC PRINT

View the library through Enterprise Guide

The next sections will cover these topics.

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.

4.4 Examining the Descriptor Portion of a SAS Data Set Using PROC CONTENTS

PROC CONTENTS generates output which provides detailed information about the contents of a SAS data set.

Syntax:

```
proc contents data=datasetname;
run;
```

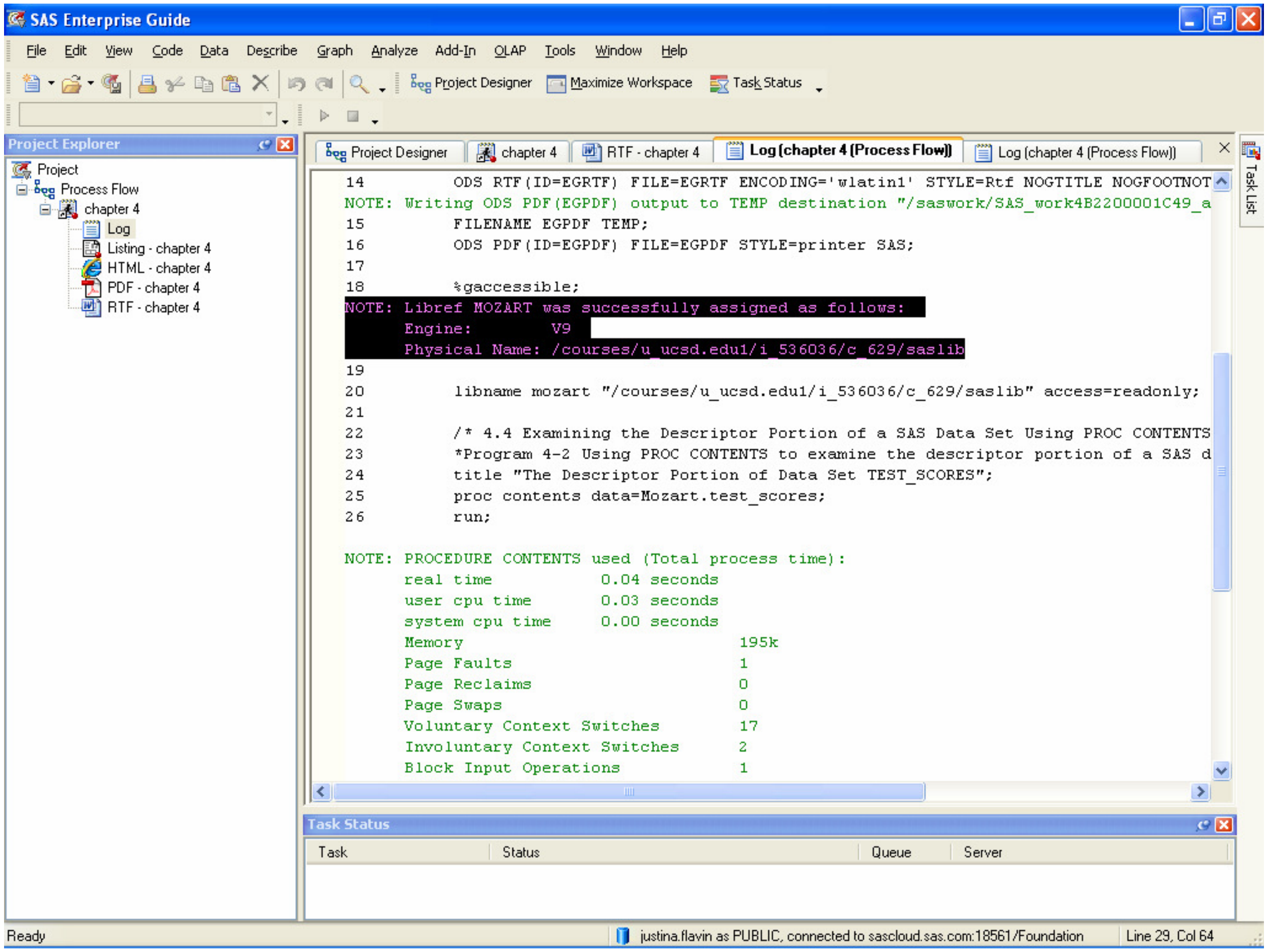
In the example, we will use PROC CONTENTS with the MOZART.TEST_SCORES data set:

```
libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib" access=readonly;

*Program 4-2 Using PROC CONTENTS to examine the descriptor portion of a SAS data set - page 56;
title "The Descriptor Portion of Data Set TEST_SCORES";
proc contents data=Mozart.test_scores;
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.

Before looking at the output, let’s look at the log.

Notice the note in the log that tells us that SAS successfully assigned the libref MOZART to the location specified in our libname statement.

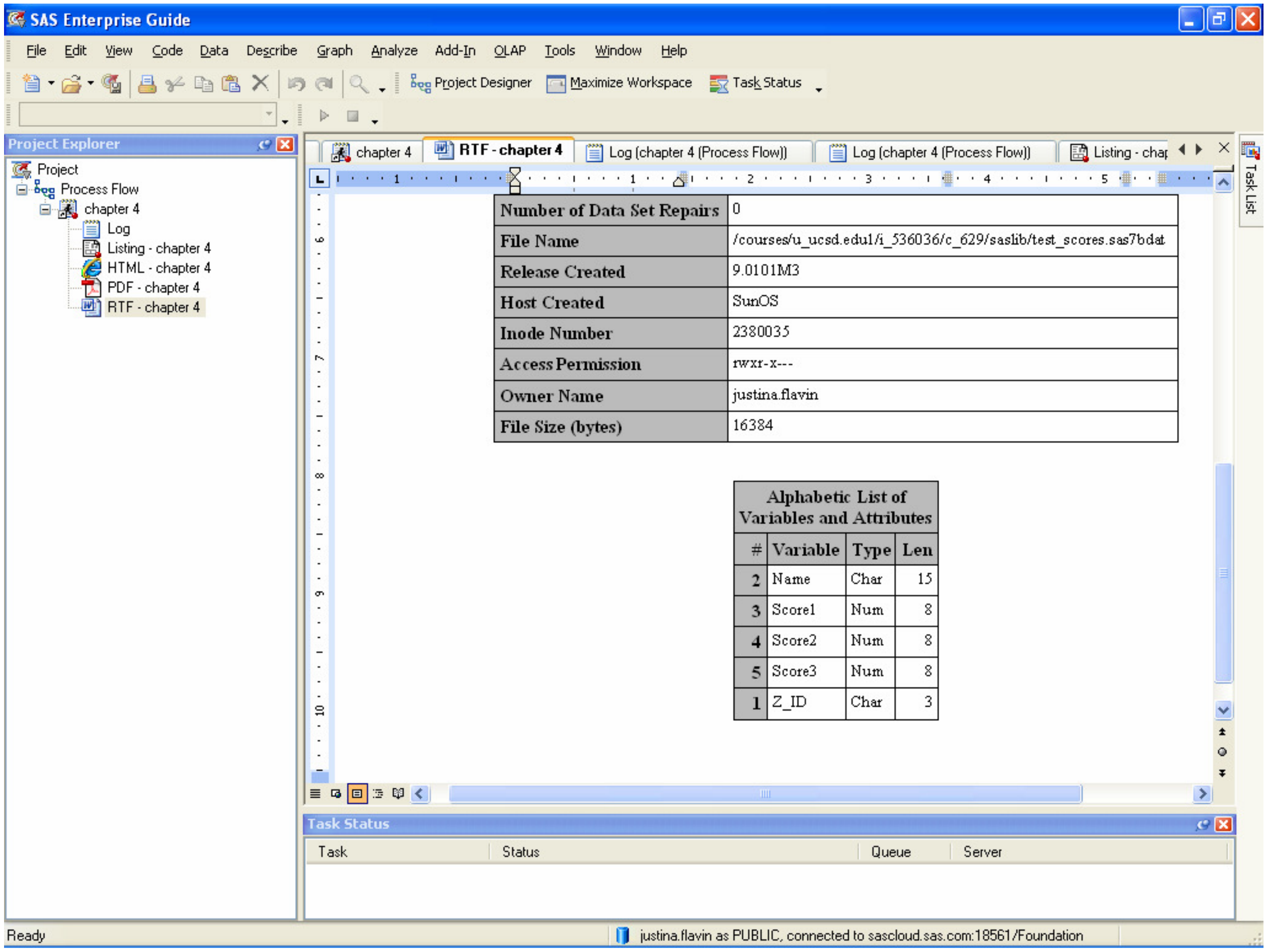

SAS Enterprise Guide
File Edit View Code Data Describe Graph Analyze Add-In OLAP Tools Window Help
Project Designer Maximize Workspace Task Status
Project Explorer
Project
Process Flow
chapter 4
Log
Listing - chapter 4
HTML - chapter 4
PDF - chapter 4
RTF - chapter 4
Project Designer
chapter 4
RTF - chapter 4
Log (chapter 4 (Process Flow))
Log (chapter 4 (Process Flow))
Task List
Task Status
Task
Status
Queue
Server
Ready
justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation
Line 29, Col 64

```
14         ODS RTF(ID=EGRTF) FILE=EGRTF ENCODING='wlatin1' STYLE=Rtf NOGTITLE NOGFOOTNOT
NOTE: Writing ODS PDF(EGPDF) output to TEMP destination "/saswork/SAS_work4B2200001C49_a
15         FILENAME EGPDF TEMP;
16         ODS PDF(ID=EGPDF) FILE=EGPDF STYLE=printer SAS;
17
18         %gaccessible;
NOTE: Libref MOZART was successfully assigned as follows:
      Engine:        V9
      Physical Name: /courses/u_ucsd.edu1/i_536036/c_629/saslib
19
20         libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib" access=readonly;
21
22         /* 4.4 Examining the Descriptor Portion of a SAS Data Set Using PROC CONTENTS
23         *Program 4-2 Using PROC CONTENTS to examine the descriptor portion of a SAS d
24         title "The Descriptor Portion of Data Set TEST_SCORES";
25         proc contents data=Mozart.test_scores;
26         run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           0.04 seconds
      user cpu time       0.03 seconds
      system cpu time     0.00 seconds
      Memory                            195k
      Page Faults                       1
      Page Reclaims                     0
      Page Swaps                        0
      Voluntary Context Switches        17
      Involuntary Context Switches      2
      Block Input Operations            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.

Scrolling through the output, we can see that there are three subsections of information about the data set.

The first section provides basic information about the data set, the second section provides system and storage information, and the third section provides detailed information about the variables which are listed in alphabetical order:



Number of Data Set Repairs	0
File Name	/courses/u_ucsd.edu1/i_536036/c_629/saslib/test_scores.sas7bdat
Release Created	9.0101M3
Host Created	SunOS
Inode Number	2380035
Access Permission	rwxr-x---
Owner Name	justina.flavin
File Size (bytes)	16384

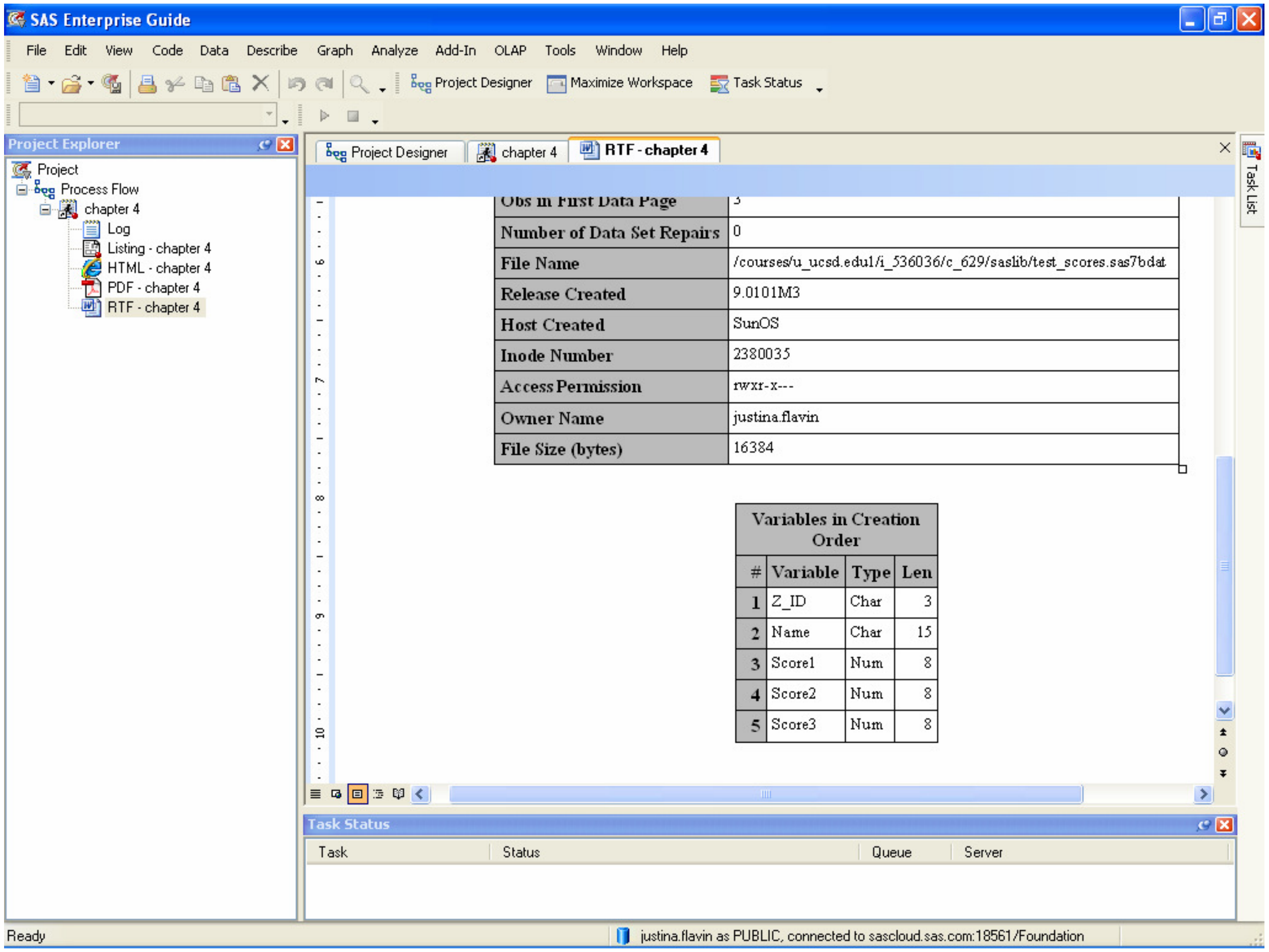
Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	Name	Char	15
3	Score1	Num	8
4	Score2	Num	8
5	Score3	Num	8
1	Z_ID	Char	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.

Next we will add the **varnum** option on the PROC CONTENTS statement.

This changes the third section of information – instead of the variables being listed alphabetically, they are now listed in the order in which they are saved in the data set.


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 4
Log
Listing - chapter 4
HTML - chapter 4
PDF - chapter 4
RTF - chapter 4
Project Designer chapter 4 RTF - chapter 4
Task Status
Task Status Queue Server
Ready
justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation

Obs in First Data Page	3
Number of Data Set Repairs	0
File Name	/courses/u_ucsd.edu1/i_536036/c_629/saslib/test_scores.sas7bdat
Release Created	9.0101M3
Host Created	SunOS
Inode Number	2380035
Access Permission	rwxr-x---
Owner Name	justina.flavin
File Size (bytes)	16384

Variables in Creation Order

#	Variable	Type	Len
1	Z_ID	Char	3
2	Name	Char	15
3	Score1	Num	8
4	Score2	Num	8
5	Score3	Num	8

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.

Finally, if you wish to have the variables listed in the output both alphabetically and in data set order, the **position** option can be used instead of the **varnum** option on the PROC CONTENTS statement as shown below:

```
title Adding the position option;
proc contents data=Mozart.test_scores position;
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.

A few more odds and ends before we finish up with PROC CONTENTS.

Suppose you really didn’t like the libref MOZART and wanted to change it.
Could you do so?
Yes, by simply changing the libref on the libname statement and reissuing the statement.

In this next example, we change MOZART to PROJ99 & reissue the libname statement.
Then in our proc contents, we reference the data set as proj99.test_scores

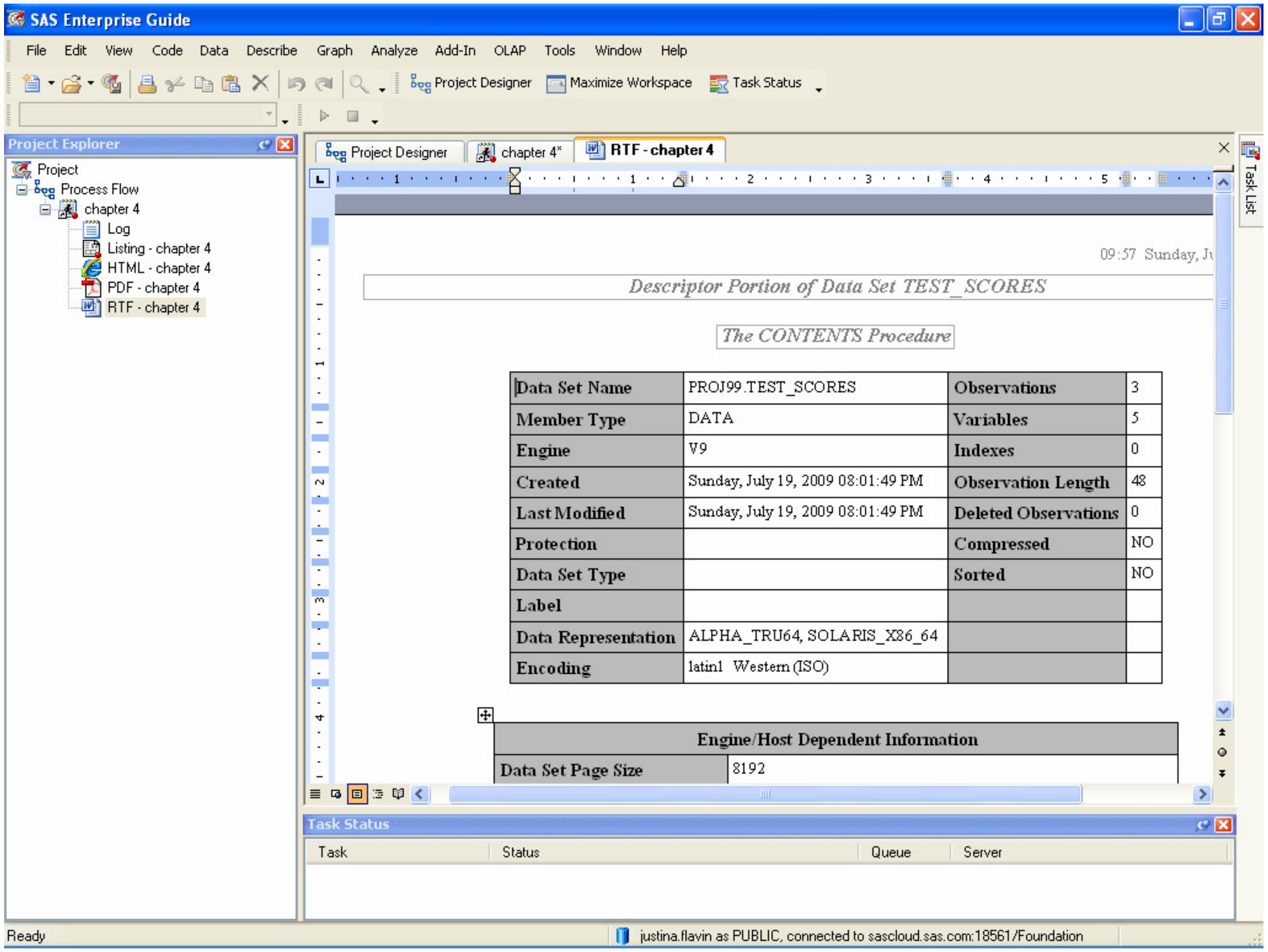
```
*Program 4-4 Using a LIBNAME in a new SAS session - page 58;
libname proj99 "/courses/u_ucsd.edu1/i_536036/c_629/saslib" access=readonly;

title "Descriptor Portion of Data Set TEST_SCORES";
proc contents data=proj99.test_scores varnum;
run;
```

One thing to note about libname statements is that “renaming” the libref as we did here doesn’t have the effect of cancelling out the old libref.
So in your SAS session, you should now have both MOZART and PROJ99 available.

They both reference the same physical location, so which you chose to use is a matter of preference.

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 now in our PROC CONTENTS output that the data set name is listed as PROJ99.TEST_SCORES



09:57 Sunday, Ju

Descriptor Portion of Data Set TEST_SCORES

The CONTENTS Procedure

Data Set Name	PROJ99.TEST_SCORES	**Observations**	3
Member Type	DATA	**Variables**	5
Engine	V9	**Indexes**	0
Created	Sunday, July 19, 2009 08:01:49 PM	**Observation Length**	48
Last Modified	Sunday, July 19, 2009 08:01:49 PM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	ALPHA_TRU64, SOLARIS_X86_64		
Encoding	latin1 Western (ISO)		

Engine/Host Dependent Information	
Data Set Page Size	8192

4.5 Listing All the SAS Data Sets in a SAS Library Using PROC CONTENTS

When you start up a SAS session, the system initializes or creates some variables.

These are known as **automatic** variables.

These are temporary variables and, like work data sets, disappear when the session is closed.

Additionally, some of these types of variables are also initialized when a data set is created.

The keyword **_all_** is one such temporary automatic variable.

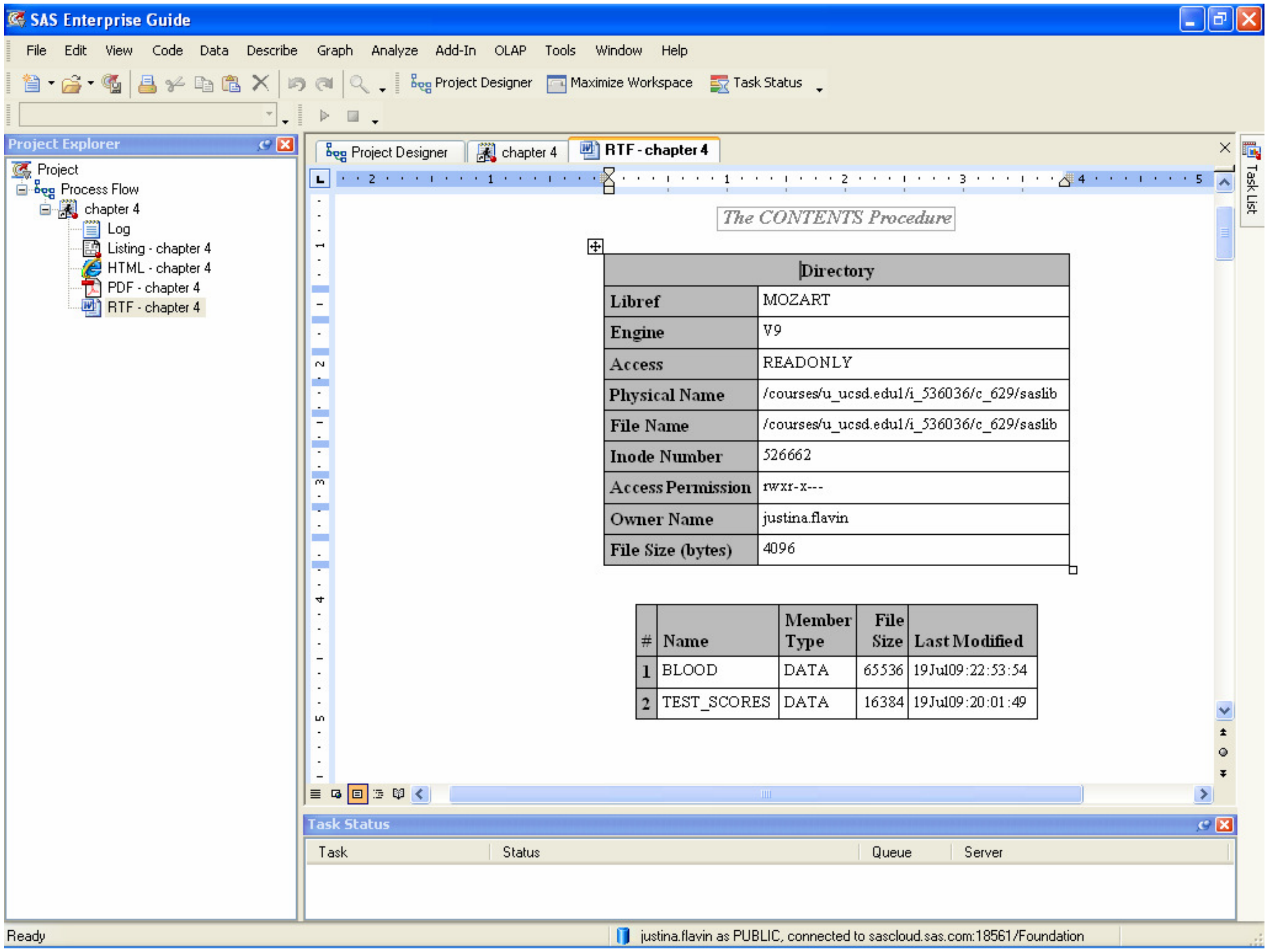
It can be used in PROC CONTENTS to reference all data sets in a library, by replacing the data set name with the keyword _all_

```
*Program 4-5 Using PROC CONTENTS to list the names of all the SAS data sets in a SAS library - page
59;
title "Listing All the SAS Data Sets in a Library";
proc contents data=Mozart._all_;
run;
```

If there are many data sets in the library, much output 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.

When the keyword _all_ is used in PROC CONTENTS, in addition to the summary information about each data set that exists in the library, SAS also provides an overview page at the beginning of the output which provides a list of the data sets (the library members).



The CONTENTS Procedure

Directory	
Libref	MOZART
Engine	V9
Access	READONLY
Physical Name	/courses/u_ucsd.edu1/i_536036/c_629/saslib
File Name	/courses/u_ucsd.edu1/i_536036/c_629/saslib
Inode Number	526662
Access Permission	rwxr-x---
Owner Name	justina.flavin
File Size (bytes)	4096

#	Name	Member Type	File Size	Last Modified
1	BLOOD	DATA	65536	19Jul09:22:53:54
2	TEST_SCORES	DATA	16384	19Jul09:20:01:49

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.

If you have a large number of data sets in the library, it may be a good idea to add the **nods** option on your PROC CONTENTS statement when listing out all of the data set information.

The **nods** option generates a list of the data set names only without any detail information.

```
proc contents data=proj99._all_ nods;
run;
```

The output generated by adding the nods option is the same output that was shown on the previous slide.

We have just seen that PROC CONTENTS provides information about the structure, contents and location of the data set.

It does not provide detail information about the individual data values.

A way to view that information is through the use of PROC PRINT

4.7 Viewing the Data Portion of a SAS Data Set Using PROC PRINT

The basics of PROC PRINT have been explained in a previous lecture.

Program 4-6 provides an example for these 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.

We have now seen how to use two procedures, PROC CONTENTS and PROC PRINT to inspect data sets.

If output is not required, we can use the Enterprise Guide menus to view the data.

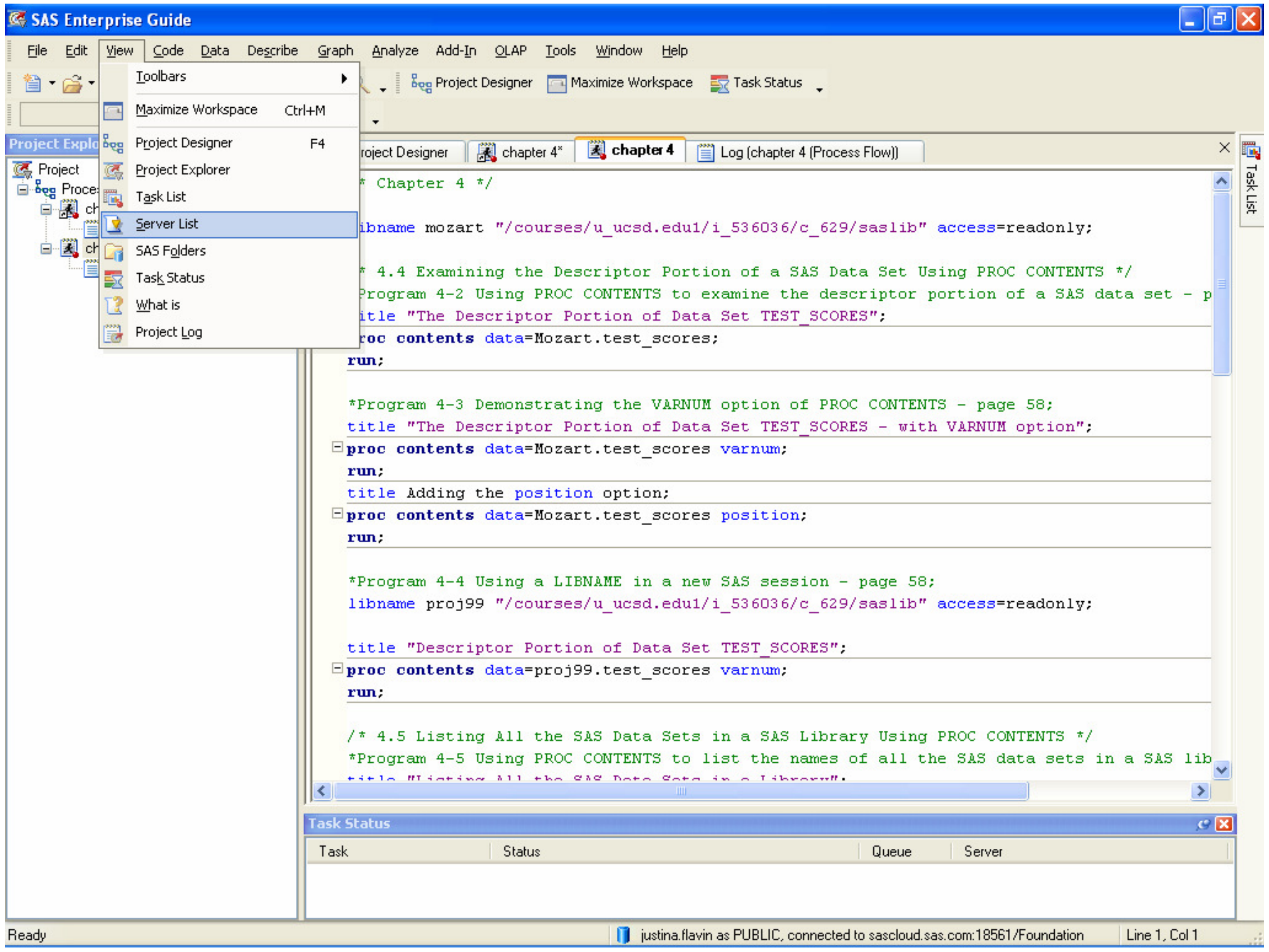
4.6 Viewing the Descriptor Portion of a SAS Data Set Using the SAS Explorer
4.8 Viewing the Data Portion of a SAS Data Set Using the SAS VIEWTABLE Window

Note: The screen shots shown in the book in these sections will be a bit different than what is show here in the slides.
The book illustrates the use of the Display Manager interface, while we are using the Enterprise Guide interface.

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.

To view the libraries and data sets available in your SAS session, you must first ensure that you have submitted a libname statement during your current SAS session. In this lecture, we have submitted two libname statements, for the librefs MOZART and PROJ99, both of which point to the same physical location.

On the menu bar, go to View -> Server List

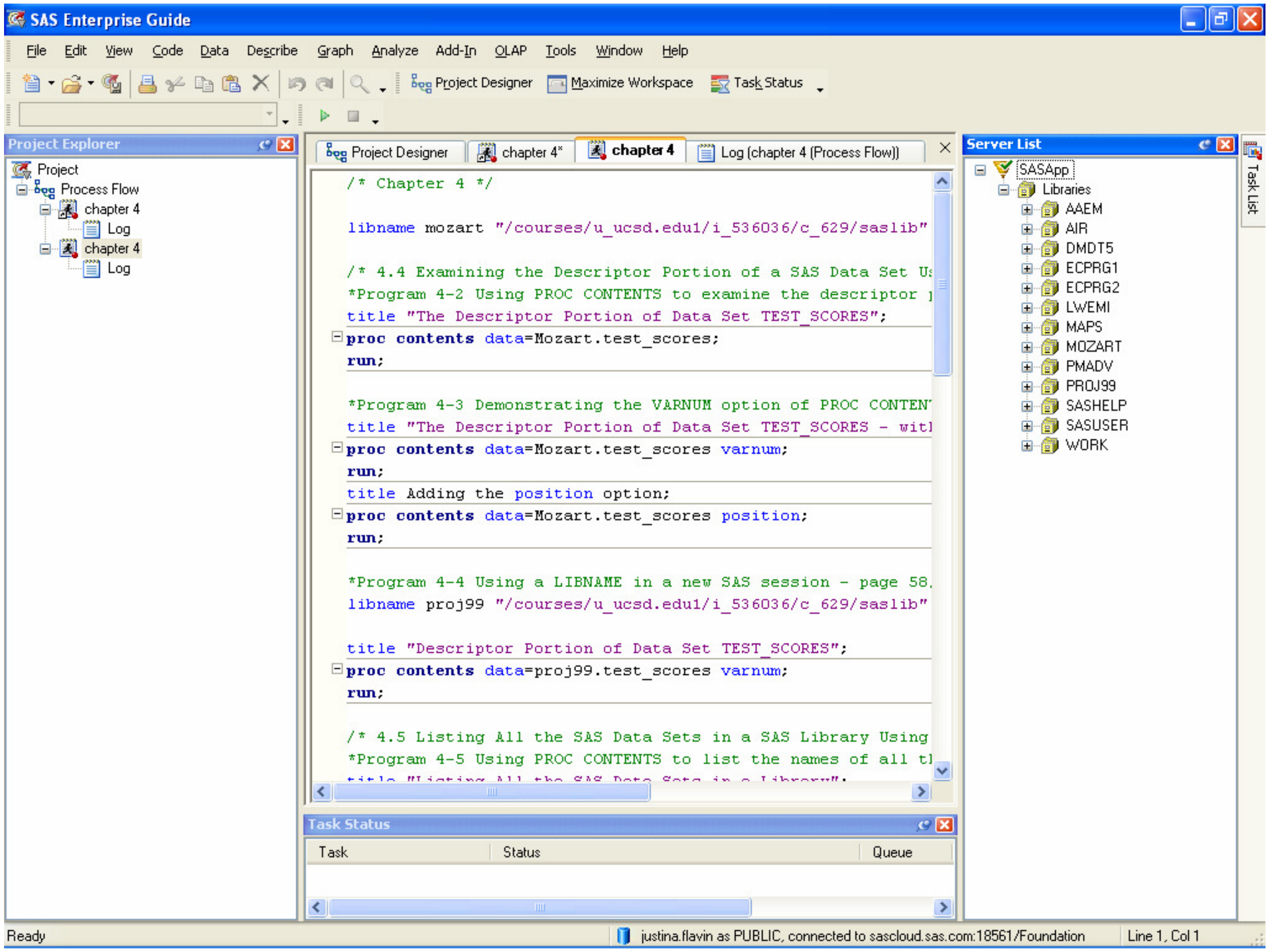


```
* Chapter 4 */

ibname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib" access=readonly;

* 4.4 Examining the Descriptor Portion of a SAS Data Set Using PROC CONTENTS */
Program 4-2 Using PROC CONTENTS to examine the descriptor portion of a SAS data set - p
itle "The Descriptor Portion of Data Set TEST_SCORES";
roc contents data=Mozart.test_scores;
run;

*Program 4-3 Demonstrating the VARNUM option of PROC CONTENTS - page 58;
title "The Descriptor Portion of Data Set TEST_SCORES - with VARNUM option";
proc contents data=Mozart.test_scores varnum;
run;
title Adding the position option;
proc contents data=Mozart.test_scores position;
run;

*Program 4-4 Using a LIBNAME in a new SAS session - page 58;
libname proj99 "/courses/u_ucsd.edu1/i_536036/c_629/saslib" access=readonly;

title "Descriptor Portion of Data Set TEST_SCORES";
proc contents data=proj99.test_scores varnum;
run;

/* 4.5 Listing All the SAS Data Sets in a SAS Library Using PROC CONTENTS */
*Program 4-5 Using PROC CONTENTS to list the names of all the SAS data sets in a SAS lib
title "Listing All the SAS Data Sets in a Library";
```

In the Server List box on the left side, click on the "+" to the left of SASApp and do the same for Libraries.

This display now shows you all of the libraries that are available to you in your SAS session.

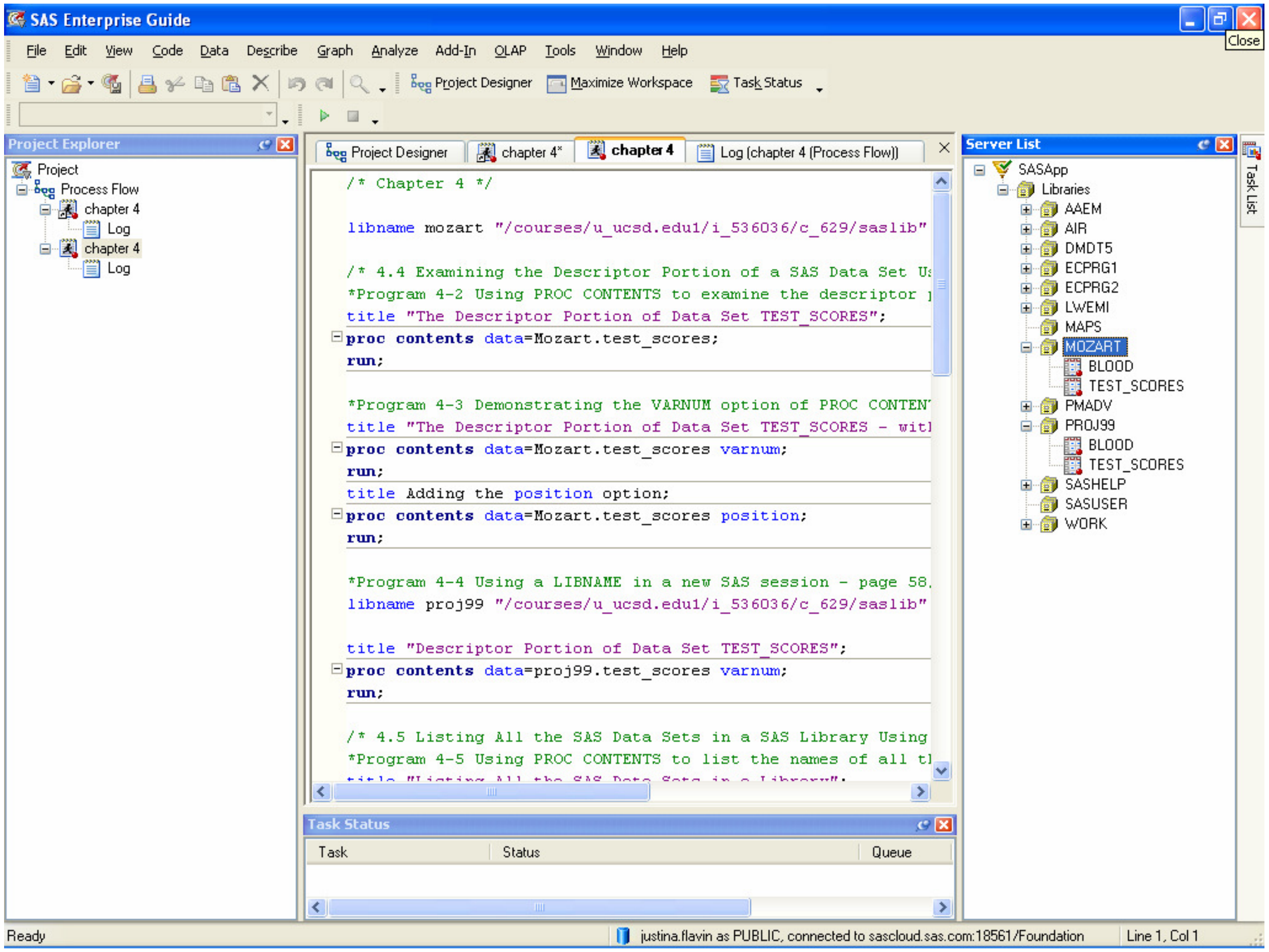
Other than MOZART and PROJ99 whose librefs we set up, the rest were predefined and automatically set up by the SAS system.



```
/* Chapter 4 */

libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

/* 4.4 Examining the Descriptor Portion of a SAS Data Set Us
*Program 4-2 Using PROC CONTENTS to examine the descriptor p
title "The Descriptor Portion of Data Set TEST_SCORES";
proc contents data=Mozart.test_scores;
run;

*Program 4-3 Demonstrating the VARNUM option of PROC CONTENT
title "The Descriptor Portion of Data Set TEST_SCORES - witl
proc contents data=Mozart.test_scores varnum;
run;
title Adding the position option;
proc contents data=Mozart.test_scores position;
run;

*Program 4-4 Using a LIBNAME in a new SAS session - page 58.
libname proj99 "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

title "Descriptor Portion of Data Set TEST_SCORES";
proc contents data=proj99.test_scores varnum;
run;

/* 4.5 Listing All the SAS Data Sets in a SAS Library Using
*Program 4-5 Using PROC CONTENTS to list the names of all tl
title "Listing All the SAS Data Sets in a Library";
```

Click on both MOZART and PROJ99 to see the contents.
Not surprisingly, these will contain the same data sets.
At the time of creating these slides, there were two data sets in the directory, BLOOD and TEST_SCORES.
You can click on any of the other libraries to see what data sets are available in those.

.



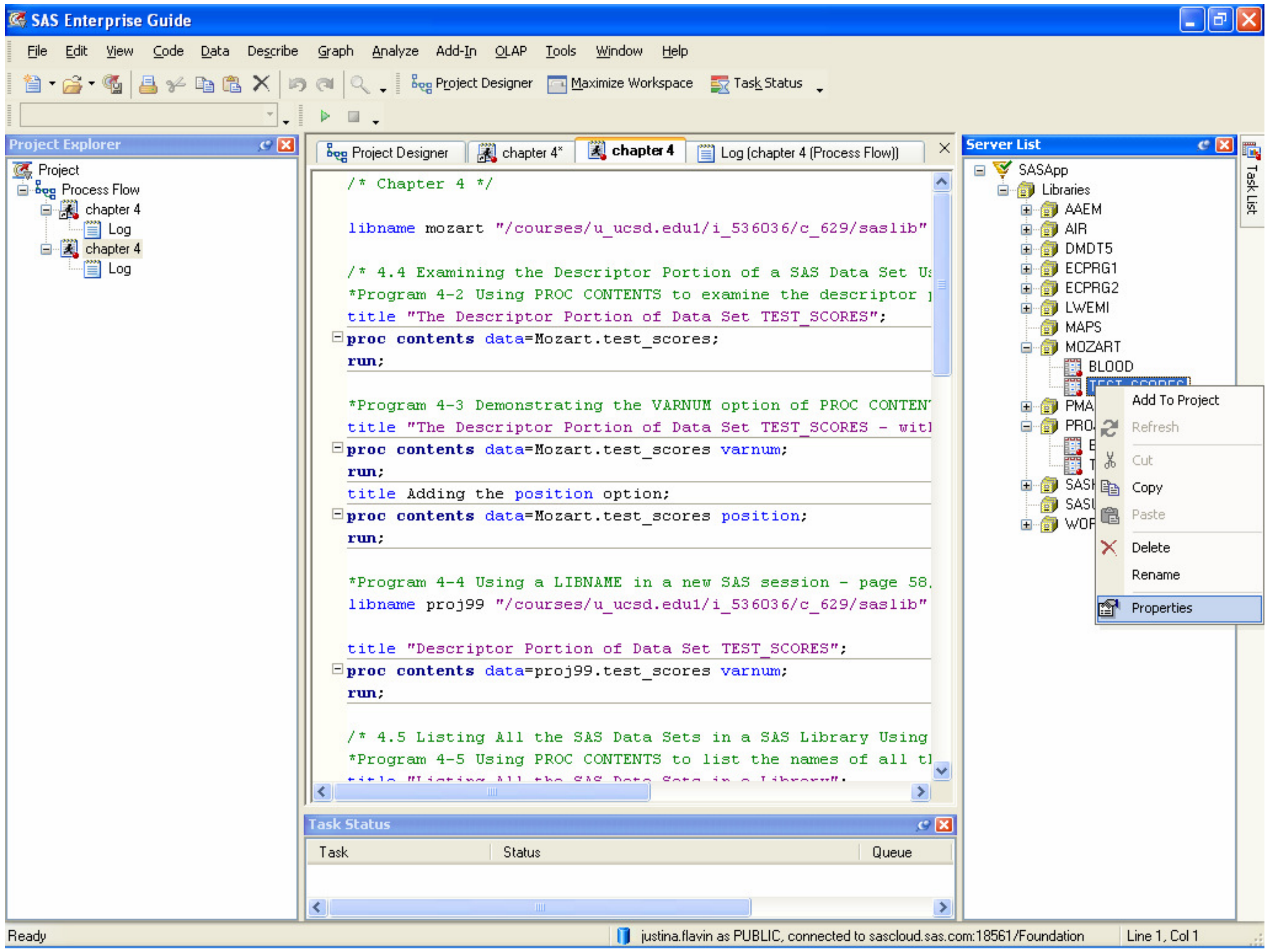
```
/* Chapter 4 */

libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

/* 4.4 Examining the Descriptor Portion of a SAS Data Set Us
*Program 4-2 Using PROC CONTENTS to examine the descriptor
title "The Descriptor Portion of Data Set TEST_SCORES";
proc contents data=Mozart.test_scores;
run;

*Program 4-3 Demonstrating the VARNUM option of PROC CONTEN
title "The Descriptor Portion of Data Set TEST_SCORES - wit
proc contents data=Mozart.test_scores varnum;
run;
title Adding the position option;
proc contents data=Mozart.test_scores position;
run;

*Program 4-4 Using a LIBNAME in a new SAS session - page 58,
libname proj99 "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

title "Descriptor Portion of Data Set TEST_SCORES";
proc contents data=proj99.test_scores varnum;
run;

/* 4.5 Listing All the SAS Data Sets in a SAS Library Using
*Program 4-5 Using PROC CONTENTS to list the names of all tl
title "Listing All the SAS Data Sets in a Library";
```

Ready   justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation   Line 1, Col 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.

To view summary information about a data set, right click on the data set to bring up the floating menu and select Properties

.

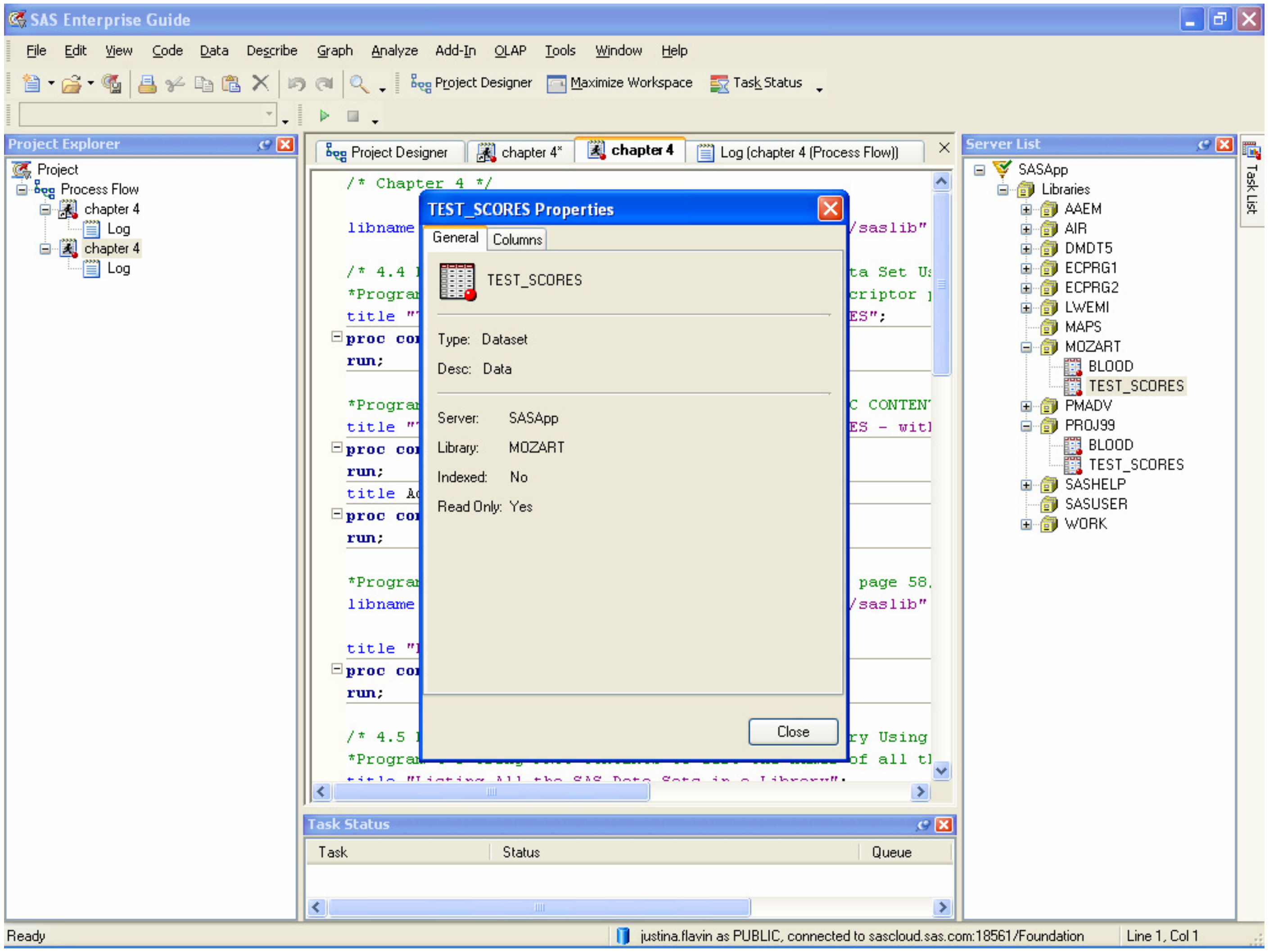


```
/* Chapter 4 */

libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

/* 4.4 Examining the Descriptor Portion of a SAS Data Set Us
*Program 4-2 Using PROC CONTENTS to examine the descriptor p
title "The Descriptor Portion of Data Set TEST_SCORES";
proc contents data=Mozart.test_scores;
run;

*Program 4-3 Demonstrating the VARNUM option of PROC CONTENT
title "The Descriptor Portion of Data Set TEST_SCORES - wit
proc contents data=Mozart.test_scores varnum;
run;
title Adding the position option;
proc contents data=Mozart.test_scores position;
run;

*Program 4-4 Using a LIBNAME in a new SAS session - page 58.
libname proj99 "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

title "Descriptor Portion of Data Set TEST_SCORES";
proc contents data=proj99.test_scores varnum;
run;

/* 4.5 Listing All the SAS Data Sets in a SAS Library Using
*Program 4-5 Using PROC CONTENTS to list the names of all t
```

Ready   justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation   Line 1, Col 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.

This brings up a floating menu which contains General information about the data set.
The Columns tab contains information about the variables.



TEST_SCORES

Type: Dataset

Desc: Data

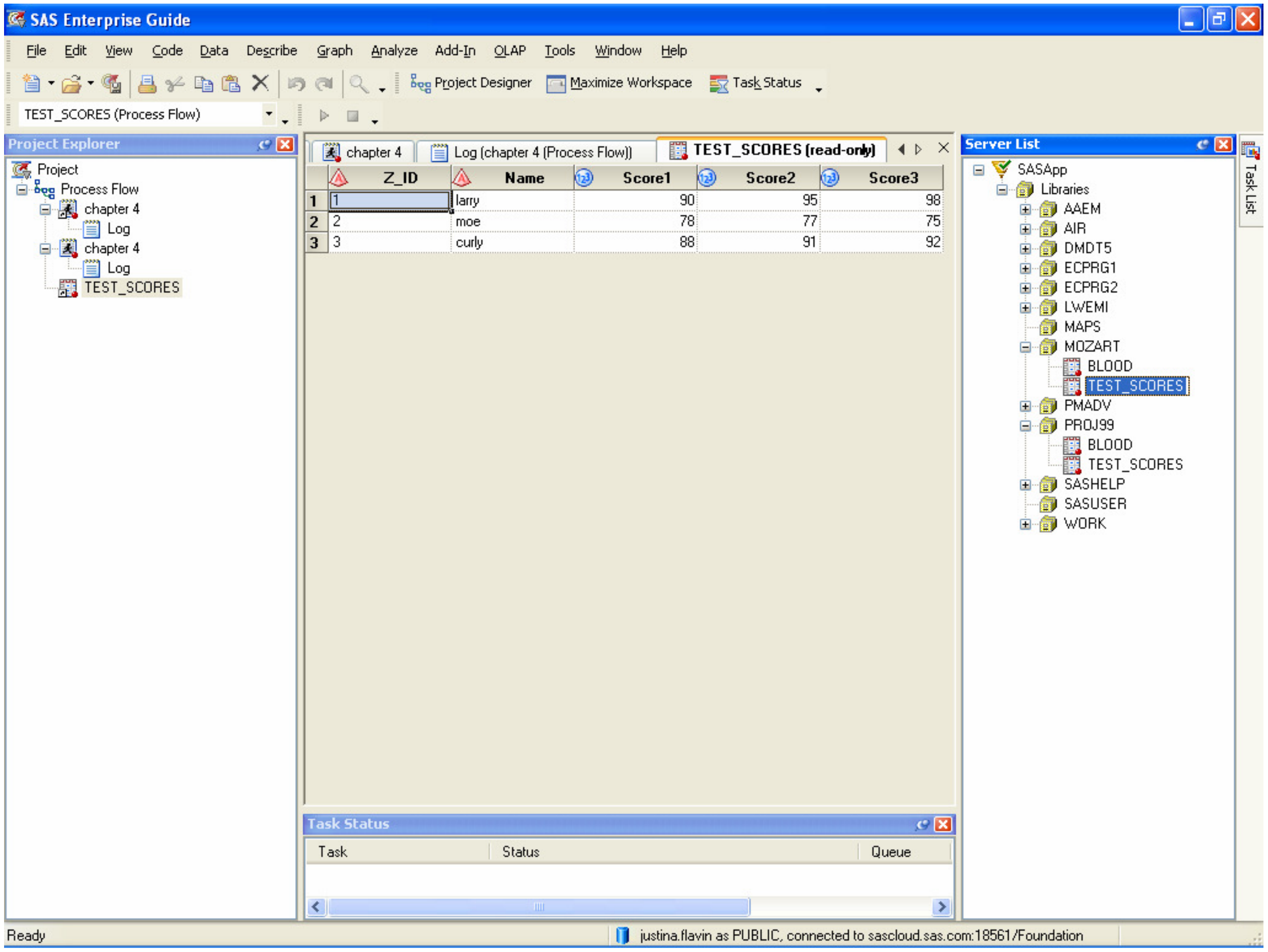
Server: SASApp

Library: MOZART

Indexed: No

Read Only: Yes

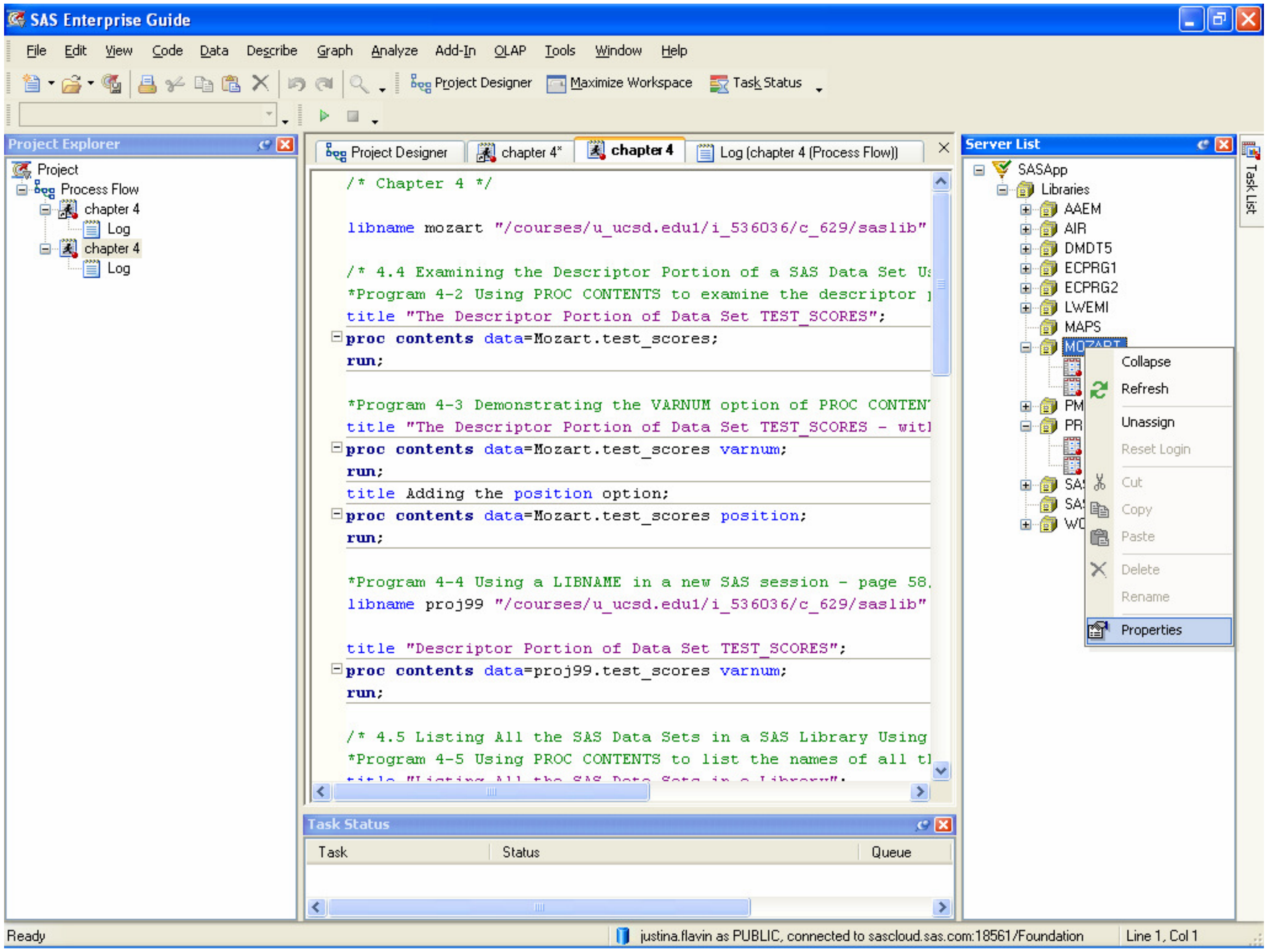
Double clicking on the data set name opens it up for viewing.



	Z_ID	Name	Score1	Score2	Score3
1	1	larry	90	95	98
2	2	moe	78	77	75
3	3	curly	88	91	92

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.

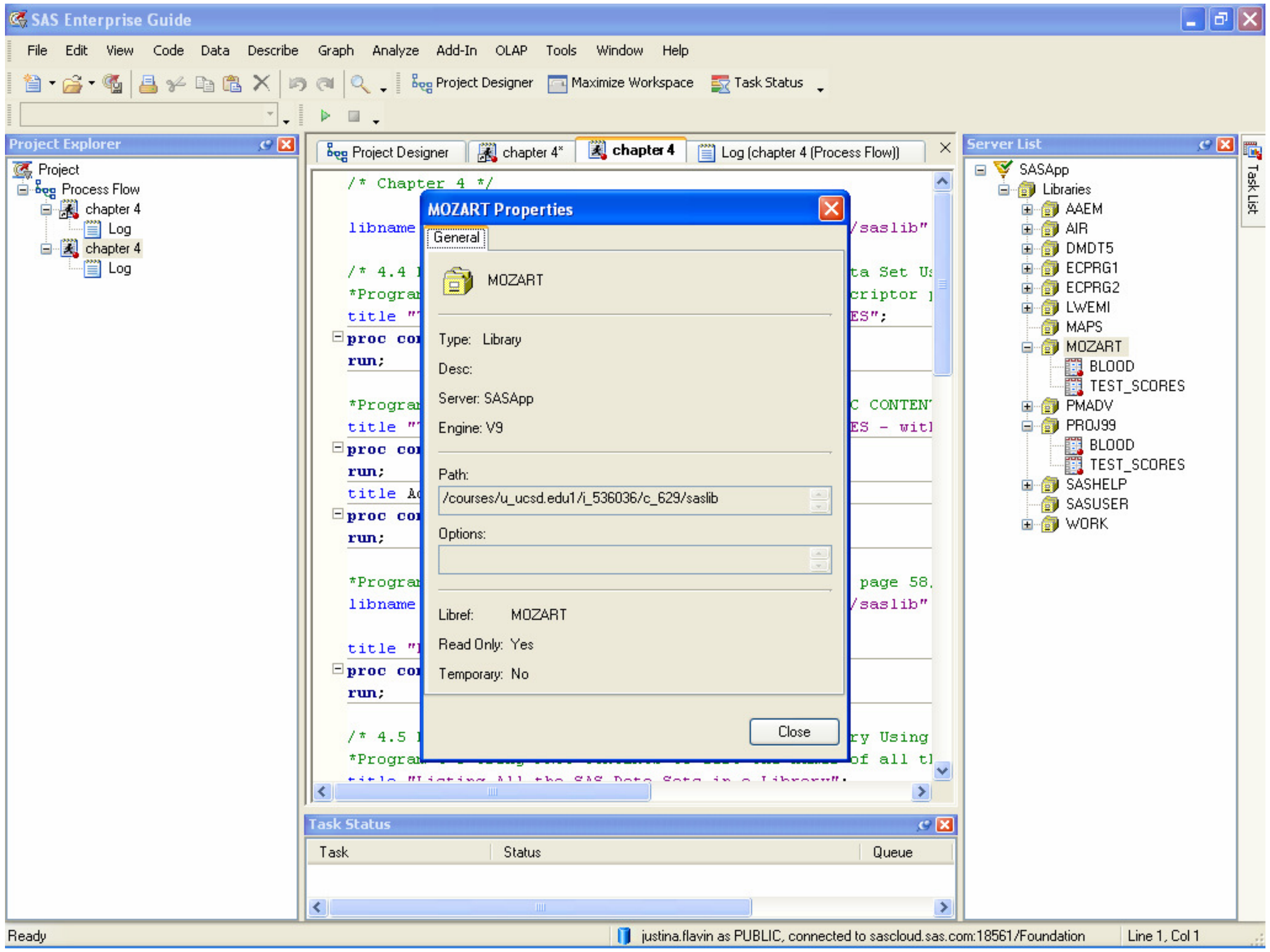
To view summary information about a library, right click on the library name to bring up the floating menu and select Properties



```
/* Chapter 4 */

libname mozart "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

/* 4.4 Examining the Descriptor Portion of a SAS Data Set Us
*Program 4-2 Using PROC CONTENTS to examine the descriptor p
title "The Descriptor Portion of Data Set TEST_SCORES";
proc contents data=Mozart.test_scores;
run;

*Program 4-3 Demonstrating the VARNUM option of PROC CONTENT
title "The Descriptor Portion of Data Set TEST_SCORES - with
proc contents data=Mozart.test_scores varnum;
run;
title Adding the position option;
proc contents data=Mozart.test_scores position;
run;

*Program 4-4 Using a LIBNAME in a new SAS session - page 58.
libname proj99 "/courses/u_ucsd.edu1/i_536036/c_629/saslib"

title "Descriptor Portion of Data Set TEST_SCORES";
proc contents data=proj99.test_scores varnum;
run;

/* 4.5 Listing All the SAS Data Sets in a SAS Library Using
*Program 4-5 Using PROC CONTENTS to list the names of all th
```

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 brings up a floating menu which contains General information about the library. Notice that the Path box defines the physical location of the library.



MOZART

Type: Library

Desc:

Server: SASApp

Engine: V9

Path:
/courses/u_ucsd.edu1/i_536036/c_629/saslib

Options:

Libref: MOZART

Read Only: Yes

Temporary: No

You can browse through any of these libraries and data sets.

The WORK library will contain all of the temporary SAS data sets which you have created in your current SAS session. Again, these data sets will disappear when you exit your SAS session, and it would be necessary to rerun code to recreate them in a new SAS session.

The data sets in the other libraries are permanent.

For the data sets in the directory that were referenced by MOZART and PROJ99, upon issuing a libref statement in a new SAS session, these will once again be available to you.

The librefs for the other libraries are automatically set up for you (by the SAS server configuration file) when the SAS session is opened, so you will never have to issue a libref statement for these.

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.

4.9 Using a SAS Data Set as Input to a DATA Step

So far we have seen how to create SAS data sets by reading in data from a file or from inline data.
Now we will see how to include data that is already in a SAS data set into a program using the SET statement.

The SET statement is used with SAS data sets. It replaces the input and infile statement that were used previously with raw data.

Syntax:

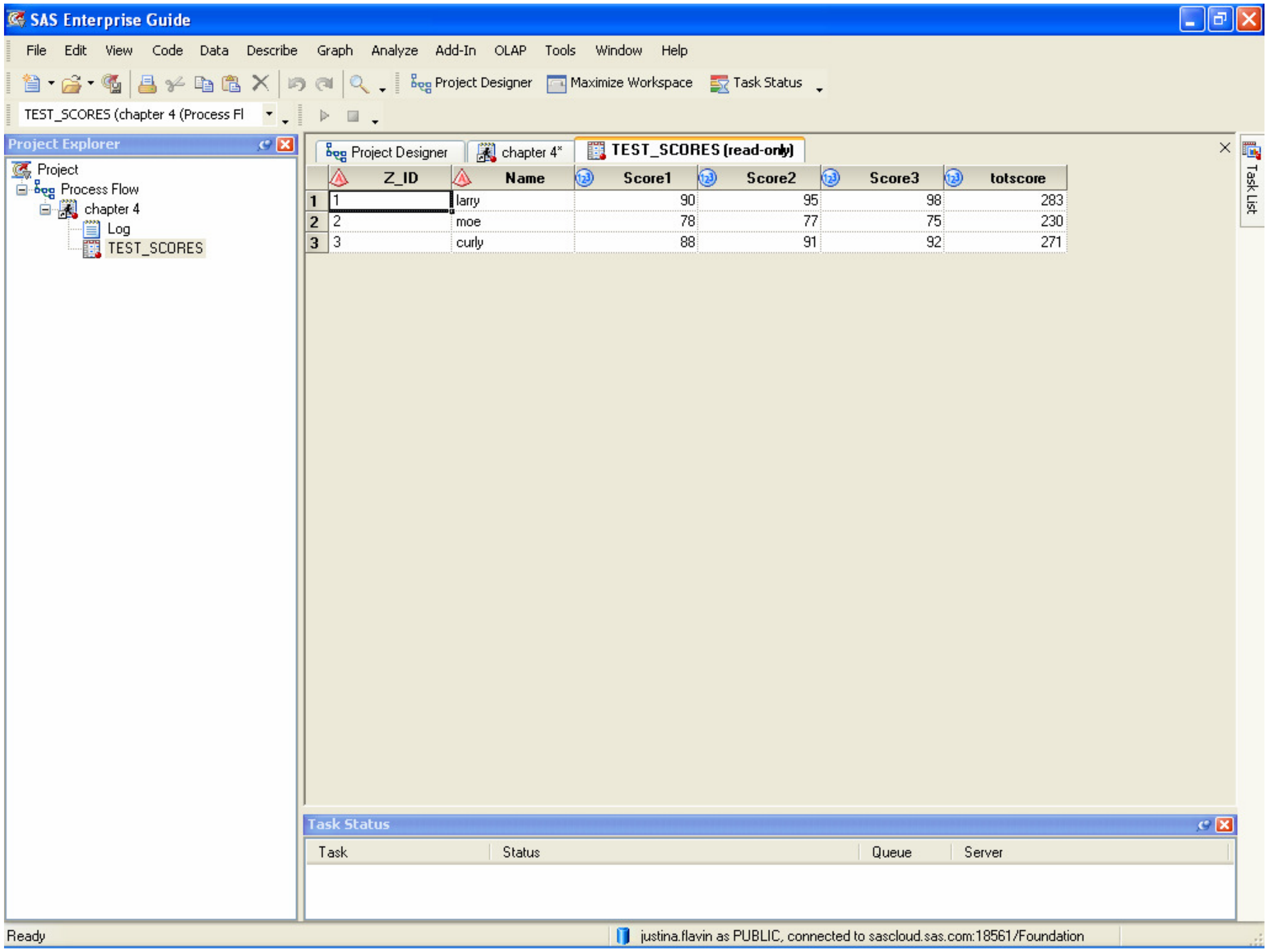
```
SET datasetname; (temporary data sets)
SET libref.datasetname; (permanent data sets)

/* 4.9 Using a SAS Data Set as Input to a DATA Step */
*Using observations from a SAS data set as input to a new SAS data set - page 66;
* this is a different example than shown in book ;
data test_scores;
  set mozart.test_scores;
  totscore=score1 + score2 + score3;
run;
```

In this example, we create a temporary data set called TEST_SCORES by setting in the permanent data set TEST_SCORES.
We then create a new variable called totscore by adding the values of score1 thru score3.
Having two data sets of the same name does not cause a problem since they are located in two different physical locations.
TEST_SCORES resides in the WORK library while MOZART.TEST_SCORES resides in the MOZART library.

Note: The book example illustrates the use of the mean function. We will skip over that for now and cover it in a later chapter.

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.

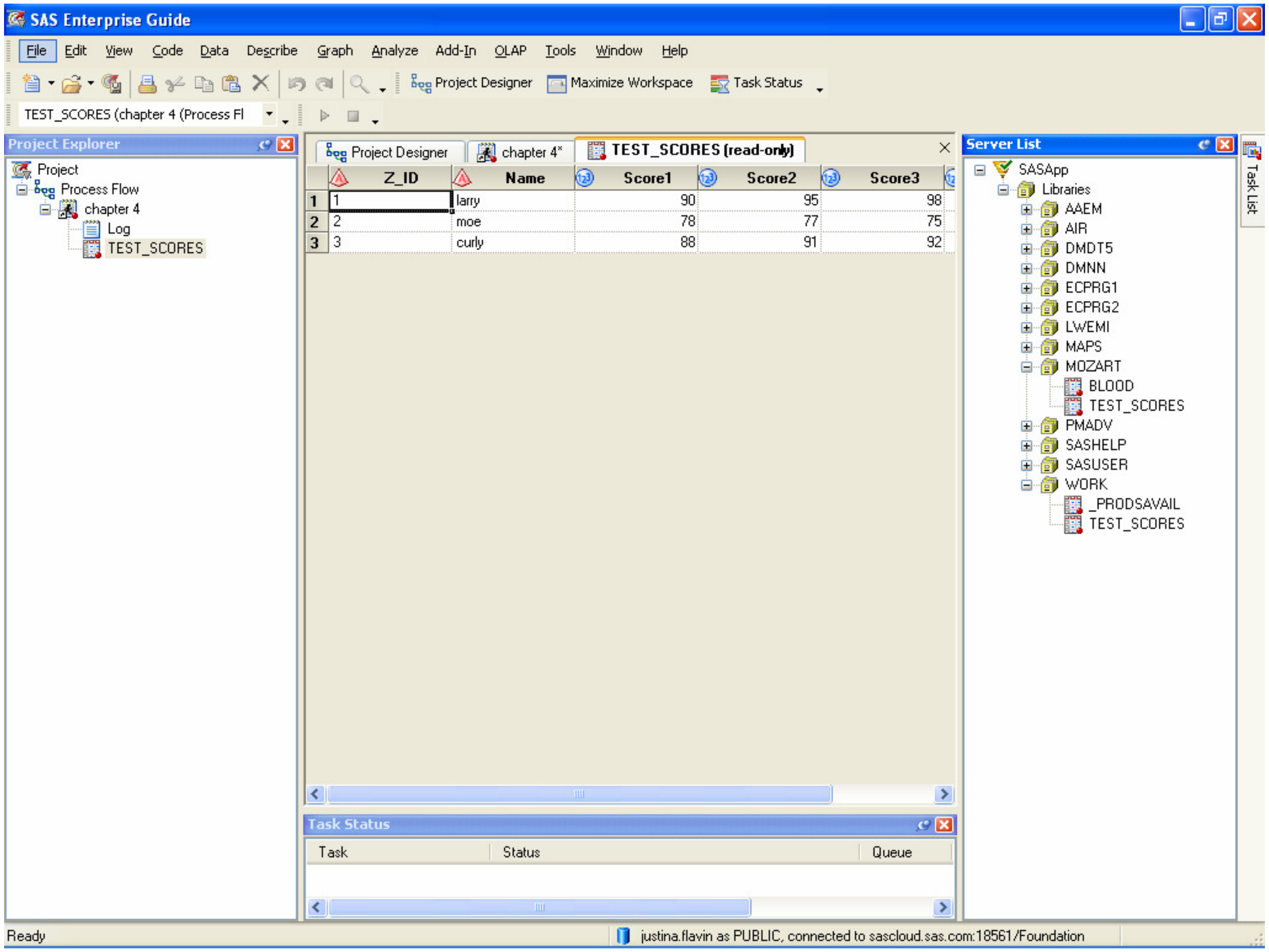
Here is our new TEST_SCORES data set.


SAS Enterprise Guide
File Edit View Code Data Describe Graph Analyze Add-In OLAP Tools Window Help
Project Designer Maximize Workspace Task Status
TEST_SCORES (chapter 4 (Process Fl
Project Explorer
Project
Process Flow
chapter 4
Log
TEST_SCORES
Project Designer
chapter 4*
TEST_SCORES (read-only)
Task List
Task Status
Task Status Queue Server
Ready
justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation

	Z_ID	Name	Score1	Score2	Score3	totscore
1	1	larry	90	95	98	283
2	2	moe	78	77	75	230
3	3	curly	88	91	92	271

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.

Looking in our Server List, we can also see the two TEST_SCORES data sets listed in both the WORK and MOZART directories.



	Z_ID	Name	Score1	Score2	Score3
1	1	larry	90	95	98
2	2	moe	78	77	75
3	3	curly	88	91	92

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.

To access permanent SAS data sets in procedures, you just need to specify the two part name.

Here are some examples using PROC FREQ and PROC MEANS:

```
proc freq data=mozart.blood;
  tables gender bloodtype;
run;

proc means data=proj99.blood;
  var chol wbc rbc;
run;
```

Using PROC FREQ and PROC MEANS is another way to view the data in your data sets and can be helpful for getting a quick look at the variable values.

Section 4.10 DATA _NULL_: A Data Set That Isn't

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.