Moving SAS programs to SAS Viya for experienced SAS programmers

Workshop

Exercises

Table of Contents

[1. Investigating your SAS Viya environment 3](#_Toc87882524)

[2. Running PROC MEANS in Compute and CAS 4](#_Toc87882525)

[3. Running DATA step in Compute or CAS 6](#_Toc87882526)

[4. Running single threaded vs. multi-threaded 7](#_Toc87882527)

[5. Upload data and programs to Viya 8](#_Toc87882528)

[6. Refactor SAS program for CAS 10](#_Toc87882529)

[Refactor section /\* 1 \*/ 10](#_Toc87882530)

[Refactor section /\* 2 \*/ 11](#_Toc87882531)

[Refactor section /\* 3 \*/ 12](#_Toc87882532)

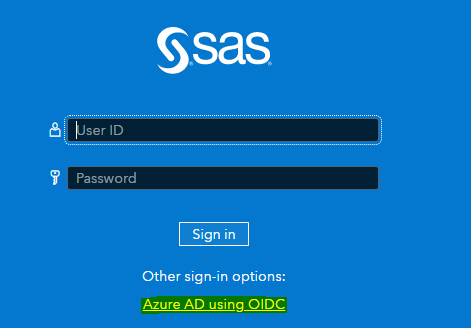
[Refactor section /\* 4 \*/ and /\* 5 \*/ 12](#_Toc87882533)

Author: Cecily Hoffritz, Data Engineering Advisory, SAS Nordics (cecily.hoffritz@sas.com)

# Investigating your SAS Viya environment

In this exercise, you gain information about the SAS Viya environment that you will be working in.

1. Access SAS Studio on the SAS internal SSEMONTHLY server: <https://ssemonthly.demo.sas.com/SASStudio>.
2. Sign in by clickling **Azure AD using OIDC**.



1. Click New 🡪 SAS program and you are ready to write code.
2. Copy the code below to SAS Studio and run the program.

/\* on SSEMONTHLY Viya server, a CAS session is already started

and global CAS libraries are already allocated.

\*/

\*cas;

\*caslib \_all\_ assign;

%put &sysvlong4.;

cas casauto listabout;

proc cas;

about;

run;

proc options group=cas;

run;

1. Read the information in the Log and Results.
   1. What is the Viya version, number of nodes, name of the active session, [CASDATALIMIT](https://go.documentation.sas.com/doc/en/pgmsascdc/v_018/lesysoptsref/p0wxc12m9hqke7n1bfcfnznsbpgo.htm) and [CASNCHARMULTIPLIER](https://go.documentation.sas.com/doc/en/pgmsascdc/v_018/lesysoptsref/n1rsugylwt5y6mn1htt45yr35mfa.htm) value?
   2. Are there any assigned CASLIBs that are not valid for use as a libref, and what does this mean?

# Running PROC MEANS in Compute and CAS

In this exercise you test the performance of PROC MEANS code run in COMPUTE and in CAS.

1. Copy the code below to SAS Studio and run it.

options msglevel=i;

cas;

libname casuser cas caslib=casuser;

\*caslib \_all\_ assign;

/\* Create test data in WORK \*/

data test;

do i = 1 to 1000000;

j=1;

output;

end;

run;

/\* Create test data in CASUSER \*/

data casuser.test;

set test;

run;

/\* Run in COMPUTE \*/

proc means data=test sum;

var j;

run;

/\* Run in CAS \*/

proc means data=casuser.test sum;

var j;

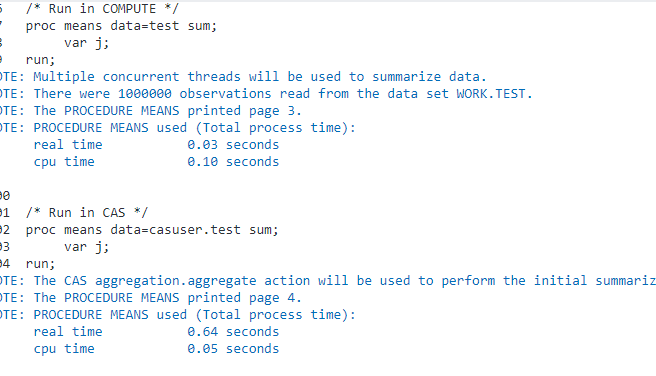
run;

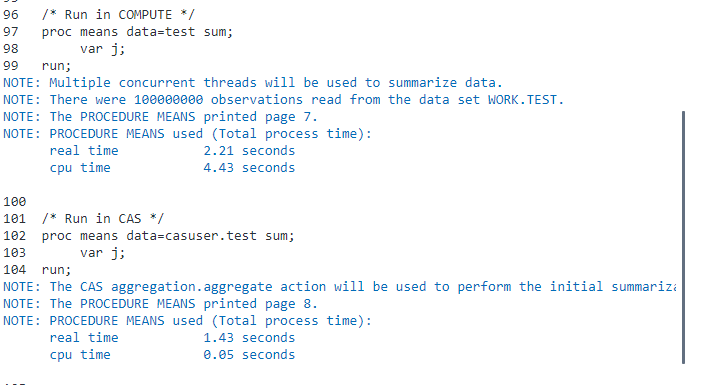
1. Open the log and make a note of the real time and cpu time for both PROC MEANS steps.
2. In the line containing do i = 1 to 1000000; modify the value to create test data with 100 million records.

Run the program again.

1. Compare your results with the ones on the next page. What is your interpretation of the results and how do they compare?

Run on SSEMONTHLY 12 November 2021





# Running DATA step in Compute or CAS

In this exercise you will test whether code runs in COMPUTE or CAS.

1. Copy the program below to SAS Studio and run it.

/\* TEST ONE \*/

/\* Get more info in the log \*/

options msglevel=i;

/\* Where does this run and what does the log say about threading? \*/

data casuser.test;

do i = 1 to 100000;

j=1;

output;

end;

run;

/\* Where does this run and what does the log say? \*/

data work.test;

set casuser.test;

run;

1. Answer the 2 questions embedded in the comments in the SAS program.
2. Copy the program below to SAS Studio and run it.

/\* TEST TWO \*/

/\* Create bigger test data \*/

data casuser.test;

do i = 1 to 10000000;

j=1;

output;

end;

run;

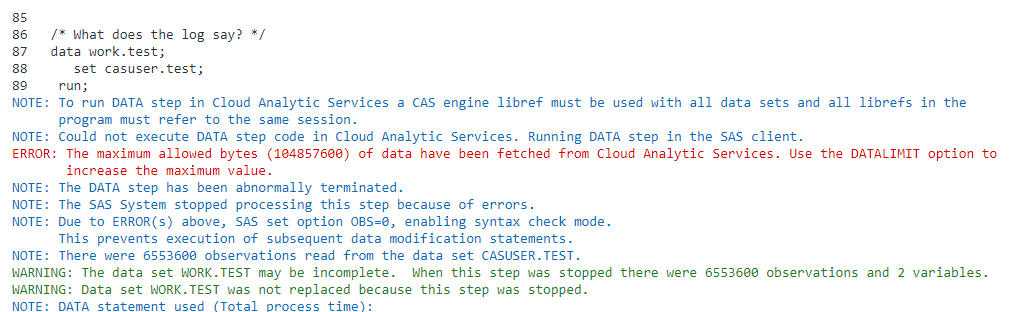
/\* What does the log say? \*/

data work.test;

set casuser.test;

run;

1. Answer the question embedded in the last comment in the SAS program. Does your log show the same as below and why? Remediate the problem and run the program again. Note that (DATALIMIT=) is a data set option.



# Running single threaded vs. multi-threaded

In this exercise you will gain an understanding of threads when running data steps in CAS.

1. Copy the program below to SAS Studio and run it.

/\* Create test data in WORK \*/

data testthreads;

do i = 1 to 100000;

j=1;

output;

end;

run;

/\* Create test data in CAS \*/

data casuser.testthreads;

set testthreads;

run;

/\* Runs in COMPUTE. How many rows? \*/

data testthreadssas(drop=i j);

retain sum\_i sum\_j;

set testthreads end=eof;

sum\_i + i;

sum\_j + j;

put \_threadid\_= \_nthreads\_=;

if eof;

run;

/\* Runs in CAS. How many rows? \*/

data casuser.testthreadscas(drop=i j);

retain sum\_i sum\_j;

set casuser.testthreads end=eof;

sum\_i + i;

sum\_j + j;

put \_threadid\_= \_nthreads\_=;

if eof;

run;

How many rows does the table testthreadssas have compared to casuser.testthreadscas?

1. Add the option /single=yes to the DATA statement of the last step that runs in CAS.

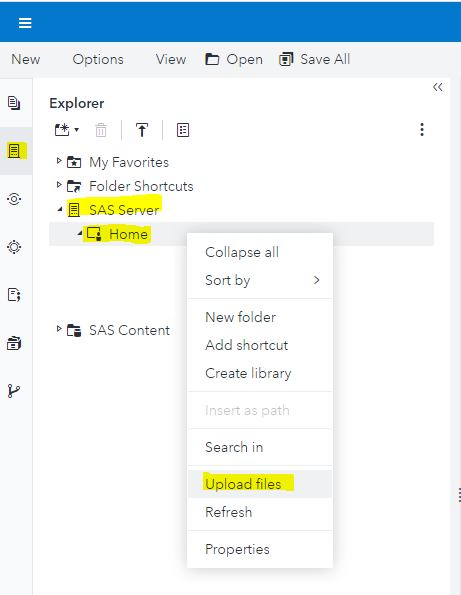
Do you get the same results as running in Compute?

# Upload data and programs to Viya

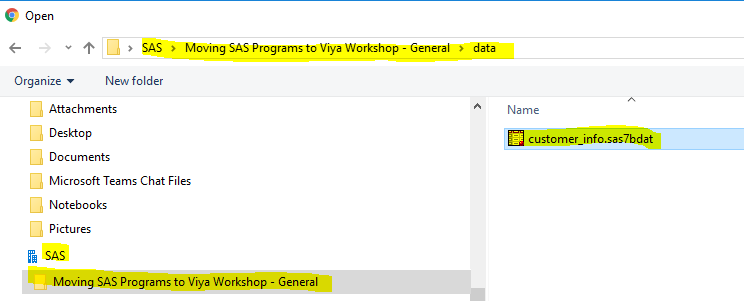
In this exercise you will be uploading a SAS program and a SAS table to your **Home** folder on Viya.

Upload SAS table to SAS Viya

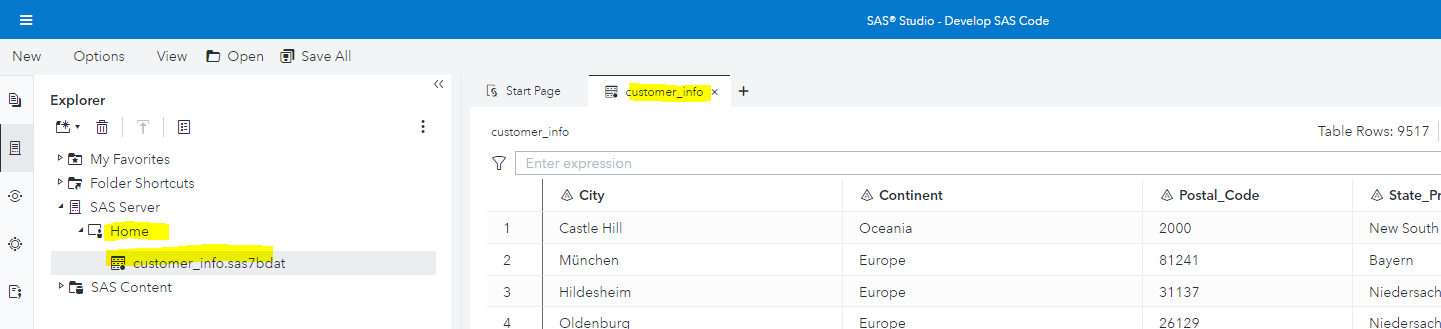
1. Sign into SAS Studio on SSEMONTHLY, if not already signed in.
2. Click the **Explorer** icon in the left pane. Right click the **Home** folder below **SAS Server** and select **Upload files**.



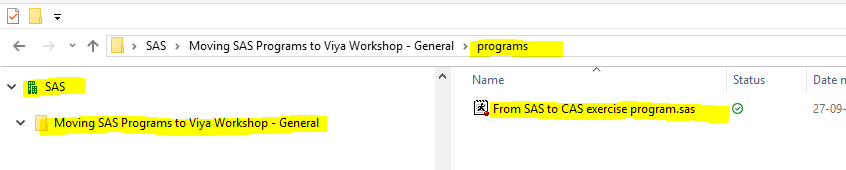
1. Upload the SAS table **customer\_info** from your synced **data** folder in Windows.



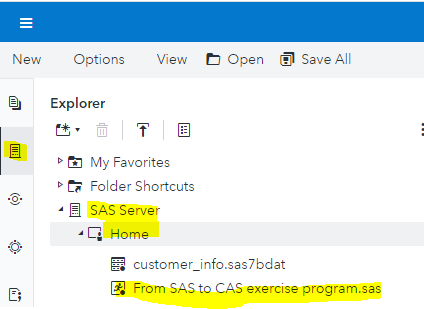
1. Verify that you can open **customer\_info** after upload.



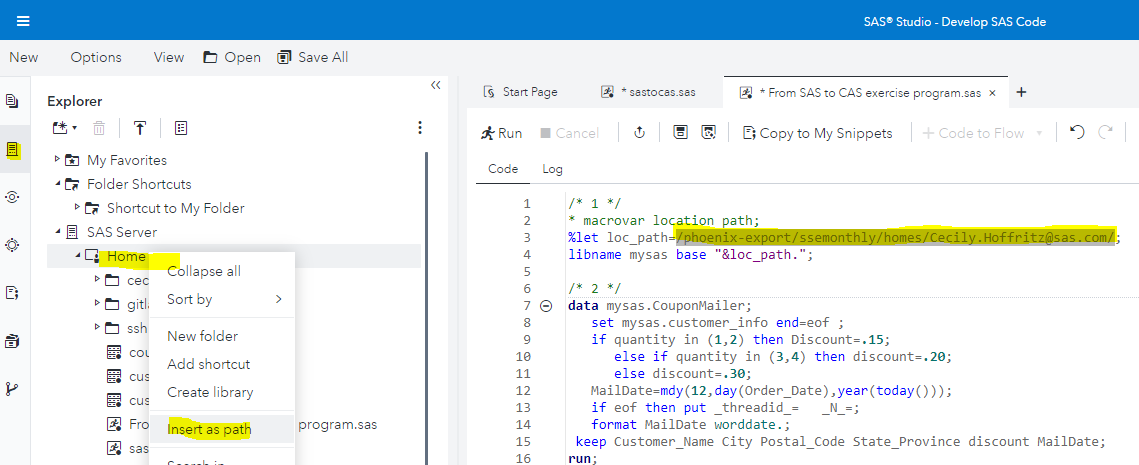
1. Upload the SAS program **From SAS to CAS exercise program.sas.** from your synced **program** folder in Windows



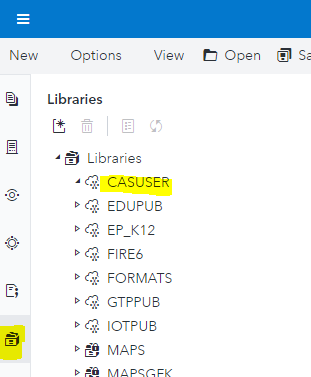
to your HOME location.



1. Open the uploaded SAS program and familiarize yourself with its contents.
2. Modify the path so that it points to HOME. You can do this where you highlight the path in the %LET statement, right click **Home** and select **Insert as path**.



1. Run the saved program to validate that it runs on Compute without errors.
2. Save the program as **sastocas.sas** in **Home**.
3. Notice that SSEMONTHLY Viya server provides an already allocated CAS session and CAS libraries, including CASUSER.

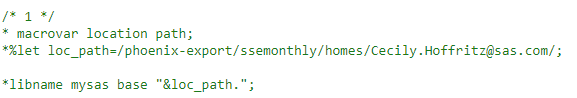


# Refactor SAS program for CAS

In this exercise you will be refactoring **sastocas.sas**. Focus today is on functional refactoring of code and not refactoring code for better performance.

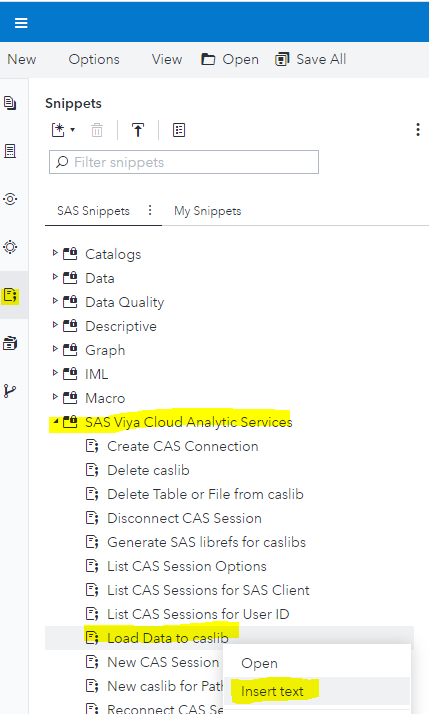
## Refactor section /\* 1 \*/

1. Comment out all code in section /\* 1 \*/



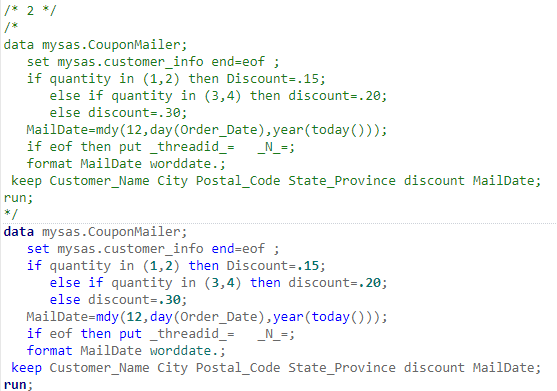
1. Add a PROC CASUTIL load step and load MYSAS.CUSTOMER\_INFO to CASUSER.

Tip: If you can’t remember PROC CASUTIL load code, use the code snippet below that contains different syntax examples.

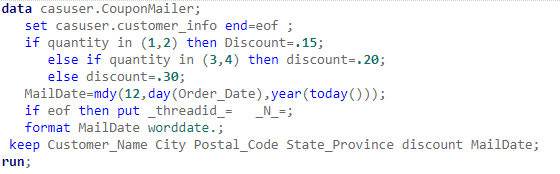


## Refactor section /\* 2 \*/

1. Make a copy of section /\* 2 \*/



1. Replace the input table to the one residing in CASUSER and save the output table to CASUSER.



1. Run the step.
   1. Did you get this message in the log?

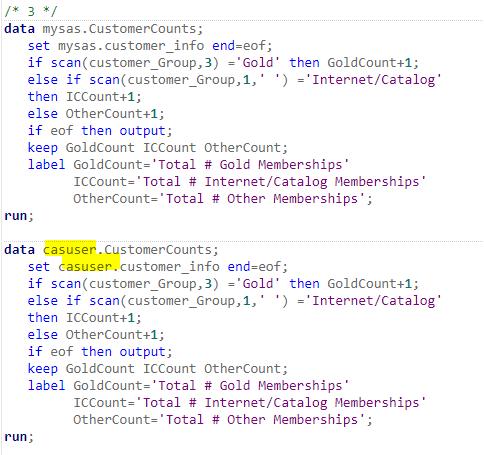


* 1. How many used threads does the log reveal, and what could this be an indication of?

1. Refactor the program so that it runs in CAS. For now, don’t worry about a potential degrade in performance with the solution you suggest, since we are currently focusing on functional conversion.

## Refactor section /\* 3 \*/

1. Make a copy of section /\* 3 \*/ and replace libraries in the copy with CASUSER.



1. Run both steps.
2. Investigate the output of each. Are they the same?
3. Refactor the code so that when it runs in CAS it produces the same result as the one you get when you run it in Compute.

## Refactor section /\* 4 \*/ and /\* 5 \*/

1. Refactor section /\* 4 \*/ and /\* 5 \*/ on your own (or with a colleague).