Ask the Expert Webinar: 10 Ways to Make Your SAS® Code Run More Efficiently

Shannon J. Moore

Sr. Technical Advisor

Shannon Moore
Sr. Technical Advisor, SAS

As a charter member of the Customer Success organization, Moore helped create the strategy and practices of this post-sales group. The team’s goal is to help with the adoption and modernization of customers' SAS solutions. Moore frequently speaks at users groups events and customer sites, either in person or via the web, on a wide range of topics including business analytics, system architecture, enhancing graphics and report output, and software updates.

Full disclosure -- closer to 25!

Processing Environment

Software

- SAS Release 9.4M7 – Enterprise Guide, SAS Studio, Display Manager System
- Windows 10 Enterprise Edition – 64 bit

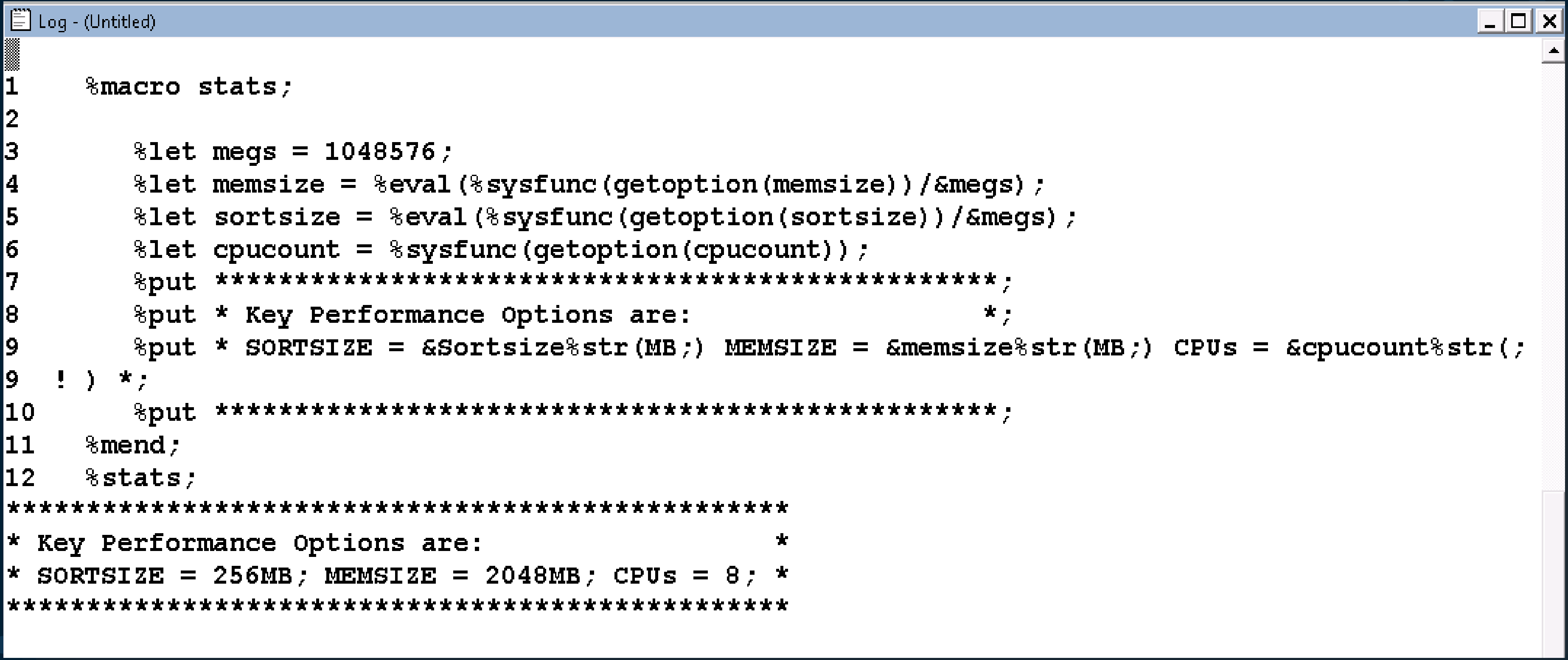
Hardware – Lenovo T470

- Intel i7-7600U @ 2.8GHz
- 2 cores – 4 logical processors
- 16 GB Physical Memory
- 2.3 GB Virtual Memory (Page File Size)

SAS Programming Best Practices

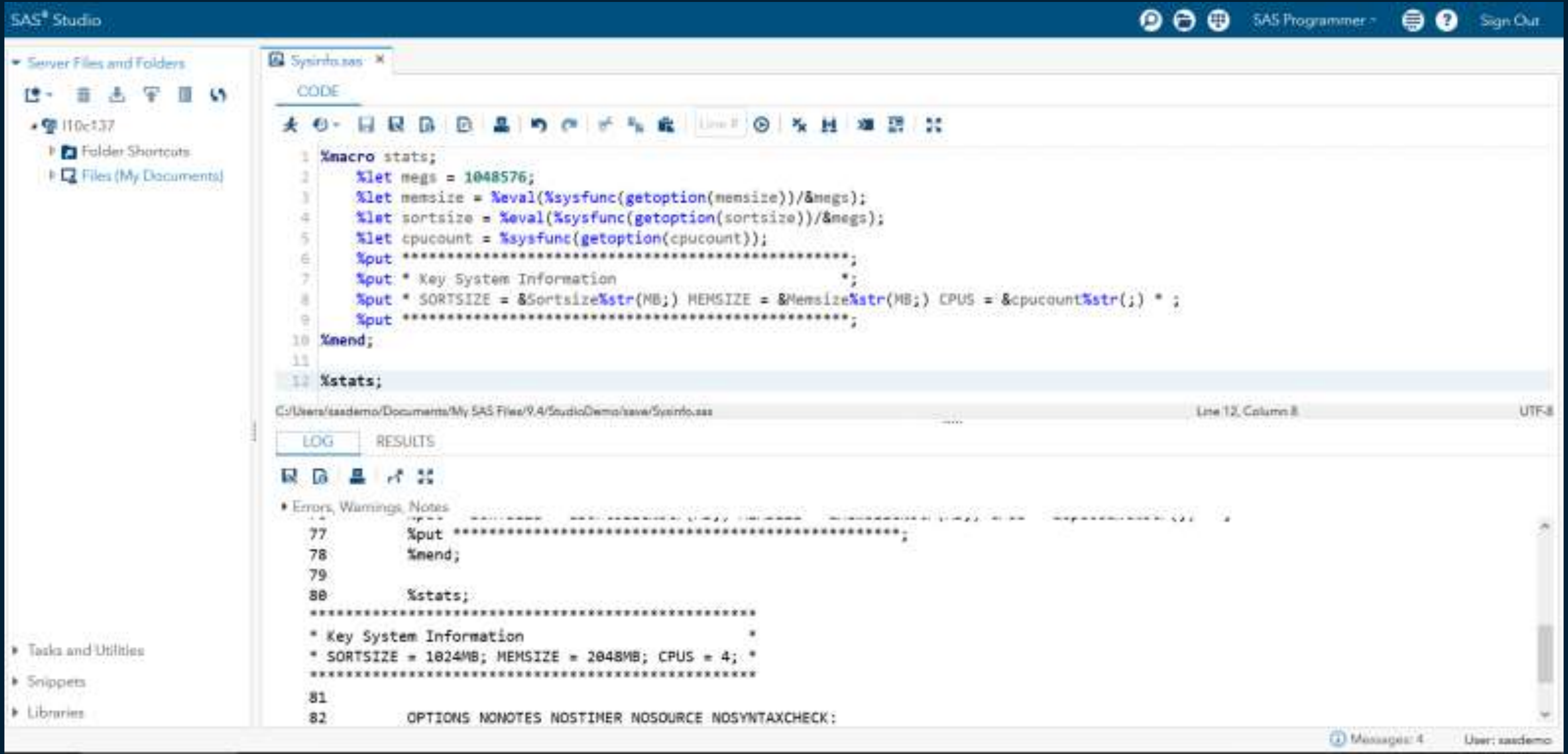
- Create Readable Code
- Basic Coding Recommendations

SAS Default Settings



```
1     %macro stats;
2
3        %let megs = 1048576;
4        %let memsize = %eval(%sysfunc(getoption(memsize))/&megs);
5        %let sortsize = %eval(%sysfunc(getoption(sortsize))/&megs);
6        %let cpucount = %sysfunc(getoption(cpucount));
7        %put *************************************************;
8        %put * Key Performance Options are:                 *;
9        %put * SORTSIZE = &Sortsize%str(MB;) MEMSIZE = &memsize%str(MB;) CPUs = &cpucount%str(;
9  ! ) *;
10       %put *************************************************;
11    %mend;
12    %stats;
*************************************************
* Key Performance Options are:                  *
* SORTSIZE = 256MB; MEMSIZE = 2048MB; CPUs = 8; *
*************************************************
```

My computer



```
%macro stats;
    %let megs = 1048576;
    %let memsize = %eval(%sysfunc(getoption(memsize))/&megs);
    %let sortsize = %eval(%sysfunc(getoption(sortsize))/&megs);
    %let cpucount = %sysfunc(getoption(cpucount));
    %put **************************************************;
    %put * Key System Information                        *;
    %put * SORTSIZE = &Sortsize%str(MB;) MEMSIZE = &Memsize%str(MB;) CPUS = &cpucount%str(;) * ;
    %put **************************************************;
%mend;

%stats;
```

C:/Users/sasdemo/Documents/My SAS Files/9.4/StudioDemo/save/Sysinfo.sas

```
77          %put **************************************************;
78          %mend;
79
80          %stats;
**************************************************
* Key System Information                         *
* SORTSIZE = 1024MB; MEMSIZE = 2048MB; CPUS = 4; *
**************************************************

81
82          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
```

Processing Environment

STIMER - Writes real-time and CPU time system performance statistics to the SAS log

Statistic	Description
Real-Time	the amount of time spent to process the SAS job. Real-time is also referred to as elapsed time.
CPU Time	the total time spent to execute your SAS code and spent to perform system overhead tasks on behalf of the SAS process. This value is the combination of the user CPU and system CPU statistics from FULLSTIMER

Note: Starting in SAS 9, some procedures use **multiple threads**. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, **CPU time might exceed real-time in your STIMER output**

Processing Environment

FULLSTIMER - Specifies whether all the performance statistics of your computer system that are available to SAS are written to the SAS log

Statistic	Description
Real-Time	the amount of time spent to process the SAS job. Real-time is also referred to as elapsed time
User CPU Time	the CPU time spent to execute SAS code
System CPU Time	the CPU time spent to perform operating system tasks (system overhead tasks) that support the execution of SAS code
Memory	the amount of memory required to run a step.
OS Memory	the maximum amount of memory that a step requested from the System

Create Readable Code

Tips for creating code that you and your co-workers will find easy to read and understand

1. Comment, Comment, Comment

Method 1:

```
/* create summary report*/
proc means data=new;
   more statements here;
run;
```

Method 2:

```
*create summary report;
proc means data=old;
   more statements here;
run;
```

Note: Method 1 may also be helpful when developing and debugging code

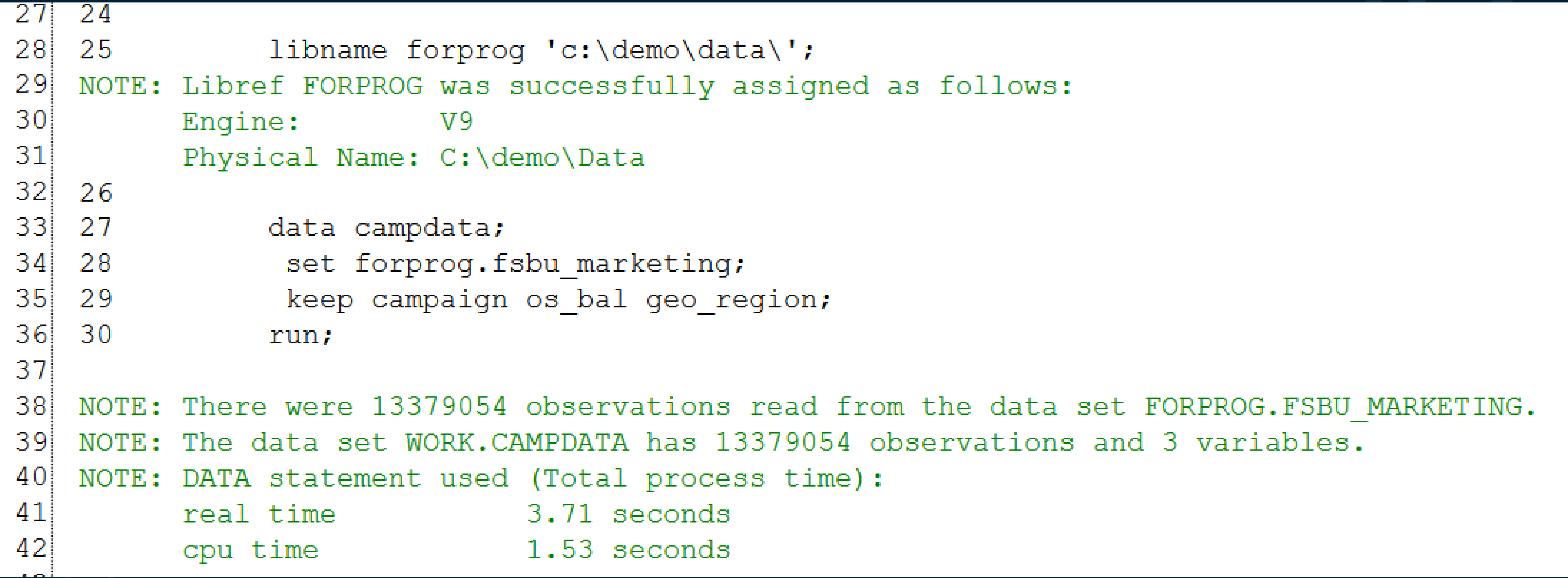
1. Comment, Comment, Comment

Method 1:

```
/*
data new;
   set old;
run;
*/
proc means data=new;
   more statements here;
run;
```

Efficiency consideration: every submission of the DATA step re-creates the SAS data

1. Comment, Comment, Comment



```
24
25         libname forprog 'c:\demo\data\';
NOTE: Libref FORPROG was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\demo\Data
26
27         data campdata;
28          set forprog.fsbu_marketing;
29          keep campaign os_bal geo_region;
30         run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.CAMPDATA has 13379054 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           3.71 seconds
      cpu time            1.53 seconds
```

Efficiency consideration: every submission of the DATA step re-creates the SAS data

2. Use recommended formatting for SAS code

Do this:

```
Data new;
  set old;
Run;
proc means data=new;
  var newvar;
  class year;
run;
```

Not this:

```
data new; set old; run;
proc means data=new;
var newvar; class year;
run;
```

If you are using Enterprise Guide, you can use **Format Code** or **Ctrl-i**

3. Use Descriptive Names

Do this:

```
data salaryinfo2012;
  set salaryinfo2011;
  newsalary=
      oldsalary+increase;
run;
```

Not this:

```
data new;
  set old;
  z=x+y;
run;
```

4. Use Underscores or Camel Case to Create Descriptive Names

Camel Case

```
data salaryInfo2012;
  set salaryInfo2011;
  newSalary=
      oldSalary+increase;
run;
```

Underscores

```
data salary_info2012;
  set salary_info2011;
  new_salary=
      old_salary+increase;
run;
```

SAS names:

- Can be 32 characters long
- Must start with a letter or underscore, continuing with numbers, letters or underscores
- Can be uppercase, lowercase or mixed case
- Are not case sensitive

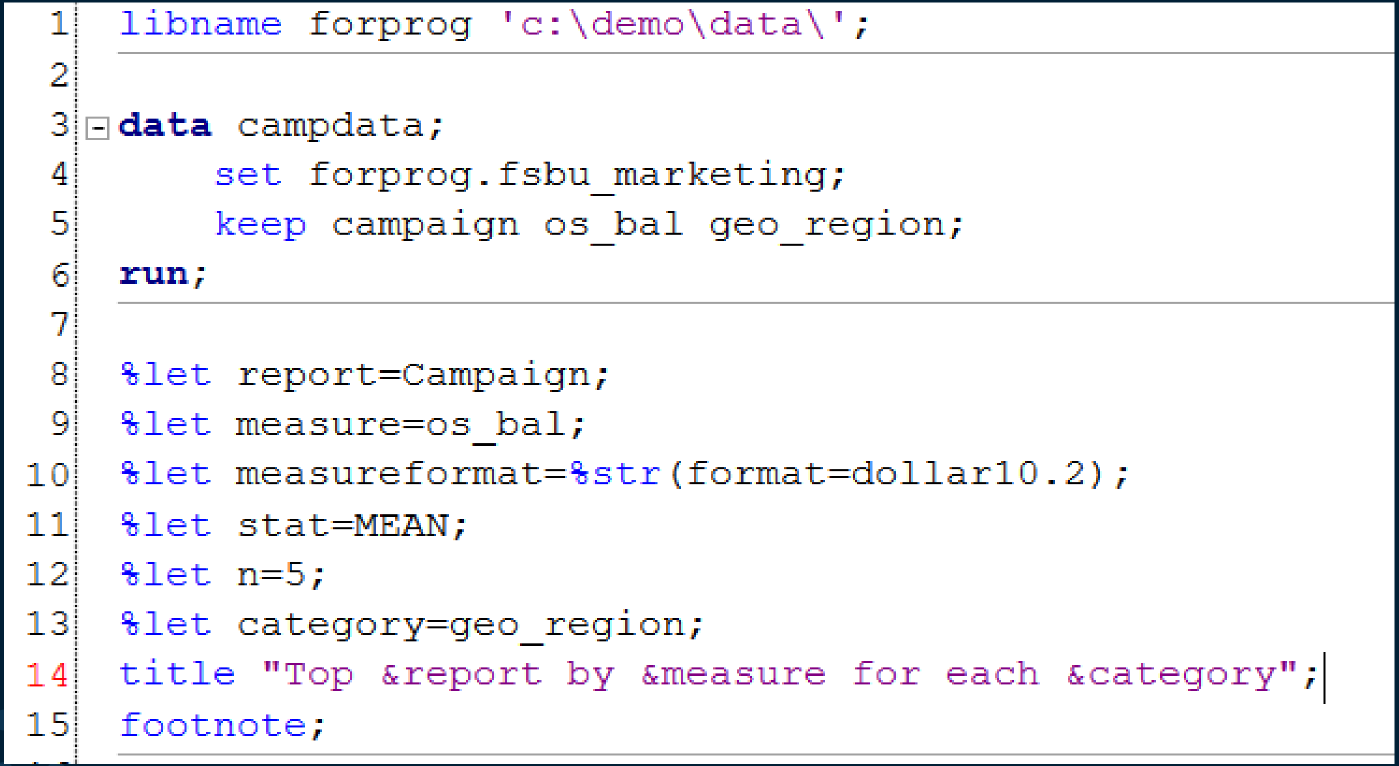
5. Put All “global” Statements at the Beginning of Your Code

- Libname statements, system options, and title, footnote statements are easier to find (and change, if necessary) if they are all in one place.

- Helpful when creating prompts for SAS Stored Processes in Enterprise Guide or SAS Management Console

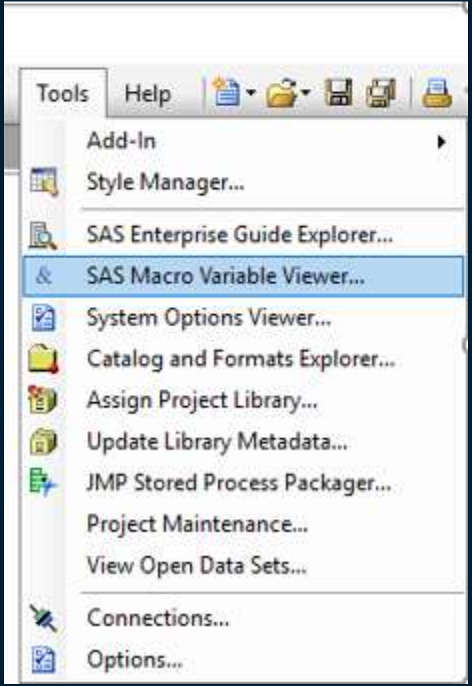
- Not a requirement of SAS language

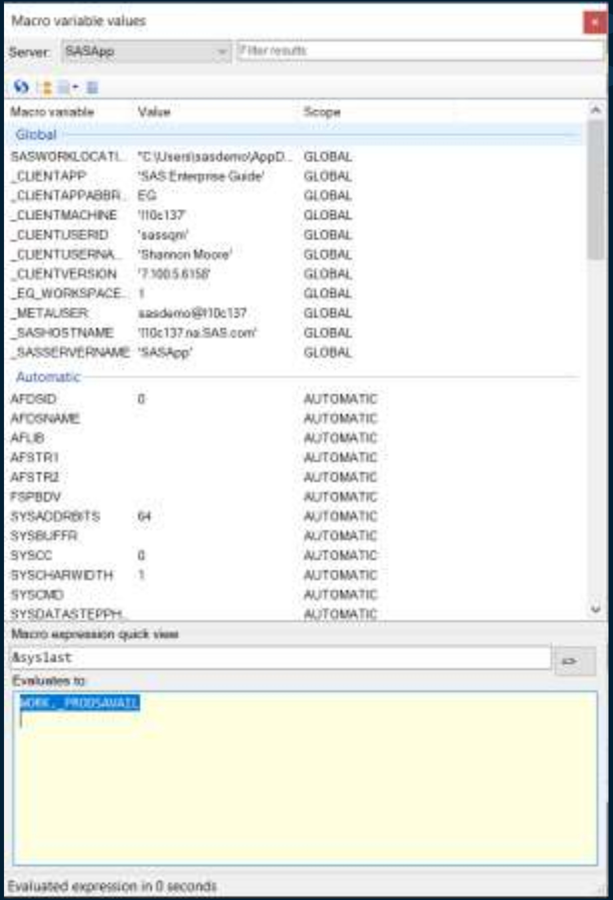
5. Put All “global” Statements at the Beginning of Your Code



```
libname forprog 'c:\demo\data\';

data campdata;
     set forprog.fsbu_marketing;
     keep campaign os_bal geo_region;
run;

%let report=Campaign;
%let measure=os_bal;
%let measureformat=%str(format=dollar10.2);
%let stat=MEAN;
%let n=5;
%let category=geo_region;
title "Top &report by &measure for each &category";
footnote;
```

5a. New in Enterprise Guide – Macro Variable Viewer


Tools
Help
Add-In
Style Manager...
SAS Enterprise Guide Explorer...
SAS Macro Variable Viewer...
System Options Viewer...
Catalog and Formats Explorer...
Assign Project Library...
Update Library Metadata...
JMP Stored Process Packager...
Project Maintenance...
View Open Data Sets...
Connections...
Options...



Macro variable	Value	Scope
Global		
SASWORKLOCATI...	"C:\Users\sasdemo\AppD...	GLOBAL
_CLIENTAPP	'SAS Enterprise Guide'	GLOBAL
_CLIENTAPPABBR...	EG	GLOBAL
_CLIENTMACHINE	'l10c137'	GLOBAL
_CLIENTUSERID	'sassqm'	GLOBAL
_CLIENTUSERNA...	'Shannon Moore'	GLOBAL
_CLIENTVERSION	'7.100.5.6158'	GLOBAL
_EG_WORKSPACE...	1	GLOBAL
_METAUSER	sasdemo@l10c137	GLOBAL
_SASHOSTNAME	'l10c137.na.SAS.com'	GLOBAL
_SASSERVERNAME	'SASApp'	GLOBAL
Automatic		
AFDSID	0	AUTOMATIC
AFDSNAME		AUTOMATIC
AFLIB		AUTOMATIC
AFSTR1		AUTOMATIC
AFSTR2		AUTOMATIC
FSPBDV		AUTOMATIC
SYSADDRBITS	64	AUTOMATIC
SYSBUFFR		AUTOMATIC
SYSCC	0	AUTOMATIC
SYSCHARWIDTH	1	AUTOMATIC
SYSCMD		AUTOMATIC
SYSDATASTEPPH...		AUTOMATIC

Macro expression quick view

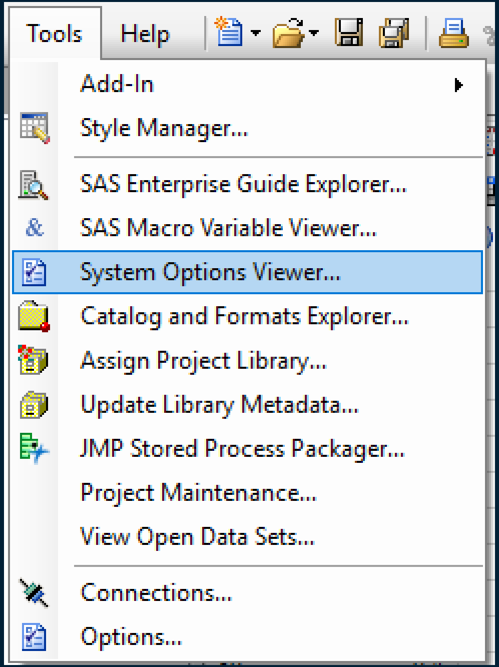
```
&syslast
```

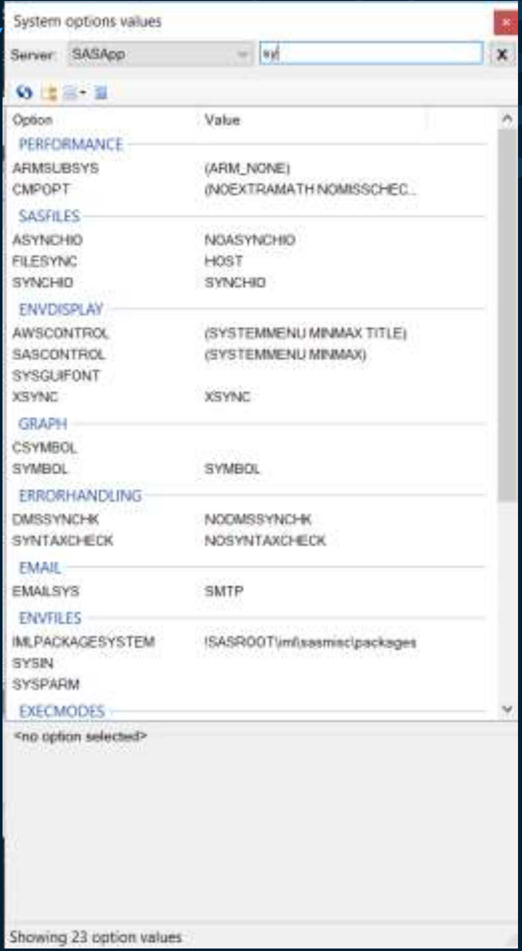
Evaluates to

```
WORK._PRODSAVAIL
```

Evaluated expression in 0 seconds

5b. SAS Options Viewer


Tools
Help
Add-In
Style Manager...
SAS Enterprise Guide Explorer...
SAS Macro Variable Viewer...
System Options Viewer...
Catalog and Formats Explorer...
Assign Project Library...
Update Library Metadata...
JMP Stored Process Packager...
Project Maintenance...
View Open Data Sets...
Connections...
Options...



Option	Value
PERFORMANCE	
ARMSUBSYS	(ARM_NONE)
CMPOPT	(NOEXTRAMATH NOMISSCHEC...
SASFILES	
ASYNCHIO	NOASYNCHIO
FILESYNC	HOST
SYNCHIO	SYNCHIO
ENVDISPLAY	
AWSCONTROL	(SYSTEMMENU MINMAX TITLE)
SASCONTROL	(SYSTEMMENU MINMAX)
SYSGUIFONT	
XSYNC	XSYNC
GRAPH	
CSYMBOL	
SYMBOL	SYMBOL
ERRORHANDLING	
DMSSYNCHK	NODMSSYNCHK
SYNTAXCHECK	NOSYNTAXCHECK
EMAIL	
EMAILSYS	SMTP
ENVFILES	
IMLPACKAGESYSTEM	!SASROOT\iml\sasmisc\packages
SYSIN	
SYSPARM	
EXECMODES	

<no option selected>

Showing 23 option values

Basic Coding Recommendations

Basic coding recommendations to increase the efficiency of your SAS programs

6. Minimize the number of times you **read** your data

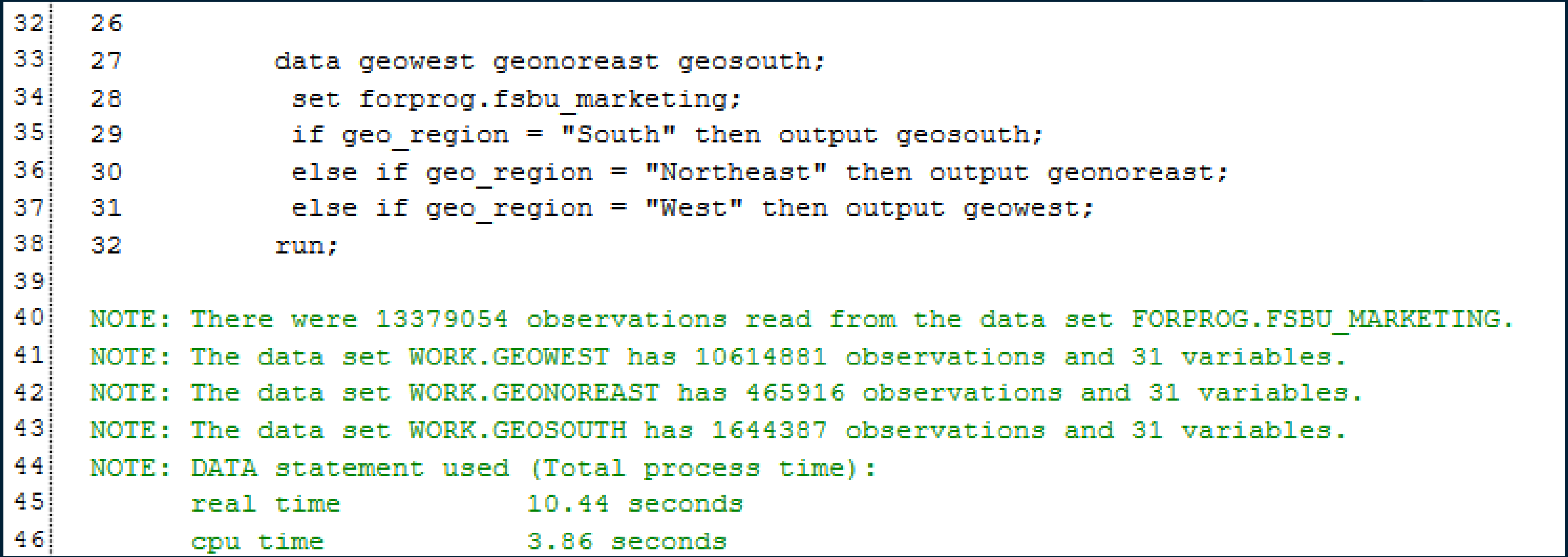
Do this:

```
data a b c;
  set old;
  if condition then
     output a;
  else if condition then
     output b;
  else if condition then
     output c;
run;
```

Not this:

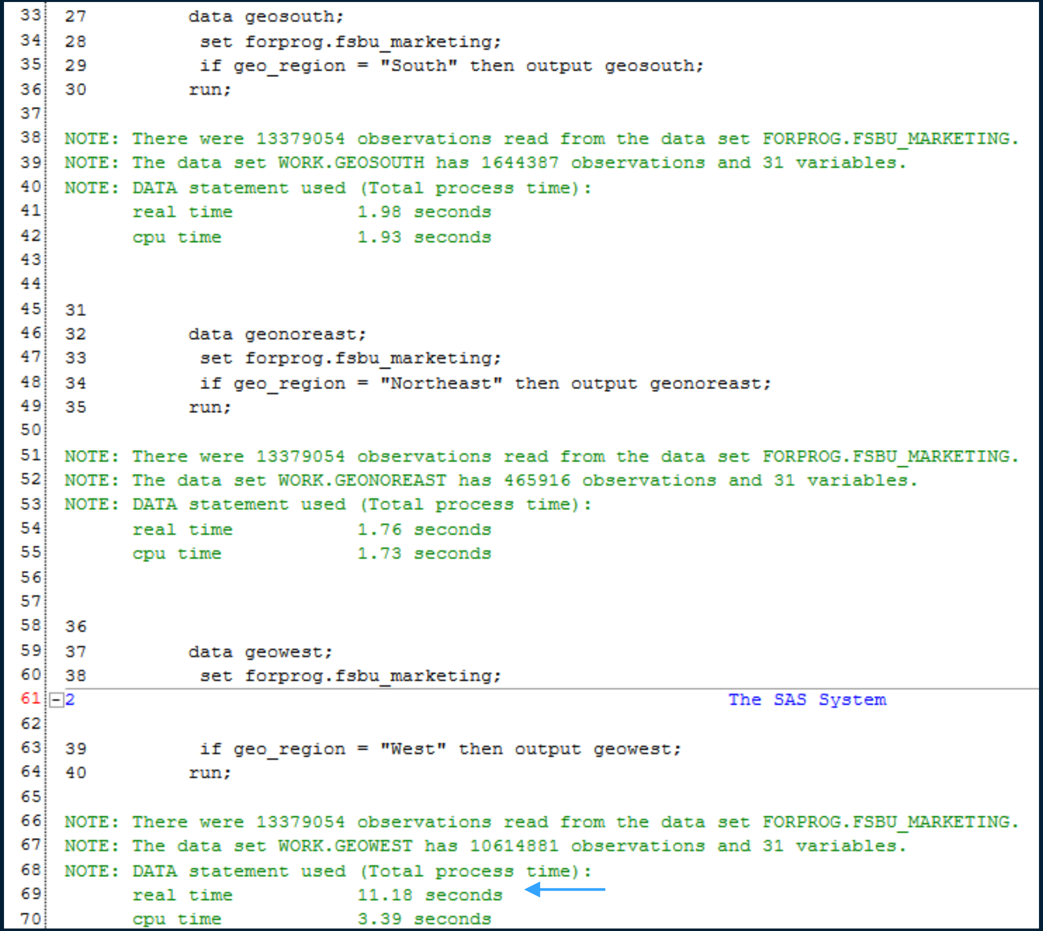
```
data a;
    set old;
    [more code]
run;
data b;
    set old;
    [more code]
run;
data c;
    set old;
    [more code]
run;
```

6. Minimize the number of times you **read** your data



```
26
27          data geowest geonoreast geosouth;
28           set forprog.fsbu_marketing;
29           if geo_region = "South" then output geosouth;
30           else if geo_region = "Northeast" then output geonoreast;
31           else if geo_region = "West" then output geowest;
32          run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEOWEST has 10614881 observations and 31 variables.
NOTE: The data set WORK.GEONOREAST has 465916 observations and 31 variables.
NOTE: The data set WORK.GEOSOUTH has 1644387 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           10.44 seconds
      cpu time            3.86 seconds
```

6. Minimize the number of times you **read** your data



```
27          data geosouth;
28           set forprog.fsbu_marketing;
29           if geo_region = "South" then output geosouth;
30          run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEOSOUTH has 1644387 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           1.98 seconds
      cpu time            1.93 seconds


31
32          data geonoreast;
33           set forprog.fsbu_marketing;
34           if geo_region = "Northeast" then output geonoreast;
35          run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEONOREAST has 465916 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           1.76 seconds
      cpu time            1.73 seconds


36
37          data geowest;
38           set forprog.fsbu_marketing;
2                                                              The SAS System

39           if geo_region = "West" then output geowest;
40          run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEOWEST has 10614881 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           11.18 seconds
      cpu time            3.39 seconds
```

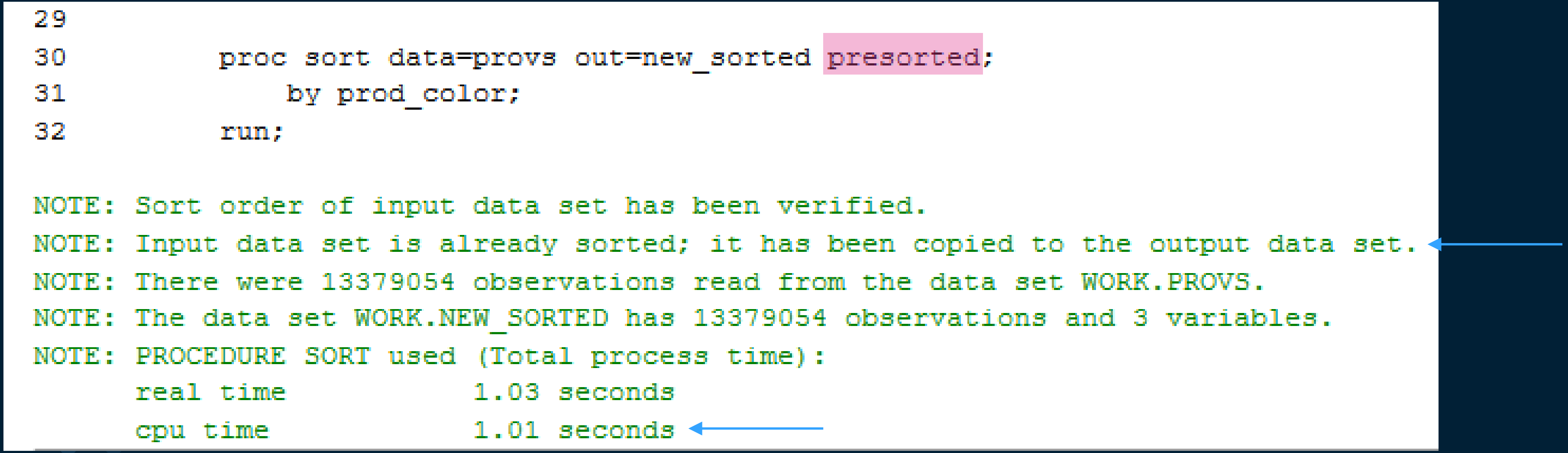
7. Limit the number of times you **sort** your data

If you think the incoming data is already sorted, use the **presorted** option on your SORT statement; the sort order will be verified

```
data new;
     infile ‘rawdata.dat’;
     input ID $ 1-4 name $ 5-25 salary 26-35;
run;

proc sort data=new out=new_sorted presorted;
    by ID;
run;
```

7. Limit the number of times you **sort** your data

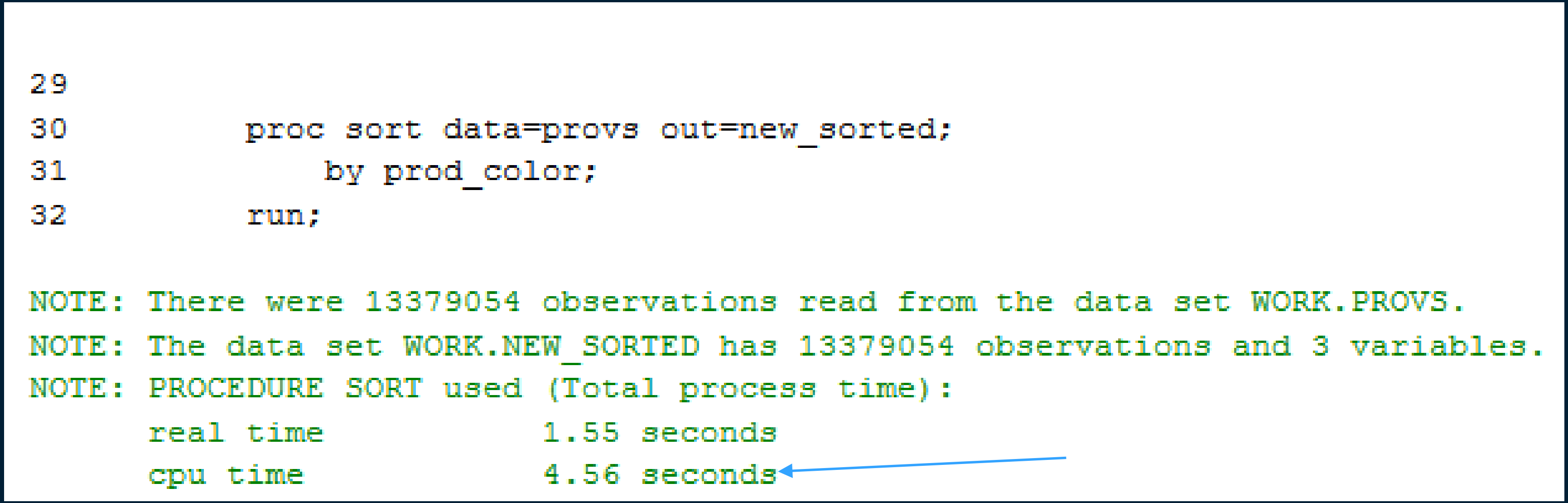
If you think the incoming data is already sorted, use the **presorted** option on your SORT statement; the sort order will be verified



```
        proc sort data=provs out=new_sorted presorted;
            by prod_color;
        run;

NOTE: Sort order of input data set has been verified.
NOTE: Input data set is already sorted; it has been copied to the output data set.
NOTE: There were 13379054 observations read from the data set WORK.PROVS.
NOTE: The data set WORK.NEW_SORTED has 13379054 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           1.03 seconds
      cpu time            1.01 seconds
```

7. Limit the number of times you **sort** your data

If you think the incoming data is already sorted, use the **presorted** option on your SORT statement; the sort order will be verified



```
          proc sort data=provs out=new_sorted;
              by prod_color;
          run;

NOTE: There were 13379054 observations read from the data set WORK.PROVS.
NOTE: The data set WORK.NEW_SORTED has 13379054 observations and 3 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           1.55 seconds
      cpu time            4.56 seconds
```

8. Use IF-THEN-ELSE instead of IF-IF-IF

Do this:

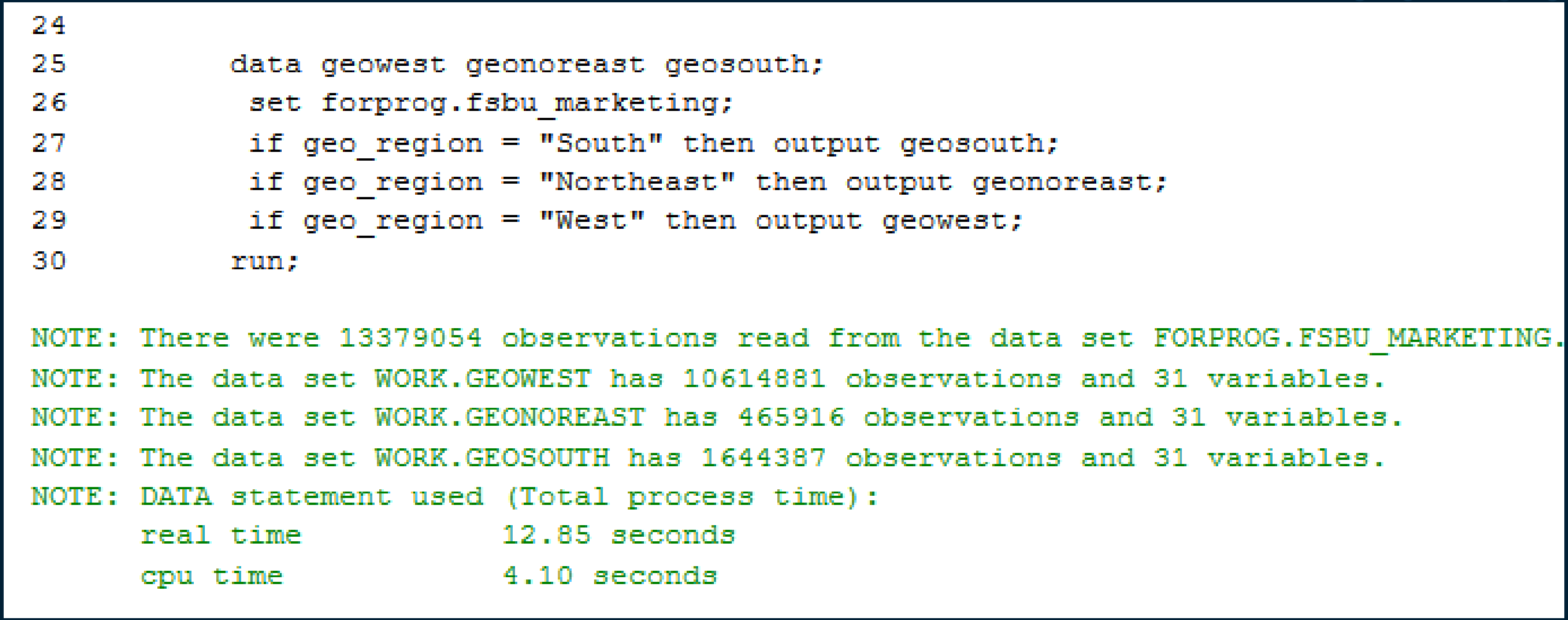
```
data new;
  set old;
  if condition then
     some action;
  else if condition then
     some other action;
  else if condition then
     some other action;
run;
```

Not this:

```
data new;
  set old;
  if condition then
     some action;
  if condition then
     some other action;
  if condition then
     some other action;
run;
```

Please note that this general recommendation relates to conditions that are **mutually exclusive**

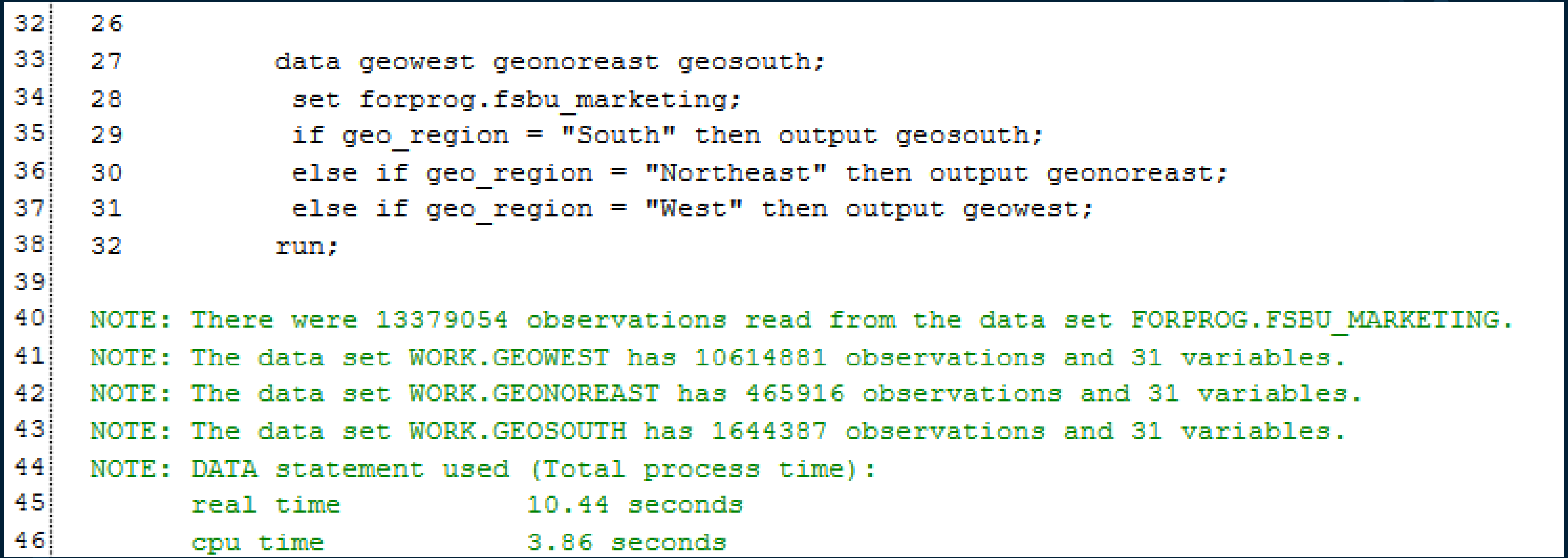
8. Use IF-THEN-ELSE instead of IF-IF-IF



```
        data geowest geonoreast geosouth;
         set forprog.fsbu_marketing;
         if geo_region = "South" then output geosouth;
         if geo_region = "Northeast" then output geonoreast;
         if geo_region = "West" then output geowest;
        run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEOWEST has 10614881 observations and 31 variables.
NOTE: The data set WORK.GEONOREAST has 465916 observations and 31 variables.
NOTE: The data set WORK.GEOSOUTH has 1644387 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           12.85 seconds
      cpu time            4.10 seconds
```

Please note that this general recommendation relates to conditions that are **mutually exclusive**

8. Use IF-THEN-ELSE instead of IF-IF-IF



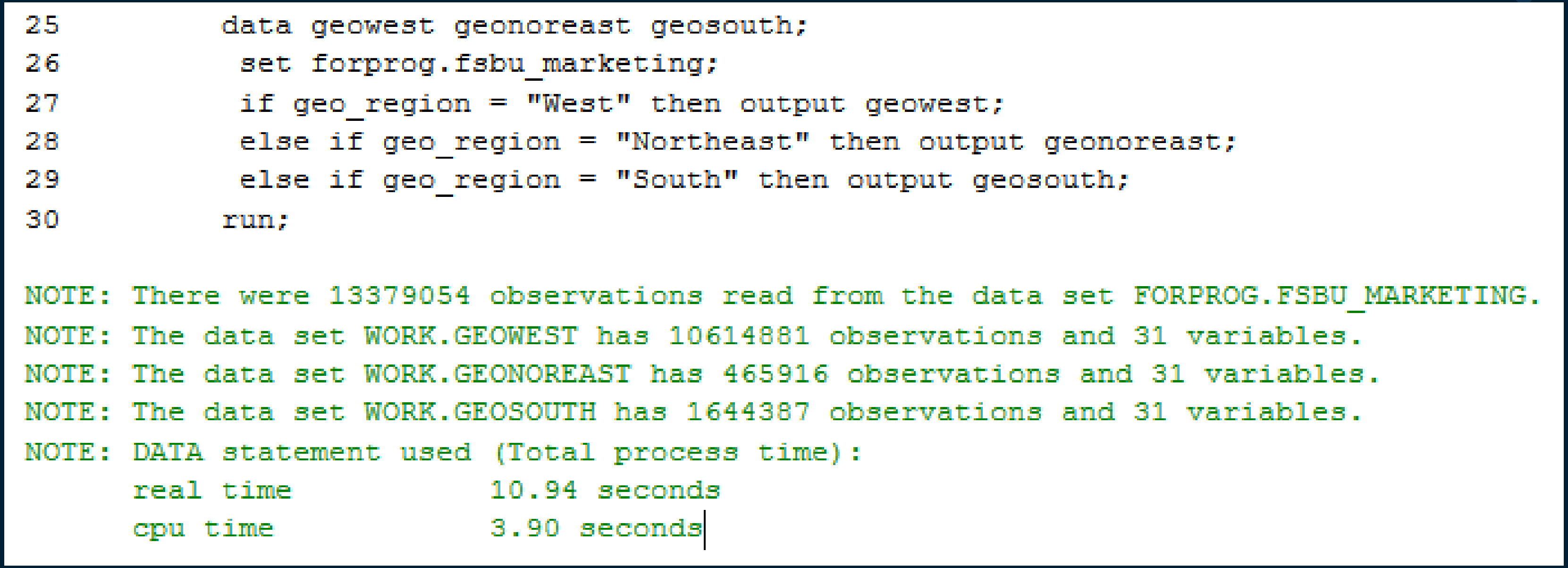
```
26
27         data geowest geonoreast geosouth;
28          set forprog.fsbu_marketing;
28          if geo_region = "South" then output geosouth;
30          else if geo_region = "Northeast" then output geonoreast;
31          else if geo_region = "West" then output geowest;
32         run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEOWEST has 10614881 observations and 31 variables.
NOTE: The data set WORK.GEONOREAST has 465916 observations and 31 variables.
NOTE: The data set WORK.GEOSOUTH has 1644387 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           10.44 seconds
      cpu time            3.86 seconds
```

Please note that this general recommendation relates to conditions that are **mutually exclusive**

9. Order of IF THEN conditions in descending order of probability

```
data new;
  set old;
  if condition occurring most often then
     some action;
  else if condition then
     some other action;
  else if condition then
     some other action;
run;
```

9. Order of IF THEN conditions in descending order of probability



```
data geowest geonoreast geosouth;
 set forprog.fsbu_marketing;
 if geo_region = "West" then output geowest;
 else if geo_region = "Northeast" then output geonoreast;
 else if geo_region = "South" then output geosouth;
run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.GEOWEST has 10614881 observations and 31 variables.
NOTE: The data set WORK.GEONOREAST has 465916 observations and 31 variables.
NOTE: The data set WORK.GEOSOUTH has 1644387 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           10.94 seconds
      cpu time            3.90 seconds
```

10. Select only the columns you need when working with SAS data

Do This:

```
data new;
   set old (drop=category
      type value ...);
   more statements here;
run;
```

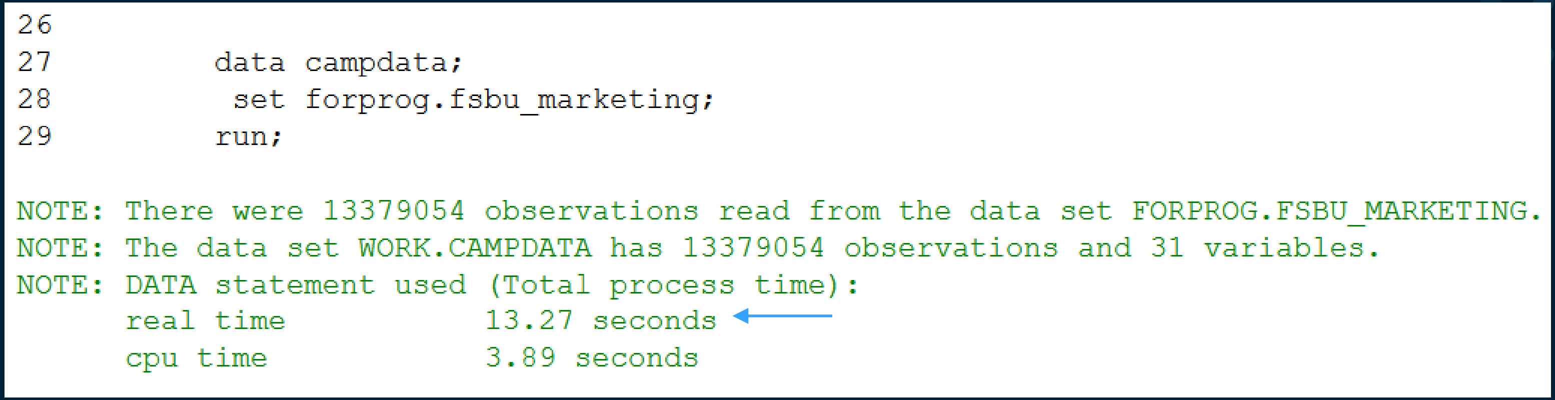
Not This:

```
data new;
   set old;
   more statements here;
run;
```

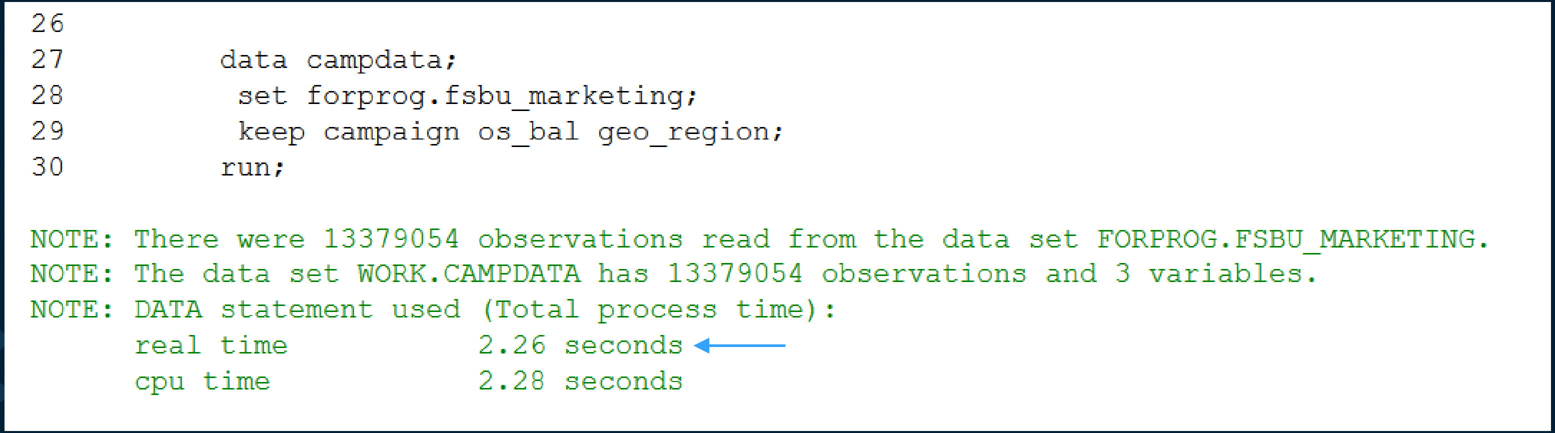
Variations:

- Use the **keep=** option if you need to keep more variables than you need to drop
- Use both **keep=** and **drop=** options to control variables on both the incoming and outgoing sides
- **Keep=** and **drop=** options can be used in PROC steps, too

10. Select only the columns you need when working with SAS data



```
26
27         data campdata;
28          set forprog.fsbu_marketing;
29         run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.CAMPDATA has 13379054 observations and 31 variables.
NOTE: DATA statement used (Total process time):
      real time           13.27 seconds
      cpu time            3.89 seconds
```



```
26
27         data campdata;
28          set forprog.fsbu_marketing;
29          keep campaign os_bal geo_region;
30         run;

NOTE: There were 13379054 observations read from the data set FORPROG.FSBU_MARKETING.
NOTE: The data set WORK.CAMPDATA has 13379054 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           2.26 seconds
      cpu time            2.28 seconds
```

11. Select only the rows you need when working with SAS data

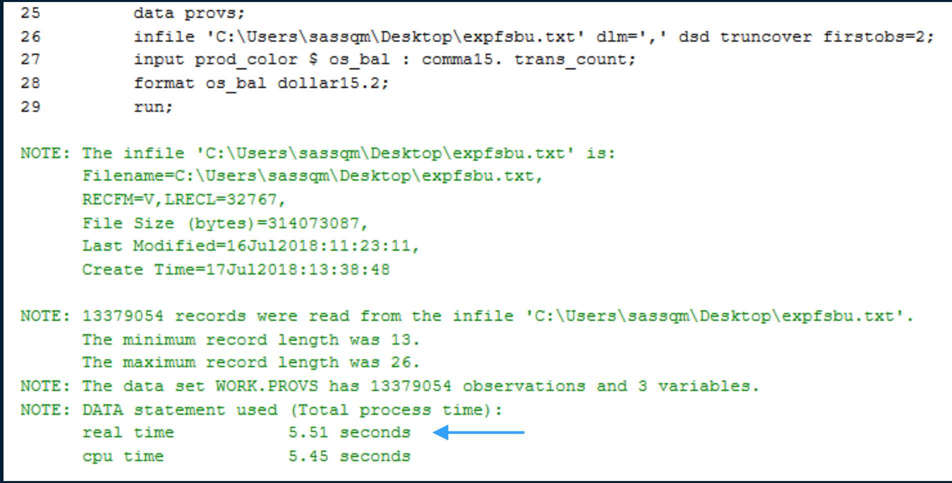
Do this:

```
data new;
   infile ‘old.dat’;
   if city=‘CLEVELAND';
   more statements here;
run;
```

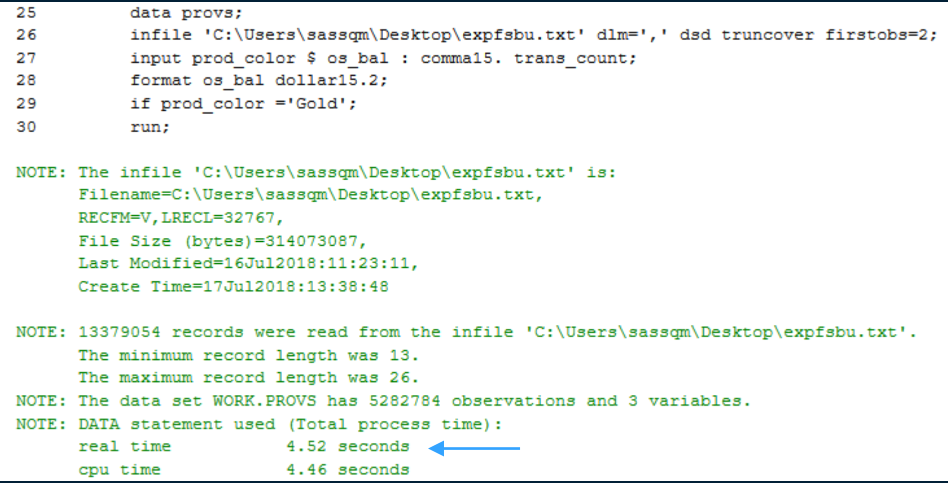
Not this:

```
data new;
   infile ‘old.dat’;
   more statements here;
run;
```

11. Select only the rows you need when working with SAS data



```
        data provs;
        infile 'C:\Users\sassqm\Desktop\expfsbu.txt' dlm=',' dsd truncover firstobs=2;
        input prod_color $ os_bal : comma15. trans_count;
        format os_bal dollar15.2;
        run;

NOTE: The infile 'C:\Users\sassqm\Desktop\expfsbu.txt' is:
      Filename=C:\Users\sassqm\Desktop\expfsbu.txt,
      RECFM=V,LRECL=32767,
      File Size (bytes)=314073087,
      Last Modified=16Jul2018:11:23:11,
      Create Time=17Jul2018:13:38:48

NOTE: 13379054 records were read from the infile 'C:\Users\sassqm\Desktop\expfsbu.txt'.
      The minimum record length was 13.
      The maximum record length was 26.
NOTE: The data set WORK.PROVS has 13379054 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           5.51 seconds
      cpu time            5.45 seconds
```



```
        data provs;
        infile 'C:\Users\sassqm\Desktop\expfsbu.txt' dlm=',' dsd truncover firstobs=2;
        input prod_color $ os_bal : comma15. trans_count;
        format os_bal dollar15.2;
        if prod_color ='Gold';
        run;

NOTE: The infile 'C:\Users\sassqm\Desktop\expfsbu.txt' is:
      Filename=C:\Users\sassqm\Desktop\expfsbu.txt,
      RECFM=V,LRECL=32767,
      File Size (bytes)=314073087,
      Last Modified=16Jul2018:11:23:11,
      Create Time=17Jul2018:13:38:48

NOTE: 13379054 records were read from the infile 'C:\Users\sassqm\Desktop\expfsbu.txt'.
      The minimum record length was 13.
      The maximum record length was 26.
NOTE: The data set WORK.PROVS has 5282784 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           4.52 seconds
      cpu time            4.46 seconds
```

12. Consider the position of the subsetting IF

Do this:

```
data new;
   infile ‘old.dat’;
   if city=‘CLEVELAND';
   more statements here;
run;
```

Not this:

```
data new;
   infile ‘old.dat’;
   more statements here;
   if city=‘CLEVELAND';
run;
```

Subset as soon as you have all necessary values in order to prevent unnecessary creation of variables and additional processing

13. If you are reading SAS data, use WHERE instead of subsetting IF

Instead of this:

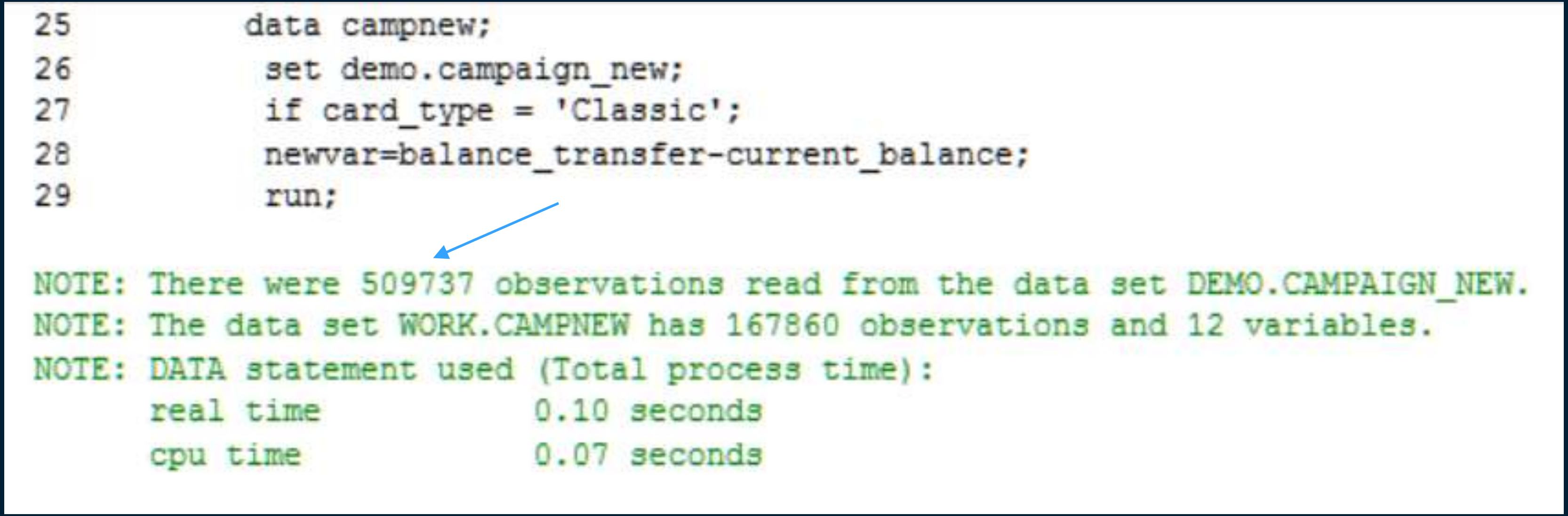
```
data new;
   set old;
   if condition;
   more statements here;
run;
```

Try this:

```
data new;
   set old;
   where condition;
   more statements here;
run;
```

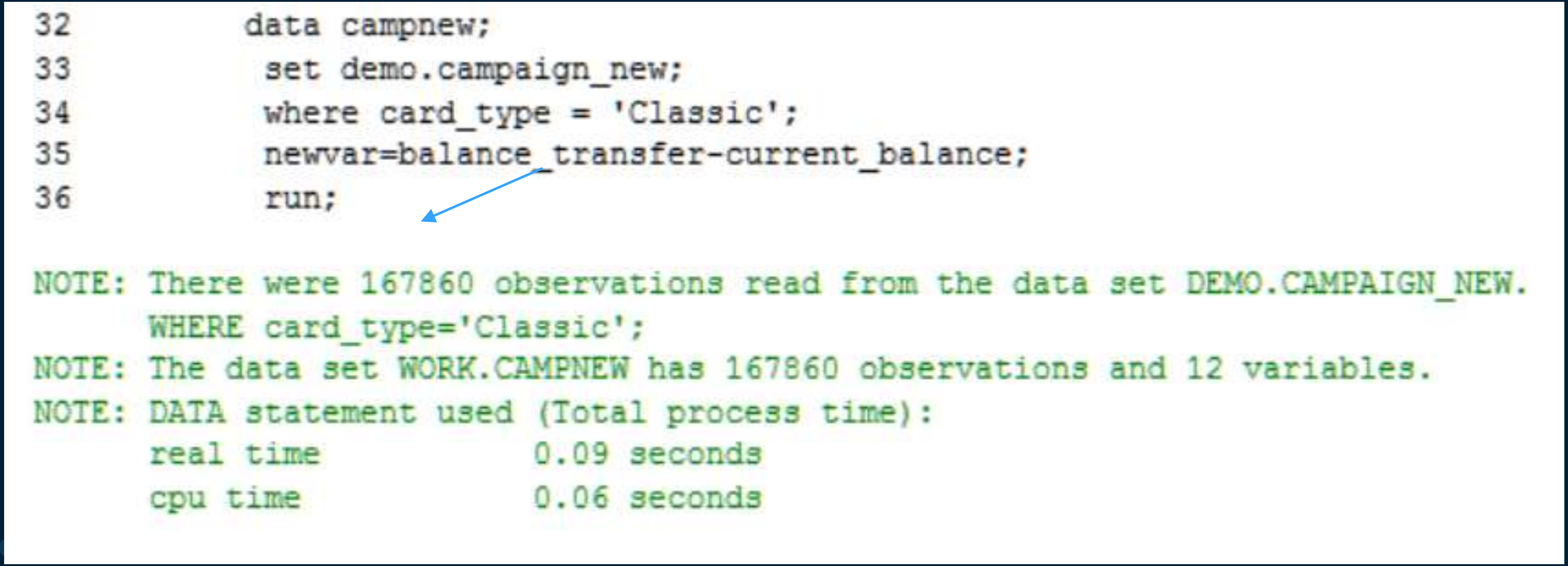
Added efficiency: When using SAS/Access engines, SAS attempts to send the **WHERE** clause to the RDBMS for evaluation rather than to SAS;
With the IF statement, SAS must do the processing

If loads entire dataset



```
data campnew;
 set demo.campaign_new;
 if card_type = 'Classic';
 newvar=balance_transfer-current_balance;
 run;

NOTE: There were 509737 observations read from the data set DEMO.CAMPAIGN_NEW.
NOTE: The data set WORK.CAMPNEW has 167860 observations and 12 variables.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      cpu time            0.07 seconds
```

Where subsets first



```
data campnew;
 set demo.campaign_new;
 where card_type = 'Classic';
 newvar=balance_transfer-current_balance;
 run;

NOTE: There were 167860 observations read from the data set DEMO.CAMPAIGN_NEW.
      WHERE card_type='Classic';
NOTE: The data set WORK.CAMPNEW has 167860 observations and 12 variables.
NOTE: DATA statement used (Total process time):
      real time           0.09 seconds
      cpu time            0.06 seconds
```

13. Where vs IF - WHERE and IF processing are not always interchangeable

- **IF** processing <u>must</u> be used with:
  - Accessing raw data using **INPUT** statements
  - With **Automatic Variables**, e.g. first.variable, last.variable, _N_, etc.
  - Using **newly created variables** in the **same DATA Step**
  - In combination with **data set options** such as OBS =, POINT = , FIRSTOBS =
  - To **conditionally execute** a statement

13. Where vs IF - WHERE and IF processing are not always interchangeable

- **WHERE** processing <u>must</u> be used to:
  - **Utilize special operators** such as LIKE or CONTAINS
  - **Filter rows** for input to SAS Procedures
  - Trigger use of **indexes***, if available
  - When sub-setting as data set option
  - When sub-setting using **PROC SQL**

*The presence of an index column on a SAS data set does not always guarantee its use in a query

13. Where vs IF

- WHERE and IF processing are applied differently in MERGE operations:
  - With WHERE processing the sub-setting takes place **before** the MERGE operation
  - With IF processing the sub-setting takes place **after** the MERGE operation

13b. IF Then vs SELECT else

IF THEN

- When there are **few conditions** to check
- The values are **not uniformly distributed**
- The values are character or the values are discrete numeric data

SELECT

- Where there is a long series of mutually exclusive conditions
- The values are numeric and are uniformly distributed

Efficiency Considerations Using the SAS® System

EFFECTIVE USE OF SAS SELECT LANGUAGE STATEMENT

14. Consider declaring variables as character when there is a storage savings

Consider Employee ID values similar to the following:

```
1015
2034
5543
6793
...
```

Compare:

```
data new;
  input ID 1-4;
```

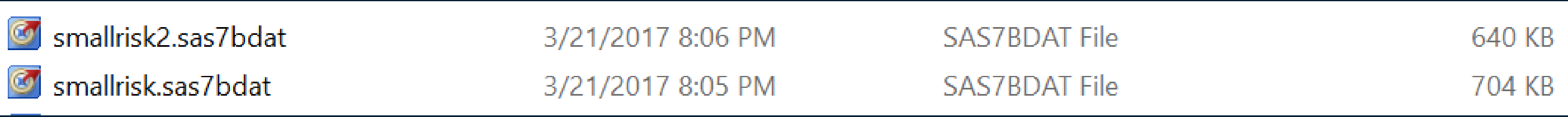
ID is numeric requiring 8 bytes of storage

```
data new;
  input ID $ 1-4;
```

ID is character requiring 4 bytes of storage

A savings of 4 bytes per observation adds up when dealing with large data

14. Consider declaring variables as character when there is a storage savings



smallrisk2.sas7bdat	3/21/2017 8:06 PM	SAS7BDAT File	640 KB
smallrisk.sas7bdat	3/21/2017 8:05 PM	SAS7BDAT File	704 KB

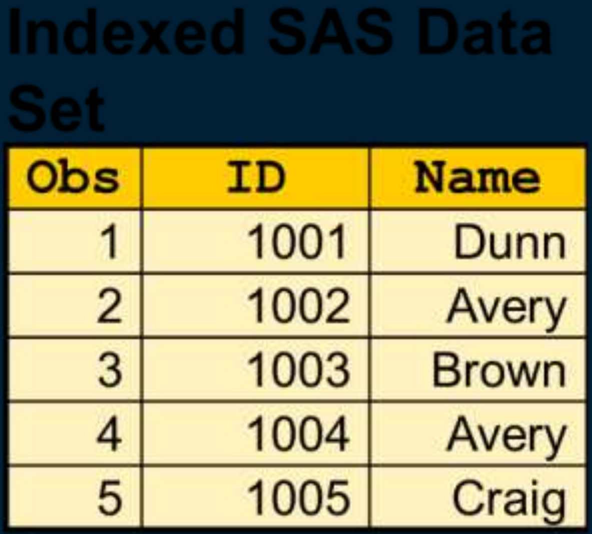
A savings of a few bytes per observation adds up when dealing with large data – 10%

15. Consider adding indexes to your data if you will be filtering it frequently

- What is an index?
  - An index is an optional **file** that you can **create** for a SAS data set in order to **provide direct access** to **specific observations**
  - In other words, an index **enables** you to **locate** an observation by **value**

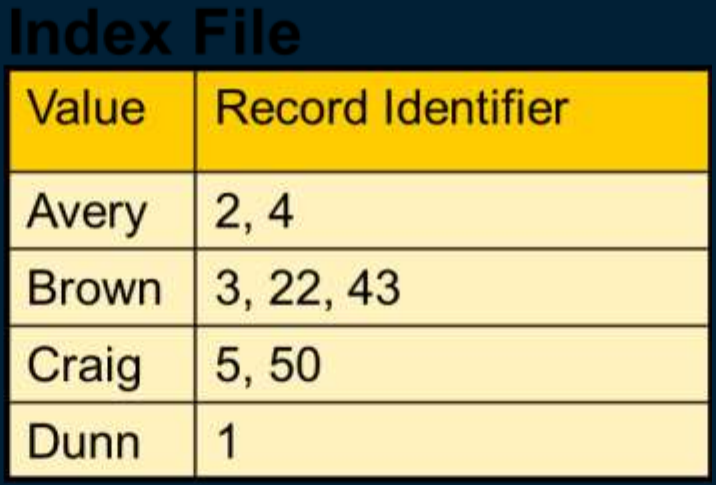
15. What Is an Index?

The index file has the **same name** as its associated data file, and a member type of INDEX



Indexed SAS Data Set

Obs	ID	Name
1	1001	Dunn
2	1002	Avery
3	1003	Brown
4	1004	Avery
5	1005	Craig



Index File

Value	Record Identifier
Avery	2, 4
Brown	3, 22, 43
Craig	5, 50
Dunn	1

15. Using Indexes

When an index is used to process a request, such as a WHERE expression, SAS

- Performs a **binary search** on the index file and positions the index to the first entry that contains a qualified value
- Uses the value's **Record Identifier** (RID) to read the observation(s) that contains the value
- **No sequential read** needed

15. When to Use Indexes?

Index guidelines:

- Indexes perform best when they retrieve **15% or fewer** rows in a table/data set
- Indexes are **not** usually **useful** if they are based on **uniformly distributed**, low cardinality columns - Male & Female example
- **Do not** create indexes on **small tables**. Sequential access is faster
- **Minimize the number of indexes** in order to reduce disk storage and update costs

15. Consider adding indexes to your data if you will be filtering it frequently

- Indexes can be **created** on the DATA statement, with PROC **SQL**, with PROC **DATASETS**, and in other ways
- Indexes can be simple or composite
- Under the right circumstances, indexes can decrease processing time
- However, indexes take up space!
- Carefully consider whether an index makes sense in the specific situation

Added efficiency: sort the data in ascending order on the key variable before indexing the data file

15. Methods for Creating Indexes

DATA step:

```
data finances(index=(stock /unique));
 more statements here
run;
```

PROC SQL:

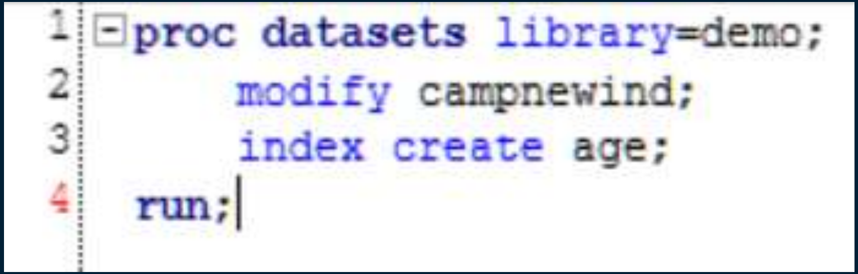
```
proc sql;
      create unique index empnum
         on employee (empnum);
                     Or

proc sql;
     create index names
         on employee(lastname,frstname);
```

15. Methods for Creating Indexes

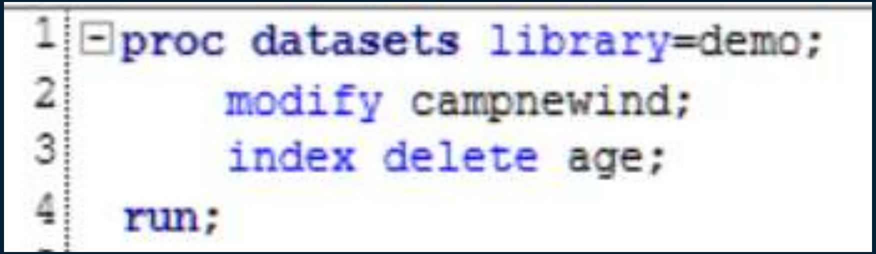
PROC DATASETS:

```
proc datasets library=mylib;
     modify my_dataset;
      index create empnum / unique;
      index create names=(lastname frstname);
run;
```

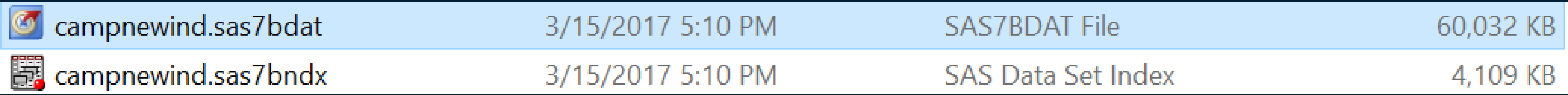
Index file is 6.9% size of dataset



```
proc datasets library=demo;
     modify campnewind;
     index create age;
run;
```

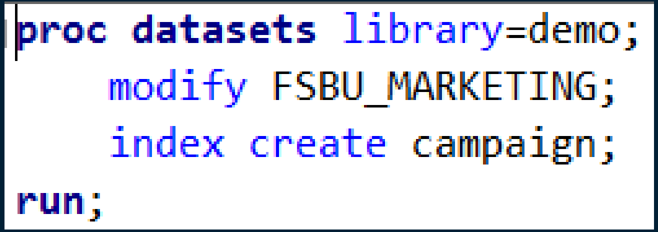


```
proc datasets library=demo;
     modify campnewind;
     index delete age;
run;
```

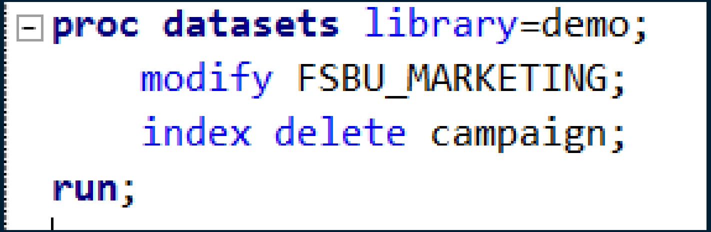


```
campnewind.sas7bdat     3/15/2017 5:10 PM     SAS7BDAT File          60,032 KB
campnewind.sas7bndx     3/15/2017 5:10 PM     SAS Data Set Index      4,109 KB
```

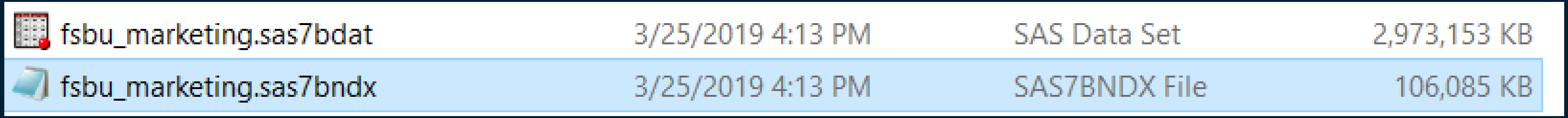
Index file is 3.6% size of dataset



```
proc datasets library=demo;
    modify FSBU_MARKETING;
    index create campaign;
run;
```

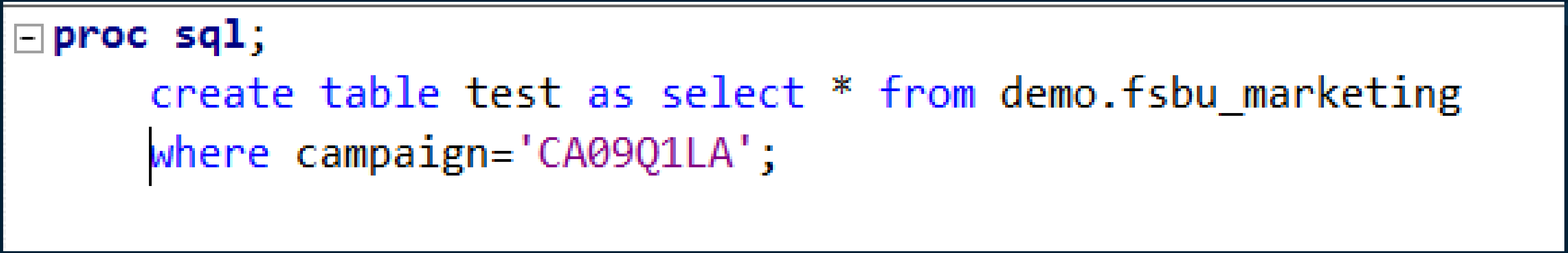


```
proc datasets library=demo;
    modify FSBU_MARKETING;
    index delete campaign;
run;
```

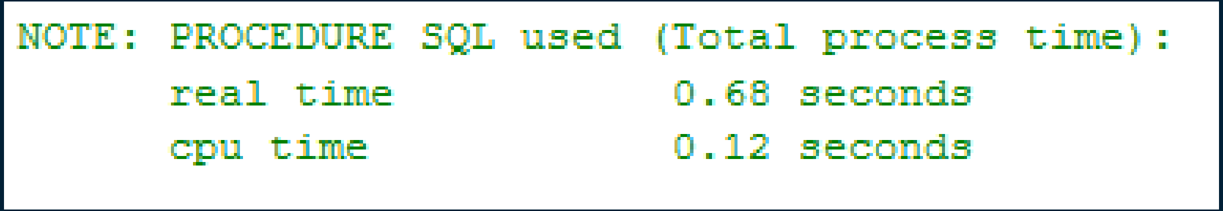


fsbu_marketing.sas7bdat	3/25/2019 4:13 PM	SAS Data Set	2,973,153 KB
fsbu_marketing.sas7bndx	3/25/2019 4:13 PM	SAS7BNDX File	106,085 KB

Indexed on campaign

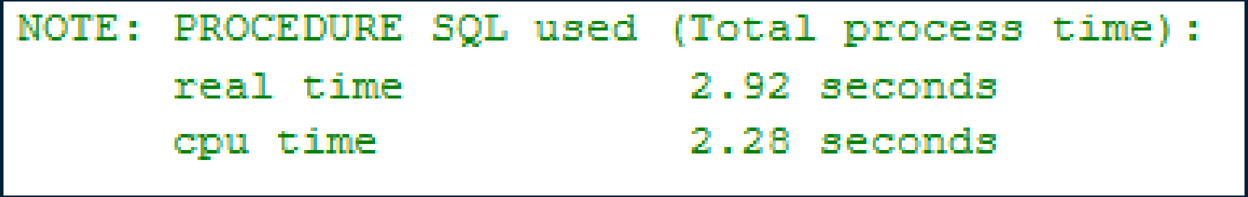


```
proc sql;
    create table test as select * from demo.fsbu_marketing
    where campaign='CA09Q1LA';
```



```
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.68 seconds
      cpu time            0.12 seconds
```

After I deleted the index



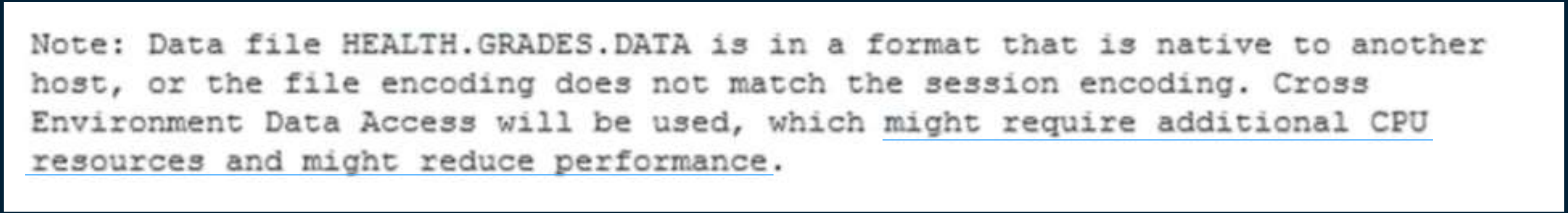
```
NOTE: PROCEDURE SQL used (Total process time):
      real time           2.92 seconds
      cpu time            2.28 seconds
```

16. Use CEDA Wisely

- Reading SAS 9.2 or earlier data sets in SAS 9.3 and 9.4 results in a translation process using **CEDA** (Cross-Environment Data Access)
- UNIX/Windows, 32-/64-bit, etc.
- Because the BASE engine **translates the data as the data is read**, multiple procedures could require SAS to read and translate the data **multiple times**. In this way, repeated translation could affect system performance
- Convert SAS data sets by using **PROC MIGRATE** or other techniques

Use CEDA Wisely

If you see the following note in your log, consider using **PROC MIGRATE** or other techniques to convert your data to your current environment



```
Note: Data file HEALTH.GRADES.DATA is in a format that is native to another
host, or the file encoding does not match the session encoding. Cross
Environment Data Access will be used, which might require additional CPU
resources and might reduce performance.
```

When You are Developing Code

Tips to save time and create efficient code

17. Test your programs with the OBS= option

```
data complicated_program;
  set sample_data(obs=50);
      many;
      many;
      many more statements here;
run;
```

NOTE: This technique **may not** adequately **test all conditions**, but should confirm the **correctness** of the overall program **logic** – and could save time and computer resources

18 .Test your programs with the PUT Statement

```
data complicated_program;
  set sample_data(obs=50);
   if condition then do;
       put ‘write value here’ value;
       other statements to execute;
   end;
run;
```

This technique allows you to test certain coding logic to determine if conditions are met as well as variable values.

19. Benchmark Production Jobs

Recommendations for benchmarking include:

- Benchmark your programs in **separate** SAS **sessions**
- Run each program **multiple** times and **average** the performance statistics
- Use **realistic data** for tests
- **Elapsed time should not be used** for benchmarking, **use CPU time**

20. Use Macro variables to Simplify Maintenance

- Use macro variables to reduce the number of changes that have to be applied manually when code needs to be updated.

This works:

```
Libname file_01 ‘c:\SGF_2018\project_dir\input_data’;

Libname file_02 ‘c:\SGF_2018\project_dir\output_data’;

Filename in_file1 ‘c:\SGF_2018\project_dir\my_text.txt’;
```

This requires less Maintenance

```
%let My_Dir = c:\SGF_2018\project_dir;

Libname file_01 “&My_dir.\input_data“;

Libname file_02 “&My_dir.\output_data“;

Filename in_file1 “&My_dir.\my_test.txt“;
```

Note: the **double quote** will cause the macro variable to be resolved; the one change is applied to all three commands.

20. Use Macro variables to Simplify Maintenance

- Use macro variables to reduce the number of changes that have to be applied manually when code needs to be updated.

This works:

```
Data array_test;
Array counters {15}
var_01-var_15;

Do I = 1 to 15;
       Counters(i) = 0;
End;
```

Three changes needed if array size changes

This requires less maintenance

```
%let max_size = 15;

Data array_test;

Array counters {&max_size} var_01
var_&max_size;

Do I = 1 to &max_size;
       Counters(i) = 0;
End;
```

Only one change here

20. Use Macro variables to Simplify Maintenance

- Use macro variables to reduce the number of changes that have to be applied manually when code needs to be updated.

This works:

```
Data myfile.Monthly_Data_for_2016_feb;
   Set myfile.Monthly_Data_for_2016_Jan;
Run;
```

Most people code the dates into file names

This requires less maintenance

```
%let old_month = 2016_Jan;
%let new_month = 2016_Feb;

Data
myfile.Monthly_Data_for_&new_month;
      Set
myfile.Monthly_Data_for_&old_month;
Run;
```

Using macro variables allows for one change at the top of the program

21: Multi-Threading

- Threaded enabled processes include
  - Base SAS engine indexing
  - Base SAS procedures: SORT, SUMMARY, MEANS, REPORT, TABULATE, and SQL
  - SAS/STAT procedures: GLM, LOESS, REG, ROBUSTREG
  - EM procedures: DMREG, DMINE
  - Eligible RDBMS Access Reads

21: Multi-Threading

- The system options THREADS, NOTHREADS, and CPUCOUNT influence threading throughout SAS
- **Options threads cpucount = <number of cores on your machine>;**
- The system option THREADS is the default in all products so that threading can occur wherever use of threading is possible and performance is improved
- NOTHREADS disables threading in Base SAS

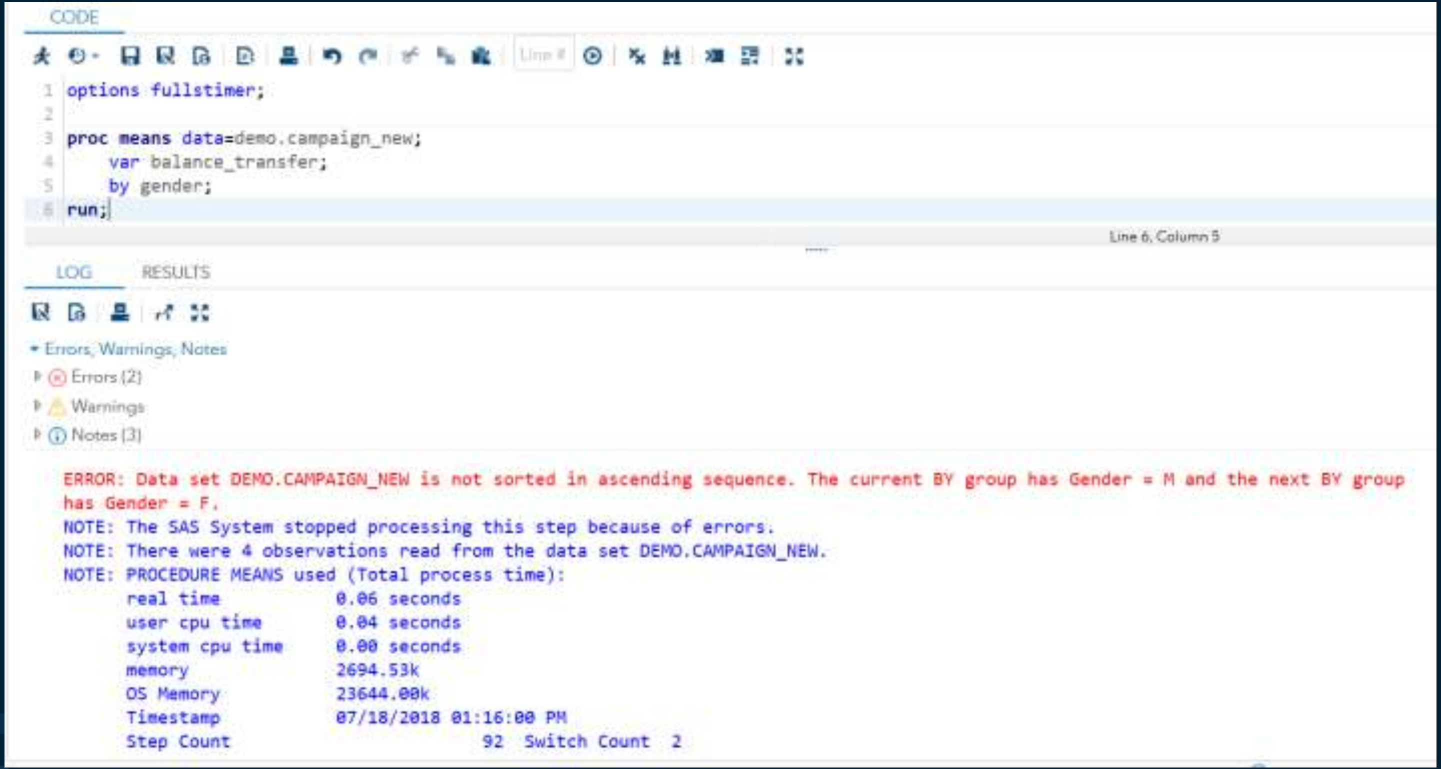
21: Multi-Threading

- Beginning with SAS 9.4: DS2 and FedSQL now enable the data step to be multi-threaded
- This is a topic for another day

22: Multi-Threading

- Use the CLASS statements in procedures that support them to avoid the need for sorting:
  - PROC MEANS
  - PROC SUMMARY
  - PROC UNIVARIATE
  - PROC TABULATE

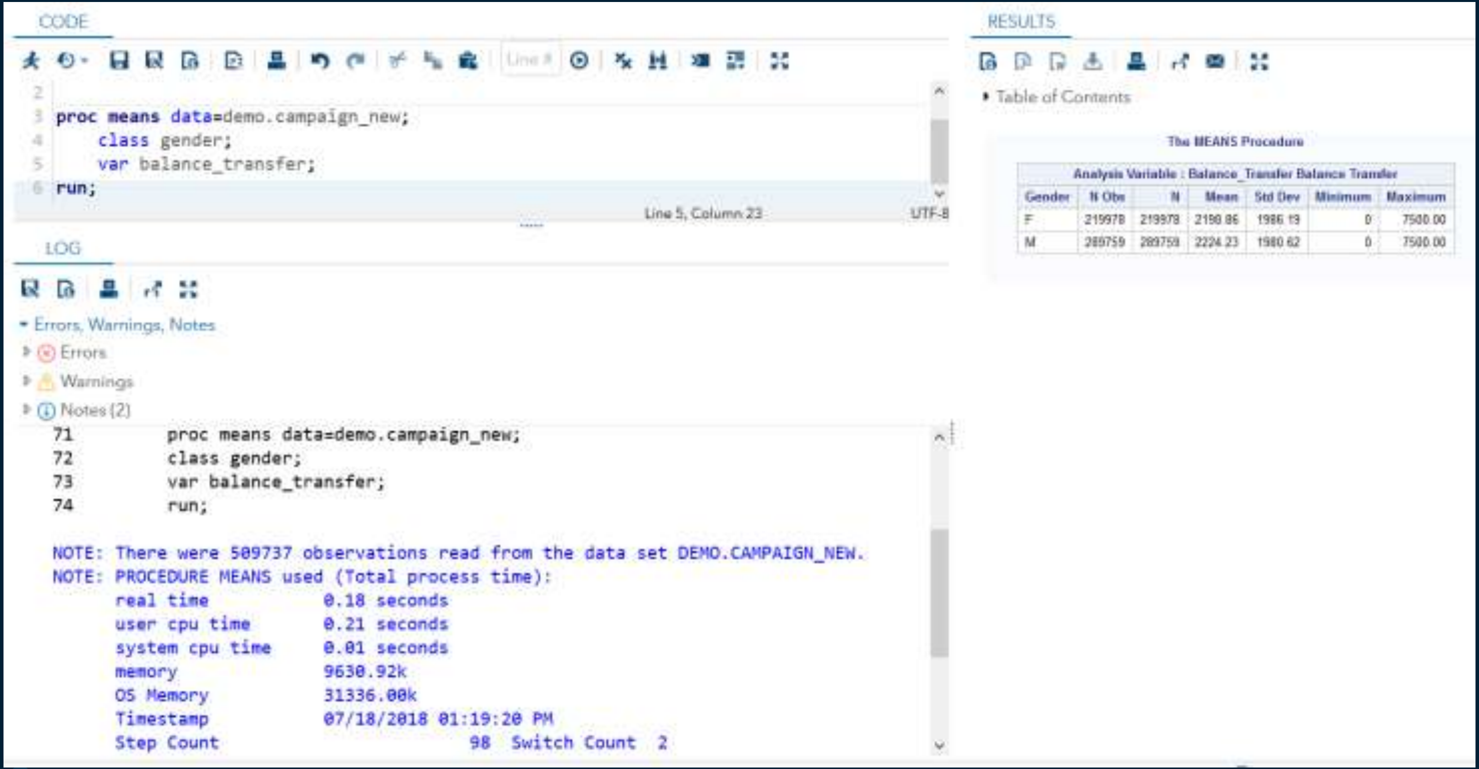
Proc Means



```
options fullstimer;

proc means data=demo.campaign_new;
    var balance_transfer;
    by gender;
run;
```

```
ERROR: Data set DEMO.CAMPAIGN_NEW is not sorted in ascending sequence. The current BY group has Gender = M and the next BY group
has Gender = F.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 4 observations read from the data set DEMO.CAMPAIGN_NEW.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.06 seconds
      user cpu time       0.04 seconds
      system cpu time     0.00 seconds
      memory              2694.53k
      OS Memory           23644.00k
      Timestamp           07/18/2018 01:16:00 PM
      Step Count                        92  Switch Count  2
```

Proc Means



CODE

```
proc means data=demo.campaign_new;
    class gender;
    var balance_transfer;
run;
```

Line 5, Column 23 UTF-8

LOG

Errors, Warnings, Notes
- Errors
- Warnings
- Notes (2)

```
71         proc means data=demo.campaign_new;
72         class gender;
73         var balance_transfer;
74         run;

NOTE: There were 509737 observations read from the data set DEMO.CAMPAIGN_NEW.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.18 seconds
      user cpu time       0.21 seconds
      system cpu time     0.01 seconds
      memory              9630.92k
      OS Memory           31336.00k
      Timestamp           07/18/2018 01:19:20 PM
      Step Count                        98  Switch Count  2
```

RESULTS

Table of Contents

The MEANS Procedure

Analysis Variable : Balance_Transfer Balance Transfer						
Gender	N Obs	N	Mean	Std Dev	Minimum	Maximum
F	219978	219978	2190.86	1986.19	0	7500.00
M	289759	289759	2224.23	1980.62	0	7500.00

23. Make things easier for yourself: efficiency also means working smarter!

- Save code and reuse it later!
- Collaborate with your co-workers to share tips and suggestions
- Meet regularly to share ideas
- Some ways SAS code fosters reusability:
  - Macro libraries
  - Stored processes – can be registered using SAS Management Console, not just Enterprise Guide
  - User-written functions and procedures
  - Repositories in SAS Studio

24. If you are using Enterprise Guide

- Auto-complete and hover help including table and column names
- Program history including comparison
- Smart highlighting
- System options viewer
- Macro Variable viewer
- Catalog and Format Viewer
- Data step debugger (7.13)
- Log Summary
- View/Use code *behind* the Tasks including code templates

24. If you are using SAS Studio

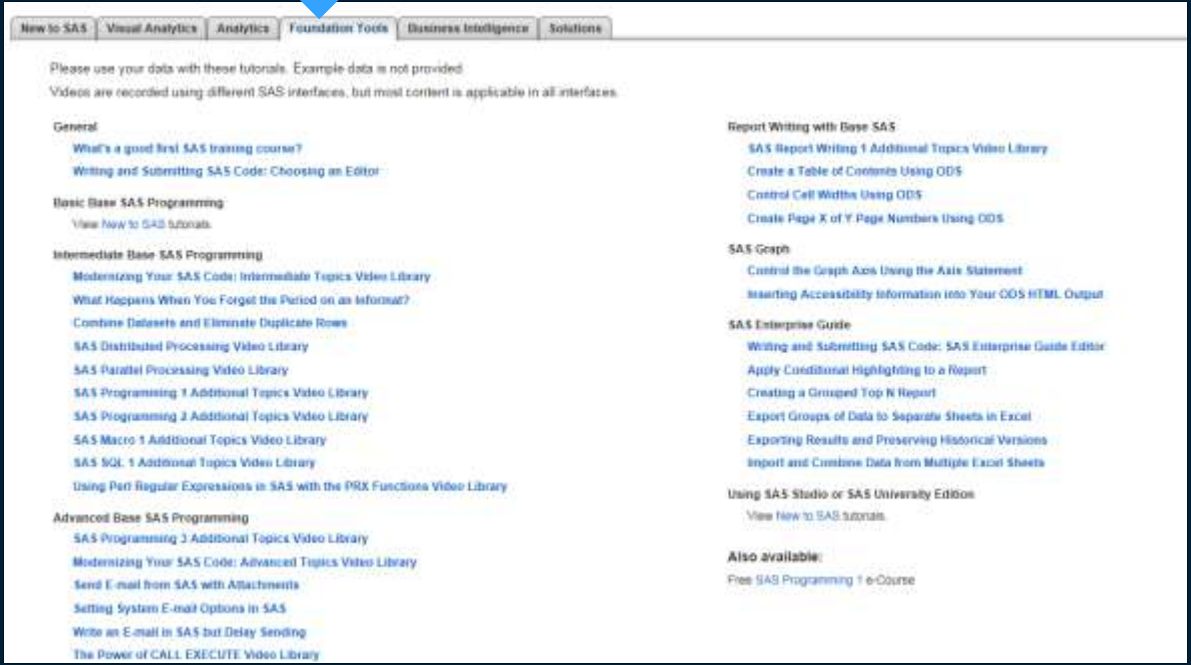
- Auto-complete and hover help including table and column names
- Program history
- Log Summary
- View/Use code *behind* the Tasks including code templates
- Code Snippets
- Custom Tasks

Resources

- http://support.sas.com
- SAS Training
- Local and regional users groups
- SAS Communities
- Your Customer Account Executive and Success Systems Engineers
- Your co-workers and peers

Online

- SAS Tutorials on Programming
- YouTube Video on SAS Programming (2-hour)



Please use your data with these tutorials. Example data is not provided.

Videos are recorded using different SAS interfaces, but most content is applicable in all interfaces.

General
- What's a good first SAS training course?
- Writing and Submitting SAS Code: Choosing an Editor

Basic Base SAS Programming
- View New to SAS tutorials.

Intermediate Base SAS Programming
- Modernizing Your SAS Code: Intermediate Topics Video Library
- What Happens When You Forget the Period on an Informat?
- Combine Datasets and Eliminate Duplicate Rows
- SAS Distributed Processing Video Library
- SAS Parallel Processing Video Library
- SAS Programming 1 Additional Topics Video Library
- SAS Programming 2 Additional Topics Video Library
- SAS Macro 1 Additional Topics Video Library
- SAS SQL 1 Additional Topics Video Library
- Using Perl Regular Expressions in SAS with the PRX Functions Video Library

Advanced Base SAS Programming
- SAS Programming 3 Additional Topics Video Library
- Modernizing Your SAS Code: Advanced Topics Video Library
- Send E-mail from SAS with Attachments
- Setting System E-mail Options in SAS
- Write an E-mail in SAS but Delay Sending
- The Power of CALL EXECUTE Video Library

Report Writing with Base SAS
- SAS Report Writing 1 Additional Topics Video Library
- Create a Table of Contents Using ODS
- Control Cell Widths Using ODS
- Create Page X of Y Page Numbers Using ODS

SAS Graph
- Control the Graph Axis Using the Axis Statement
- Inserting Accessibility Information into Your ODS HTML Output

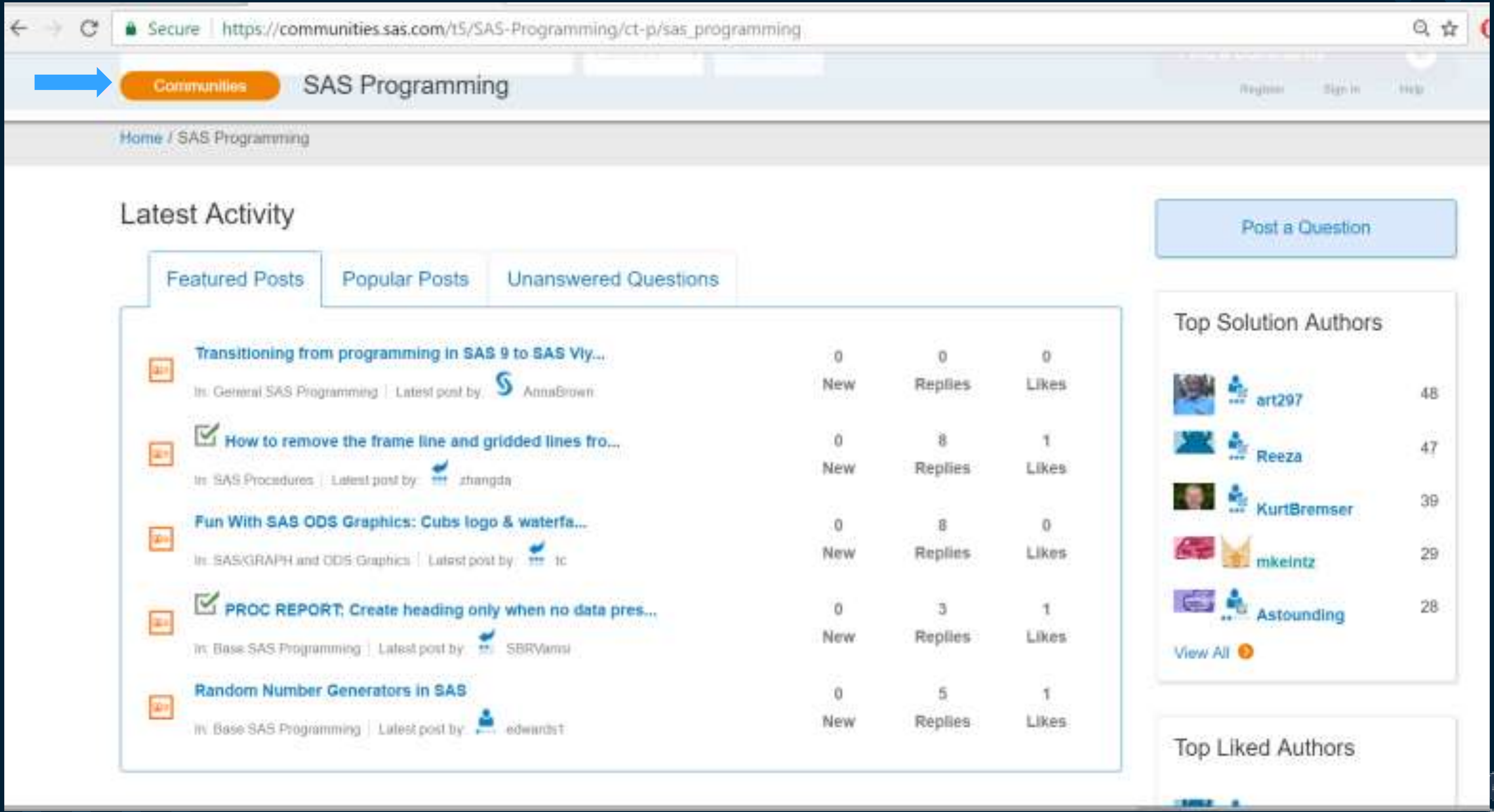
SAS Enterprise Guide
- Writing and Submitting SAS Code: SAS Enterprise Guide Editor
- Apply Conditional Highlighting to a Report
- Creating a Grouped Top N Report
- Export Groups of Data to Separate Sheets in Excel
- Exporting Results and Preserving Historical Versions
- Import and Combine Data from Multiple Excel Sheets

Using SAS Studio or SAS University Edition
- View New to SAS tutorials.

Also available:

Free SAS Programming 1 e-Course

Online


https://communities.sas.com/t5/SAS-Programming/ct-p/sas_programming

Communities SAS Programming

Register Sign In Help

Home / SAS Programming

Latest Activity

Post a Question

Featured Posts | Popular Posts | Unanswered Questions

Transitioning from programming in SAS 9 to SAS Viy... In: General SAS Programming | Latest post by AnnaBrown	0 New	0 Replies	0 Likes
✓ How to remove the frame line and gridded lines fro... In: SAS Procedures | Latest post by zhangda	0 New	8 Replies	1 Likes
Fun With SAS ODS Graphics: Cubs logo & waterfa... In: SAS/GRAPH and ODS Graphics | Latest post by tc	0 New	8 Replies	0 Likes
✓ PROC REPORT: Create heading only when no data pres... In: Base SAS Programming | Latest post by SBRVamsi	0 New	3 Replies	1 Likes
Random Number Generators in SAS In: Base SAS Programming | Latest post by edwards1	0 New	5 Replies	1 Likes

Top Solution Authors

art297	48
Reeza	47
KurtBremser	39
mkeintz	29
Astounding	28

View All

Top Liked Authors

SAS Global Forum Papers

- Leave Your Bad Code Behind: 50 Ways to Make Your SAS® Code Execute More Efficiently William E Benjamin Jr, Owl Computer Consultancy, LLC
- SAS® Shorts: Valuable Tips for Everyday Programming Jeff McCartney and Raymond Hu, Social and Scientific Systems, Inc., Bethesda, MD
- Productivity Tips for SAS® Enterprise Guide® Users Jennifer First and Steven First, Systems Seminar Consultants, Madison, WI, United States
- Tips and Techniques for the SAS® Programmer Helen Carey, Carey Consulting, Kaneohe, HI, Ginger Carey, Carey Consulting, Kaneohe, HI

Videos for topics A-Z

- SAS Software YouTube Channel
  - http://www.youtube.com/user/SASsoftware?feature=watch

Thank you for your time and for using SAS®!

sas.com

Explore Helpful Resources

[Ask the Expert](#)
View other user webinars that provide insights into using SAS products to make your job easier.

[FREE Training](#)
Learn from home – free for 30 days. Get software labs to practice and online support if needed.

[SAS Support Communities](#)
Ask questions, get answers and share insights with SAS users.

[SAS Analytics Explorers](#)
An exclusive platform to collaborate, learn and share your expertise. Gain access to a diverse network to advance your career. Special rewards and recognition exclusively for SAS users.

[SAS Users YouTube Channel](#)
A plethora of videos on hundreds of topics, just for SAS users.

[Newsletters](#)
Get the latest SAS news plus tips, tricks and more.

[Users Groups](#)
Meet local SAS users, network and exchange ideas – virtually.

[SAS Profile](#)
If you haven't already done so, create your SAS Profile to access free training, SAS Support Communities, technical support, software downloads, newsletters and more.