SESUG Paper 138-2020

Code Generating Code: An alternative to Macros

David B. Horvath, MS, CCP

ABSTRACT

While the macro capability within SAS® is very powerful, there are times that the complexity of the code you want to generate make it easier to directly generate code rather than use macros. Some of the advantages and disadvantages of each approach will be discussed as well as examples on how to implement. Unfortunately, the logging of macro-generated code is difficult to read (including decoupling of Timer statistics). While directly generating code allows the use of all SAS capabilities, it does add code complexity (but easier to read in the log). Direct generated code also allows you to split execution environment -- interactive code generation while execution takes place in a background/batch/operating system command line mode.

INTRODUCTION

SAS has many powerful features – the macro processer which allows you to generate (or edit) code while your program is running – on the fly. As powerful as macro is, it has a few limitations:

- CARDS or DATALINES are not allowed
- When there are multiple steps within a macro, the notes appear at the bottom. This has been corrected in newer versions but for many years this was the one that bothered me the most.
- It can be difficult to code because it is a different language
- You do not have all the functionality of the main SAS language

One technique that I will use when facing complex code is the use of generated code.

Put simply, you write SAS code that creates the statements you want to execute, they are written to a file, and that file is brought into your program with the `%include` statement.

I have a larger example that creates a few datasets using plain code, macro, and finally generated code. These are fairly trivial examples. Then, using the metadata tables, determine available data and then generate code to modify those datasets. That last example is a situation where generated code comes in handy. I've used this technique to communize years of data, organized into monthly libraries, where formats have changed over those years into singular tables for migration to other storage media. For one dataset, I might do the work by hand. But when you have hundreds over seven years, you don't want to type that much.

CREATING THE TABLE IN REGULAR CODE

The first dataset is created using plain old regular code. To create multiple files, you would have to replicate the code with different months as needed.

```
options source source2 fullstimer merror mprint mprintnest symbolgen;
libname sas_data "/folders/myshortcuts/sas_data";
libname myfolder "/folders/myfolders";

%let month=202009;
data sas_data.file_202009;
   infile cards dlm=",";
   third=&month.;
   input first second $ ;
   output;
cards;
1,list,202009
2,list2,202009
3,list3,202009
run;

title "Regular file creation for 202009";
proc print data=sas_data.file_202009;
run;
```

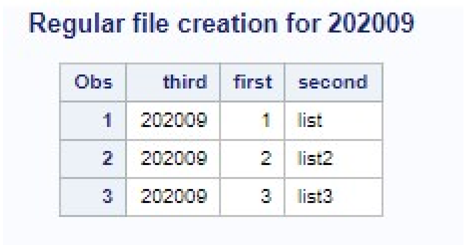
The options and libnames apply throughout all examples. All examples were created using the SAS University Edition.

Figure 1 Normal Code Log Output

```
 77         %let month=202009;
 78         data sas_data.file_202009;
 79            infile cards dlm=",";
 80            third=&month.;
 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 81
 82            input first second $ ;
 83            output;
 84         cards;

 NOTE: The data set SAS_DATA.FILE_202009 has 3 observations and 3 variables.
 NOTE: DATA statement used (Total process time):
       real time           0.09 seconds
       user cpu time       0.01 seconds
       system cpu time     0.01 seconds
       memory              419.65k
       OS Memory           30112.00k
       Timestamp           09/28/2020 04:21:18 PM
       Step Count                        52  Switch Count  1
       Page Faults                       0
       Page Reclaims                     69
       Page Swaps                        0
       Voluntary Context Switches        92
       Involuntary Context Switches      7
       Block Input Operations            0
       Block Output Operations           0


 88         run;
 89
 90         title "Regular file creation for 202009";
 91         proc print data=sas_data.file_202009;
 92         run;

 NOTE: There were 3 observations read from the data set SAS_DATA.FILE_202009.
 NOTE: PROCEDURE PRINT used (Total process time):
       real time           0.18 seconds
       user cpu time       0.16 seconds
       system cpu time     0.01 seconds
       memory              2229.18k
       OS Memory           31392.00k
       Timestamp           09/28/2020 04:21:19 PM
       Step Count                        53  Switch Count  0
       Page Faults                       0
       Page Reclaims                     353
       Page Swaps                        0
       Voluntary Context Switches        17
       Involuntary Context Switches      2
       Block Input Operations            0
       Block Output Operations           8
```

Note that there are NOTEs after each executed step (data and proc). With macros, those appear at the end of the macro execution.

Figure 2 Resulting Output



Regular file creation for 202009

Obs	third	first	second
1	202009	1	list
2	202009	2	list2
3	202009	3	list3

MACRO CODE

There are two components to a macro – the definition and invocation. Because the CARDS statement is not allowed within a macro, the invocation is more complicated and the proc print has to be hand coded.

```
%macro filebuild(month);
data _null_; /* extra step to show notes */
   put "Text string";
run;

data sas_data.mfile_&month.;
   infile cards dlm=",";
   third=&month.;

   input first second $ ;
   output;

%mend;

/* You can't put cards statement in a macro
https://communities.sas.com/t5/SAS-Programming/DATALINES-statement-inside-a-
macro/td-p/37960
*/
%filebuild(202009);
cards;
1,list,202009
2,list2,202009
3,list3,202009
;
run;
```

```
title "Macro file creation for 202009";
proc print data=sas_data.mfile_202009;
run;
```

In this example, a second data step was added to show that NOTEs now appear after each step. Prior versions were much harder to read because they would not appear until after all the MPRINT results.

Figure 3 Macro code definition and invocation

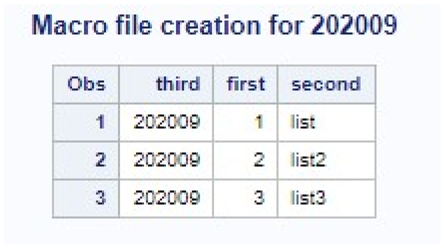
```
 95         %macro filebuild(month);
 96         data _null_; /* extra step to show lack of notes */
 97            put "Text string";
 98         run;
 99
 100        data sas_data.mfile_&month.;
 101           infile cards dlm=",";
 102           third=&month.;
 103
 104           input first second $ ;
 105           output;
 106
 107        %mend;
 108
 109        /* You can't put cards statement in a macro
 110        https://communities.sas.com/t5/SAS-Programming/DATALINES-
statement-inside-a-macro/td-p/37960
 111        */
 112        %filebuild(202009);
 MPRINT(FILEBUILD):   data _null_;
 MPRINT(FILEBUILD):   put "Text string";
 MPRINT(FILEBUILD):   run;

 Text string
 NOTE: DATA statement used (Total process time):
       real time           0.00 seconds
       user cpu time       0.00 seconds
       system cpu time     0.00 seconds
       memory              460.96k
       OS Memory           31904.00k
       Timestamp           09/28/2020 04:32:39 PM
       Step Count                        169  Switch Count  0
       Page Faults                       0
       Page Reclaims                     73
       Page Swaps                        0
       Voluntary Context Switches        0
       Involuntary Context Switches      1
       Block Input Operations            0
       Block Output Operations           0


 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 MPRINT(FILEBUILD):   data sas_data.mfile_202009;
 MPRINT(FILEBUILD):   infile cards dlm=",";
 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 MPRINT(FILEBUILD):   third=202009;
 MPRINT(FILEBUILD):   input first second $ ;
 MPRINT(FILEBUILD):   output;
```

```
113       cards;

NOTE: The data set SAS_DATA.MFILE_202009 has 3 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      user cpu time       0.01 seconds
      system cpu time     0.01 seconds
      memory              660.78k
      OS Memory           31904.00k
      Timestamp           09/28/2020 04:32:39 PM
      Step Count                        170  Switch Count  1
      Page Faults                       0
      Page Reclaims                     42
      Page Swaps                        0
      Voluntary Context Switches        95
      Involuntary Context Switches      3
      Block Input Operations            0
      Block Output Operations           0


117       ;
118       run;
119
120       title "Macro file creation for 202009";
121       proc print data=sas_data.mfile_202009;
122       run;
```

```
 NOTE: There were 3 observations read from the data set
SAS_DATA.MFILE_202009.
 NOTE: PROCEDURE PRINT used (Total process time):
       real time           0.05 seconds
       user cpu time       0.05 seconds
       system cpu time     0.00 seconds
       memory              717.12k
       OS Memory           31904.00k
       Timestamp           09/28/2020 04:32:39 PM
       Step Count                        171  Switch Count  0
       Page Faults                       0
       Page Reclaims                     59
       Page Swaps                        0
       Voluntary Context Switches        16
       Involuntary Context Switches      2
       Block Input Operations            0
       Block Output Operations           0
```

The output is the same as expected. The advantage of a macro approach to this code would be having to write less code. By invoking the macro multiple times, you save typing (development and maintenance) time. For complex code, once you debug it, you can keep reusing it.

Figure 4 Resulting Output


Macro file creation for 202009

Obs	third	first	second
1	202009	1	list
2	202009	2	list2
3	202009	3	list3

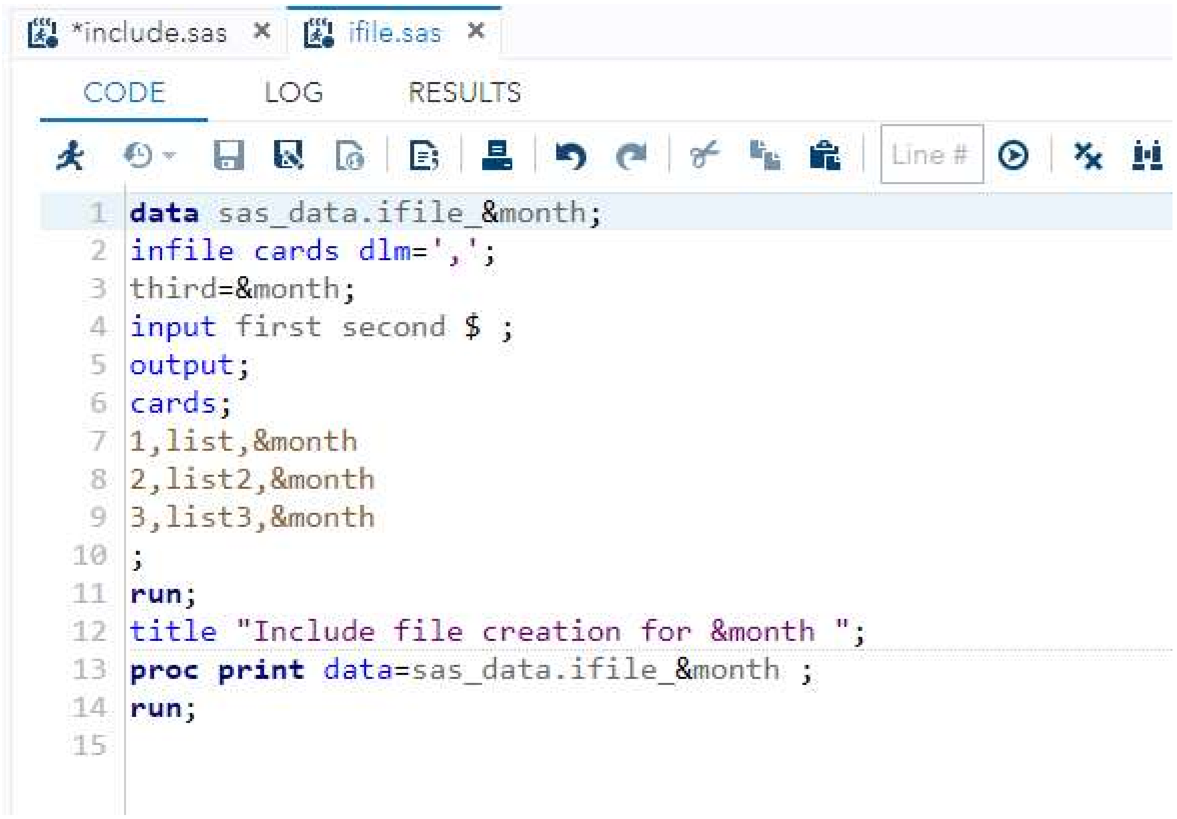
GENERATING CODE

With generated code, your data step writes the statements into a text file that is later included into the program at a later time. In this case, &Month within the base data step is not resolved – it will be resolved when the generated code is included.

```
%let month=;
data _null_;
   file "/folders/myshortcuts/sas_data/ifile.sas";
   put 'data sas_data.ifile_&month;';
   put "infile cards dlm=',';";
   put 'third=&month;';
   put "input first second $ ;";
   put "output;";
   put "cards;";
   put '1,list,&month';
   put '2,list2,&month';
   put '3,list3,&month';
   put ";";
   put "run;";
   put 'title "Include file creation for &month ";';
   put 'proc print data=sas_data.ifile_&month ;';
   put "run;";
run;

/* But you can put cards in an include file
*/
%let month=202009;
%include "/folders/myshortcuts/sas_data/ifile.sas";
%let month=202010;
%include "/folders/myshortcuts/sas_data/ifile.sas";
```

The resulting code (stored in ifile.sas) is editable if necessary. In the following Figure, it was opened within Studio.

Figure 5 Resulting Code



```
data sas_data.ifile_&month;
infile cards dlm=',';
third=&month;
input first second $ ;
output;
cards;
1,list,&month
2,list2,&month
3,list3,&month
;
run;
title "Include file creation for &month ";
proc print data=sas_data.ifile_&month ;
run;
```

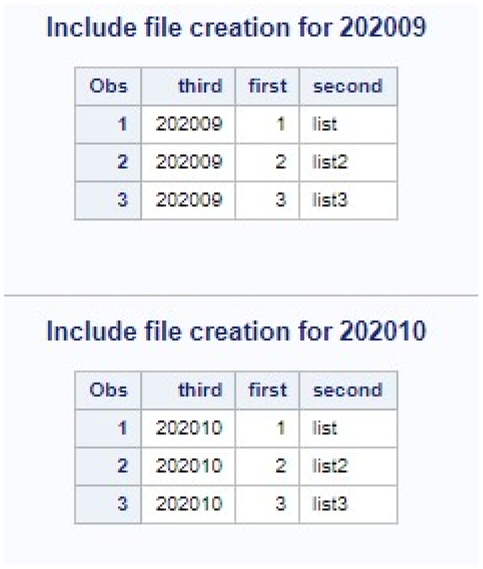
On every invocation (every %include) you have full macro capabilities as well as log reporting (NOTES):

Figure 6 Execution of Generated Code

```
146         %let month=;
147         data _null_;
148            file "/folders/myshortcuts/sas_data/ifile.sas";
149            put 'data sas_data.ifile_&month;';
150            put "infile cards dlm=',';";
151            put 'third=&month;';
152            put "input first second $ ;";
153            put "output;";
154            put "cards;";
155            put '1,list,&month';
156            put '2,list2,&month';
157            put '3,list3,&month';
158            put ";";
159            put "run;";
160            put 'title "Include file creation for &month ";';
161            put 'proc print data=sas_data.ifile_&month ;';
162            put "run;";
163         run;
 NOTE: The file "/folders/myshortcuts/sas_data/ifile.sas" is:
      Filename=/folders/myshortcuts/sas_data/ifile.sas,
      Owner Name=root,Group Name=vboxsf,
      Access Permission=-rwxrwx---,
      Last Modified=28Sep2020:12:32:39
 NOTE: 14 records were written to the file
"/folders/myshortcuts/sas_data/ifile.sas".
       The minimum record length was 1.
       The maximum record length was 42.
 NOTE: DATA statement used (Total process time):
       real time           0.02 seconds
       user cpu time       0.01 seconds
       system cpu time     0.00 seconds
       memory              489.53k
       OS Memory           31904.00k
       Timestamp           09/28/2020 04:32:39 PM
       Step Count                        172  Switch Count  0
       Page Faults                       0
       Page Reclaims                     32
       Page Swaps                        0
       Voluntary Context Switches        23
       Involuntary Context Switches      0
       Block Input Operations            0
       Block Output Operations           0


 164
 165
 166        /* But you can put cards in an include file
 167        */
 168        %let month=202009;
 169        %include "/folders/myshortcuts/sas_data/ifile.sas";
 NOTE: %INCLUDE (level 1) file /folders/myshortcuts/sas_data/ifile.sas is
file /folders/myshortcuts/sas_data/ifile.sas.
 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 170       +data sas_data.ifile_&month;
 171       +infile cards dlm=',';
 172       +third=&month;
 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 173       +input first second $ ;
 174       +output;
 175       +cards;

 NOTE: The data set SAS_DATA.IFILE_202009 has 3 observations and 3 variables.
 NOTE: DATA statement used (Total process time):
       real time           0.06 seconds
       user cpu time       0.00 seconds
       system cpu time     0.01 seconds
       memory              555.59k
       OS Memory           32160.00k
       Timestamp           09/28/2020 04:32:40 PM
       Step Count                        173  Switch Count  1
       Page Faults                       0
       Page Reclaims                     33
       Page Swaps                        0
       Voluntary Context Switches        93
       Involuntary Context Switches      5
       Block Input Operations            0
       Block Output Operations           0
 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 179       +;
 180       +run;
 181       +title "Include file creation for &month ";
 182       +proc print data=sas_data.ifile_&month ;
 SYMBOLGEN:  Macro variable MONTH resolves to 202009
 183       +run;

 NOTE: There were 3 observations read from the data set
SAS_DATA.IFILE_202009.
 NOTE: PROCEDURE PRINT used (Total process time):
       real time           0.06 seconds
       user cpu time       0.05 seconds
       system cpu time     0.00 seconds
       memory              553.93k
       OS Memory           32160.00k
       Timestamp           09/28/2020 04:32:40 PM
       Step Count                        174  Switch Count  0
       Page Faults                       0
       Page Reclaims                     51
       Page Swaps                        0
       Voluntary Context Switches        17
       Involuntary Context Switches      9
       Block Input Operations            0
       Block Output Operations           0


 NOTE: %INCLUDE (level 1) ending.
 184        %let month=202010;
 185        %include "/folders/myshortcuts/sas_data/ifile.sas";
 NOTE: %INCLUDE (level 1) file /folders/myshortcuts/sas_data/ifile.sas is
file /folders/myshortcuts/sas_data/ifile.sas.
 SYMBOLGEN:  Macro variable MONTH resolves to 202010
 186       +data sas_data.ifile_&month;
 187       +infile cards dlm=',';
 188       +third=&month;
 SYMBOLGEN:  Macro variable MONTH resolves to 202010
 189       +input first second $ ;
 190       +output;
 191       +cards;

 NOTE: The data set SAS_DATA.IFILE_202010 has 3 observations and 3 variables.
 NOTE: DATA statement used (Total process time):
       real time           0.07 seconds
       user cpu time       0.01 seconds
       system cpu time     0.01 seconds
       memory              555.78k
       OS Memory           32160.00k
       Timestamp           09/28/2020 04:32:40 PM
       Step Count                        175  Switch Count  1
       Page Faults                       0
       Page Reclaims                     30
       Page Swaps                        0
       Voluntary Context Switches        96
       Involuntary Context Switches      6
       Block Input Operations            0
       Block Output Operations           0


 SYMBOLGEN:  Macro variable MONTH resolves to 202010
 195       +;
 196       +run;
 197       +title "Include file creation for &month ";
 198       +proc print data=sas_data.ifile_&month ;
 SYMBOLGEN:  Macro variable MONTH resolves to 202010
 199       +run;

 NOTE: There were 3 observations read from the data set
SAS_DATA.IFILE_202010.
 NOTE: PROCEDURE PRINT used (Total process time):
       real time           0.05 seconds
       user cpu time       0.04 seconds
       system cpu time     0.00 seconds
       memory              611.18k
       OS Memory           32416.00k
       Timestamp           09/28/2020 04:32:40 PM
       Step Count                        176  Switch Count  0
       Page Faults                       0
       Page Reclaims                     33
       Page Swaps                        0
       Voluntary Context Switches        16
       Involuntary Context Switches      6
       Block Input Operations            0
       Block Output Operations           0


 NOTE: %INCLUDE (level 1) ending.
 200
```

And, as expected, the output is the same as the prior examples.

Figure 7 Resulting Output



Include file creation for 202009

Obs	third	first	second
1	202009	1	list
2	202009	2	list2
3	202009	3	list3

Include file creation for 202010

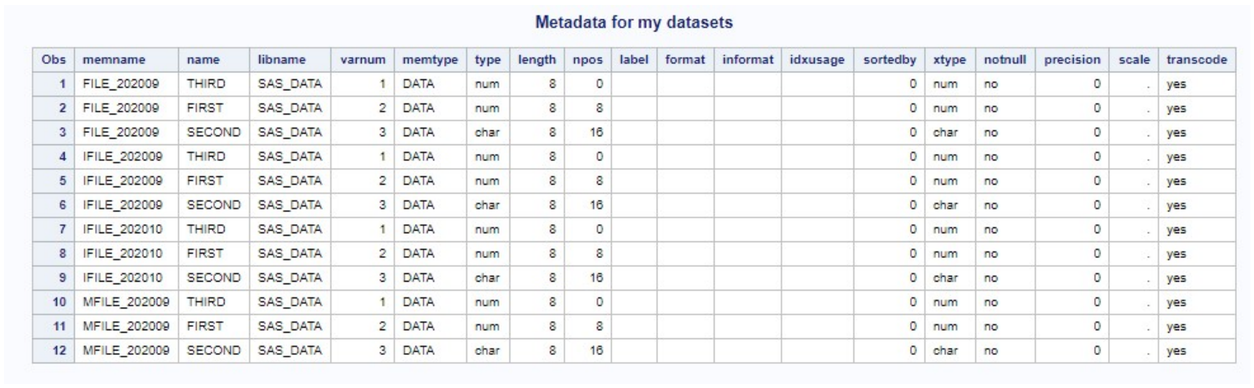
Obs	third	first	second
1	202010	1	list
2	202010	2	list2
3	202010	3	list3

A MORE COMPLEX EXAMPLE

These examples have been very simple. A better example is where I need to execute large blocks of code based on datasets in a library. The first thing I need to do is get the metadata I need for my processing. Since I only care about the examples created earlier in the program, I limit the dataset names in the where clause:

```
proc sql feedback stimer;
      create table sas_data.ddl as select upcase(memname) as memname,
            upcase(name) as name, upcase(libname) as libname, varnum,
            memtype, type, length, npos, label, format, informat, idxusage,
            sortedby, xtype, notnull, precision, scale, transcode
            from sashelp.vcolumn
            where upcase(memname) like '%FILE_%';

title "Metadata for my datasets";
proc print data=sas_data.ddl (obs=100);
run;
```

Figure 8 Metadata Values for these examples



Metadata for my datasets

Obs	memname	name	libname	varnum	memtype	type	length	npos	label	format	informat	idxusage	sortedby	xtype	notnull	precision	scale	transcode
1	FILE_202009	THIRD	SAS_DATA	1	DATA	num	8	0					0	num	no	0	.	yes
2	FILE_202009	FIRST	SAS_DATA	2	DATA	num	8	8					0	num	no	0	.	yes
3	FILE_202009	SECOND	SAS_DATA	3	DATA	char	8	16					0	char	no	0	.	yes
4	IFILE_202009	THIRD	SAS_DATA	1	DATA	num	8	0					0	num	no	0	.	yes
5	IFILE_202009	FIRST	SAS_DATA	2	DATA	num	8	8					0	num	no	0	.	yes
6	IFILE_202009	SECOND	SAS_DATA	3	DATA	char	8	16					0	char	no	0	.	yes
7	IFILE_202010	THIRD	SAS_DATA	1	DATA	num	8	0					0	num	no	0	.	yes
8	IFILE_202010	FIRST	SAS_DATA	2	DATA	num	8	8					0	num	no	0	.	yes
9	IFILE_202010	SECOND	SAS_DATA	3	DATA	char	8	16					0	char	no	0	.	yes
10	MFILE_202009	THIRD	SAS_DATA	1	DATA	num	8	0					0	num	no	0	.	yes
11	MFILE_202009	FIRST	SAS_DATA	2	DATA	num	8	8					0	num	no	0	.	yes
12	MFILE_202009	SECOND	SAS_DATA	3	DATA	char	8	16					0	char	no	0	.	yes

You can use this technique on any data source that contains metadata, not just SAS datasets. I've used this technique on XML files as well with the SAS XML engine.

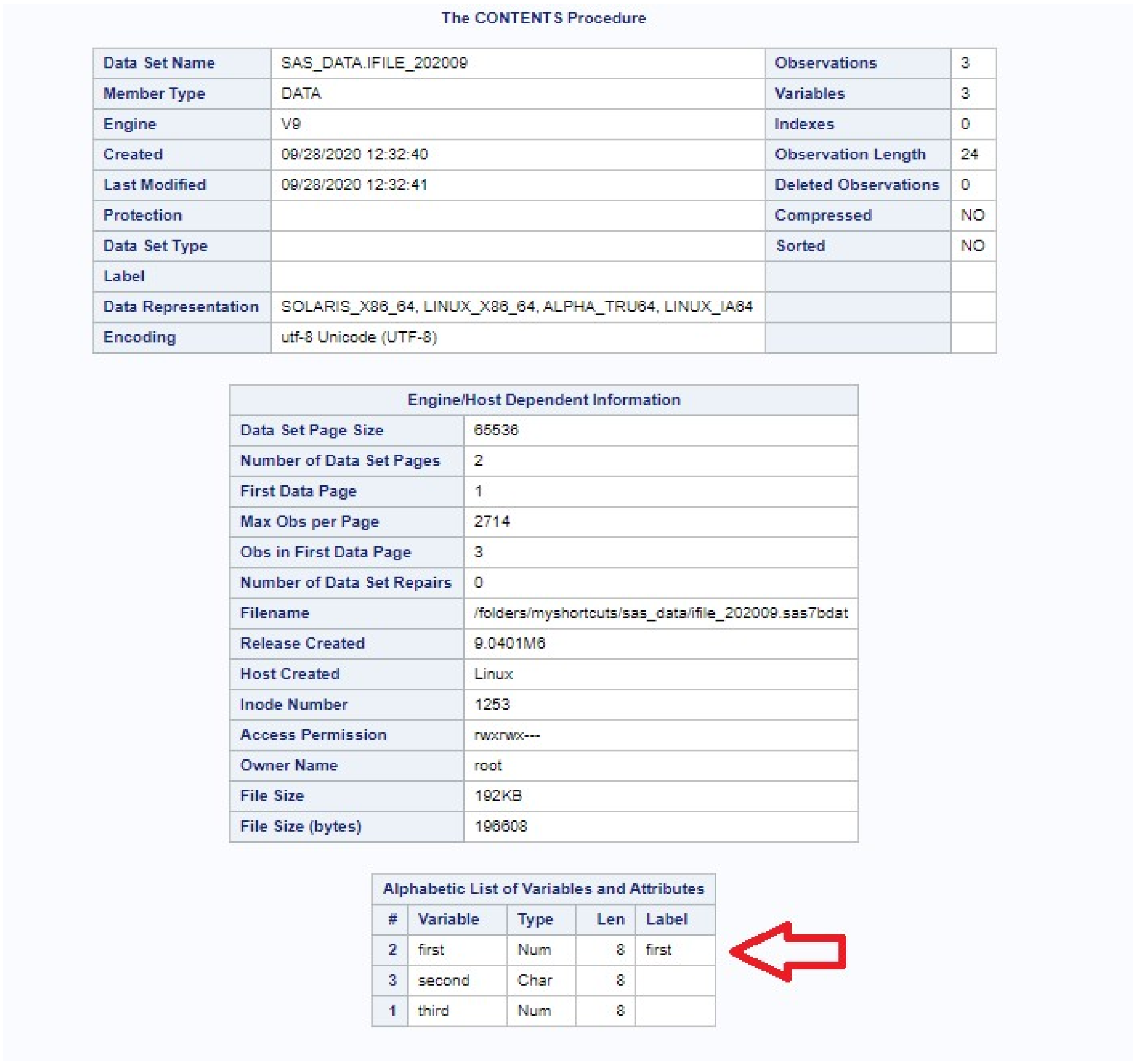
GETTING THE CODE TO WORK

The first thing I want to do is figure out exactly what my code has to do and create an example. I want to add labels to all of these that will be upcase(name). The proc sql alter statement seems to be the easiest (but certainly not only choice). Proc contents shows the change that occurred.

I can keep working on it at this level until I get it right – without worrying about the generation part.

```
proc sql feedback stimer;
   alter table sas_data.ifile_202009 modify first label="first";

Title "labels added to one column, one table";
proc contents data=sas_data.ifile_202009; run;
```

Figure 9 Showing the Alter worked



The CONTENTS Procedure

Data Set Name	SAS_DATA.IFILE_202009	Observations	3
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	09/28/2020 12:32:40	Observation Length	24
Last Modified	09/28/2020 12:32:41	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	2714
Obs in First Data Page	3
Number of Data Set Repairs	0
Filename	/folders/myshortcuts/sas_data/ifile_202009.sas7bdat
Release Created	9.0401M6
Host Created	Linux
Inode Number	1253
Access Permission	rwxrwx---
Owner Name	root
File Size	192KB
File Size (bytes)	196608

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
2	first	Num	8	first
3	second	Char	8	
1	third	Num	8	

Once it is working, it would be commented out since the alteration should be generated by the real code.

GENERATING AND EXECUTING THE RESULTING CODE

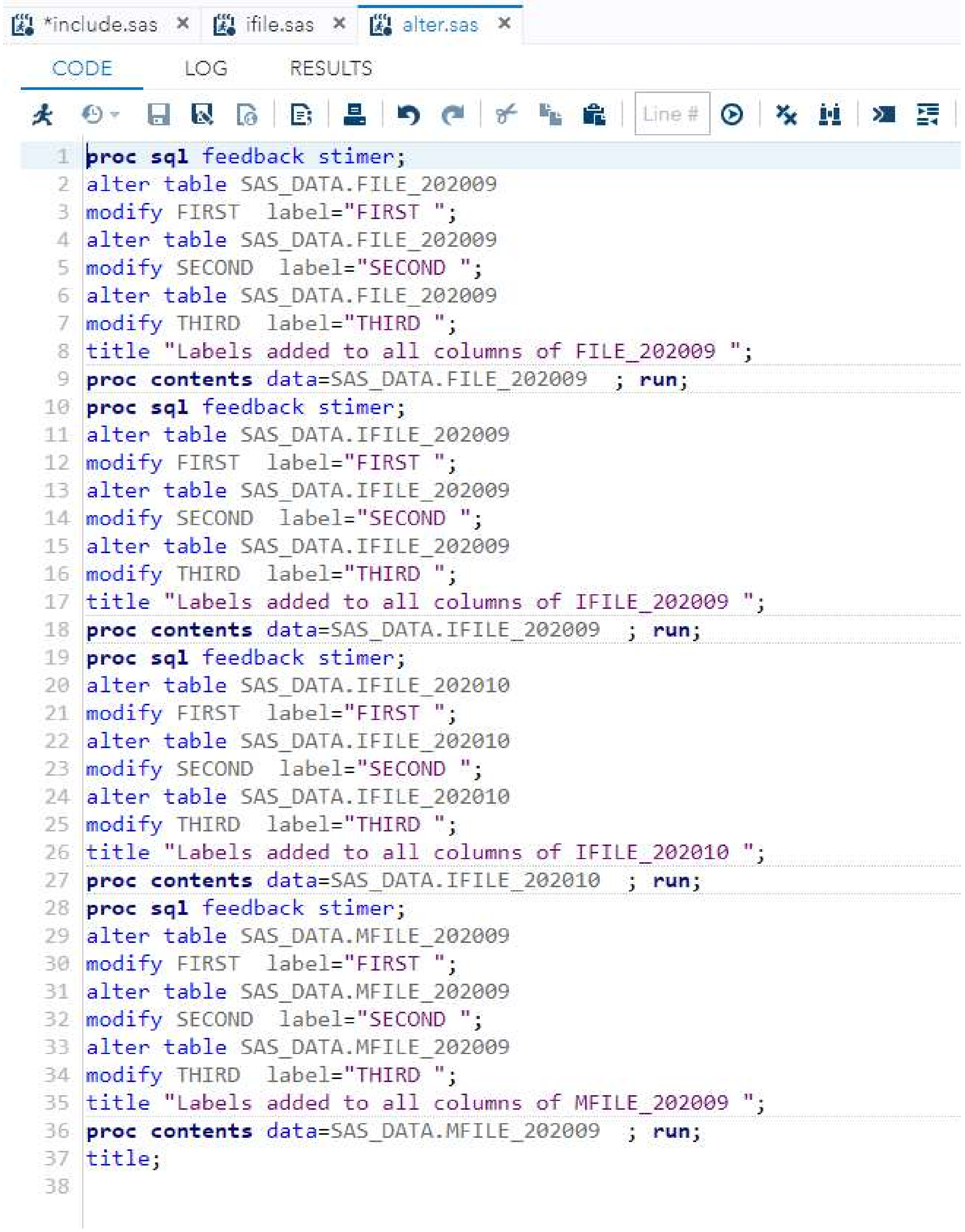
Now, I need to generate statements that look like that for all the fields in the metadata I extracted. Since I want to include proc contents and minimize the generated code, I need some ordering within the data (so I can use first. and last.

```
proc sort data=sas_data.ddl;
   by memname name;
run;

data _null_;
   file "/folders/myshortcuts/sas_data/alter.sas";
   set sas_data.ddl end=EOF;
   length fullname $ 32;
   retain fullname;

   by memname name;
   if first.memname then do;
      fullname=cats(libname,".",memname);
      put 'proc sql feedback stimer;';
   end;

   put "alter table " fullname ;
   put "modify " name ' label="' name '";';

   if last.memname then do;
      put 'title "Labels added to all columns of ' memname '";';
      put "proc contents data=" fullname " ; run;";
   end;

   if EOF then
      put "title;";

run;

%include "/folders/myshortcuts/sas_data/alter.sas";
```

The resulting code, stored in alter.sas is pretty simple. As I already mentioned, these are fairly simple examples – a lot of work to create a little bit of code. If this was the real situation, I'd probably write it once and then copy/paste as needed (or even create a macro that accepts the table name). But when I have hundreds or even thousands of datasets, the upfront development time is readily saved.

Another advantage is that I can edit the generated code file if necessary. For instance, if I did not want to alter a particular dataset that otherwise met the selection criteria, I could delete the statements that reference it.

Figure 10 Generated Code to Alter Datasets



```
proc sql feedback stimer;
alter table SAS_DATA.FILE_202009
modify FIRST  label="FIRST ";
alter table SAS_DATA.FILE_202009
modify SECOND  label="SECOND ";
alter table SAS_DATA.FILE_202009
modify THIRD  label="THIRD ";
title "Labels added to all columns of FILE_202009 ";
proc contents data=SAS_DATA.FILE_202009  ; run;
proc sql feedback stimer;
alter table SAS_DATA.IFILE_202009
modify FIRST  label="FIRST ";
alter table SAS_DATA.IFILE_202009
modify SECOND  label="SECOND ";
alter table SAS_DATA.IFILE_202009
modify THIRD  label="THIRD ";
title "Labels added to all columns of IFILE_202009 ";
proc contents data=SAS_DATA.IFILE_202009  ; run;
proc sql feedback stimer;
alter table SAS_DATA.IFILE_202010
modify FIRST  label="FIRST ";
alter table SAS_DATA.IFILE_202010
modify SECOND  label="SECOND ";
alter table SAS_DATA.IFILE_202010
modify THIRD  label="THIRD ";
title "Labels added to all columns of IFILE_202010 ";
proc contents data=SAS_DATA.IFILE_202010  ; run;
proc sql feedback stimer;
alter table SAS_DATA.MFILE_202009
modify FIRST  label="FIRST ";
alter table SAS_DATA.MFILE_202009
modify SECOND  label="SECOND ";
alter table SAS_DATA.MFILE_202009
modify THIRD  label="THIRD ";
title "Labels added to all columns of MFILE_202009 ";
proc contents data=SAS_DATA.MFILE_202009  ; run;
title;

```

I do include a final title statement so that the “MFILE_202009” title wouldn’t appear on any subsequent outputs in case I forget to set them later on. The log example only includes the first block to reduce space used.

Figure 11 Execution of the Included Code (First Dataset only)

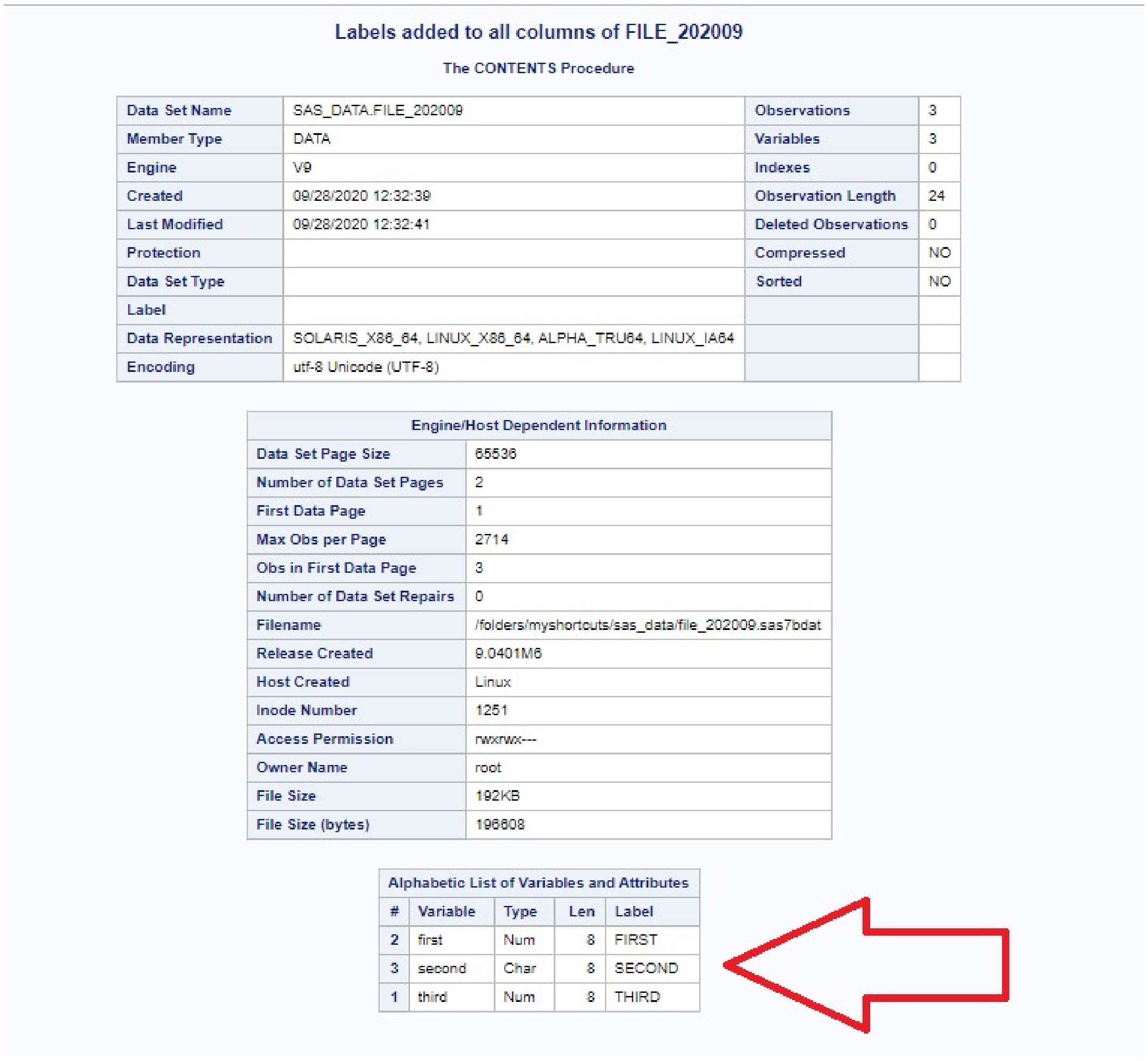
```
 249        %include "/folders/myshortcuts/sas_data/alter.sas";
 NOTE: %INCLUDE (level 1) file /folders/myshortcuts/sas_data/alter.sas is
file /folders/myshortcuts/sas_data/alter.sas.
 250       +proc sql feedback stimer;
 NOTE: SQL Statement used (Total process time):
       real time           0.00 seconds
       user cpu time       0.00 seconds
       system cpu time     0.00 seconds
       memory              12.43k
       OS Memory           33696.00k
       Timestamp           09/28/2020 04:32:41 PM
       Step Count                        183  Switch Count  0
       Page Faults                       0
       Page Reclaims                     3
       Page Swaps                        0
       Voluntary Context Switches        3
       Involuntary Context Switches      1
       Block Input Operations            0
       Block Output Operations           0

 251       +alter table SAS_DATA.FILE_202009
 252       +modify FIRST  label="FIRST ";
 NOTE: Table SAS_DATA.FILE_202009 has been modified, with 3 columns.
 NOTE: SQL Statement used (Total process time):
       real time           0.01 seconds
       user cpu time       0.00 seconds
       system cpu time     0.01 seconds
       memory              238.00k
       OS Memory           33696.00k
       Timestamp           09/28/2020 04:32:41 PM
       Step Count                        183  Switch Count  0
       Page Faults                       0
       Page Reclaims                     14
       Page Swaps                        0
       Voluntary Context Switches        37
       Involuntary Context Switches      0
       Block Input Operations            0
       Block Output Operations           0

 253       +alter table SAS_DATA.FILE_202009
 254       +modify SECOND  label="SECOND ";
 NOTE: Table SAS_DATA.FILE_202009 has been modified, with 3 columns.
 NOTE: SQL Statement used (Total process time):
       real time           0.02 seconds
       user cpu time       0.00 seconds
       system cpu time     0.00 seconds
       memory              228.34k
       OS Memory           33696.00k
       Timestamp           09/28/2020 04:32:41 PM
       Step Count                        183  Switch Count  0
       Page Faults                       0
       Page Reclaims                     0
```

```
      Page Swaps                        0
      Voluntary Context Switches        46
      Involuntary Context Switches      0
      Block Input Operations            0
      Block Output Operations           0

255       +alter table SAS_DATA.FILE_202009
256       +modify THIRD  label="THIRD ";
NOTE: Table SAS_DATA.FILE_202009 has been modified, with 3 columns.
NOTE: SQL Statement used (Total process time):
      real time           0.02 seconds
      user cpu time       0.01 seconds
      system cpu time     0.00 seconds
      memory              225.06k
      OS Memory           33696.00k
      Timestamp           09/28/2020 04:32:41 PM
      Step Count                        183  Switch Count  0
      Page Faults                       0
      Page Reclaims                     1
      Page Swaps                        0
      Voluntary Context Switches        43
      Involuntary Context Switches      1
      Block Input Operations            0
      Block Output Operations           0

257       +title "Labels added to all columns of FILE_202009 ";
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
      memory              14.21k
      OS Memory           33696.00k
      Timestamp           09/28/2020 04:32:41 PM
      Step Count                        183  Switch Count  1
      Page Faults                       0
      Page Reclaims                     3
      Page Swaps                        0
      Voluntary Context Switches        6
      Involuntary Context Switches      5
      Block Input Operations            0
      Block Output Operations           0


258       +proc contents data=SAS_DATA.FILE_202009  ; run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           0.17 seconds
      user cpu time       0.14 seconds
      system cpu time     0.01 seconds
      memory              857.90k
      OS Memory           33696.00k
      Timestamp           09/28/2020 04:32:41 PM
      Step Count                        184  Switch Count  0
      Page Faults                       0
      Page Reclaims                     42
      Page Swaps                        0
      Voluntary Context Switches        42
```

```
        Involuntary Context Switches      2
        Block Input Operations            0
        Block Output Operations           8
```

And finally, the resulting proc contents shows that labels have been attached to these columns (again, only the first dataset is included).

Figure 12 Result of Generated Code



Labels added to all columns of FILE_202009

The CONTENTS Procedure

Data Set Name	SAS_DATA.FILE_202009	Observations	3
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	09/28/2020 12:32:39	Observation Length	24
Last Modified	09/28/2020 12:32:41	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	2714
Obs in First Data Page	3
Number of Data Set Repairs	0
Filename	/folders/myshortcuts/sas_data/file_202009.sas7bdat
Release Created	9.0401M6
Host Created	Linux
Inode Number	1251
Access Permission	rwxrwx---
Owner Name	root
File Size	192KB
File Size (bytes)	196608

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
2	first	Num	8	FIRST
3	second	Char	8	SECOND
1	third	Num	8	THIRD

CONCLUSION

You have many choices in the way you decide to solve a problem within SAS. Even the choice of using proc datasets or proc sql to alter a dataset. In addition to copy/paste code replication or even macros, you can programmatically generate code that later gets executed.

ACKNOWLEDGMENTS

I want to thank the organizers of this great conference – I would have preferred to present in person but certainly understand the need for the conference to be paper-only. I will miss the interaction with other speakers and attendees. I also want to thank my employer for their past willingness to allow me to expand my horizons by attending SESUG. Lastly but certainly not least, I want to thank my spouse Mary who doesn't complain when I spend so much time at the keyboard working on documents like this.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David B. Horvath, CCP
+1-610-859-8826
dhorvath@cobs.com
http://www.cobs.com
LinkedIn: https://www.linkedin.com/in/dbhorvath/