**PhUSE 2015**

**Paper CC07**

# Slim Down Your Data

Mickael Borne, 4Clinics, Montpellier, France

## ABSTRACT

We developed a package of SAS® macro-programs that was developed to automatically resize character variables of all SAS datasets in a project directory. The allotted character variable length/size can significantly impact the size of the corresponding SAS dataset file. Adjusting this length to the maximum length observed is recommended for electronic submission of clinical trial data. The context of the package's development i.e. electronic clinical data submission and the relationship between SAS dataset size and the variable length declaration are discussed. Programming topics such as SAS dictionary tables, SAS component language (SCL) functions, and the Call Execute statement are also covered.

## INTRODUCTION

In submitting clinical data to regulatory authorities such as the US Food and Drug Administration (FDA), the file size of the SAS datasets often need to be substantially reduced, or squeezed, to be within the maximum size permitted. SAS7BDAT files (i.e., the physical files containing the SAS dataset) must be less than 1 GB in size, and it is recommended that the character variable lengths fit the longest value observed.

Here, I present a package of SAS macros that together can be used to squeeze all the datasets of a database. In addition to describing each component in this SAS package, I demonstrate the relationship between the variable length declaration and SAS dataset file size. I also describe some programming tricks SAS dictionary tables, SCL functions, and the Call Execute facility implemented in the package.

## ELECTRONIC SUBMISSION OF CLINICAL TRIAL DATA

In 2008, the FDA announced that the Electronic Common Technical Document (eCTD) developed by International Conference on Harmonisation Multidisciplinary Group 2 Expert Working Group would be the preferred format for electronic submissions. The FDA's Center for Drug Evaluation and Research (CDER) strongly encourages sponsors to consider the implementation and use of data standards, developed by the Clinical Data Interchange Standards Consortium (CDISC), for the submission of new drug applications. CDER released several guidelines related to the electronic submission of clinical trial data, which recommended that "allotted character variable length/size for each column in a dataset should be the maximum length used" for that variable.[1]

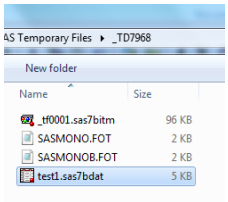
An analysis conducted by CDER in 2011 and based on studies tabulated by the eData Team between January 2010 and February 2011 identified correlation between dataset sizes and allotted column variable length.[2] This analysis was done after the eData team, which is responsible for electronic submissions made to the FDA, determined that the principal issue for these submissions was waste of file space, mainly due to allotted variable lengths.

However, issues with the lengths of variables are not CDISC or eCTD compliance issues; they are specific FDA submission requirements. In the most recent version of the free tool (Open CDISC community v2.0.2) used by the FDA to validate standard data (from CDISC models), all the rules (such as the SD1082 error of the Open CDISC validator v1.4) related to excess variable lengths have been removed.[3]

## THE RELATIONSHIP BETWEEN VARIABLE LENGTH AND SAS DATASET FILE SIZE

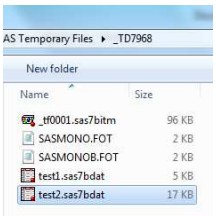
The correlation between file size and the length of variables length can be demonstrated by the following simple empirical experiment.

```
data WORK.TEST1 ;
  length MyKey 8. VarA VarB VarC VarD VarE VarF $ 2 ;
  delete ;
run ;
```



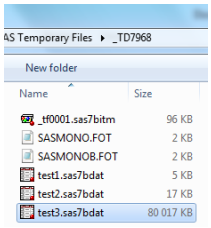

➔ Creates an empty SAS7BDAT file of 5 kB size.

```
data WORK.TEST2 ;
  length MyKey 8. VarA VarB VarC VarD VarE VarF $ 200 ;
  delete ;
run ;
```



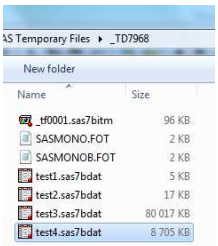

➔ Even without any observations, the increase in variable length increased the SAS7BDAT file size from 5 kB to 17 kB.

```
data WORK.TEST3(label='Test3 Dataset') ;
  length MyKey 8. VarA VarB VarC VarD VarE VarF $ 200
         VarG VarH $ 8 VarI VarJ $ 40 VarK $ 80
         VarL $ 100 ;
  label MyKey='Identifier' VarA='Parity' ;
  do MyKey=1 to 50000 ;
    if (mod(MyKey,2)) then VarA="Uneven" ;
    else VarA="Even" ;
    output ;
  end ;
run ;
```




➔ With 50,000 observations and inappropriate variable lengths, the file size becomes huge (80,017 kB).

```
data WORK.TEST4(label='Test4 Dataset') ;
  length MyKey 8. VarA VarB VarC VarD VarE VarF $ 20
         VarG VarH $ 8 VarI VarJ $ 4 VarK $ 8
         VarL $ 10 ;
  label MyKey='Identifier' VarA='Parity' ;
  do MyKey=1 to 50000 ;
    if (mod(MyKey,2)) then VarA="Uneven" ;
    else VarA="Even" ;
    output ;
  end ;
run ;
```




➔ Here, an appropriate variable length declaration reduces the SAS7BDAT file size by a factor of 10, from 80,017 kB to 8,705 kB).

## DESCRIPTION OF THE SAS RESIZE VARIABLE PACKAGE

This package is made up of four SAS macros. The full code of these macros is presented in Appendix A.
Optimization of dataset size by optimizing variable size can only be accomplished if the maximum size of the data in question is known. For this reason, these macros must be used in a post-processing step.

#### GLOBAL PROCESS

1. Identification of all the datasets to be squeezed
2. For each dataset, the character lengths of the variables are identified
3. For each variable, the maximal observed length is calculated
4. For each variable, the length is modified

#### REMARKS

- All the datasets will be replaced by the squeezed one.
- All variables with a name beginning with a '__' will be dropped.
- In a new document released in June 2015, the CDER recommends that the length "should be set to the maximum length of the variable used across all datasets in the study". [7] Because this procedure is implemented within each dataset, any variable common to multiple datasets has the potential of having different lengths in different datasets. A SAS macro program that allows this new requirement to be met can be found in Appendix B.
- Any programming transformations that manipulate resized data must take these lengths into account (e.g. appending data, key variables).

#### DORESIZEVAR MACRO

- Macro Description: Calls ResizeVariables macro for each dataset of a database
- Macro Process: *This macro*:
  - Creates a list with the name of each SAS dataset of a given library (parameter of the macro), and
  - For each entry of the list calls the ResizeVariables macro.

#### RESIZEVARIABLES MACRO

- Macro Description: Resizes character variables to the maximum length (based on actual stored data)
- Macro Process: *This macro*:
  - Verifies the existence of the input dataset (parameter of the macro),
  - Retrieves some information about this dataset by calling the DsInfo macro,
  - If there is at least one observation, the input dataset is scanned and for all the character variables identified using the VarInfo macro, the length of the longest value observed is calculated, and
  - Using Call Execute functionalities, all the character variables are resized.

#### DSINFO MACRO

- Macro Description: Returns the Label, Observation number, and Variable number of a dataset as global macro-variable values
- Macro Process: *This macro*:
  - Uses SCL functions to return the numbers of variables (VarNb) and observations (ObsNb), and the label (DsLabel) of the input dataset (parameter of the macro) as global macro variables.

#### VARINFO MACRO

- Macro Description: Returns the label, length, type, number, and name of a variable as global macro-variable values
- Macro Process: *This macro*:
  - Uses SCL functions to return the following as global macro variables: the number (VarNum), name (VarName), type (VarType), length (VarLength), and label (VarLabel) of a given variable identified either by its number (a parameter of the macro) or by its name (parameter of the macro).

## FOCUS ON SOME PROGRAMMING TECHNIQUES USED IN THE SAS PACKAGE

#### DICTIONARY TABLES

Dictionary tables are SAS datasets containing information (libraries, tables, files, options, etc.) related to the active SAS session. SAS dictionary tables are read-only tables created and maintained by SAS and are located in a DICTIONARY library. DICTIONARY is a 10 letter libref, but SAS libref values are limited to 8 characters. Therefore, views are needed to get access to the dictionary tables in DATA or PROC (other than PROC SQL) steps. These views are in the SASHELP library.[4]

The doResizeVar macro presented here uses the SASHELP.VTABLES view of the DICTIONARY.VTABLES table to retrieve the list of datasets in a given database: the datasets to treat with the ResizeVariables macro.

#### SCL FUNCTIONS

SCL is primarily used in SAS/AF® programming to develop applications that work within the SAS environment. Some of SCL functions allow SCL code to access and manipulate SAS datasets and, as such, have access to the SAS dataset metadata. [5]

List of SCL functions used in DsInfo & VarInfo macros

| SCL Function Name | Description |
|---|---|
| open | Opens a SAS data set |
| close | Closes a SAS data set |
| attrn | Returns the value of a numeric attribute for a SAS data set |
| attrc | Returns the value of a character attribute for a SAS data set |
| varname | Returns the name of a SAS data set variable |
| varnum | Returns the number of a variable's position in a SAS data set |
| vartype | Returns the data type of a SAS data set variable |
| varlen | Returns the length of a SAS data set variable |
| varlabel | Returns the label that is assigned to a SAS data set variable |

#### CALL EXECUTE

Call Execute allows using the values in a SAS dataset as parameters of a macro. Call Execute is a facility of the DATA step, which allows executing SAS code generated by the DATA step. Also, the data from the DATA step can be used as part of the executable code in the Call Execute. The process generated and executed by Call Execute will be repeated from every observation of the input dataset. [6]

*Caution*: if Call Execute is used to call a macro and there is either a PROC SQL with INTO or a DATA step with CALL SYMPUT inside the macro to create a new macro variable, then the creation of the macro variable is delayed until the macro finished execution. As a result, the macro variable cannot be used inside the same macro. The %NRSTR function must be used around the argument to Call Execute to resolve this problem.

The ResizeVariables macro of the SAS package presented here uses the Call Execute facility to create the output of a squeezed dataset after the input dataset is scanned and the maximal observed length is calculated for each character variable.

## CONCLUSION

This SAS package is a very friendly tool for squeezing SAS dataset files by resizing the character lengths of variables to fit them within the maximum length used. The length adjustment is done without losing the contents of the character variables. These macros were developed in the context of the electronic submission of clinical trial data. Indeed, it is currently recommended by the FDA to resize the character variables in submitted datasets. We have found that adjusting the widths of variables is an effective alternative to dataset compression and can be used to squeeze datasets into the required file size limits. Such a macro package or a similar technique has many other applications, such as in data archiving to reduce file sizes and costs. Reducing the size of SAS datasets may also resolve resource limitations and facilitate completing long SAS jobs or tasks. For these reasons, this SAS package for adjusting the lengths of character variables to fit the longest value may be the solution needed for many users wishing to reduce dataset file size.

## REFERENCES

[1] “Study Data Specifications.” FDA. Available at: http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM312964.pdf

[2] Chhatre, Dhananjay. “SDTM Column Resizing: Background and Industry Testing Results.” Available at: http://www.cdisc.org/stuff/contentmgr/files/0/4f05d8426369051905a247002c87e38e/files/dhananjay_chhatre__session_9.pdf

[3] OpenCDISC. Available at: www.opencdisc.org

[4] Peter Eberhardt and Ilene Brill, “How Do I Look it Up If I Cannot Spell It: An Introduction to SAS® Dictionary Tables”, SUGI 31 Paper 259. Available at: http://www2.sas.com/proceedings/sugi31/259-31.pdf

[5] Edward Foster, “Create your own Functions using SAS/MACRO and SCL”, PhUSE 2006 Paper CS06. Available at: http://www.lexjansen.com/phuse/2006/cs/CS06.pdf

[6] Artur Usov, “Call Execute: Let Your Program Run Your Macro”, PhUSE 2014 Paper CC06. Available at: http://www2.sas.com/proceedings/sugi31/259-31.pdf

## ACKNOWLEDGMENTS

I would like to thank my colleagues at 4Clinics for their support and valuable comments. Special thanks to Kurt Liittschwager for editing the English.

## RECOMMENDED READING

PhUSE Data Sizing Best Practices Recommendation. Available at: (http://www.phusewiki.org/wiki/index.php?title=Data_Sizing_Best_Practices_Recommendation)

[7] Study Data Technical Conformance Guide Available at (http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM384744.pdf)

## CONTACT INFORMATION

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

Mickael Borne
4Clinics
8, rue de la Terrasse
Paris / 75017
Work Phone: +33 1 42 86 64 57
Fax: +33 1 42 86 64 65
Email: mborne@4clinics.com
Web: www.4clinics.com

## APPENDIX A: SAS MACRO PROGRAMS

```
%MACRO doResizeVar(     /* Call ResizeVariables macro for each dataset of a database */
                   DBIn=            /* Input Database Name */
                   ) / des='Call ResizeVariables macro for each dataset of a database' ;
/*************************************************************************************
Program name:          doResizeVar
Platform:              Windows
Version:               9.2
Author(s):             Mickael Borne (4Clinics)
Date:                  25-04-2013
Purpose:               Call ResizeVariables macro for each dataset of a database (each dataset will
                       be replace)
Parameters:            DBIn: Libref of the input database
Assumptions:           need DsInfo, VarInfo and ResizeVariables macros
Sample of use:
data WORK.TEST1 ;
  length SC1 SC2 SC3 SC4 $ 50 SC5 SDSC $ 15 ;
  SC1='kgzekfg' ;SDSC=' ejfkjg' ;output ;
  SC1='kgzekfg' ;SDSC='kjjgfk ' ;output ;
  SC1='kgzekfg' ;SDSC='kjjgfk ejfkjg' ;output ;
run ;
data WORK.TEST2 ;
  length SC1 $ 15 SC2 SC3 $ 3 SC4 SC5 SDSC $ 10 ;
  SC1='kgzekfg' ;output ;
run ;
data WORK.TEST3 ;
  length SC1 SC2 SC3 SC4 SC5 SDSC $ 100 ;
  SC1='kgzekfg' ;output ;
run ;
%doResizeVar(DBIn=WORK) ;
*************************************************************************************/
  %local Tmp KO DsNumber OccId DsName ;
  %let Tmp=cmodif_ ;
  proc sql noprint ;
    create table WORK.&Tmp.vTable as
      select * from SASHELP.VTABLE
      where upcase(compress(LIBNAME))=upcase(compress("&DBIn.")) ;
    %let KO=1 ;
    select distinct 0 into :KO from WORK.&Tmp.vTable ;
  quit ;
  %if (&KO.) %then %do ;
     %goto exitDRV ;
  %end ;
  data WORK.&Tmp.vTable ;
    set WORK.&Tmp.vTable ;
    OccId=_N_ ;
    call symput('DsNumber',OccId) ;
  run ;
  %do OccId=1 %to &DsNumber. ;
     data _NULL_ ;
       set WORK.&Tmp.vTable(where=(OccId=&OccId.)) ;
       call symput('DsName',MEMNAME) ;
     run ;
     %ResizeVariables(DsIn=%sysfunc(compress(%bquote(&DBIn..&DsName.))),
                      DsOut=%sysfunc(compress(%bquote(&DBIn..&DsName.)))) ;
  %end ;

  %exitDRV:
  %*====== Delete local Datasets ====== ;
  proc datasets library=WORK nolist ;delete &TMP.: ;run ;quit ;

%MEND doResizeVar ;
```

```
%MACRO ResizeVariables( /* Resize character variables to the maximum length (based on actual
stored data) */
                        DsIn=,          /* Input Dataset Name */
                        DsOut=          /* Output Dataset Name */
                       ) / des='Resize character variables to the maximum length (based on
actual stored data)' ;
/*************************************************************************************
Program name:          ResizeVariables
Platform:              Windows
Version:               9.2
Author(s):             Mickael Borne (4Clinics)
Date:                  25-04-2013
Purpose:               Resize character variables to the maximum length (based on actual stored data)
Parameters:            DsIn: Input Dataset Name
                       DsOut: Output Dataset Name
Assumptions:           need DsInfo and VarInfo macros
Sample of use:
data WORK.TEST ;length SC1 SC2 SC3 SC4 SC5 TOTO SDSC $ 10 ;SC1='kgzekfg' ;output ;run ;
%ResizeVariables(DsIn=WORK.TEST, DsOut=WORK.TEST2) ;
*************************************************************************************/
  %local Tmp ;
  %let Tmp=ResV_ ;

  %if (not %sysfunc(exist(%bquote(&DsIn.)))) %then %do ;
     %put WARNING: The dataset %str(%')%sysfunc(compress(&DsIn.))%str(%') does not exist.
;
     %goto exitRV ;
  %end ;

  %DsInfo(DsIn=&DsIn.) ;
  %if (%length(%bquote(&DsLabel.))>0) %then %let DsLabel= label=%sysfunc(strip(&DsLabel.))
;

  %if (&ObsNb.>0) %then %do ;
     options varlenchk=nowarn ;
     data WORK.&Tmp.TEMP(compress=Y) ;
       set &DsIn. end=last ;
       length __VARLGTH $ 12600 ;
       retain __1-__&VarNb. 1 ;
       %do VarCt=1 %to &VarNb. ;
          %VarInfo(DsIn=&DsIn.,Number=&VarCt.) ;
          %if (&VarType.=C) %then __&VarCt.=max(__&VarCt.,length(&VarName.)) ;
          %else if (_n_=1) then __&VarCt.=&VarLength. ; ;
       %end ;
       if last then do ;
         %do VarCt=1 %to &VarNb. ;
            %VarInfo(DsIn=&DsIn.,Number=&VarCt.) ;
            (__VARLGTH=cat(strip(__VARLGTH)," &VarName. ", %if (&VarType.=C) %then '$ ', ;
strip(put(__&VarCt.,best.))) ;
         %end ;
         call execute("data &DsOut.(&DsLabel.) ; length "||strip((__VARLGTH))||"; set
&Tmp.TEMP ; run ;") ;
         drop __: ; %* All the variables with a name beginning by __ were dropped * ;
       end ;
     run ;
  %end ;
  %else %do ;
     data &DsOut.(&DsLabel.);
       set &DsIn. ;
     run ;
  %end ;
  options varlenchk=warn ;

  %exitRV:
  %*====== Delete local Datasets ====== ;
  proc datasets library=WORK nolist ;delete &TMP.: ;run ;quit ;

%MEND ResizeVariables ;
```

```
%MACRO DsInfo(          /* Return the Label, Observation number, Variable number of a dataset as
global macro-variable values */
              DsIn=         /* Input dataset */
              ) / des='Return the Label, Observation number, Variable number of a dataset
as global macro-variable values' ;
/*************************************************************************************
Program name:          DsInfo
Platform:              Windows
Version:               9.2
Author(s):             Mickael Borne (4Clinics)
Date:                  25-04-2013
Purpose:               Return the Label, Observation number, Variable number of a dataset as global
Parameters:            DsIn: Input Dataset Name
Assumptions:           none
*************************************************************************************/
  %global VarNb ObsNb DsLabel ;%local DsId RC ;
  %let DsId=%sysfunc(open(&DsIn.,i)) ;
  %let VarNb=%sysfunc(attrn(&DsId.,nvars)) ;
  %let ObsNb=%sysfunc(attrn(&DsId.,nobs)) ;
  %let DsLabel=%sysfunc(attrc(&DsId.,label)) ;
  %let RC=%sysfunc(close(&DsId.)) ;
%MEND DsInfo ;

%MACRO VarInfo(         /* Return the label, length, type, number and name of a variable as global
macro-variable values */
               DsIn=,         /* Input dataset */
               Number=,       /* Number of the variable's position in the input SAS data set */
               Name=          /* Name of the variable */
               ) / des='Return the label, length, type, number and name of a variable as
global macro-variable values' ;
/*************************************************************************************
Program name:          VarInfo
Platform:              Windows
Version:               9.2
Author(s):             Mickael Borne (4Clinics)
Date:                  25-04-2013
Purpose:               Return the label, length, type and name of a variable as global macro-variable
                       values
Parameters:            DsIn: Input Dataset Name
                       Number: Number of the variable's position in the input SAS data set
                       Name: Name of the variable
Assumptions:           none
Sample of use:
data WORK.TEST ;
  label TestTPT='Time Point' Param='Parameter' ;
  do Subect=1 to 10 ;
    do TestTPT=30, 60, 240, 780 ;
      do Param='P1', 'P3', 'P6', 'P9' ;
        Value=ranuni(0) ;output ;
      end ;
    end ;
  end ;
run ;
%VarInfo(DsIn=WORK.TEST,Number=1) ;
%put VarNum=&VarNum. VarName=&VarName. VarType=&VarType. VarLength=&VarLength. VarLabel=&VarLabel. ;
%VarInfo(DsIn=WORK.TEST,Name=Param) ;
%put VarNum=&VarNum. VarName=&VarName. VarType=&VarType. VarLength=&VarLength. VarLabel=&VarLabel. ;
*************************************************************************************/
  %global VarNum VarName VarType VarLength VarLabel ;%local DsId RC ;
  %let DsId=%sysfunc(open(&DsIn.,i)) ;
  %if (%length(%bquote(&Number.))>0) %then %do ;
     %let VarName=%sysfunc(varname(&DsId.,&Number.)) ;
     %let VarNum=&Number. ;
  %end ;
  %else %do ;
     %let VarNum=%sysfunc(varnum(&DsId.,%upcase(&Name.))) ;
     %let VarName=%sysfunc(strip(&Name.)) ;
  %end ;
  %let VarType=%sysfunc(vartype(&DsId.,&VarNum.)) ;
  %let VarLength=%sysfunc(varlen(&DsId.,&VarNum.)) ;
  %let VarLabel=%sysfunc(varlabel(&DsId.,&VarNum.)) ;
 %let rc=%sysfunc(close(&DsId.)) ;
%MEND VarInfo ;
```

## APPENDIX B: ADDITIONAL MACRO

```
%MACRO setMaxLengthDeclared(                        /* Harmonization of the lengths for character
variables present in more than one dataset of a given database */
                            DbIn=                   /* Harmonization of the lengths for
character variables present in more than one dataset of a given database */
                          , Report=0                /* Boolean to ask a report of modifications
(done if Report=1) */
                            ) / des='' ;
/*************************************************************************************
Program name:          setLengthStatement
Platform:              Windows
Version:               9.2
Author(s):             Mickael Borne
Date:                  13-08-2015
Purpose:               Harmonization of the lengths for character variables present in more than one
dataset of a given database
Parameters:            DbIn: Libref of the input database
                       Report: Boolean to ask a report of modifications
Sample of use:
data WORK.TEST1 ;
  length SC1 $ 7 SC2 $ 1 SDSC $ 13 ;
  call missing(SC2) ;
  SC1='kgzekfg' ;SDSC=' ejfkjg' ;output ;
  SC1='kgzekfg' ;SDSC='kjjgfk ' ;output ;
  SC1='kgzekfg' ;SDSC='kjjgfk ejfkjg' ;output ;
run ;
data WORK.TEST2 ;
  length SC1 $ 5 SC2 SC3 $ 3 SDSC $ 1 ;
  call missing(SDSC) ;
  SC1='kgzek' ;SC2='hgh' ;SC3='pof' ;output ;
run ;
data WORK.TEST3 ;
  length SC1 $ 4 SC2 $ 1 SC3 $ 4 SC4 $ 5 SC5 $ 6 SDSC $ 1 ;
  call missing(SC2 SDSC) ;
  SC1='kgze' ;SC3='PKGD' ;SC4='GHtye' ;SC5='lkfhty' ;output ;
run ;
%setMaxLengthDeclared(DbIn=WORK,Report=1) ;
*************************************************************************************/
  %local Tmp ;
  %let Tmp=SMLD_ ;

  %if (&Report.) %then %do ;
     title "Before modification of lengths" ;
     proc sql ;
       select LIBNAME, MEMNAME, NAME, LENGTH
         from SASHELP.VCOLUMN
         where upcase(LIBNAME) eq compress(upcase("&DbIn."))
         group by NAME having count(*)>1
         order by NAME, LENGTH desc ;
     quit ;
     title ;
  %end ;

  proc sql ;
    %** Extract from vColumn dictionary ** ;
    create table WORK.&Tmp.VCOLUMN as
      select LIBNAME, MEMNAME, NAME, LENGTH
      from SASHELP.VCOLUMN
      where upcase(LIBNAME) eq compress(upcase("&DbIn."))
      group by NAME having count(*)>1
      order by NAME, LENGTH desc ;
    %** List of variables to resize ** ;
    create table WORK.&Tmp.TODO as
      select l.*, r.LENGTH
      from (select LIBNAME, MEMNAME, NAME from WORK.&Tmp.VCOLUMN group by NAME having
LENGTH<max(LENGTH)) as l
      left join (select NAME, max(LENGTH) as LENGTH from WORK.&Tmp.VCOLUMN group by NAME)
as r
      on l.NAME=r.NAME
```

```
      order by MEMNAME, NAME ;
  quit ;

  %** Resize variable using Call execute and SQL procedure ** ;
  data _NULL_ ;
    set WORK.&Tmp.TODO ;
    length COMMAND $ 1000 ;
    COMMAND=catx(' ',"proc sql ;alter
table",cats(LIBNAME,'.',MEMNAME),"modify",NAME,cats("char(",put(LENGTH,best12.),")"),";qui
t ;") ;
    call execute(COMMAND) ;
  run ;

  %*====== Delete local Datasets ====== ;
  proc datasets library=WORK nolist ;
    delete &TMP.: ;
  run ;
  quit ;

  %if (&Report.) %then %do ;
     title "After modification of lengths" ;
     proc sql ;
       select LIBNAME, MEMNAME, NAME, LENGTH
         from SASHELP.VCOLUMN
         where upcase(LIBNAME) eq compress(upcase("&DbIn."))
         group by NAME having count(*)>1
         order by NAME, LENGTH desc ;
     quit ;
     title ;
  %end ;
%MEND setMaxLengthDeclared ;
```