SAS Macros are the Cure for Quality Control Pains

Gary McQuown
Data and Analytic Solutions, Inc.
Fairfax, VA

Abstract:

Quality control should be an important part of all projects. Even though we all know "Garbage In .. Garbage Out"; Quality Control is almost always under appreciated, often under funded and occasionally even overlooked. Macros are the ideal tool for automating these essential but tedious tasks, saving both time and money, and promoting the task so that it gets the attention it deservers. This presentation will cover macros designed to detect and report on data quality errors and will hopefully be useful to anyone interested in the topic.

INTRODUCTION

The purpose of this article is three fold. The first is to introduce the reader to the concept of Data Quality Control and it's value to any IT project or process. This process is a frequent topic at "high levels" but seldom discussed with those who do the actual programming, where it will do the most good.

Secondly it is intended to encourage the use of SAS macros when and where they are most appropriate. Macros lend themselves to tasks that are repetitious and well defined. They are ideal for quality control work because (if designed and run correctly) they greatly reduce the introduction of human error the resources required. As we all know, documentation, error checking, and data cleaning all essential tasks, but they are all too often delayed until the very end or delegated to those with the least understanding of the process.

Lastly it is intended to direct the reader to the wealth of SAS Macro related code available on the web. The quantity and quality of SAS code placed in public domain is quite impressive and continues to grow. When supplemented with the ongoing exchanges on SAS-L, the volume of "free" assistance and material is truly amazing.

I. DATA QUALITY CONTROL

Data Quality Control has been described as an ongoing effort for the validation, improvement and facilitation of the Extraction, Transformation and Loading (ETL) process to insure that the data meets the business needs. Dirty data is simply any data that is not usable, regardless of the reason. As the size and scope of data collected continues to increase in geometrical proportions, so have the issues and importance of maintaining its quality and integrity. To facilitate and monitor the ETL process, a host of Quality Control related programs must be written, maintained and run on an ongoing basis. Fortunately a significant portion of the programming burden can be relieved with the use of SAS macros. Even more fortunate is the availability of SAS macros on the web.

In respect to the ETL process, the Data Quality Control process can be broken down into the following components (figure 1).

A. Problem Definition
B. Identify Data Issues
C. Analyze
D. Improve

Inherent in each step is the necessity to document the activity and the outcome. If data sets and/or values are modified for any reason or in any way, a record of the action should be maintained and reviewed. It is through this process that one learns how the entire process can be improved upon so that it is more efficient, cost effective and responsive to the requirements of the project.

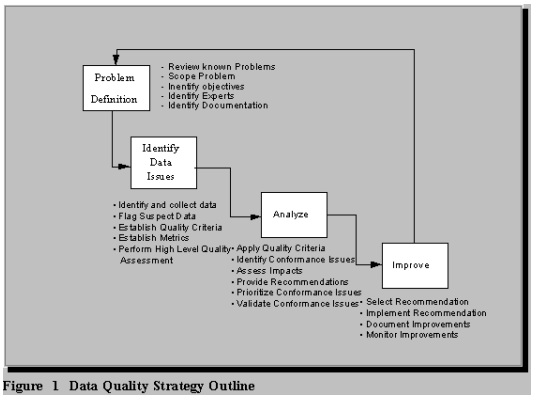

Figure 1 Data Quality Strategy Outline

A) Problem Definition: The first step in the process is to define any existing problems and to identify those responsible for resolving the conflict. This task requires an understanding of the business process, the data, and the operating environment as well as the problem to be resolved. Detailed documentation and project specs are the primary tools used in this phase. Poor communication and "protectionism" are the most common issues to overcome.

It is important to keep in mind that data is often used by multiple groups for multiple reasons. The "correct" structure for one will probably be incorrect for another. Communication between groups can identify common modifications that all groups will benefit from (and share in the cost of implementing).

B) Identify Data Issues:
Once the scope and nature of the problem is fully understood, it should be possible to identify relevant data issues. Using the guidelines established in the business specs, quality criteria and metrics are established to identify questionable values. Questionable values are then flagged for further analysis. It is normally easiest to correct / modify non-conforming values as close to the source of corruption as possible. In some instances, it may be possible to request data in a modified format that alleviates the necessity for additional cleaning / processing. Another solution may be altering the method of data transference or adding additional software that converts data between formats.

According to Andrew Ippilito, data should be evaluated in regards to six quality characteristics:

1) *Accuracy*
a) The measure or degree of agreement between a data value (and set of values) and a source assumed to be correct.
b) A qualitative assessment of freedom from error.

2) *Completeness*
a) The degree to which values are present in the attributes that require them.

3) *Consistency*
a) Data are maintained so they are free from variation or contradiction.
b) The measure of the degree to which a set of data satisfies a set of constraints.

4) *Timeliness*
a) The extent to which a data item or multiple items are provided at the time required or specified
b) A synonym for currency, the degree to which specified values are up to date.

5) *Uniqueness*
a) The ability to establish the uniqueness of a data record (and data key values).

6) *Validity*
a) The quality of the maintained data is rigorous enough to satisfy the acceptance requirements of the classification criteria.
b) A condition where the data values pass all edits for acceptability, producing desired results.

If the data is found to meet the acceptance criteria established for the project, then the requirements themselves must be examined to determine if they are correct.

Samples of the data are often used to expedite this process. The most common SAS tools used to delve into the nature and structures of the data are the Procedures: Contents, Format, Freq, Means, Print and Univariate. There are a number of macros available to assist with this process.

C) Analyze

Working with sample data, now is the time to apply the quality criteria and to identify conformance issues. At this point, some macros may be helpful, but at least some of the programming will have to be "ad hoc." Additional input may need to be added to determine if the programs will properly handle other non-conforming values. Once the data has been modified, you should then assess its impact to confirm that the new values meet the required criteria. It is also important to determine if the changes made will have an unintended effect on other processing. It is not unusual to create additional issues for yourself or others while attempting to correct something considered relatively minor. After the impact has been established, recommendations on the proper procedure can be made. However, before they are implemented, any outstanding conformance issues should be prioritized and validated.

D) Improve
This is the phase where macros can be most beneficial. The process has been carefully defined with a means of measuring its success or failure. Although there will be a number of different actions taken, most are highly redundant. If performed by an individual over a lengthy period of time or on a large number of data sets, the results would probably be tainted by human error. Anther factor to consider is that although it may take longer to write and develop macros, they are extremely efficient in their use.

To proceed in this step, the appropriate recommendation is selected. Further documentation should be performed to assure that the proper procedures are implemented. Additional documentation in the form of exception data sets and summary reports are often useful (necessary) to document changes by variable or dataset.

The documentation can then be used to determine if the desired results were achieved and that the data meets the specified criteria. The entire process should be monitored to determine when and where additional modifications should be made. Special attention should be given when new data or data sources are introduced, or when the scopes of the 'downstream' processes are altered.

II. SAS MACROS

The following section is a smorgasbord of programs, code and Internet links. The selection provided is only a small example of the open code SAS Macros that are available on the web. All are free, but offer no warranty. Most offer some documentation, but the user is required to take it upon themselves to determine if the program is appropriate for their use and will not conflict with other processes on their system. Fortunately most are simple enough to follow that this should not be a difficulty.

DATA CLEANING
TIP00000 - **Cleansing Macro**, Data Scrubbing routine (see tip 00128 for more), just one schema technique by Charles Patridge (www.sconsig.com)

%macro cleanse(schlib=work, schema=, strlen=50,var=,target=target,replace=replace, case=nocase);

Author: Charles Patridge
Email: Charles_S_Patridge@prodigy.net
Version: 2.1 (sug. by Ian Whitlock)

REMOVE OUTLIERS
Outlier
www.spikeware.com

%outlier
(data = _SAS_dataset_name_,
out = _SAS_output_dataset_name
var = _variable_to_screen
pass = _number_of_passes
except = _exception_report_data_set_,
mult = _multiplier_of_standard_deviations_)

The %OUTLIER macro completes outlier screens based on statistical values of a numeric variable in a SAS data set. It is set up to remove any outlier records that are within a given number of Standard Deviations from the mean, and will run that screen a given number of times. For example, a "3-Pass-2" outlier screen will remove any values outside 3 standard deviations from the mean, and will run that outlier screen twice. The given numbers can be any integer.

Trims (Removes Extra Spaces) in File
Trimfile
www.spikeware.com

%trimfile
(infile = input file name
file = output file name
leading = remove leading blanks YES / NO
compbl =compress blanks YES / NO
lrecl = length of sub-variable
outlrecl = output lrecl)

The %TRIMFILE macro is used to remove leading blanks, trailing blanks, and/or multiple concurrent blanks in a file. This macro was created with two uses in mind:

1. Compress an HTML file of extraneous blanks
2. Remove trailing blanks from a program file or other flat file

DOCUMENTS DUPLICATES
TIP000367- **KeepDbls**
By Jim Groeneveld
(www.sconsig.com)

%MACRO KeepDbls (SourceDs=_LAST_, TargetDs=, Overwrit=N, *IdList=, Where=);*

Moves duplicate observations to another file.

EXAMINES AND REPORTS ON SAS LOG
Log Filter
www.ratcliffe.co.uk/rest_logfilt.htm
Log Filter checks your log for errors, warnings, and other "interesting" messages. It then displays what it finds in its summary window. Double-click on a row and it'll reposition the log window to display the message in context (if it's an external log file, it'll open it in a viewer window and position it for you).

The following SAS Macro is provided for use, or as a templet for use and abuse. Many of the macros found on the web will be of a similar format.

/**********************************

EXCEPTION AND PERCENT NON-CONFORMING REPORT

NAME: EAP_RPT.SAS

BY: Gary McQuown

(www.dasconsultants.com)

TO: Produce frequencies identifying questionable data. Values will be compared to the specified format and judged to be G "Good", M "Missing" or N "Non Conforming. An output file will be produced with cross tabs of the variable name by the result of the comparison. The output file will be the same as the input file, preceded by the letter Q and ending in an LST extension. The output file can be redirected to a different directory.

EXAMPLE:

%EAP_RPT (
_VARS=VAR1 VAR2,
_FMTS=$Z9F $Z18F,
LIBIN=IN,
LIBOUT=OUT,
DSN=NEW200306);

_VARS= list of character variables to
review .. paired with _FMTS
_FMTS= list of formats to apply ...
paired with _VARS
LIBIN= SAS library of input data set
DSN= Name of input SAS data set
LIBOUT= SAS library of output data set

```
* EXAMPLE OF APPROPRIATE FORMAT;
PROC FORMAT LIBRARY=LIBRARY;
          VALUE $Z18F
          '0'-'18'    = 'G'
          ' '         = 'M'
          OTHER  = 'N';
RUN;
```

**********/

```
%MACRO EAP_RPT
  (_VARS=, _FMTS=, LIBIN=, DSN=, LIBOUT=);
%LOCAL I _VARS _FMTS LIBIN DSN LIBOUT ;

DATA    EXCEPTION
          (KEEP = VARNAME VALUE FORMAT)
           PCT    (KEEP = VARNAME QUALITY );
 SET &LIBIN..&DSN (KEEP=&_VARS. );

%LET I = 1;
  %DO %WHILE ((%SCAN(&_VARS, &I))>' ');
  %LET _VAR = %SCAN(&_VARS, &I);
  %LET _FMT = %SCAN(&_FMTS, &I);

     QUALITY =  PUT( &_VAR , &_FMT.. ) ;

           DATASET="&DSN";
           VARNAME="&_VAR";
           VALUE  = &_VAR ;
           FORMAT ="&_FMT";

      IF QUALITY ^= 'G' THEN OUTPUT
      EXCEPTION;
   %LET I = %EVAL(&I+1);
    OUTPUT PCT;
   %END;

     LABEL
           DATASET="DATASET"
           VARNAME="VARIABLE/NAME"
           VALUE  ="VALUE"
           QUALITY="ASSESSMENT";
RUN;

PROC SORT DATA=PCT;
   BY VARNAME QUALITY;
RUN;

TITLE1 "DATA QUALITY REPORT";
TITLE2 "DATA SET: &DSN";
FOOTNOTE "G=GOOD, M=MISSING, N=NON
CONFORMING";

PROC FREQ DATA=PCT        ;
  BY VARNAME;
  TABLES VARNAME*QUALITY / LIST
  OUT=&LIBOUT..Q&DSN. MISSING ;
RUN;
TITLE;
 %MEND EAP_RPT;
```

SAS ON THE WEB

The following is a short list of websites that contain archives of SAS programs for public use. Most of the locations also contain links to other SAS related sites that may also contain yet more programs.

www.sas.com
The SAS Institute Web Site

www.sconsig.com
SAS Consultants SUG -- this site contains tips, tricks, articles, and a salary survey.

www.ratcliffe.co.uk
More tips and tricks

www.stattechservices.com
Downloadable SAS utility macros

www.spikeware.com
Downloadable SAS utility macros

www.listserv.uga.edu/archives/sas-l.html
The SAS ListServ

www.nesug.org
North East SUG -- with proceedings

www.sesug.org
South East SUG -- with proceedings

www.dasconsultants.com
Articles and links

www.mcw.edu/pcor/rsparapa/sasmacro.html
Large number of SAS utility macros

www.math.yorku.ca/SCS/friendly.html
SAS related articles and programs

www.stat.ncsu.edu/sas/samples/index.html
Sample SAS programs

CONCLUSION

The Internet is an excellent tool for locating SAS programs, material related to Data Quality Control, or both. Data Quality Control is essential to the success of any analytic process that is itself dependent upon the quality of the data. By utilizing "pubic domain" SAS code, it is possible to make the Quality Control process more productive and efficient. The user will also learn a thing or two about SAS macros along the way.

Publications:

Ron Cody, Cody's Data Cleaning Techniques Using SAS Software, SAS Institute BBU 1999

Janet Stuelpner, "Mrs. Clean Tackles Dirty Data" SESUG Proceedings, 2002

Ralph Kimball, "Dealing with Dirty Data", DBMS Online, September 1996.

Acknowledgements: Special thanks to Dr. Dawn Li and Hunter Nichols for their assistance.

About the Author:

Gary McQuown is a co-owner of Data and Analytic Solutions, Inc, a SAS Alliance Partner, located in Fairfax, Virginia (http://www.dasconsultants.com). He has programmed in SAS long enough to learn the difference between a programmer and a developer. He provides SAS solutions for large and small firms in the US and Europe. Previous presentations have been made at NESUG, SESUG, SSU and MWSUG. He can be contacted at gmcquown@dasconsultants.com.

Appendix: Additional Macros:

```
*************************************************************************;
**                                                                     **;
** Dictnary.sas - Creates dictionary with formats for SAS datasets     **;
**                                                                     **;
** Example:                                                            **;
** libname sasdata 'r:\cprubrst\main\saslib';                          **;
** libname library 'r:\cprubrst\main\saslib';                          **;
** options fmtsearch=(sasdata);                                       **;
** %include 'c:\Projects\sasprgms\dictnary.sas';                       **;
** %dictnary(datalib=sasdata,dataset=patient,fmtlib=sasdata);          **;
**                                                                     **;
** N.B. If fmtlib=library is not used, then must include               **;
** options fmtsearch=( );                                             **;
**                                                                     **;
*************************************************************************;
**  Dictionary may be sorted by:                                       **;
**      - variable order using order=varnum or npos, or                **;
**      - alphabetically using order=name                              **;
*************************************************************************;

%macro dictnary(datalib=sasdata,dataset=_all_,
            fmtlib=library,formats=Yes,order=varnum);
 proc format;
  value type_fmt 1='Num' 2='Char' .='   ';
  value var_num .='   ';

 *** Makes dataset of contents of SAS Dataset *;
 proc contents data=&datalib..&dataset
     memtype=data noprint out=work.contents
     (keep=memname memlabel varnum npos name label type format);
 run;

 %if %upcase(&formats)=YES %then %do;
 *** Creates DS of Formats *;
 proc format lib=&fmtlib cntlout=work.formats
     (keep=fmtname start end label type);
 data work.formats;
  set work.formats;
  if type='C' then fmtname='$'||fmtname;       * Makes FMMNAME compatable *;
  rename /* fmtname=format */ label=fmtlabel;
  drop type;

 *** Joins Contents and Formats ***;
```

```
proc sql;
 create table work.combine as select * from
    work.contents a left join work.formats b on
    a.format=b.fmtname order by memname, &order, start;
quit;
%end;
%else %do;
proc sort data=work.contents out=work.combine;
  by memname memlabel &order;
%end;

*** Changes to Improve Appearance ***;
data work.dictnary;
  set work.combine;
  by memname memlabel &order;

  *** More Compact Listing ***;
  start=compress(start);
  end=compress(end);
  if ((start ne ' ') and (start ne end))
     then range=compress(start||'-'||end);
     else range=compress(start);
  label range='Range';

  *** Omits Repeats of Name, Label, etc. ***;
  if first.&order=0 then do;
     varnum=.;
     name  ='   ';
     label ='   ';
     type  =.;
     format='   ';
  end;
  output;
  *** Adds a empty line between variables ***;
  if last.&order=1 then do;
     varnum=.;
     name  ='   ';
     label ='   ';
     type  =.;
     format='   ';
     range='  ';
     fmtlabel='  ';
     output;
   end;

*** Prints Dictionary ***;
```

```
 ods proclabel="&dataset"; * Creates Bookmarks *;
 proc print data=work.dictnary label uniform;
   by memname memlabel; pageby memname;
   id &order;
   var name label type format range fmtlabel;
   format type type_fmt. &order var_num.
         label $char40. fmtlabel $char20.;
   label memname='SAS Dataset'
         memlabel='Dataset Label'
         varnum='Order'
         type='Type'
         format='Format'
         fmtlabel='Format Label';
 run;
%mend dictnary;



%macro CONTENTS(lib);
/*************************************************************************
NAME: CONTENTS

TO: Create a SAS file with the Contents of All SAS files in a given Libname

INPUT: Header of all files in a given libname
OUTPUT: Work file (cont_&lib) with contents of all file in given libname

PARMS
LIB = Libname to review

use: %CONTENTS(PLRSDIS);

NOTE: The output can be easily put into an Excel Pivot Table.

BY: Gary McQuown mcquown@dasconsultants.com

*************************************************************************/


%let lib=%upcase(&lib);
proc sql;
  create table dirlist as
    select memname from sashelp.vtable
      where memtype="DATA" and libname="&lib";
   quit;
%let no=0;
```

```
data _null_;
  set dirlist;
  call symput("d"!!compress(put(_n_,8.)),memname);
  call symput("no",_n_);
run;

proc sql noprint;
select memname into: list separated by "|" from dirlist;
quit;

data cont_&lib;
 run;

%let i = 1;
%do %until(%scan(&list,&i,|) = );
%let dsn = %scan(&list,&i,|);

proc contents data=&lib..&dsn noprint out=cont;
run;

data cont_&lib;
   set cont_&lib cont;
run;

%let i = %eval( &i + 1 );
%end;

%mend CONTENTS;


%macro cont_compare(dsn1, dsn2);
/************************************************************
NAME:  CONT_COMPARE
TO  :  compare field names in two data sets
BY  :  Gary McQuown www.DASconsultants.com
FOR :

INPUT:
dsn1 = full name of first SAS data set (data_1.thisone)
dsn2 = full name of second SAS data set (data_1.thatone)

OUTPUT:  temp file work.compare
        print out of results

PARMS: none
```

EXAMPLE: %cont_compare(temp_1.tpr_tmp0107, data_1.medb_4_dly_trend);

MODS:

```
**************************************************************/
proc contents data=&dsn1 noprint out=one ;run;
data one; set one; name=upcase(name); run;
proc sort data=one; by name; run;

proc contents data=&dsn2 noprint out=two ; run;
data two; set two; name=upcase(name); run;
proc sort data=two; by name; run;

data compare (keep= name issue one two type_: length_: );
  merge one (in=aa rename=(type=type_1 length = length_1 ))
       two (in=bb rename=(type=type_2 length = length_2 ));
                  length issue $20.;
by name;

if aa=bb then
do;
         issue="IN BOTH";
         if type_1 ^= type_2     then issue = compbl(issue ||"/ TYPE");
         if length_1 ^= length_2 then issue=compbl(issue ||"/ LENGTH");
end;

if aa then ONE = "Y"; else ONE = "N";
if bb then TWO = "Y"; else TWO = "N";

label one = "IN &DSN1"
     two = "IN &DSN2"
           type_1 = "TYPE &DSN1"
           type_2 = "TYPE &DSN2"
           length_1="LENGTH &DSN1"
           length_2="LENGTH &DSN2"
           issue  = "ISSUE:"
;
run;
title1 "FIELD COMPARISON";
title2 "&DSN1 to &DSN2";
title3 "ISSUES";
proc print data=compare (where=(issue ^=" ")) split=" ";
         var name issue;
run;

title3 "ALL FIELDS";
```

```
proc print data=compare split=" ";
        var name type_1 type_2 length_1 length_2 issue;
run;
title;

%mend cont_compare;


%macro compare(dsn1=, dsn2=, by=, clist=_character_, nlist=_numeric_, out=);
/*** COMPARE.SAS
To compare two files with the same variables and create a third file that
contains only:
1) the variables to be compared
2) the "by" variables used to join the files
3) a list of variables (starting with x) that flag matching/non matching values

by Gary McQuown www.dasconsultants.com

Input: any two files with the same variables (must have same name and type)

Ouput: user named file with comparison of the named variables
      report of matching and missing values

PARMS:

dsn1= first data set (libname and name )
dsn2= second data set (libame and name)
by= = by variable or list of by variables
clist= list of character variables to compare.  Default is all Character variables.
nlist= list of numeric variables to compare. Defaultis all Numeric variables.
out= name of output data set (libname and name )


data a;
 a = 1; b = 6; c="x"; output;
 a = 2; b = 7; c="x"; output;
 a = 3; b = 8; c="x"; output;
run;

data b;
 a = 1; b = 6; c="y"; output;
 a = 2; b = 7; c="x"; output;
 a = 3; b = 7; c="x"; output;
run;
 %compare (dsn1=a, dsn2=b, by=a, clist=c, nlist=b, out=compare);
```

Logic:
1) create lists/array of variables to process.
2) rename all variables in the second data set
3) merge the data sets on the by variables.
4) compare
5) freq results

******/

```
%macro rename (varlist) ;
%local i ;
%let i = 1 ;
%do %while (%scan(&varlist,&i) ^= %str( )) ;
%trim(%scan(&varlist,&i)) = _%trim(%scan(&varlist,&i))
%let i = %eval(&i+1) ;
%end ;
%mend rename ;

%macro _list (varlist, del) ;
%local i ;
%let i = 1 ;
%do %while (%scan(&varlist,&i) ^= %str( )) ;
        &del.%trim(%scan(&varlist,&i))
%let i = %eval(&i+1) ;
%end ;
%mend _list ;

proc sort data=&dsn1 (keep = &by &nlist &clist) out=dsn1;
by &by;
run;

proc sort data=&dsn2 (keep = &by &nlist &clist) out=dsn2
 (rename=(
   %if &clist ^= %then %do;  %rename(&clist ) %end;
   %if &nlist ^= %then %do; %rename(&nlist )  %end;
  ));
by &by;
run;

data &out (drop=i );
 merge dsn1 (in=aa)
       dsn2 (in=bb);
by &by;

%if &nlist ^= %then %do;
```

```
array n_old(*)     &nlist;
array n_new(*)              %_list(&nlist,_ ) ;
array n_delta(*) $       %_list(&nlist,x) ;

do i = 1 to dim(n_old);
  if missing(n_old(i)) and missing(n_new(i)) then n_delta(i) = "M"; else
  if n_old(i) = n_new(i) then n_delta(i) = "G"; else
  if missing(n_old(i))   then n_delta(i) = "1"; else
  if missing(n_new(i))   then n_delta(i) = "2"; else
  if n_old(i) ^= n_new(i) then n_delta(i) = "B";

end;
%end;

%if &clist ^= %then %do;
array c_old(*)     &clist ;
array c_new(*)          %_list(&clist,_) ;
array c_delta(*)        %_list(&clist,x)  ;

do i = 1 to dim(c_old);
  if missing(c_old(i)) and missing(c_new(i)) then c_delta(i) = "M"; else
  if c_old(i) = c_new(i) then c_delta(i) = "G"; else
  if missing(c_old(i))   then c_delta(i) = "1"; else
  if missing(c_new(i))   then c_delta(i) = "2"; else
  if c_old(i) ^= c_new(i) then c_delta(i) = "B";
end;

%end;

run;

proc format;
   value $ match
    "G" = "Match         "
          "B" = "Non Matching  "
          "M" = "Both Missing  "
          "1" = "First Missing "
          "2" = "Second Missing";
   value missing
      . ="Missing    "
          other="Non Missing";
   value $ missing
     ' '="Missing    "
          other="Non Missing";
run;
```

```
Title1 "Matching and Missing";
title2 "File1:&dsn1";
Title3 "File2:&dsn2";

proc freq data=&out;
  tables x: / missing;
  format x: $match.;
run;

title1 "Missing in Input File";
title2 "File1:&dsn1";
proc freq data=&dsn1;
  tables &nlist &clist / missing;
  format _numeric_ missing. _character_ $missing.;
run;

title1 "Missing in Input File";
title2 "File1:&dsn2";
proc freq data=&dsn2;
  tables &nlist &clist / missing;
  format _numeric_ missing. _character_ $missing.;
run;

%mend compare;

/** example of  Compare use **/
%compare(dsn1=plrsdis.wc, dsn2=plrsdis.tm1_wc_pol_term_st, by=pol_num eff_yr state,
nlist=premium, out=test);


/** example of CK_Missing use **/
%macro ck_missing(dsn=&syslast, path=&_QC_base., var=_all_, fmt=_numeric_ m_n. _character_
$m_c.);



/*******************************************************
CK_MISSING.sas

To: Evaluate variables in a data set in regards to missing and non missing status.
The default is to evaluate all variables for the last data set accessed.

Parms:
DSN = libname and name of data set. Default is the last read/created.
path= path to directory where QC info is stored.
var = list of variables to be evaluated.  If blank, all variables will be evaluated.
```

fmt = format statment. If blank, a standard format for missing values will
be applied.
Format example:
fmt=var1 fmtx. var2 fmty. var3 $fmtz.

Note: A format will be applied to all fields that immediatly proceeds it. So multiple
fields that use the same format can be grouped together (see fico1, fico2 and fico3 below).

Do not use commas

Character formats start with a dollar sign ($) and all formats end with a period (.)

by Gary McQuown www.dasconsultants.com

USAGE:

```
** will evaluate all variables in the last created data set **
%ck_missing;

** will evaluate all variables in a given data set **
%ck_missing (dsn=mylib.recentfile);

** to evaluate selected variables with certain formats **
%ck_missing(   dsn=mylib.recentfile,
            var=UPB FICO1 FICO2 FICO3 CHANNEL,
            fmt=UPB upb. FICO1 FICO2 FICO3 fico. CHANNEL $chnl.
         );
*******************************************************/

data _null_;

    x = "&dsn";
    one=scan("&dsn",1,".");
    two=scan("&dsn",2,",");
    if two ^= ' ' then sdsn= two; else sdsn=one;
    call symput("hdsn", hdsn);
run;

proc format;
 value m_n
              .    = "Missing"
        0    = "Zero   "
                   Other= "Other  ";

value $ m_c
         " " ="Blank  "
```

```
    "Missing" ="Missing"
    "MISSING" ="Missing"
    "UNKNOWN" ="Unknown"
    "Unknown" ="Unknown"
    "UNK"     ="Unknown"
    "Unk"     ="Unknown"
          Other    ="Other  ";
run;

ods html close;
ods html body = "&path.\QC_&hdsn._&sysdate..xls";

title1 "Missing Check for %upcase(&dsn) ";
title2 "Run on &sysdate by &sysuserid";

proc freq data=&dsn ;
  tables &var /missing list ;
format &fmt ;
run;

ods html close;
title;

%mend ck_missing;



%macro time(process);
/** utility macro to log time of each process **/
/** to be used for efficiency reporting       **/
%global ttotal;

data time_temp;
 length process $25 ;
 process      = "&process";
 time_start   = datetime();
 time_started =put(time_start,datetime18.);
;
call symput('ttime_started', time_started);
run;

data time_log;

    %if %sysfunc(exist(time_log, data)) %then
    %do;
       set time_log time_temp;
```

```
    %end; %else
    %do;
       set time_temp ;
    %end;
 ;
duration =max(0, time_start - lag(time_start));
lag_p = lag(process);

  call symput ('tduration', duration);
  call symput ('tlag_p', lag_p);
run;

proc sql noprint;
 select round(sum(duration)/60,.01) into: ttotal from time_log;
quit;

%put **********************************************;
%put Run time for &tlag_p = &tduration seconds ;
%put Process &process stated at &ttime_started;
%put Total Processing time = &ttotal minutes;
%put **********************************************;

%mend time;

%macro mk_fmt (dsn=, start=,label=, fmtname=, type=C, Library=work, Other=OTHER);

/** mk_formats.sas

To: Create a format from a SAS data set.

Parms:
DSN
START   =Unique key value  ie. SSN
LABEL   =Value to be associated with start ie. Full Name with SSN
FMTNAME    =Name of Format (sans "."), must be seven or less in length
TYPE  = C or N for Character or Numberic
Library =libname of Format Library  (default =work)
Other   =Value to supply for missing (default =OTHER)
By: Gary McQuown mcquown@DASconsultants.com  wwww.DASconsultants.com

**/

/** note: the following should be used: options fmtsearch=(fmtlib) nofmterr;  **/


DATA temp1 ; set &dsn  end=last;
```

```
  if missing(&start) then delete;

    start   =trim(left(&start.));
    fmtname ="&fmtname.";
    type    ="&type.";
    label   =&label;
      output;
    if last then
    do;
      start = "OTHER";
     %if &type = C %then label = "&OTHER"; %else label = . ; ;
        output;
    end;
    keep start fmtname type label;
run;

proc sort data=temp1 nodupkey; by start; run;
proc format cntlin=temp1 library=&library; run;

%mend mk_fmt ;
```