SAS® Macros and the Power of Design
Dalia C. Kahane, Westat, Rockville, MD

ABSTRACT:
This paper demonstrates how you can use SAS macros to design a complex system that is yet efficient and flexible. It shows how you may utilize macros to perform tasks that need to be repeated, with variations across tasks and across different datasets. A careful design allows you to take charge of and respond to changes and variations in requirements.

The system of macros described here includes a %DRIVER and supporting macros. Together, they declare needed macro variables, prepare required SAS datasets, perform varying tasks based on current needs, and report the results of the processes.

Two important design features are put forward. First, SAS macros facilitate the use of metadata tables that provide relevant values for the current run. Second, modularized design allows each macro to be separately programmed and tested. The macros can then serve as building blocks, with various combinations of macros being used to complete different tasks.

INTRODUCTION
As part of our Quality Assurance Plan for a longitudinal Cancer Screening Trial, data gathered from multiple sources are examined for consistency and accuracy. Many edit checks have been specified. Some of the edit checks are relevant to all studies but others are relevant only to a single study and/or to a specific phase in the study. It is easy to recognize the importance of a system that responds to the changing requirements based on study and tasks.

The trial includes the following features:
- Multiple studies (Prostate, Lung, Colon and Ovarian Cancer Studies)
- Multiple medical centers collecting data
- More than 100,000 participants
- More than 30 Forms per participant (Screening Exams, DE, TI, Admin)
- Hundreds of edit checks to date, reviewing hundreds of data fields
- Some of the edit checks are relevant to all four studies but others are relevant only to a single study and/or to a specific phase in the study (examples are given in the cfgTask table in Appendix A)
- The trial includes varying types of data (For example: “person” data with one record per participant; “main” data with one record per participant per phase; data about events such as screening exams, diagnostic procedures, and treatments with multiple records per participant per phase.)

Here are some example situations from our QA Plan:
- At regular intervals, users need to run all edit checks that are valid for the Prostate data.
- Similar runs are required for the Lung, Ovarian and Colon studies, each separately.
- Periodically, users need to run a few specific edit checks that need immediate and urgent correction.
- The study manager wants to see a quick summary of errors generated by the latest edit checks (added in the last month) that are unique to the Lung study.

The Challenge:
- Started with one study but knew that additional studies were coming.
- Knew that edit requirements were in flux.
- Ongoing review of data from multiple centers means that additional edit checks are to be expected.
- Multiple clients and their numerous requests lead to ongoing changes in priorities.
- Had to provide an efficient way to accommodate the changing requirements.

The Response:
The %DRIVER macro system. It was developed to allow users the freedom to run some or all of the edit checks, as needed. This system responds to the following needs:
- Each study has a "main" dataset and may have one or more other datasets.
- Some edits are to be performed for all studies, others only for a specific study or phase in a study.
- Reports need to be generated for each study, showing relevant data for that study.

The %DRIVER macro system incorporates the following important design features:
- Modularized design of macros (i.e. macros serve as building blocks). Each of these macro blocks is separately programmed, tested and executed. A given combination of these blocks serves to complete the specific request at a given run.
- Metadata tables provide macro variables with their appropriate values for a given run.
SYSTEM STRUCTURE AND PROCESSING

Overall structure of the %Driver system:

The following metadata tables are necessary for correct application of the system:

- **cfgStudy**: this table contains data for setting up macro variables based on the Study. Table columns include: Study, Var Name, Var Description, and Var Value. The Var Name column provides the "macro variable name" and the Var Value provides the "macro variable value".
- **cfgTask**: table contains, per task: Task, Study Name, General Rule, InMain, and Versions.

THE %DRIVER MACRO

Parameters:

- **STUDY** → Study for which current run is to be performed (e.g. P for Prostate)
- **TASKS** → List of tasks to be performed in current run (default=ALL)
- **PATH** → Full path and folder name for storing results of current run
- **METAPATH** → Full path and folder name for storage of metadata tables

The following steps are performed by the %DRIVER macro:

- **Declare** and assign values to macro variables based on metadata stored in the cfgStudy and cfgTask tables. Done via call to a %SetVars macro.
- **Prepare** the data relevant to the current study and current edits, via calls to %Get&DAT macros, where &DAT is based on a list of datasets declared in the macro variable &DATLIST. At a minimum, &DATLIST contains one dataset representing the Main dataset for the current study.
- **Perform** the edit checks included in &TASKLIST for the current run. Calls to %Chk&TASK macros.
- **Report** results of edit checks (after the individual check results were brought back into the Main dataset for summary and reporting.)

---

1. Example records for both metadata tables are shown in Appendix A.
2. %Driver, %SetVars and example macros %GetMain, %GetDEProc, %Chk05, %Chk07 and %Chk29 are shown in Appendix B.
EXAMPLE CALLS TO %DRIVER

The parameter values provided upon calling the %DRIVER macro determine what data is examined as well as what edit checks are to be performed in the current run.

Examples:

- To run the edit checks for the Prostate data, using all the edit checks that are valid for Prostate, and saving the current edit results in the appropriate network folder, the following call is made:
  %DRIVER (STUDY=P, TASKS=ALL, PATH= k:\mhgrp\plco\Prostate\Round1, METAPATH= k:\mhgrp\plco\Metadata)

- To run the edit checks for the Prostate data but for a few specific edit checks that need immediate and urgent fixes, the following call is made:
  %DRIVER (STUDY=P, TASKS=05 07 29, PATH=k:\mhgrp\plco\Prostate\Expedited05, METAPATH=k:\mhgrp\plco\Metadata)

THE %SETVARS MACRO

This macro sets up all macro variables based on the metadata tables for Studies and Tasks.

Parameters:

- STUDY → Study for which current run is to be performed (e.g. P for Prostate)
- METAPATH → Full path to folder that contains the metadata tables
- METASTUDY → Name of table that contains metadata related to studies (default=cfgStudy)
- METATASK → Name of table that contains metadata related to tasks (default=cfgTask)

This macro performs the following steps:

- Declaring needed macro variables
- Reading in the relevant Study metadata.
- Assigning Study metadata to macro variables via CALL SYMPUTX
- Reading in the relevant Tasks metadata
- Assigning Tasks metadata to macro variables via CALL SYMPUTX

A few of the important macro variables assigned values in %SetVars:

- DATLIST – determines what datasets get created for the current run
- TASKLIST – list of valid tasks for the study and current run
- InTASKS – subset of TASKLIST: tasks that are contained within the Main dataset
- OutTASKS – subset of TASKLIST: tasks that need data from datasets other than Main

THE %GET&DAT MACROS

Looping through &DATLIST, each distinct %Get&DAT macro gets the specified relevant data. As shown in the cfgStudy table, the &DATLIST is determined based on the current study. For example: the %GetMain macro gets all data associated with the “main” table for that study (one record per participant), %GetDEProcs macro gets the data relevant to diagnostic medical procedures per participant; the %GetSurg macro gets the data relevant to the surgical treatments performed per participant, etc.

THE %CHK&TASK MACROS

Two lists of tasks are created in %SetVars, in the macro variables &InTASKS and &OutTASKS (see Appendix B). A task may be very simple and may require only a few lines of code, or it may be complex and may include several steps.

The loop through &InTASKS is performed within the “Main data step”. Each %Chk&TASK macro that is part of the &InTASKS list, assigns a &TASK indicator value of True if an error is found that corresponds to the edit check being performed. For example: we have 20 edits that need to be performed on each record in data Main. We have two choices: (a) to run through data Main 20 times or (b) to run through it once performing the 20 checks. Decision: we run as many or as few checks as are relevant to the &Study and current run, going through the Main dataset only once! Each task described in cfgTask is marked as belonging to &InTASKS or &OutTASKS based on the InMain column.

The loop through the &OutTASKS list is done “outside the Main data step”. Each %Chk&TASK macro that is part of the &OutTASKS list, may use data elements from the Main dataset, but always needs additional or exclusive data from other &DATLIST datasets. An &OutTASKS %Chk&TASK assigns a &TASK indicator but also creates a ChkDat&TASK dataset that contains the merge variables &KEYVARS and the &TASK indicator. The edit results contained in the &TASK indicator will be merged back onto each relevant Main record.

Note that great flexibility is gained because the &InTASKS and the &OutTASKS lists are both constructed at the start of the run by %SetVars and include only the tasks that are considered valid for the specific study and the current run.
MAINTAINING THE SYSTEM

WHEN A NEW STUDY IS ADDED:
For each new study, new records are added to the cfgStudy table:
- The new records should reflect all metadata relevant to this study.
- Macro variables with their associated values need to be added in a manner similar to the example records shown in cfgStudy for already existing studies.
- The new records relevant to the new study may include some new macro variables that are needed for tasks that are relevant only to this new study.

WHEN A NEW TASK IS REQUESTED:
Each edit check is considered a TASK. The following steps need to be performed when a new edit check is requested:
- Add the new TASK with name, description and complete metadata in the cfgTask table.
- Set the InMain indicator. I. e. make sure to indicate whether variables related to this TASK are all contained within the Main SAS dataset. This determines whether the task is considered as part of &InTASKS or &OutTASKS.
- Prepare a new %Chk&TASK macro. This macro shall contain the code needed to identify cases which fail the required edit check. Program and test this macro code.

Once the TASK has been added to the cfgTask table and the macro has been programmed and tested, the call to this macro will automatically be performed by %DRIVER because this new TASK will be added to the &TASKLIST created in %SetVars.

WHEN CHANGING REQUIREMENTS FOR AN EXISTING TASK:
The steps to follow depend on the following:
- The changed requirements still deal with the same data elements that are already obtained under the old requirements: the only thing that needs to be revised and tested is the macro block of code for that &TASK.
- The changed requirements demand that new data elements be obtained: it is possible that a new %Get&DAT macro will be needed or that an existing one needs revision. In addition, the %Chk&TASK macro code will need to be revised.

CONCLUSION
The method described here for the utilization of SAS macros demands careful design but enables you to build systems that are easy to program, test, execute and maintain. This method was developed in response to the situation of multiple studies with multiple edit-checks of data, but may be implemented in any instance where similar (even if not identical) tasks need to be done in multiple instances and/or with multiple datasets.

Examples:
- SAS datasets need to be created from raw data files. There are data elements that are identical within all these data files (e.g. key fields such as IDs); there are other fields that appear in a subset of these files (e.g. all medical diagnostic forms, demographics, occupational history, etc.); there are additional fields that are unique to a specific dataset. Such datasets may be built by putting together detailed metadata tables as well as various blocks of macro code. Such design will ensure that all data elements that are identical will be declared in an identical manner across datasets, will be checked in a similar fashion, etc.
- Multiple reports need to be generated. These reports describe similar data at the top part of the page per case and unique data at the bottom part of the page. Through the use of metadata and proper design, you may create macros to handle the common parts and others to handle the unique parts of the report.
DISCLAIMER
The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

ACKNOWLEDGEMENTS
I would like to thank Marianne Whitlock whose macro programming inspired the creation of this system and whose encouragement, support, and reviews of this paper were invaluable.
I would also like to thank Mike Rhoads, Judy Walsh, and Fran Cohen for their help in reviewing this paper.

CONTACT INFORMATION
If you have any questions or comments about the paper, you can reach the author at:
Dalia Kahane, Ph.D.
Westat
1650 Research Blvd.
Rockville, MD 20850
Kahaned1@Westat.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are registered trademarks or trademarks of their respective companies.
APPENDIX A
METADATA TABLES EXAMPLE RECORDS

cfgStudy - Metadata relevant to different studies:

<table>
<thead>
<tr>
<th>Study</th>
<th>VarName</th>
<th>VarDescription</th>
<th>VarValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>ORGAN</td>
<td>Organ-site relevant to study</td>
<td>Prostate</td>
</tr>
<tr>
<td>P</td>
<td>INLIB</td>
<td>Libname for input SAS datasets</td>
<td>\</td>
</tr><tr>
<td>k25\vol2502\mhgrp\plco\Prostate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>DATList</td>
<td>List of datasets associated with study</td>
<td>Main DEprocs</td>
</tr>
<tr>
<td>P</td>
<td>MAINVars</td>
<td>List of variables to keep from Main data</td>
<td><em>all</em></td>
</tr>
<tr>
<td>P</td>
<td>KEYVars</td>
<td>Key variables for merge between Main and other datasets</td>
<td>PID STUDYYR</td>
</tr>
<tr>
<td>P</td>
<td>DEformlist</td>
<td>Complete list of forms for this Organ</td>
<td>(&quot;DEP&quot;,&quot;DEP2&quot;,&quot;DEP3&quot;)</td>
</tr>
<tr>
<td>P</td>
<td>Vformlist</td>
<td>Complete list of Verbatim Ftypes</td>
<td>associated with this Organ</td>
</tr>
<tr>
<td>P</td>
<td>DIAGCNFM</td>
<td>DiagEval shows confirmed cancer</td>
<td>IN (2,4,5)</td>
</tr>
<tr>
<td>P</td>
<td>DIAGPATH</td>
<td>DiagEval shows confirmed pathologically</td>
<td>IN (4,5)</td>
</tr>
<tr>
<td>P</td>
<td>TNMVars</td>
<td>Complete list of TNM variables per form</td>
<td>TNMCLASS PRITUMOR NODALINV DISTMETA STAGPATH STAGCLIN</td>
</tr>
<tr>
<td>P</td>
<td>STAGEVAR</td>
<td>Staging lead variable for Clinical Staging</td>
<td>STAGCLIN</td>
</tr>
<tr>
<td>P</td>
<td>DXDD</td>
<td>Diagnosis Day</td>
<td>PROSDXDA</td>
</tr>
<tr>
<td>P</td>
<td>DXMM</td>
<td>Diagnosis Month</td>
<td>PROSDXMO</td>
</tr>
<tr>
<td>P</td>
<td>DXYYYY</td>
<td>Diagnosis Year</td>
<td>PROSDXYR</td>
</tr>
<tr>
<td>P</td>
<td>HISTTYPO</td>
<td>HISTTYPE - Other specify</td>
<td>11</td>
</tr>
<tr>
<td>P</td>
<td>HISTTYPDU</td>
<td>HISTTYPE - Unknown</td>
<td>12</td>
</tr>
<tr>
<td>P</td>
<td>HISTGRDU</td>
<td>HISTGRD - Unknown</td>
<td>5</td>
</tr>
<tr>
<td>P</td>
<td>OTHCODES</td>
<td>Dx/STG codes w Other(specify)</td>
<td>02,08,10,11,19,25,27,88</td>
</tr>
<tr>
<td>P</td>
<td>BIOCODES</td>
<td>Dx/STG codes indicating Biopsy</td>
<td>05,10,15,16,20</td>
</tr>
<tr>
<td>L</td>
<td>Organ</td>
<td>Organ-site relevant to study</td>
<td>Lung</td>
</tr>
<tr>
<td>L</td>
<td>INLIB</td>
<td>Libname for input SAS datasets</td>
<td>\</td>
</tr><tr>
<td>k25\vol2502\mhgrp\plco\Lung</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>DATList</td>
<td>List of datasets associated with study</td>
<td>Main DEprocs</td>
</tr>
<tr>
<td>L</td>
<td>MAINVars</td>
<td>List of variables to keep from Main data</td>
<td><em>all</em></td>
</tr>
<tr>
<td>L</td>
<td>KEYVars</td>
<td>Key variables for merge between Main and other datasets</td>
<td>PID STUDYYR</td>
</tr>
<tr>
<td>L</td>
<td>DEformlist</td>
<td>Complete list of forms for this Organ</td>
<td>(&quot;DEL2&quot;,&quot;DEL3&quot;)</td>
</tr>
<tr>
<td>L</td>
<td>Vformlist</td>
<td>Complete list of Verbatim Ftypes</td>
<td>associated with this Organ</td>
</tr>
</tbody>
</table>
**cfgTask** - Metadata relevant to different tasks per study:

<table>
<thead>
<tr>
<th>Task</th>
<th>Study</th>
<th>Scope</th>
<th>Status</th>
<th>In Main</th>
<th>Phase</th>
<th>General rule describes if to investigate case</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>L</td>
<td>L</td>
<td>A</td>
<td>0</td>
<td>2,3</td>
<td>Some malignancy confirmed Pathologically/Histologically/Cytologically but no Biopsy procedure is recorded in the other Dx/STG procedures.</td>
</tr>
<tr>
<td>02</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>0</td>
<td>0</td>
<td>Rule actually repeated Probcat_1 results.</td>
</tr>
<tr>
<td>04</td>
<td>P</td>
<td>P</td>
<td>A</td>
<td>1</td>
<td>3</td>
<td>Photo = Yes but diagnosis day is not available. Cancelled because dealt with through general DIFs.</td>
</tr>
<tr>
<td>05</td>
<td>P</td>
<td>PLCO</td>
<td>A</td>
<td>1</td>
<td>1,2,3</td>
<td>Histopathologic Type Unknown</td>
</tr>
<tr>
<td>05</td>
<td>L</td>
<td>PLCO</td>
<td>A</td>
<td>1</td>
<td>1,2</td>
<td>Same description as for Prostate</td>
</tr>
<tr>
<td>05</td>
<td>C</td>
<td>PLCO</td>
<td>A</td>
<td>1</td>
<td>1,2,3</td>
<td>Same description as for Prostate</td>
</tr>
<tr>
<td>05</td>
<td>O</td>
<td>PLCO</td>
<td>A</td>
<td>1</td>
<td>1,2,3</td>
<td>Same description as for Prostate</td>
</tr>
<tr>
<td>06</td>
<td>O</td>
<td>O</td>
<td>A</td>
<td>0</td>
<td>3</td>
<td>Surgery without same day Approach</td>
</tr>
<tr>
<td>07</td>
<td>P</td>
<td>P</td>
<td>A</td>
<td>1</td>
<td>1,2,3</td>
<td>Gleason score not available (see also Probcat_45 and Probcat_46)</td>
</tr>
<tr>
<td>29</td>
<td>P</td>
<td>PLO</td>
<td>A</td>
<td>0</td>
<td>2,3</td>
<td>Any procedure with date after the date of death (MKS 8783). See also related Probcat_39.</td>
</tr>
<tr>
<td>29</td>
<td>L</td>
<td>PLO</td>
<td>A</td>
<td>0</td>
<td>2</td>
<td>Same description as for Prostate</td>
</tr>
<tr>
<td>29</td>
<td>O</td>
<td>PLO</td>
<td>A</td>
<td>0</td>
<td>2,3</td>
<td>Same description as for Prostate</td>
</tr>
<tr>
<td>30</td>
<td>L</td>
<td>L</td>
<td>A</td>
<td>1</td>
<td>1,2</td>
<td>Special test for HISTTYPE</td>
</tr>
</tbody>
</table>

**Task** = Identifier of edit-check task  
**Study** = P for Prostate, L for Lung, etc.  
**Scope** = Shows scope of relevance of rule (i.e. task relevant to which studies)  
**Status** = A for Active, C for Cancelled, H for Hold.  
**InMain** = Designates task as an In-Main or Out-of-Main edit-check.  
**Phase** = 3 study phases overall - task may be relevant to a single of multiple phases.
Macro code for %Driver:

%Macro Driver (STUDY=,TASKS=ALL,FOLDER=);

%local J;
/* set macro parameters based metadata in cfgStudy and cfgTask tables */
%SetVars((STUDY=&STUDY,METAPATH=&METAPATH)
/* get the datasets relevant to this study. Loop through each DAT in DATLIST */
%let J = 1;
%let DAT=%scan (&DATLIST,&J);
%do %while (&DAT ne);
%put starting macro ==> Get&DAT;
%Get&DAT(STUDY=&STUDY)
%let J = %eval(&J+1) ;
%let DAT=%scan(&DATLIST,&J);
%end;
/* within Main loop through the list of &InTASKS. Call each %Chk&TASK macro. */
data Main;
set Main;
%let J=1;
%let TASK=%scan (&InTASKS,&J);
%do %while (&TASK ne);
%put Starting macro ==> Chk&TASK;
%Chk&TASK()
%let J = %eval(&J+1) ;
%let TASK=%scan(&InTASKS,&J);
%end;
run;
/* outside of Main */
/* Loop through the list of &OutTASKS. Call each %Chk&TASK macro. */
/* ChkDat&TASK dataset gets created that shows result of edit check */
%let J=1;
%let TASK=%scan(&OutTASKS,&J);
%do %while (&TASK ne);
%put starting macro ==> Chk&TASK;
%Chk&TASK()
%let J = %eval(&J+1) ;
%let TASK=%scan(&OutTASKS,&J);
%end;
/* combine Main and "Other" results */
data Main;
merge Main(in=a) /* for all checks done "outside", get the relevant datasets */
%let J=1;
%let TASK=%scan(&OutTASKS,&J);
%do %while (&TASK ne);
ChkDat&TASK ()
%let J = %eval(&J+1) ;
%let TASK=%scan(&OutTASKS,&J);
%end;
by &KEYVARS;
if a;
run;
/* create edit check report via call to %ReportIt */
%mend Driver;
Macro code for %SetVars:

```
%macro SetVars(STUDY=,METAPATH=,METASTUDY=cfgSTUDY,METATASK=cfgTask);
   %GetCfgStudy(PATH=&METAPATH, STUDY=&STUDY) /* get metadata for STUDY */
   /* get list of study-related macro variables */
   proc sql noprint;
      select distinct VarName into: MacVars separated by ' ' from &METASTUDY;
   quit;
   /* declare relevant macro variables */
   %global &MacVars ;
   data _null_;
      set &METASTUDY;
      call symputx( VarName, VarValue ) ; /* assign values to macro variables */
   run;
   %GetCfgTask(PATH=&METAPATH,STUDY=&STUDY) /* get metadata for TASKS */
   %Global CHKLIST InTASKS OutTASKS ;
   %Local VALIDTASKS TASKLIST ;
   /* get list of valid tasks from the cfgTask table */
   proc sql noprint;
      select distinct Task into: VALIDTASKS separated by ' ' from &METATASK;
   quit;
   /* the &TASKS variable contains the current list of tasks requested by user */
   /* if default of ALL is used - replace with all valid tasks for current study */
   %if &TASKS=ALL %then %let TASKS=&VALIDTASKS;
   /* assign values to the task-related macro variables */
   data _null_;
      length TaskList InTasks OutTasks $200 CheckList $500 CheckTask $20;
      retain TaskList InTasks OutTasks CheckList "";
      if EOF then do;
         call symputx("TASKLIST",TaskList);
         call symputx("CHKLIST", CheckList);
         call symputx("InTASKS", InTasks);
         call symputx("OutTASKS",OutTasks);
      end;
   run;
   /* review the requested tasks against cfgTasks data for current study */
   if index("&TASKS",strip(Task) ) > 0 then do;
      TaskList = catx(" ",TaskList,Task) ;
      CheckTask="Check"|| strip(Task) ;
      CheckList = catx(" ",CheckList,CheckTask) ;
      if InMain=1 then InTasks = catx(" ",InTasks,Task) ;
      else OutTasks = catx(" ",OutTasks,Task) ;
   end;
%put TASKS (requested) ==> &TASKS;
%put TASKLIST (requested and valid for &ORGAN) ==> &TASKLIST ;
%put CHKLIST (array of check indicators) ==> &CHKLIST ;
%put INTASKS (checked within Main) ==> &InTASKS ;
%put OUTTASKS (checked outside Main) ==> &OutTASKS ;
%mend SetVars ;
```
Examples of %Get&DAT macros:

%macro GetMain(STUDY=);

/ * Get Main data per &STUDY */
proc sort data=INLIB.&STUDY.Main out=Main;
by &KEYVARS;
run;

data Main (keep = &KEYVARS &MAINVARS);
set Main ;
/* calc the cancer SY because we do not want to display on PCIF */
/* events that occured 2+ years after Cancer was confirmed */
CancerSY = 99;
CConfirm = 0 ; /* cancer confirmed indicator */
ConfPath = 0 ; /* cancer confirmed pathologically indicator */
/* set up edit-check indicators array */
array Checks(*) &CHKLIST;
do i = 1 to dim(Checks);
    Checks(i) = 0;
end;
if DIAGEVAL &DIAGCNFM then do; /* cancer Confirmed cases */
    CConfirm = 1;
    CancerSY = input(STUDYYR,2.);
/* set indicator for "cancer confirmed pathologically" */
    if DIAGEVAL &DIAGPATH then ConfPath=1;
end;
else CConfirm = 0;
run;
/* get randomization and status information needed for the form */
/* rndgroup, rnddate, DOB, and age are added for each problem case */
proc sort nodupkey
    data=INLIB.prsn (keep=p1d rnddate rndgroup gender DOB)
    out=prsn;
    by PID;
run;
proc sort nodupkey
    data=INLIB.stat (keep=p1d MOB Vstatus Gstatus Vdate Gdate Ddate)
    out=stat;
    by PID;
run;
/* add rndgroup, rnddate, DOB, and age are added for each problem case */
data Main;
merge Main(in=a)
    prsn(in=b)
    stat(in=c)
    ;
    by PID;
if a;
run;
%mend GetMain;
%macro GetDEprocs(STUDY=);
  /* get all data relevant to diagnostic procedures */
  %local J PROCLIST SBCDLIST PREFIXLIST PREFIX CARD PROC;
  %if &STUDY=P %then %do;
    %let PROCLIST = PSA DRE TRUS BIO OTH ;          /* all procs */
    %let SBCDLIST = 2 3 4 5 6 0 0 ;                 /*SBCDs for procs */
    %let PREFIXLIST = P R T B O ;                   /*SBCDs for procs */
  %end;
  %else %if &STUDY=L %then %do;
    %let PROCLIST = OTH ;                            /* all procs */
    %let SBCDLIST = 2 ;                        /*SBCDs for procs */
    %let PREFIXLIST = P  ;                      /*SBCDs for procs */
  %end;
  %let J=1;
  %let PROC=%scan(&PROCLIST,&J);
  %let CARD=%scan(&SBCDLIST,&J);
  %let PREFIX=%scan(&PREFIXLIST,&J);
  %do %while (&PROC ne );  /* for each procedure in PROCLIST */
    %if &PROC=DSS or &PROC=DSS3 %then %do;
      %let PROCDS = &PROC.MAIN;
    %end;
    %else %do;
      %let PROCDS = &STUDY.0&CARD;
    %end;
    data &PROC;
      set INLIB.&PROCDS ;
      if &PREFIX.DAY = 99 or &PREFIX.MONTH = 99 or &PREFIX.MONTH = . or
      &PREFIX.YEAR = 9999 or &PREFIX.YEAR = .
      then ProcDate=.;
      else ProcDate = MDY(&PREFIX.MONTH,&PREFIX.DAY,&PREFIX.YEAR);
      if &PREFIX.DAY = . then CharDAY = "**";
      else CharDAY = put(&PREFIX.DAY,Z2.) ;
      if &PREFIX.MONTH = . then CharMONTH = "**";
      else CharMONTH = put(&PREFIX.MONTH,Z2.) ;
      if &PREFIX.YEAR = . then CharYEAR = "****";
      else CharYEAR = put(&PREFIX.YEAR,Z4.) ;
      Proc_Date = catx("/",CharMONTH,CharDAY,CharYEAR) || " E";
      run;
      %let J = %eval(&J+1);
      %let PROC=%scan(&PROCLIST,&J);
      %let CARD=%scan(&SBCDLIST,&J);
      %let PREFIX=%scan(&PREFIXLIST,&J);
  %end;
  data DEprocs;
    set &PROCLIST ;
    run;
  proc sort data=DEprocs;
    by PID STUDYYR;
    run;
  %mend GetDEprocs;
Examples of &InTASKS edit check macros:

%macro Chk05(FORM=);
    /* check if HISTTYPE is unknown */
    if CConfirm = 1 and HISTTYPE = &HISTTYPU
    then Check05 = 1;
%mend Chk05;

%macro Chk07(X=);
    /* relevant to Prostate only – check Gleason score*/
    if ConfPath and Gleason = 99 then Check07 = 1;
%mend Chk07;

An example macro of &OutTASKS edit checks:

%macro Chk29(FORM=);
    /* assumes that &GetDEProcs macro was already run */
    /* find procedures where date shows done after Date of Death */
    data ChkDat29Multi;
        merge Main(in=a keep=PID Ddate)
            DEprocs(in=b keep=PID STUDYYR ProcDate Proc_Date)
        ;
        by PID;
        if a;
        length Probcat $20;
        if Ddate>. and ProcDate>. then do;
            if ProcDate > Ddate then do;
                Check29=1;
                Probcat="Probcat_29";
                output ChkDat29Multi;
            end;
        end;
    run;
    proc sort nodupkey data=ChkDat29Multi
        out=ChkDat29(keep = PID STUDYYR Check29);
        by PID STUDYYR;
    run;
%mend Chk29;