PharmaSUG China 2015 - 50

The loop within: A macro on looping variables and observations

Eduard Joseph Siquioco, PPD Inc., Philippines

ABSTRACT

Producing the same outputs for different treatments, patients, phases can be tedious when typing a lot of codes when in fact you only need one. This looping macro will help in producing these kinds of outputs for SDTM or TLFs. This macro will also show a procedure where variables can be used in the looping calls for a more customized transposition of data set. The paper will introduce the basics of looping and iterating inside macros and will also include a walkthrough of the macro being used for variables, and observations.

Keywords: Macro, Looping, Dynamic loop ends

INTRODUCTION & INSPIRATION

In most studies, programmers are plagued with dealing programming one unique derivation or specification and repeating this across different patients, treatments, or variables. If you will not do a macro looping on these kinds of situations, you will have to program each one for each scenario to suit your needs. Programming each one would require more time and effort. There will also be cases that derivation or specifications would be updated; this would result in updating the programs repeatedly and would cause more time and effort for the update.

Macro looping is a very efficient way to avoid these scenarios to save time and effort. Contrary to data step looping, which is limited to looping inside the specified data set, macro looping is very flexible. It can loop outside a data step and it can also run an indefinite amount of code.

MACRO LOOPING BASICS

In order to create a macro loop, we will need to create a macro that will have the following elements. Please see the table below to familiarize with the required elements of macro looping.

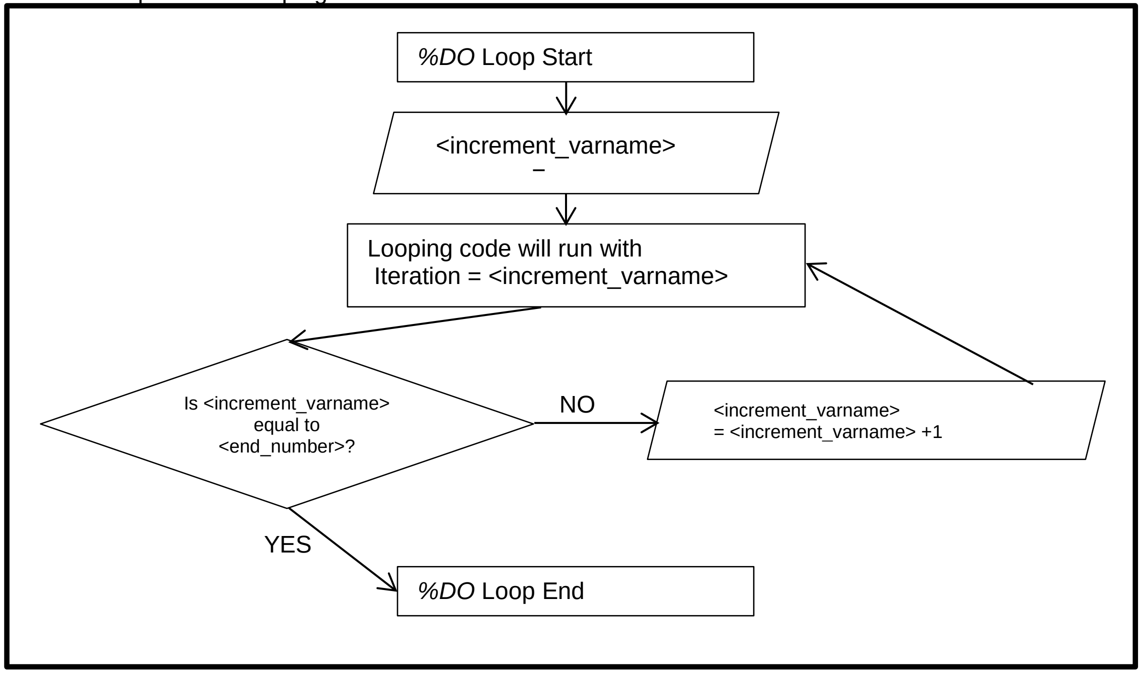
Keyword	**Description**
%MACRO <macro_name> (param =)	Start of a macro code
%MEND <macro_name>	End of a macro code
%DO <increment_varname> =	Specify the start of the macro looping. The % symbol is important to make SAS® distinguish it from the data step 'do' statement
<start_number> %TO <end_number>	Specify the start and end iterations of the %DO loop. The % symbol is important to make SAS distinguish it from the data step 'to' statement. This is only work in conjunction with a %DO statement.
%END	Specify the end of the macro loop. All codes between the %DO and %END would run at every iteration of the macro loop

Example 1: Simple macro loop code and log message

```
%macro simple_loop();
 %do counter= 1 %to 5;
   put The current value of counter is &counter.;
 %end;
%mend;
```

```
%simple_loop;
The current value of counter is 1
The current value of counter is 2
The current value of counter is 3
The current value of counter is 4
The current value of counter is 5
```

Illustration 1: Simple Macro looping flowchart


%DO Loop Start
<increment_varname> –
Looping code will run with Iteration = <increment_varname>
Is <increment_varname> equal to <end_number>?
NO
<increment_varname> = <increment_varname> +1
YES
%DO Loop End

Simple macro looping does not solve the problem of looping between variables and observations but we can use the logic of the simple macro looping and customize it to the specifications.

MASTER LOOP: A MACRO LOOPING FOR VARIABLES AND OBSERVATIONS

1.) Create a data set containing your unique looping value. The looping value will be used to subset the data inputs in your data steps or procs.
   a. Patient list
   b. Variable list
      i. A variable list can be made through the *PROC CONTENTs* step.
      ii. Subset the list to your desired variables

Code 1b: Getting the list of variables desired

```
proc contents data=source1 noprint
              out=source1_contents(keep = name);
 *The variable name in the output data set is the list of
  the variables present in the input data set;
run;

*For this example. I would get all date variables;
proc sql;
        create table var_list as
        select distinct name
        from source1_contents
        where name ? 'dt';
 *Commonly, variables with dt would be enough to cover all
  date variables
quit;
```

2.) Create a dynamic end loop count for robustness. Input the created variable as the end value of the do loop.

Code 2: Creating the robust end loop

```
proc sql noprint;
        select count(distinct name) into: totcount
        from var_list;
quit;
%put The total count of variables is &totcount.;
```

3.) Create a data _null_ step in order to assign the first value in the looping.
   a. The data _null_ step uses the data step code in order to read an input data set and use it in creating the macro variable

Code 3: Assigning the end loop to the robust end variable and assigning the macro variable for needed for programming

```
%macro masterloop();
        %do counter= 1 %to &totcount.;

        data _null_;
                set var_list;
                if _n_=&counter.;
                call symput("var_value", name);
        run;

        %put &var_value;
        *Insert desired code here;
        %end;
%mend;
```

There is an automatic variable created by SAS in every data set: *_n_*, this variable holds the value of the ordering of records. The first observation will have *_n_* equal to 1 and second observation equal to 2 and so on. This automatic variable is used by the macro code in order to determine which value it will assign to the macro variable to be used in the code.

A CALL SYMPUT is used in order to transfer information from the data set into a macro variable usable after the DATA _NULL_ step.

POTENTIAL APPLICATIONS

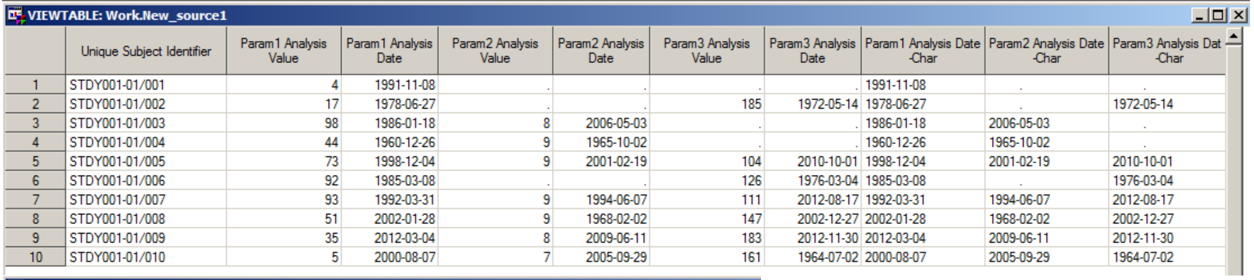
1.) Global date conversion in a library

This macro will create new date variables converting all dates into character values. This will certainly be useful when transforming variables into another format with minimal programmer input. The code will utilize two-stage looping. First stage looping will be for the data set and second-stage of looping will be for the date variables.

i) Use PROC CONTENTS to create a data set with data set names, variables, and label
ii) Create “data set names” data set for first-stage of looping and subset by data sets with dates
iii) Assign macro variable for end of first stage looping
iv) Create a data set with a count of the number of date variables by data set name
v) Run the looping for the first stage and incorporate the second stage looping inside by using the ***master loop macro*.**

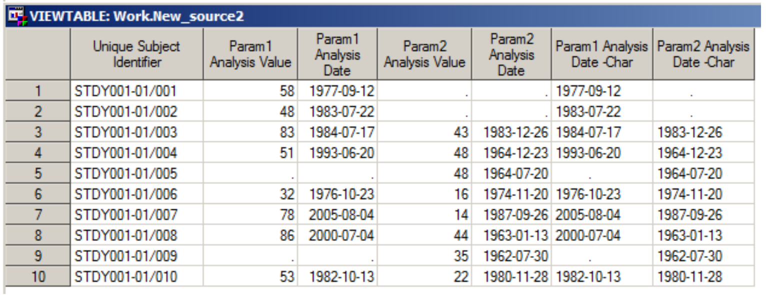
Macro call:
```
%glb_dt_conv(inlib=testlib,outlib=work,prefix=new_);
```

Macro call output: All data sets in the testlib library will have new variables with character dates and appended “-Char” to their labels



VIEWTABLE: Work.New_source1

	Unique Subject Identifier	Param1 Analysis Value	Param1 Analysis Date	Param2 Analysis Value	Param2 Analysis Date	Param3 Analysis Value	Param3 Analysis Date	Param1 Analysis Date -Char	Param2 Analysis Date -Char	Param3 Analysis Dat -Char
1	STDY001-01/001	4	1991-11-08	.	.	.	.	1991-11-08	.	.
2	STDY001-01/002	17	1978-06-27	.	.	185	1972-05-14	1978-06-27	.	1972-05-14
3	STDY001-01/003	98	1986-01-18	8	2006-05-03	.	.	1986-01-18	2006-05-03	.
4	STDY001-01/004	44	1960-12-26	9	1965-10-02	.	.	1960-12-26	1965-10-02	.
5	STDY001-01/005	73	1998-12-04	9	2001-02-19	104	2010-10-01	1998-12-04	2001-02-19	2010-10-01
6	STDY001-01/006	92	1985-03-08	.	.	126	1976-03-04	1985-03-08	.	1976-03-04
7	STDY001-01/007	93	1992-03-31	9	1994-06-07	111	2012-08-17	1992-03-31	1994-06-07	2012-08-17
8	STDY001-01/008	51	2002-01-28	9	1968-02-02	147	2002-12-27	2002-01-28	1968-02-02	2002-12-27
9	STDY001-01/009	35	2012-03-04	8	2009-06-11	183	2012-11-30	2012-03-04	2009-06-11	2012-11-30
10	STDY001-01/010	5	2000-08-07	7	2005-09-29	161	1964-07-02	2000-08-07	2005-09-29	1964-07-02



VIEWTABLE: Work.New_source2

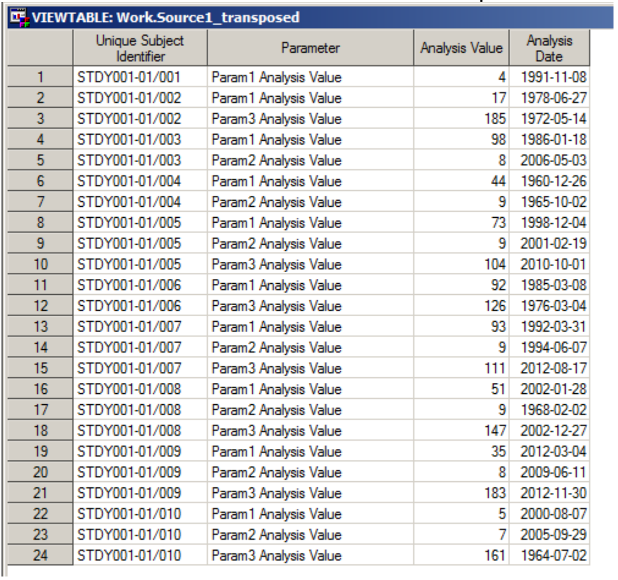
	Unique Subject Identifier	Param1 Analysis Value	Param1 Analysis Date	Param2 Analysis Value	Param2 Analysis Date	Param1 Analysis Date -Char	Param2 Analysis Date -Char
1	STDY001-01/001	58	1977-09-12	.	.	1977-09-12	.
2	STDY001-01/002	48	1983-07-22	.	.	1983-07-22	.
3	STDY001-01/003	83	1984-07-17	43	1983-12-26	1984-07-17	1983-12-26
4	STDY001-01/004	51	1993-06-20	48	1964-12-23	1993-06-20	1964-12-23
5	STDY001-01/005	.	.	48	1964-07-20	.	1964-07-20
6	STDY001-01/006	32	1976-10-23	16	1974-11-20	1976-10-23	1974-11-20
7	STDY001-01/007	78	2005-08-04	14	1987-09-26	2005-08-04	1987-09-26
8	STDY001-01/008	86	2000-07-04	44	1963-01-13	2000-07-04	1963-01-13
9	STDY001-01/009	.	.	35	1962-07-30	.	1962-07-30
10	STDY001-01/010	53	1982-10-13	22	1980-11-28	1982-10-13	1980-11-28

2.) Transposition and assigning variables

In order to transform a data from horizontal to vertical we can to use a PROC TRNSPOSE step, but the PROC TRNSPOSE step cannot immediately reassign the variables into another variable. For example, we have three result columns with three different dates for each. Transposing this data set would make many columns which would then be deleted after reassigning. The *%custom_transpose* macro will do all of these with added functionalities that would make programming easier.

Using the macro call:
```
%custom_transpose(inds=source1, inlib=testlib, outds= source1_transposed);
```

Macro call output: A data set with transposed records mapping respective AVAL and DATE variables into new variables. The label will be the value for para



VIEWTABLE: Work.Source1_transposed

	Unique Subject Identifier	Parameter	Analysis Value	Analysis Date
1	STDY001-01/001	Param1 Analysis Value	4	1991-11-08
2	STDY001-01/002	Param1 Analysis Value	17	1978-06-27
3	STDY001-01/002	Param3 Analysis Value	185	1972-05-14
4	STDY001-01/003	Param1 Analysis Value	98	1986-01-18
5	STDY001-01/003	Param2 Analysis Value	8	2006-05-03
6	STDY001-01/004	Param1 Analysis Value	44	1960-12-26
7	STDY001-01/004	Param2 Analysis Value	9	1965-10-02
8	STDY001-01/005	Param1 Analysis Value	73	1998-12-04
9	STDY001-01/005	Param2 Analysis Value	9	2001-02-19
10	STDY001-01/005	Param3 Analysis Value	104	2010-10-01
11	STDY001-01/006	Param1 Analysis Value	92	1985-03-08
12	STDY001-01/006	Param3 Analysis Value	126	1976-03-04
13	STDY001-01/007	Param1 Analysis Value	93	1992-03-31
14	STDY001-01/007	Param2 Analysis Value	9	1994-06-07
15	STDY001-01/007	Param3 Analysis Value	111	2012-08-17
16	STDY001-01/008	Param1 Analysis Value	51	2002-01-28
17	STDY001-01/008	Param2 Analysis Value	9	1968-02-02
18	STDY001-01/008	Param3 Analysis Value	147	2002-12-27
19	STDY001-01/009	Param1 Analysis Value	35	2012-03-04
20	STDY001-01/009	Param2 Analysis Value	8	2009-06-11
21	STDY001-01/009	Param3 Analysis Value	183	2012-11-30
22	STDY001-01/010	Param1 Analysis Value	5	2000-08-07
23	STDY001-01/010	Param2 Analysis Value	7	2005-09-29
24	STDY001-01/010	Param3 Analysis Value	161	1964-07-02

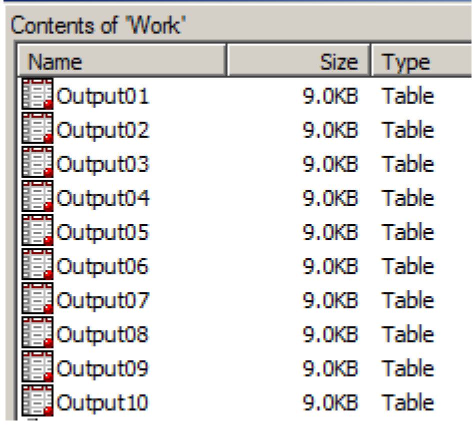
3.) Patient Listings

Before creating the output files for patient listings, we need to subset our data set either before starting our programming or after programming based on the specifications. Programmers need to create one data set for each patient. That is very time consuming to if the macro looping is not used. Using the *%Patient_List* macro programmers will only have to input their final data set and the macro will output each patient in different data sets.

Macro call:

```
%patient_list(inds=source1_transposed, outds=output);
```

Macro call output: Iterated datasets with one patient value each



Contents of 'Work'

Name	Size	Type
Output01	9.0KB	Table
Output02	9.0KB	Table
Output03	9.0KB	Table
Output04	9.0KB	Table
Output05	9.0KB	Table
Output06	9.0KB	Table
Output07	9.0KB	Table
Output08	9.0KB	Table
Output09	9.0KB	Table
Output10	9.0KB	Table

CONCLUSION

Programming dynamic macros for looping can be very tedious and time consuming but once you finish the macro, these techniques will be very natural and easy to understand for the programmer. The master loop macro might be complicated at glance but the logic it follows is simple to understand. It will familiarize it's users on the different techniques in approaching looping.

REFERENCES

Tian, Yunchao. 2014. "The Power of CALL SYMPUT - DATA Step Interface by Example"
Proceedings of the 29th Annual SAS Users Group International Conference.
Available at www2.sas.com/proceedings/sugi29/052-29.pdf

SAS 9.2 Online product documentation. Available at support.sas.com/documentation/92/index.html

ACKNOWLEDGMENTS

I would like to thank PPD Manila Biostatistics and Programming for encouraging us to submit papers in PharmaSUG and to Debbie Castillo for reviewing this paper.

CONTACT INFORMATION

Thank you for taking time reading my work. If you have questions or comments please contact me:

Eduard Joseph Siquioco
PPD
9th Floor, Sun Life Centre. 5th Avenue corner Rizal Drive.
Bonifacio Global City. 1634 Taguig City,. Philippines.
E-mail: EduardJoseph.Siquioco@ppdi.com

APPENDIX

Code A.1: Master Loop Macro

```
%macro MASTER_LOOP(inds=, inlib=);

data &inds.;
        set &inlib.&inds.;
run;

*Extract the variables from the target data set;
proc contents data=&inds. out=&inds._vars(keep=name label);
run;

proc sql;
*Create a data set with the needed variables only;
        create table &inds._varlist as
        select distinct name, label
        from &inds._vars
        where UPCASE(name) ? 'AVAL';

*If patient looping, create a data set with the patient list;
*Also applicable for other lists such as treatment groups/population flags;
        create table &inds._ptlist as
        select distinct usubjid
        from &inds;

*Get the number of variables;
        select count(*) into: totalvars
        from &inds._varlist;

*Get the number of patients;
        select count(*) into: totalobs
        from &inds._ptlist;
quit;


%do counter= 1 %to &totalvars.; *Replace to &totalobs if patient looping;
        data _null_;
        set &inds._varlist;*Replace varlist to ptlist if patient looping;
        if _n_= &counter.;
        call sympут("currentvalue", compress(name));*Replace name into usubjid if patient looping ;
        run;
        %PUT THE CURRENT VALUE IS &CURRENTVALUE.;
%end;

%mend;

%MASTER_LOOP(INDS=SOURCE1,INLIB=WORK) ;
```

Code A.3: glb_dt_conv (Global date conversion) Marco

```
%macro glb_dt_conv(inlib=,outlib=,prefix=);

proc contents noprint data=&inlib.._all_ out =&inlib._contents (keep=memname name label);
run;

proc sql noprint;
         *Subset the data set to contain only date variables;
         create table &inlib._dtvars as
         select *
         from &inlib._contents
         where name ? 'dt' or upcase(label) ? 'DATE';

         *Get the number of data sets with date variables in the library;
         select count(distinct memname) into: totalds
         from &inlib._dtvars;

         *Create the data set with data set names only;
         create table &inlib._ds as
         select distinct memname
         from &inlib._dtvars;

         *Count the number of date variables per data set;
         create table &inlib._dtcnt as
         select memname, count(*) as dtcnt
         from &inlib._dtvars
         group by memname;
quit;

****STAGE 1 LOOPING: DATA SETS;
%do counter1 = 1 %to &totalds.;

         *Assign data set name;
         data _null_;
                  set &inlib._ds;
                  if _n_=&counter1.;
                  call sympût("ds_name", compress(memname));
         run;

         proc sql noprint;
                  *Assign the macro variable for end of counter 2;
                  select dtcnt into: totalvars
                  from &inlib._dtcnt
                  where memname="&ds_name.";

                  *Create a data set with variables only of the current data set;
                  create table &inlib._dtvars&counter1. as
                  select *
                  from &inlib._dtvars
                  where memname="&ds_name.";
         quit;

         *Initalize the data set needed in outlib.;
         data &outlib..&prefix.&ds_name.;
                  set &inlib..&ds_name;
         run;

         **********STAGE 2 LOOPING: VARIABLES;
         %do counter2 = 1 %to &totalvars;
         *Loop though the variables;
                  data _null_;
                           set &inlib._dtvars(where=(memname="&ds_name."));
                           if _n_=&counter2.;
                           call sympût("var_name", compress(name));
                           call sympût("new_label", strip(label));
                  run;

                  data &outlib..&prefix.&ds_name.;
                           set &outlib..&prefix.&ds_name.;
                           &var_name.C=put(&var_name.,yymmdd10.);
                           label  &var_name.C="&new_label. -Char";
                  run;
         %end;
%end;
proc data sets; delete testlib:; quit;
%mend;
```

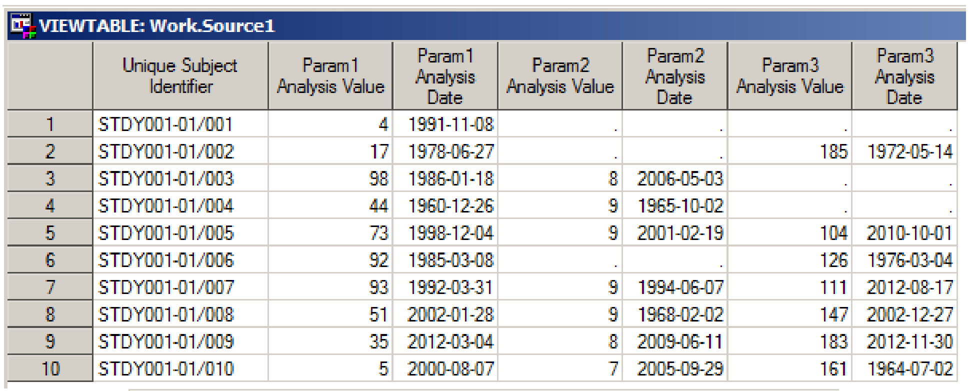
Code A.4: Custom Transpose Macro

```
%macro custom_transpose(inds=, inlib=, outds=);

data &inds.;
        set &inlib..&inds.;
run;

*Extract the variables from the target data set;
proc contents data=&inds. out=&inds._vars(keep=name label);
run;

proc sql noprint;
*Create a data set with the needed variables only;
        create table &inds._varlist as
        select distinct name, label
        from &inds._vars
        where UPCASE(name) ? 'AVAL';

        select count(name) into: totalvars
        from &inds._varlist;
quit;


%do counter= 1 %to &totalvars.;

        data _null_;
        set &inds._varlist;
        if _n_= &counter.;
        call symput("aval_var", compress(name));
        call symput("label_var", strip(label));
        run;

        data &inds._pretrans&counter.;
                set &inds.;

                if not missing(&aval_var.) then do;
                        param="&label_var.";
                        aval=&AVAL_VAR.;
                        adt=&inds.dt&counter.;
                        format adt yymmdd10.;
                output;
                end;
                keep usubjid param aval adt;
        run;
%end;

                data &outds.;
                        set %do counter2=1 %to &totalvars.; &inds._pretrans&counter2. %end; ;
                *Another do loop is added to input the individual data sets in the output data set;
                        label PARAM='Parameter'
                                  ADT='Analysis Date'
                                  AVAL='Analysis Value';
                        proc sort;
                        by usubjid param;
                run;

                proc data sets; delete  &inds._pretrans:  &inds._var:; quit;
%mend;
```

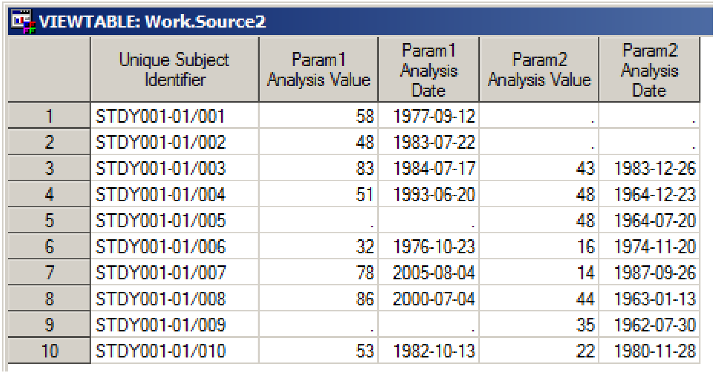
Code A.5: Patient Listing Macro

```
%macro patient_list(inds=,outds=);
proc sql noprint;
        create table &inds._ptlist as
        select distinct usubjid
        from &inds;

        select count(*) into: totalobs
        from &inds._ptlist;
quit;

%do counter= 1 %to &totalobs.;
        data _null_;
        set &inds._ptlist;
        if _n_= &counter.;
        call symput("currentpt", strip(usubjid));
        run;

        data &outds.&counter.;
        set &inds.;
        where usubjid="&currentpt.";
        run;
%end;
%mend;
```

Data set A.1: Dummy Data sets


VIEWTABLE: Work.Source1

	Unique Subject Identifier	Param1 Analysis Value	Param1 Analysis Date	Param2 Analysis Value	Param2 Analysis Date	Param3 Analysis Value	Param3 Analysis Date
1	STDY001-01/001	4	1991-11-08	.	.	.	.
2	STDY001-01/002	17	1978-06-27	.	.	185	1972-05-14
3	STDY001-01/003	98	1986-01-18	8	2006-05-03	.	.
4	STDY001-01/004	44	1960-12-26	9	1965-10-02	.	.
5	STDY001-01/005	73	1998-12-04	9	2001-02-19	104	2010-10-01
6	STDY001-01/006	92	1985-03-08	.	.	126	1976-03-04
7	STDY001-01/007	93	1992-03-31	9	1994-06-07	111	2012-08-17
8	STDY001-01/008	51	2002-01-28	9	1968-02-02	147	2002-12-27
9	STDY001-01/009	35	2012-03-04	8	2009-06-11	183	2012-11-30
10	STDY001-01/010	5	2000-08-07	7	2005-09-29	161	1964-07-02


VIEWTABLE: Work.Source2

	Unique Subject Identifier	Param1 Analysis Value	Param1 Analysis Date	Param2 Analysis Value	Param2 Analysis Date
1	STDY001-01/001	58	1977-09-12	.	.
2	STDY001-01/002	48	1983-07-22	.	.
3	STDY001-01/003	83	1984-07-17	43	1983-12-26
4	STDY001-01/004	51	1993-06-20	48	1964-12-23
5	STDY001-01/005	.	.	48	1964-07-20
6	STDY001-01/006	32	1976-10-23	16	1974-11-20
7	STDY001-01/007	78	2005-08-04	14	1987-09-26
8	STDY001-01/008	86	2000-07-04	44	1963-01-13
9	STDY001-01/009	.	.	35	1962-07-30
10	STDY001-01/010	53	1982-10-13	22	1980-11-28