```
%macro mu_transpose(
 in_data =
,in_where =

,transpose_by =
,transpose_vars =
,transpose_id =
,spanning_var=

,out_data =

,debug = &_default_debug
,help = &_default_help
) / store source  des='V1.0.0.2';

%*************************************************************************
*************
FILENAME:    MU_TRANSPOSE.SAS
DEVELOPER:   (b) (6)
PLATFORM:    SAS 9.1.3, 9.2 on PC
MACROS USED: mu_wordscan.sas
ASSUMPTIONS: Not sure if there are any?
DESCRIPTION:
This macro:
        1. outputs a dataset from the transpose procedure transposing the user
specified variables
           into observations.
        2. creates an output dataset that will contain a variable for each
combination of the
           spanning variable (if applicable) and the ID variable.
        3. creates an output dataset where the name of each resulting transpose
variable will be
           prefixed with the name of the user specified transpose variable(s)
followed by the value
           of the spanning variable (if applicable) and the value of the ID
variable, all separated
           by an "_".

USAGE NOTES:

IN_DATA = [REQUIRED] - Input dataset that needs to be transposed. In the form
LIB.DATA, where lib may be
                                            omitted if it is WORK.
                                            (no default)
IN_WHERE = [OPTIONAL] - Subsetting clause to be used to subset input dataset.
                                                 (no default)
TRANSPOSE_BY = [REQUIRED] - The list of variables to use in the by statement of the
PROC TRANSPOSE.
                                                          Delimiter is a space. NOTE:
Any variables left out of this parameter
```

```
                                                                  will not be in the output
dataset.
                                                                  (no default)
TRANSPOSE_VARS = [REQUIRED] - The list of variables to use in the var statement of
the PROC TRANSPOSE.
                                                                    Delimiter is a space.
                                                                    (no default)
TRANSPOSE_ID = [REQUIRED] - The variable whose unformatted value will be used to
name the transposed variables.
                                                                  Only 1 variable should be
passed here.
                                                                  (no default)
SPANNING_VAR = [OPTIONAL] - The variable which represents the spanning header in
your output (if applicable).
                                                                  (no default)
OUT_DATA = [REQUIRED] - The name of the output dataset.
                                                        (default=&in_data._trans)
DEBUG = [OPTIONAL] - Set to Y(es) to output helpful hints to the log.
                                          (default=&_default_debug)
HELP = [OPTIONAL] - Set to Y(es) to save all intermediate datasets and to turn on
mprint. Set to N(o) to delete
                                        all intermediate datasets and to not turn
on mprint (maintains original option setting).
                                        (default=&_default_help)
************************************************************************************
*************;

%PUT ---------------------------------------------- ;
%PUT INFO: (&SYSMACRONAME) ;
%PUT INFO: Version 1.0 ;
%PUT START;
%PUT ---------------------------------------------- ;


%mu_help_debug;

%* get a list of the datasets in the WORK library before starting to enable cleanup
at
   the end of this macro;

%md_workinfo(
      debug        = &debug
    ,_workdata     = WORK_DATASETS_DATA
    ,_workview     = WORK_DATASETS_VIEW
    ,_mprinttoggle = mprint_setting
   ) *;
```

```
%****************************************;
%** Create default if parameter is missing **;
%****************************************;
%if &out_data = %str( ) %then %do;
        %let out_data = &in_data._trans;
%end;
%****************************************;


%*********************************************;
%** Check if REQUIRED parameters are populated **;
%*********************************************;
%mu_check_req_parameters(
 parameters_to_check = in_data transpose_by transpose_vars transpose_id out_data
,help=no
);
%*********************************************;


%*****************************************************************;
%** Check for invalid dataset or invalid variables in a dataset **;
%*****************************************************************;
%mu_check_data_and_var_exist(
 data_to_check = &in_data
,vars_to_check_in_all_data =
,vars_to_check_in_respective_data = &transpose_by &transpose_vars &transpose_id
,abort_if_does_not_exist = yes
,help=no
);
%*****************************************************************;


%*************************************************************************
**********;
%** Sort the input dataset by the specified by vars, just in case it was not
already sorted **;
%*************************************************************************
**********;
proc sort data=&in_data
                  out=&in_data._sorted;
  by &transpose_by;
  %if %sysevalf(%superq(in_where)=,boolean)=0 %then %do; where &in_where; %end;
  format &transpose_id;
run;
%*************************************************************************
**********;


%*******************************************************;
%** Find all unique values of the SPANNING_VAR and then **;
```

```
%** Break up string of SPANNING_VALS into individual    **;
%** macro vars in order to do separate transposes       **;
%*********************************************************;
%if &spanning_var ne %then %do;
data &in_data._forspan;
  set &in_data._sorted;

  format &spanning_var;
run;

proc contents data=&in_data._forspan
                          out=&in_data._cont(keep=name type length

where=(compress(upcase(name))=compress(upcase("&spanning_var.")))) noprint;
run;
proc sql noprint;
        select distinct(&spanning_var) into :spanning_vals separated by ' ' from
&in_data._forspan;
        select type into :spanning_type from &in_data._cont;
quit;
%mu_wordscan(string=&spanning_vals, root=spanval, numw=numspvals);
%end;
%else %do;
%let numspvals=1;
%end;
%*********************************************************;


%*********************************************************;
%** Break up string of transpose variables into individual **;
%** macro vars in order to do separate transposes          **;
%*********************************************************;
%mu_wordscan(string=&transpose_vars, root=tvar, numw=numtvar);

%do i=1 %to &numspvals;

%do j=1 %to &numtvar;

%if &spanning_var ne %then %do;
%let prefix&i&j=&&tvar&j.._&&spanval&i.._;
%end;
%else %do;
%let prefix&i&j=&&tvar&j.._;
%end;
%put &&&&prefix&i&j;

proc transpose data=&in_data._sorted
                               out=&in_data._trans&i&j
                         prefix=&&&&prefix&i&j;
  by &transpose_by;
```

```
  var &&tvar&j;
  id &transpose_id;
  %if &spanning_var ne %then %do;
  where &spanning_var=
                %if &spanning_type=1 %then %do;
                &&spanval&i
                %end;
                %if &spanning_type=2 %then %do;
                "&&spanval&i"
                %end;
                ;
  %end;
run;

%end;

%end;
%*************************************************************;


%*************************************************************;
%** Merge each resulting transpose dataset back together **;
%*************************************************************;
data &out_data;
  merge %do i=1 %to &numspvals;
                %do j=1 %to &numtvar;
                &in_data._trans&i&j(keep=&transpose_by &&&&prefix&i&j.:)
                %end;
                %end;
                ;
   by &transpose_by;
run;

proc sort data=&out_data;
   by &transpose_by;
run;


%*************************************************************;



%md_clean_and_reset
( debug      = &debug
,_workdata   = %str(&WORK_DATASETS_DATA &out_data)
,_workview   = %str(&WORK_DATASETS_VIEW)
,resetmprint = &mprint_setting
) *;
```

```
%PUT ------------------------------------------------ ;
%PUT INFO: (&SYSMACRONAME) ;
%PUT INFO: Version 1.0 ;
%PUT END;
%PUT ------------------------------------------------ ;


%mend mu_transpose;
```