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

,trtvar = &_default_trtvar.
,DEFINE_TOTAL_GROUPS = &_default_define_total_groups.
,TRTVAR_PRELOADFMT = &_default_trtvar_preloadfmt.

,usubjid = &_default_usubjid.

,subgroup =

,byvars =
,byvars_preloadfmt =
,byvars_sort_order =
,byvars_sort_asc_or_desc =

,bign_in_data = BIGN

,catvar =
,catvar_preloadfmt=
,display_zero_row = Yes
,catvar_sort_order =
,catvar_sort_asc_or_desc = Ascending
,reorder_where =
,reorder_first_last =
,_section_ = 1
,_order1_start = 1
,indent_column1 =
,header =
,header_indent = 0
,catvar_first_or_last =
,catvar_across = No

,get_subj_count = Yes
,get_event_count = No

,count_subj_var = N
,count_event_var = N_EVENT

,out_data = CATEGORICAL_COUNT
,debug =
,help =
) /store source des='V1.0.0.11' ;

%*****************************************************************
FILENAME:    MA_COUNT_CATEGORICAL.SAS
DEVELOPER:   (b) (6)
PLATFORM:    SAS 9.1.3, 9.2 on PC
MACROS USED: mu_wordscan.sas, mu_get_maximum_length.sas
```

ASSUMPTIONS: Macro MA_BIGN must be called before calling this macro
DESCRIPTION:

This macro:

1. outputs a dataset as specified in parameter &OUT_DATA (default is CATEGORICAL_COUNT)
containing counts categorized according to SUBGROUPS and TREATMENTS and CATEGORICAL
VARIABLE.
Examples of when macro would be called:

a) For Demographic table, count number of male and female subjects, per treatment group
and subgroup, using categorical variable GENDER.

b) For Demographic table, count number of subjects in the various races, per treatment
group and subgroup, using categorical variable RACE.

2. Final dataset will have following variables:

&SUBGROUP: will exist only if parameter SUBGROUP is populated.

By variables: Variables specified in parameter BYVARS.

BYVARX_ORDER: order number within each BYVAR variable

&BYVAR_FMTD: Formatted version of BYVARS

BIGN_DUMMY_FLAG: actually created from macro MA_BIGN, brought in for reporting purposes.

CATVAR: Categorical variable.

COL1: Row header.

&COUNT_SUBJ_VAR (Default = N): subject level numeric counts.

&COUNT_EVENT_VAR (Default = N_EVENT): event level numeric counts.

SECTION: section of table, used for ordering and creating blank lines between
different sections of a table.

ORDER1: order number within a section.

SKIPVAR: same as _SECTION_.

INDENT: number of spaces to indent COL1 in final output.

```
         HEADER_FLAG: created only if parameter header is populated.  Indicates
line is a
                      header.

USAGE NOTES:

IN_DATA = input data used to calculate counts.  In the form LIB.DATA, where lib may
be ommitted if WORK
    (default = blank) REQUIRED)
IN_WHERE = subsetting clause
    (default = blank) (OPTIONAL)

TRTVAR = the treatment group variable. Used in the CLASS statement of the PROC
MEANS.
    (default = &_default_trtvar) (REQUIRED)

DEFINE_TOTAL_GROUPS = used to define (sub)total group(s), this parameter must be
defined as follows:
     list of treatment variable values = subtotal 1 ! list of treatment variable
values = subtotal 2 ! ...
     (default = &_default_define_total_groups) (OPTIONAL)


    For example, the values of TRTVAR are 'A' 'B' and 'C' and the analysis calls
for a subtotal of 'A' and
    'B' and an overall total of all three treatments.  Set the parameter as
    define_total_groups = 'A' 'B' = 'subtotal' ! 'A' 'B' 'C' = 'total'
        The (sub)total value(s) must be of the same type as the exisiting variable
    The (sub)totals must be separated by an exclamation point (!)
    The (sub)total values are then used in the TRTVAR_PRELOADFMT (see below)


TRTVAR_PRELOADFMT = the format which assigns the order of the treatments across the
page.  If no
    format is listed, SAS will generate a warn-ing that PRELAODFMT will have no
effect,
    ie, treatment groups which are not already in the data will not be
created/dummied.
    Also, note that any (sub)total groups should be defined here.
    (default = &_default_trtvar_preloadfmt) (OPTIONAL - *but highly recommended)

    For example, as above, the values of TRTVAR are 'A' 'B' and 'C' and the
analysis calls
    for the columns to be displayed as
        'A', 'B', 'Subtotal of A&B', 'C', 'Total'
    set the format as
      value $_trt <<<--fmtname of user choice
        'A' = '1'
        'B' = '2'
```

```
        'subtotal' = '3'
        'C' = '4'
        'total' = '5'

        NOTE - 'subtotal' and 'total' were created with DEFINE_TOTAL_GROUPS.  User
can pick
        whatever makes sense.

USUBJID = the variable(s) used to uniquely identify a subject.
(default=&_default_usubjid) (REQUIRED)

SUBGROUP = subgroups variable(s) to use in the BY statement of the PROC MEANS.
  (default = blank) (OPTIONAL)

BIGN_IN_DATA =  name of "BIGN" dataset.  This dataset will be used to determine
which subgroups will
    be displayed in final output. (default = BIGN) (REQUIRED)

    For example:

    1) You are trying to create a SAE table of the Safety population and
subgrouping by COUNTRY.

    2) There are subjects from Nepal in the Safety population, but none of them
have any SAEs,
       and it is required that there be a page for Nepal in the final output that
states,
       something to the effect "No subjects met criteria" or displays zeros for the
       "Number of Subjects with SAEs" row.


BYVARS = by variable to obtain counts (typically LBTESTCD VISIT)- space delimited.
    (default = Blank) (OPTIONAL)

BYVARS_PRELOADFMT =  the formats associated with BYVARS to preload in PROC MEANS -
space delimited.
        If do not want a format with a respective BYVARS then use the indicator
word BLANK.  For example,
        if BYVARS = LBTESTCD VISIT and there is no preload format associated with
BYVARS but there is a preload
        format associaged with VISIT, called VISFMT, then call would be:

                        ,BYVARS = LBTESTCD VISIT
                        ,BYVARS_PRELOADFMT = BLANK VISFMT

    If format is BLANK, the BYVAR will be used in the BY statement rather than a
CLASS statement in PROC MEANS.
        This format serves two main functions:

    1) it preloads, meaning that zero counts can be generated even if there are no
```

```
subjects that
    meet the criterion for a particular row.

    2) The format is used for display purposes.

        Multilabel formats are supported but user should ascertain results are as
expected.

    (default = blank) (OPTIONAL)

BYVARS_SORT_ORDER = Value should be I, F, N.
    I = sort by internal value of BY variable.
    F = sort by formatted value of BY variable.
        N = NOTSORTED option used when creating format specified in
BYVARS_PRELOADFMT - BY variable will be sorted in same order
                as when format was created.

    If parameter is F, but PRELOADFMT is BLANK, then defaults to I - an alert will
be
    issued.

        If parameter is N, but PRELOADFMT was created without using the NOTSORTED
option, then an alert
        will be issued and parameter will default to I.

    (default = I) (OPTIONAL)

BYVARS_SORT_ASC_OR_DESC = (D)escending or (A)scending, specify in which way the
    BY variable should be sorted. Default is ascending.
        (default = A(scending)) (OPTIONAL)


CATVAR = categorical variable to obtain counts - only one allowed.
    (default = Blank) (REQUIRED)

CATVAR_PRELOADFMT =  the format associated with CATVAR to preload in PROC MEANS.
If blank, then
    CATVAR will be used in the BY statement rather than a CLASS statement in PROC
MEANS. This format serves
    two main functions:

    1) it preloads, meaning that zero counts can be generated even if there are no
subjects that
    meet the criterion for a particular row.

    2) The format is used for display purposes.

    If the intent is to use the format for only the second purpose (display) then
you can remove the zero rows by setting
    the parameter DISPLAY_ZERO_ROW = No.  Multilabel formats are supported but user
```

```
should ascertain results are as expected.

    (default = blank) (OPTIONAL)

DISPLAY_ZERO_ROW = set to N(o) to remove rows that have zero counts for all
treatment groups.
    (default = Yes) (OPTIONAL)

CATVAR_SORT_ORDER = Value should be I, F, N.
    I = sort by internal value of CATVAR.
    F = sort by formatted value of CATVAR.
        N = NOTSORTED option used when creating format specified in
CATVAR_PRELOADFMT - CATVAR will be sorted in same order
                as when format was created.

    If goal is to sort by the numeric frequency of the categories, the macro
MR_ORDER_NESTED
    should be called after calling this macro to sort by the numeric frequency and
recalculate
    _SECTION_ and _ORDER1_.

    If parameter is F, but CATVAR_PRELOADFMT is blank, then defaults to I - an
alert will be
    issued.

        If parameter is N, but CATVAR_PRELOADFMT was created without using the
NOTSORTED option, then an alert
        will be issued and parameter will default to I.

    (default = I) (REQUIRED)

CATVAR_SORT_ASC_OR_DESC = (D)escending or (A)scending, specify in which way the
    categorical variable should be sorted. Default is ascending.
        (default = A(scending)) (REQUIRED)

REORDER_WHERE = specify the records that you like to reorder differently than above
        (default = blank) (OPTIONAL)

REORDER_FIRST_LAST = specify (F)irst or (L)ast for the location that you would like

    the records identified by REORDER_WHERE to fall.
    (default = blank) (REQUIRED, if REORDER_WHERE is populated)


_SECTION_ = section number of row - used to create blank lines between sections of
    final table. Should be a positive integer.
    (default = 1) (REQUIRED)
```

```
INDENT_COLUMN1 = Number of spaces to indent first column of categorical count rows.

    If blank:
        - default = 0 if parameter &header is not populated.
        - default = &header_indent + 2 if parameter &header is populated.

    (default = Blank) (OPTIONAL, but highly recommended)


HEADER = text of header to be displayed above count row.
    (default = blank) (OPTIONAL)

HEADER_INDENT = number of spaces to indent header.
    (default = 0) (REQUIRED)


CATVAR_FIRST_OR_LAST = values should be either F(irst) or L(ast)
    (default = blank) (Optional)

    F(irst) = The record with the least value of CATVAR per subject will be kept
for obtaining
              subject level counts. Note that if there are records with missing
CATVAR in the input
              dataset, then these records will be kept.  If this is not desired,
then user will need
              to either pre-process the input dataset or remove records with
missing CATVAR using
              the IN_WHERE parameter.

    L(ast) = The record with the highest value of CATVAR per subject
             will be kept for obtaining subject level counts.

    If missing, parameter will have no effect on which record is kept for counting.



GET_SUBJ_COUNT =  set to Y(es) if count of number of subject is desired.
                  If set to Y(es) then variable &COUNT_SUBJ_VAR will be in the
final dataset.
    (default = Yes) (OPTIONAL)

GET_EVENT_COUNT =  set to Y(es) if count of number of events (as opposed to count
of number of
                   subjects with event) is desired for CATVAR. If set to Y(es) then
variable
                   &COUNT_EVENT_VAR will be in the final dataset.
    (default = No) (OPTIONAL)

COUNT_SUBJ_VAR = name of subject level count variable
    (default = N)
```

```
COUNT_EVENT_VAR = name of event level count variable
    (default = N_EVENT)



CATVAR_ACROSS =  set to Y(es) if CATVAR should be displayed across on a page,
    rather than vertically (underneath each nested variable).  If Y(es) then text
of HEADER
    will be in COL1. Indentation will be according to parameter HEADER_INDENT, not
    INDENT_COLUMN1.
    (default = No) (OPTIONAL)

COUNT_VAR = name of count variable
    (default = N)

OUT_DATA = the output data set created.
    (default = CATEGORICAL_COUNT) (REQUIRED)

HELP = set to Y(es) to output helpful hints to the log
    (Default = &_default_help)

DEBUG = set to Y(es) to save all intermediate datasets and to turn on mprint. Set
to NO to delete
    all intermediate datasets and to not turn on mprint (maintains original option
setting)
    (Default = &_default_debug)


**************************************************************************;

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


%**** STEP 1 - PARAMETER CHECKS;

%global &sysmacroname._RC;
%let &sysmacroname._RC = 0;

%let abort = no;
%mu_help_debug;

%* get current setting of mprint and if set to DEBUG mode turn mprint on;


%if "&bign_in_data" eq "" %then %do;
```

```
    %let bign_in_data = BIGN;
%end;

%mu_check_data_and_var_exist(
data_to_check = &bign_in_data
,vars_to_check_in_all_data =
,vars_to_check_in_respective_data = &subgroup &trtvar
,abort_if_does_not_exist = yes
,help = no
);

%* 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
   ) *;


%* check if required parameters USUBJID TRTVAR CATVAR and IN_DATA are blank;
%mu_check_req_parameters(
 parameters_to_check = usubjid trtvar catvar in_data
,help = no
);

%* check for invalid IN_DATA or invalid variables in IN_DATA;
%mu_check_data_and_var_exist(
data_to_check = &in_data
,vars_to_check_in_all_data =
,vars_to_check_in_respective_data = &usubjid &subgroup &catvar &trtvar &byvars
,abort_if_does_not_exist = yes
,help = no
);



%md_byvar_check(in_data=&in_data
               ,byvars=&byvars
               ,byvars_preloadfmt=&byvars_preloadfmt
               ,byvars_sort_order=&byvars_sort_order
               ,byvars_sort_asc_or_desc=&byvars_sort_asc_or_desc);



%** check for missing TRTVAR_PRELOADFMT;
```

```
%if &trtvar_preloadfmt eq %str( ) %then %do;
    %put
------------------------------------------------------------------------------
-----------;
    %put ALERT_I: No format has been specified in the TRTVAR_PRELOADFMT parameter.;
    %put ALERT_I: If %upcase(&TRTVAR) is not already formatted PRELOADFMT will have
no effect.;
    %put
------------------------------------------------------------------------------
-----------;
%end;
%else %do;
    data _null_;
        call symput("trtvar_preloadfmt", compress("&trtvar_preloadfmt", '.'));
    run;
%end;

%if &catvar_preloadfmt ne %str( ) %then %do;

    data _null_;
        call symput("catvar_preloadfmt", compress("&catvar_preloadfmt", '.'));
    run;

%end;




%* populate with defaults if certain parameters are blank;

%if "&header" ne "" %then %do;
    %if "&header_indent" eq "" %then %do;
        %let header_indent = 0;
    %end;
%end;

%if "&indent_column1" eq "" %then %do;
    %if "&header" eq "" %then %do;
        %let indent_column1 = 0;
    %end;
    %else %do;
        %let indent_column1 = %eval(&header_indent + 2);
    %end;
%end;

%if "&display_zero_row" eq "" %then %do;
    %let display_zero_row = Y;
%end;
```

```
%if "&out_data" eq "" %then %do;
    %let out_data = CATEGORICAL_COUNT;
%end;

%if "&catvar_sort_order" eq "" %then %do;
    %let catvar_sort_order = I;
%end;

%if %upcase(%substr(&catvar_sort_order, 1, 1)) = F and "&catvar_preloadfmt" eq ""
%then %do;
    %put ALERT_I: Parameter CATVAR_PRELOADFMT is missing, but parameter
CATVAR_SORT_ORDER;
    %put ALERT_I: is &catvar_sort_order.  Since there is no format provided, the
categorical;
    %put ALERT_I: variable cannot be sorted by the formatted value. Categorical
variable;
    %put ALERT_I: will be sorted by internal value.;
    %let catvar_sort_order = I;
%end;

%if %upcase(%substr(&catvar_sort_order, 1, 1)) ne I and
    %upcase(%substr(&catvar_sort_order, 1, 1)) ne F and
        %upcase(%substr(&catvar_sort_order, 1, 1)) ne N %then %do;
        %let catvar_sort_order = I;
%end;

%if "&catvar_sort_asc_or_desc" eq "" %then %do;
    %let catvar_sort_asc_or_desc = A;
%end;

%if "&_section_" eq "" %then %do;
    %let _section_ = 1;
%end;

%if "&catvar_first_or_last" eq "" %then %do;
    %let catvar_first_or_last = I; %*I is for ignore;
%end;

%if "&subgroup" ne "" %then %do;
    %mu_wordscan(string=&subgroup,root=subgroupvar, numw=num_subgroup);
%end;

%if "&catvar_across" eq "" %then %do;
    %let catvar_across = N;
%end;

%if "&count_subj_var" eq "" %then %do;
    %let count_var = N;
%end;
```

```
%if "&count_event_var" eq "" %then %do;
    %let count_var = N_EVENT;
%end;

%if "&get_subj_count" eq "" %then %do;
    %let get_subj_count = Y;
%end;

%if "&get_event_count" eq "" %then %do;
    %let get_event_count = N;
%end;









%let temp_macro_name = &sysmacroname;

%macro check_integer(param_name =, value=) ;
 %let RE1=%sysfunc(prxparse(#\D#)) ;
 %put &RE1. ;
  %if %sysfunc(prxmatch(&re1.,&value))>0 %then %do ;
      %put ALERT_P: A NON-DIGIT character was found for parameter = &param_name
Value found was &value..;
      %put ALERT_P: &param_name must be a positive integer. Macro will abort;
      %let &temp_macro_name._RC = 1;
  %end ;
  %else %do ;
    %put INTEGER value for &param_name found! ;
  %end ;
%mend ;

%check_integer(param_name=_SECTION_, value=&_section_);

%if &&&sysmacroname._RC = 0 %then %do;
    %if %upcase(%substr(&get_subj_count,1,1)) = Y and
        %upcase(%substr(&get_event_count,1,1)) = Y and
        %upcase(&count_subj_var) = %upcase(&count_event_var) %then %do;

            %let &SYSMACRONAME._RC = 2;
            %put ------------------------------------------------------------;
            %put ALERT_P: &SYSMACRONAME - Both parameters COUNT_SUBJ_VAR and
COUNT_EVENT_VAR;
            %put ALERT_P: &SYSMACRONAME - HAVE THE SAME VALUE of &COUNT_SUBJ_VAR.;
            %put ALERT_P: &SYSMACRONAME - These two parameters must have unique
```

```
values.;
            %put ALERT_P: &SYSMACRONAME - Program will ABORT.;
            %put ------------------------------------------------------------;
    %end;
%end;

%if &&&sysmacroname._RC = 0 %then %do;
    %if %upcase(%substr(&get_subj_count,1,1)) = N and
        %upcase(%substr(&get_event_count,1,1)) = N %then %do;

            %let &sysmacroname._RC = 3;
            %put ------------------------------------------------------------;
            %put ALERT_P: &SYSMACRONAME - Both parameters GET_SUBJ_COUNT and
GET_EVENT_COUNT are No.;
            %put ALERT_P: &SYSMACRONAME - At least one must be Yes.;
            %put ALERT_P: &SYSMACRONAME - Program will ABORT.;
            %put ------------------------------------------------------------;
    %end;
%end;


%if &&&sysmacroname._RC = 0 %then %do;
    %* check for missing REORDER_FIRST_LAST when REORDER_WHERE is populated;
    %if %upcase(&reorder_where) ne %str( ) %then %do;
        %if &reorder_first_last = %str( ) %then %do;
            %let &sysmacroname._RC = 4;
            %put ------------------------------------------------------------;
            %put ALERT_P: REORDER_WHERE is not missing but parameter
REORDER_FIRST_LAST is missing.;
            %put ALERT_P: Program will ABORT.;
            %put ------------------------------------------------------------;
        %end;
    %end;
%end;




%if &&&sysmacroname._RC = 0 %then %do;

    %put ------------------------------------------------------------;
    %put USER NOTE: &SYSMACRONAME - Using dataset &IN_DATA for counting;
    %put ------------------------------------------------------------;

%*** STEP 2 - GET THE DATA,CREATE THE DUMMY COUNTING VARIABLE, APPLY FORMATS,
CHECK FOR MISSING VALUES ***;
```

```
    data dsin;
        set &in_data;
        %if %sysevalf(%superq(in_where)=,boolean)=0 %then %do; where &in_where;
%end;
        retain _counter 1;
        %if "&trtvar_preloadfmt" ne "" %then %do;
            format &trtvar &trtvar_preloadfmt..;
        %end;
        %if "&catvar_preloadfmt" ne "" %then %do;
            format &catvar &catvar_preloadfmt..;
        %end;
        %if "&byvars" ne "" and "&byvars_preloadfmt" ne "" %then %do;
            %do i=1 %to &num_byvars;
                %if "&&fmt&i" ne "BLANK" %then %do;
                        format &&byvar&i &&fmt&i... ;
                %end;
            %end;
        %end;
    run;

    data dsin;
        set dsin;
        %if "&subgroup" ne "" %then %do;
            %do i = 1 %to &num_subgroup;
                if missing(&&subgroupvar&i) then put "ALERT_I: SUBGROUP variable
%upcase(&&subgroupvar&i) is missing at record " _n_ ;
            %end;
        %end;

        if missing(&trtvar) then put "ALERT_I: TRTVAR variable %upcase(&trtvar) is
missing at record " _n_  "Record will be removed before proceeding";
        if missing(&trtvar) then delete;

        %if "&catvar_preloadfmt" ne "" %then %do;
            new_catvar = put(&catvar,&catvar_preloadfmt..);
        %end;
    run;









    %if "&catvar_preloadfmt" ne "" %then %do;

        %* determine if CATVAR_PRELOADFMT is a mulilabel format or not.  If it is,
         then add mlf option in class statement in PROC MEANS below.;
```

```
        %* also determine one possible value of the catvar variable for situations
with 0 records.;

        proc format lib=work

cntlout=catvar_format(where=(fmtname=upcase(compress("&catvar_preloadfmt",'$'))));
        run;

        data catvar_mlf;
            set catvar_format;
            if _n_=1;
            call symputx('catvar_start',start);
            if index(hlo,'M') gt 0;
        run;

        %mu_nobs(catvar_mlf);




                %* determine if CATVAR_PRELOADFMT is using NOTSORTED option;

                %if %upcase(%substr(&catvar_sort_order, 1, 1)) = N %then %do;
                data catvar_notsorted;
                    set catvar_format;
                    if index(hlo,'S') gt 0;
                run;

                        %mu_nobs(catvar_notsorted);

                        %if &catvar_notsorted_nobs = 0 %then %do;
                                %put ALERT_I: &SYSMACRONAME: Parameter
CATVAR_SORT_ORDER has N (NOTSORTED) specified. However, ;
                                %put ALERT_I: &SYSMACRONAME: the format
CATVAR_PRELOADFMT was not created using the NOTSORTED option.;
                                %put ALERT_I: &SYSMACRONAME: option.
CATVAR_SORT_ORDER will default to I (INTERNAL).;
                                %let catvar_sort_order = I;
                        %end;
                %end;

    %end;



    %if "&trtvar_preloadfmt" ne "" %then %do;
```

```
        %* determine one possible value of the catvar variable for situations with
0 records.;

        proc format lib=work

cntlout=trtvar_format(where=(fmtname=upcase(compress("&trtvar_preloadfmt",'$'))));
        run;

        data trtvar_format;
            set trtvar_format;
            if _n_=1;
            call symputx('trtvar_start',start);
        run;

    %end;









    %mu_nobs(dsin);

    %* if input dataset has no observations after subsetting then create dummy
dataset
        with count zero;

        %if &dsin_nobs eq 0 %then %do;

        %put NOTE: Dataset &in_data contains 0 observations after subsetting.;

        %* get format associated with subgroup variable;

        %if "&subgroup" ne "" %then %do;
            %mu_var_attributes(datasets=&in_data, variables=&subgroup);

            data _null_;
                call symput("muva_in_data", tranwrd("&in_data", ".", "_"));
            run;

            %do i = 1 %to &num_subgroup;
                %put Macro variable &muva_in_data._&&subgroupvar&i.._FMT resolves
to &&&&&muva_in_data._&&subgroupvar&i.._FMT  ;
            %end;
        %end;
```

```
        *Bring in input dataset without subsetting and later replace all counts
with 0.;

        proc sort data=&in_data(keep = &subgroup &byvars &usubjid &catvar &trtvar)
out=_dsin nodupkey;
            by &subgroup &byvars &catvar &trtvar;
        run;



        %mu_nobs(_dsin);

                %if &_dsin_nobs eq 0 %then %do;

            *If non-subsetted input data also contains zero observations then fool
program to think that data exists.;

            %if "&subgroup" ne "" %then %do;

                proc sort data = &bign_in_data(keep = &subgroup) out = dummy;
                    by &subgroup;
                run;

            %end;


            %mu_var_attributes(
             datasets=_dsin
            ,variables= &usubjid &catvar &trtvar
            ,debug=N
            ,help=N
            );
                %put Macro variable _dsin_&usubjid._len resolves to
&&_dsin_&usubjid._len  ;
                %put Macro variable _dsin_&catvar._len resolves to
&&_dsin_&catvar._len  ;
                %put Macro variable _dsin_&trtvar._vartyp resolves to
&&_dsin_&trtvar._vartyp  ;

            data dsin;
                %if "&subgroup" ne "" %then %do;
                    set dummy;
                    format %do i = 1 %to &num_subgroup; &&subgroupvar&i.
&&&&&muva_in_data._&&subgroupvar&i.._FMT    %end; ;
                %end;
                %if "&&_dsin_&usubjid._vartyp" eq "C" %then %do;
                    length &usubjid $ &&_dsin_&usubjid._len ;
                    &usubjid='A';
                %end;
                %else %do;
```

```
        &usubjid=1;
    %end;
    label &usubjid = "&&_dsin_&usubjid._lbl";

    %if "&&_dsin_&catvar._vartyp" eq "C" %then %do;
        length &catvar $ &&_dsin_&catvar._len;
        %if "&catvar_preloadfmt" ne "" %then %do;
            &catvar="&catvar_start";
        %end;
        %else %do;
            &catvar='A';
        %end;
    %end;
    %else %do;
        %if "&catvar_preloadfmt" ne "" %then %do;
            &catvar=&catvar_start;
        %end;
        %else %do;
            &catvar=1;
        %end;
    %end;
    label &catvar = "&&_dsin_&catvar._lbl";

    %if "&&_dsin_&trtvar._vartyp " eq "C" %then %do;
        length &trtvar $ &&&trtvar._len;
        %if "&trtvar_preloadfmt" ne "" %then %do;
            &trtvar="&trtvar_start";
        %end;
        %else %do;
            &trtvar='A';
        %end;
    %end;
    %else %do;
        %if "&trtvar_preloadfmt" ne "" %then %do;
            &trtvar=&trtvar_start;
        %end;
        %else %do;
            &trtvar=1;
        %end;
    %end;
    label &trtvar = "&&_dsin_&trtvar._lbl";

run;

data dsin;
    set dsin;
    retain _counter 1;
    %if "&trtvar_preloadfmt" ne "" %then %do;
        format &trtvar &TRTVAR_PRELOADFMT..;
    %end;
```

```
                %if "&catvar_preloadfmt" ne "" %then %do;
                    format &catvar &catvar_preloadfmt..;
                    new_catvar = put(&catvar,&catvar_preloadfmt..);
                %end;
            run;

                        %if "&byvars" ne "" and "&byvars_preloadfmt" ne "" %then
%do;
                                %do i=1 %to &num_byvars;
                                        %if "&&fmt&i" ne "BLANK" %then %do;
                                                proc format lib=work

cntlout=byvar_format(where=(fmtname=upcase(compress("&&fmt&i",'$'))));
                                                run;

                        data byvar_format;
                                                        set byvar_format;
                                                        &&byvar&i = start;
                                                        for_merge=1;
                                                        keep  &&byvar&i for_merge;
                        run;

                                                data dsin;
                                                        set dsin;
                                                        for_merge=1;
                                                run;

                        proc sql noprint;
                                create table dsin_ as
                                select a.*, b.&&byvar&i
                                from dsin a, byvar_format b
                                where a.for_merge = b.for_merge;
                        quit;

                        data dsin(drop=for_merge);
                                set dsin_;
                        run;
                                        %end;
                                        %else %do;
                                                proc contents data=&in_data
out=contents(keep=name type) noprint;
                                                run;

                                                data contents;
                                                        set contents;
                                                        where
upcase(name)=upcase("&&byvar&i");
                                                        call
symputx('vartype',compress(put(type,best.)));
                                                run;
```

```
                                                data dsin;
                                                        set dsin;
                                                        %if &vartype = 1 %then %do;
                                                                &&byvar&i = 1;
                                                        %end;
                                                        %else %do;
                                                                &&byvar&i = 'A';
                                                        %end;
                                                run;
                                        %end;
                                %end;
                        %end;

                        %else %if "&byvars" ne "" and "&byvars_preloadfmt" eq ""
%then %do;
                                %do i=1 %to &num_byvars;
                                        proc contents data=&in_data
out=contents(keep=name type) noprint;
                                        run;

                                        data contents;
                                                set contents;
                                                where
upcase(name)=upcase("&&byvar&i");
                                                call
symputx('vartype',compress(put(type,best.)));
                                        run;

                                        data dsin;
                                                set dsin;
                                                %if &vartype = 1 %then %do;
                                                        &&byvar&i = 1;
                                                %end;
                                                %else %do;
                                                        &&byvar&i = 'A';
                                                %end;
                                        run;
                                %end;
                        %end;
                %end;

                %else %do;
            data dsin;
                set _dsin;
                retain _counter 1;
                %if "&trtvar_preloadfmt" ne "" %then %do;
                    format &trtvar &TRTVAR_PRELOADFMT..;
                %end;
                %if "&catvar_preloadfmt" ne "" %then %do;
```

```
                    format &catvar &catvar_preloadfmt..;
                    new_catvar = put(&catvar,&catvar_preloadfmt..);
                %end;
                        %if "&byvars" ne "" and "&byvars_preloadfmt" ne "" %then
%do;
                            %do i=1 %to &num_byvars;
                                %if "&&fmt&i" ne "BLANK" %then %do;
                                        format &&byvar&i &&fmt&i... ;
                                %end;
                            %end;
                        %end;
            run;
        %end;
        %end;
```

```
%*** STEP 3 - CREATE SUBTOTALS AND TOTAL GROUPS AS REQUESTED ***;

    *get the type and length of the treatment variable;
    %let dsid_in_data = %sysfunc(open(dsin));
    %let trtvar_type = %sysfunc(vartype(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &trtvar))));
    %let trtvar_length = %sysfunc(varlen(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &trtvar))));
    %let close_in_data = %sysfunc(close(&dsid_in_data));

    %if &define_total_groups eq %str( ) or %index(&define_total_groups, =) eq 0
%then %do;
        %put ALERT_I:
------------------------------------------------------------------------------
--;
        %put ALERT_I: DEFINE_TOTAL_GROUPS paramenter is missing or does not specify
a total.;
        %put ALERT_I: No subtotal and/or total groups will be calculated.;
        %let define_total_groups = ;
        %if &help eq Y %then %do;
            %put HELP: To define subtotal and/or total groups enter the definition
of the group as:;
            %put HELP: list of treatments = new group ! list of treatments = new
group.;
            %put HELP: For example: TRTC is in the data with values of 'A' 'B' and
'C' and the analysis;
            %put HELP: calls for a subtotal of 'A' and 'B' along with an overall
total of all three treatments.;
```

```
            %put HELP: set DEFINE_TOTAL_GROUPS = 'A' 'B' = 'subtotal' ! 'A' 'B' 'C'
= 'total';
            %put HELP: The left side of the equals sign must be in the syntax of
the actual data and will be;
            %put HELP: used in an IN statement, ie, TRTC in ('A' 'B')  (no
commas!).;
            %put HELP: The right side of the equals sign will be the value of the
new group and must be of the same;
            %put HELP: type as the existing treatment group variable.;
            %put HELP: Separate each subtotal/total definition with the exclamation
point (!);
            %put HELP: Make sure to include these new groups in the value statement
of the format that will be ;
            %put HELP: assigned to the TRTVAR_PRELOADFMT parameter.;
        %end;
        %put ALERT_I:
------------------------------------------------------------------------------
--;
    %end;
    %else %do;
        %mu_wordscan(string=&define_total_groups, root=total, numw=numtotals,
delim=!)
        %do i = 1 %to &numtotals;
            %mu_wordscan(string=&&total&i, root=side&i._, numw=numside,
delim=%str(=))
        %end;
    %end;


        %if &define_total_groups ne %str( ) %then %do;
        ** START CROSSOVER SUBTOTAL / TOTAL GROUPS;
            %if &trtvar_type eq C %then %do;
                * may need to redefine the length of the treatment variable -
consider the new treatment group names;
                data _null_;
                    call symput('new_trtvar_length',
                            trim(left(put(max(%do i = 1 %to &numtotals;
length(&&side&i._2), %end; &trtvar_length), 8.))));
                run;
            %end;

            %* cycle through each definition of subtotal/total and create new
observations for each subject in each group;
            %do i = 1 %to &numtotals;
                * keep one record per subject per group of treatments;
                proc sort data=dsin out=total&i(drop=&trtvar);
                    by &usubjid;
                    where &trtvar in (&&side&i._1);
                run;
```

```
            * create the new treatment group;
            data total&i;
                %if &trtvar_type eq C %then %do;
                    length &trtvar $ &new_trtvar_length;
                %end;
                set total&i;
                retain &trtvar &&side&i._2;
            run;
        %end;

        *set the new treatment groups together with the original data;
        data dsin;
            %if &trtvar_type eq C %then %do;
                length &trtvar $ &new_trtvar_length;
            %end;
            set dsin %do i = 1 %to &numtotals; total&i %end; ;
        run;

        proc sort data=dsin;
            by &subgroup &usubjid &trtvar;
        run;

    *END OF SUBTOTAL / TOTAL GROUPS;
    %end;




%*** STEP 4 - COUNT;

    %* keep records with highest or lowest value of CATVAR per subject;

    %if %upcase(%substr(&catvar_first_or_last, 1, 1)) = F %then %do;
        proc sort data=dsin;
            by &subgroup &byvars &trtvar &usubjid &catvar;
        run;

        proc sort data=dsin out=data_for_count nodupkey;
            by &subgroup &byvars &trtvar &usubjid;
        run;
    %end;

    %else %if %upcase(%substr(&catvar_first_or_last, 1, 1)) = L %then %do;
        proc sort data=dsin;
            by &subgroup &byvars &trtvar &usubjid descending &catvar;
```

```
    run;

    proc sort data=dsin out=data_for_count nodupkey;
        by &subgroup &byvars &trtvar &usubjid;
    run;
%end;

%else %if "&catvar_preloadfmt" ne "" %then %do;
    proc sort data=dsin out=data_for_count nodupkey;
        by &subgroup &byvars &trtvar &usubjid new_catvar;

    run;
%end;

%else %do;
    proc sort data=dsin out=data_for_count nodupkey;
        by &subgroup &byvars &trtvar &usubjid &catvar;
    run;
%end;

    proc sort data=data_for_count;
        by &subgroup
                %if "&byvars" ne "" %then %do;
                        %do i=1 %to &num_byvars;
                                %if "&&fmt&i" eq "BLANK" %then %do;
                                        &&byvar&i
                                %end;
                        %end;
                %end;
                &catvar;
    run;

proc means data=data_for_count noprint nway completetypes missing;
    by &subgroup
            %if "&byvars" ne "" %then %do;
                    %do i=1 %to &num_byvars;
                            %if "&&fmt&i" eq "BLANK" %then %do;
                                    &&byvar&i
                            %end;
                    %end;
            %end;
            %if "&catvar_preloadfmt" eq "" %then %do;
                    &catvar
            %end;
            ;

            %if "&byvars" ne "" %then %do;
                    %do j=1 %to &num_byvars;
                            %if "&&fmt&j" ne "BLANK" %then %do;
                                    class &&byvar&j / preloadfmt
```

```
                                    %if &&byvar_mlf&j._nobs gt 0 %then %do;
                                            mlf
                                    %end;
                                    ;
                            %end;
                    %end;
            %end;

    %if "&catvar_preloadfmt" ne "" %then %do;
          class &catvar / preloadfmt
        %if &catvar_mlf_nobs gt 0 %then %do;
            mlf
        %end;
        ;
    %end;
    class &trtvar / preloadfmt mlf;
    var _counter;
    output out=cat_count(drop=_type_ _freq_) n=n ;
run;


%if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;

    proc sort data=dsin;
        by &subgroup
                    %if "&byvars" ne "" %then %do;
                            %do i=1 %to &num_byvars;
                                    %if "&&fmt&i" eq "BLANK" %then %do;
                                            &&byvar&i
                                    %end;
                            %end;
                    %end;
                    &catvar;
    run;

    proc means data=dsin noprint nway completetypes missing;
        by &subgroup
                    %if "&byvars" ne "" %then %do;
                            %do i=1 %to &num_byvars;
                                    %if "&&fmt&i" eq "BLANK" %then %do;
                                            &&byvar&i
                                    %end;
                            %end;
                    %end;
                    %if "&catvar_preloadfmt" eq "" %then %do;
                            &catvar
                    %end;
                    ;

                    %if "&byvars" ne "" %then %do;
```

```
                                %do j=1 %to &num_byvars;
                                        %if "&&fmt&j" ne "BLANK" %then %do;
                                                class &&byvar&j / preloadfmt
                                                %if &&byvar_mlf&j._nobs gt 0 %then
%do;
                                                        mlf
                                                %end;
                                                ;
                                        %end;
                                %end;
                        %end;

            %if "&catvar_preloadfmt" ne "" %then %do;
                class &catvar / preloadfmt
                %if &catvar_mlf_nobs gt 0 %then %do;
                    mlf
                %end;
                ;
            %end;
            class &trtvar / preloadfmt mlf;
            var _counter;
            output out=cat_event_count(drop=_type_ _freq_) n=n_event ;
        run;

                proc sort data=cat_count;
                        by &subgroup
                        %if "&byvars" ne "" %then %do;
                                %do i=1 %to &num_byvars;
                                        &&byvar&i
                                %end;
                        %end;
                        &catvar &trtvar;
                run;

                proc sort data=cat_event_count;
                        by &subgroup
                        %if "&byvars" ne "" %then %do;
                                %do i=1 %to &num_byvars;
                                        &&byvar&i
                                %end;
                        %end;
                        &catvar &trtvar;
                run;


        data cat_count;
            merge cat_count cat_event_count;
                        by &subgroup
                        %if "&byvars" ne "" %then %do;
                                %do i=1 %to &num_byvars;
```

```
                                        &&byvar&i
                                %end;
                        %end;
                        &catvar &trtvar;
        run;
    %end;




    %* the following algorithm deals with situations when there are subgroups in
BIGN
       that are not present in the subsetted input data and if there are zero
observations
       after subsetting the input data;

        %if "&byvars" eq "" %then %do;

                %if "&subgroup" ne "" %then %do;

                %* get subgroups in the data;

                proc sort data=cat_count(keep=&subgroup) out=subgroups_in_data
nodupkey;
                    by &subgroup;
                run;

                %if "&catvar_preloadfmt" ne "" %then %do;
                    %* this lets us know the values for &catvar found in the data
and format, including
                       if there are records with missing &catvar - this will be
used later to get the
                       cartesian product of &subgroup and &catvar and &trtvar;

                    proc sort data=cat_count(keep=&catvar) out=catvar_dummy
nodupkey;
                        by &catvar;
                    run;

                    data catvar_dummy;
                        set catvar_dummy;
                        for_merge=1;
                    run;
                %end;

                %* get subgroups and trtvars in BIGN data;
```

```
                proc sort data = &bign_in_data(keep = &subgroup &trtvar
bign_dummy_flag)
                                    out=bign_dummy;
                    by &subgroup &trtvar;
                run;

                proc sort data=cat_count;
                    by &subgroup &trtvar;
                run;



                %if "&catvar_preloadfmt" ne "" %then %do;

                    data temp;
                        merge cat_count bign_dummy;
                        by &subgroup &trtvar;
                    run;

                    proc sort data=temp(keep=&subgroup &trtvar) out=subgroups_dummy
nodupkey;
                        by &subgroup &trtvar;
                    run;

                    data subgroups_dummy;
                        set subgroups_dummy;
                        for_merge=1;
                    run;

                    %* get cartesian product of all subgroups, catvar, and trtvar
that exist in
                       either the subsetted input data or in BIGN;

                    proc sql noprint;
                        create table dummy as
                        select a.&subgroup, b.&catvar, a.&trtvar
                        from subgroups_dummy a, catvar_dummy b
                        where a.for_merge = b.for_merge
                        order by &subgroup,&catvar,&trtvar;
                    quit;

                    proc sort data=cat_count;
                        by &subgroup &catvar &trtvar;
                    run;

                    data cat_count;
                        merge cat_count dummy;
                        by &subgroup &catvar &trtvar;
                        if n = . then n = 0;
```

```
                    %if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;

                        if n_event = . then n_event = 0;
                    %end;
                run;

                proc sort data=cat_count;
                    by &subgroup &trtvar;
                run;

                *get bign_dummy_flag;

                data cat_count;
                    merge cat_count bign_dummy;
                    by &subgroup &trtvar;
                run;

                proc sort data=cat_count;
                    by &subgroup &catvar &trtvar;
                run;
            %end;

            %else %if "&catvar_preloadfmt" eq "" %then %do;
                data cat_count;
                    merge cat_count bign_dummy;
                    by &subgroup &trtvar;
                run;
            %end;
            %end;
        %end;


        %else %if "&byvars" ne "" %then %do;

            %if "&subgroup" ne "" %then %do;

                    proc sort data=cat_count out=temp(drop=&subgroup)
nodupkey;*here;
                            by
                            %do i=1 %to &num_byvars;
                                %if "&&fmt&i" ne "BLANK" %then %do;
                                    &&byvar&i
                                %end;
                            %end;
                            %if "&catvar_preloadfmt" ne "" %then %do;
                                &catvar
                            %end;
                            &trtvar
                            ;
                    run;
```

```
                        data temp;
                                set temp;
                                %do i=1 %to &num_byvars;
                                        %if "&&fmt&i" eq "BLANK" %then %do;
                                                call missing (&&byvar&i);
                                        %end;
                                %end;
                                %if "&catvar_preloadfmt" eq "" %then %do;
                                        call missing(&catvar);
                                %end;
                                n=0;
                                %if %upcase(%substr(&get_event_count,1,1)) = Y
%then %do;
                                        n_event=0;
                                %end;
                                for_merge=1;
                        run;

                proc sort data=cat_count(keep=&subgroup) out=subgroups_in_data
nodupkey;
                    by &subgroup;
                run;

                        proc sort data = &bign_in_data(keep = &subgroup)
                                        out=subgroups_in_bign nodupkey;
                            by &subgroup;
                        run;

                        data subgroups_only_in_bign;
                                merge subgroups_in_bign(in=a)
subgroups_in_data(in=b);
                                by &subgroup;
                                if a and not b;
                                for_merge=1;
                        run;

             proc sql noprint;
                    create table dummy as
                    select a.&subgroup, b.*
                    from subgroups_only_in_bign a, temp b
                    where a.for_merge = b.for_merge;
                        quit;

                        data cat_count(drop=for_merge);
                                set cat_count dummy;
                        run;

                        proc sort data=cat_count;
                                by &subgroup
```

```
                                %do i=1 %to &num_byvars;
                                        &&byvar&i
                                %end;
                                &catvar
                                ;
                        run;



                %end;

        %end;


    %if &dsin_nobs eq 0 %then %do;

        %put USER NOTE: Dataset did not have any observations after subsetting.;
        %put USER NOTE: Counts will be zero for all treatment groups and all
subgroups (if any).;
        data cat_count;
            set cat_count;

        %if "&catvar_preloadfmt" ne "" %then %do;
            n = 0;
            %if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;
            n_event = 0;
            %end;
        %end;
        %else %do;
            &catvar='';
            n = 0 ;
            %if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;
            n_event = 0;
            %end;
        %end;
        run;

    %end;

    %* determine if CATVAR is numeric or character in dataset CAT_COUNT - note that
if
       CATVAR_PRELOADFMT is a multilabel format, CATVAR will now be character, even
if CATVAR was
       numeric in the input dataset;

    %let dsid_cat_count = %sysfunc(open(cat_count));
    %let catvar_type =
        %sysfunc(vartype(&dsid_cat_count, %sysfunc(varnum(&dsid_cat_count,
&catvar))));
    %let close_cat_count = %sysfunc(close(&dsid_cat_count));
```

```
    data cat_count;
        set cat_count;
        %if %upcase(%substr(&catvar_across, 1, 1)) ne Y %then %do;
            %if "&catvar_preloadfmt" ne "" %then %do;
                %if &catvar_mlf_nobs eq 0 %then %do;
                   COL1 = put(&catvar,&catvar_preloadfmt..);
                %end;
                %else %do;
                                        COL1 = &catvar;
                %end;
            %end;

            %else %if &catvar_type eq C %then %do;
                COL1 = &catvar;
            %end;
            %else %do;
                COL1 = put(&catvar,best.);
            %end;
            _indent_ = &indent_column1;
        %end;
        %else %do;
            COL1 = "&header";
            _indent_ = &header_indent;
        %end;
    run;

    %* remove rows with zero count - an entire group must be all zero counts in
order to remove it -
                will only work if BYVARS not populated at this point;

        %if "&byvars" eq "" %then %do;

            %if %upcase(%substr(&display_zero_row, 1, 1)) = N and
"&catvar_preloadfmt" ne "" and &dsin_nobs NE 0 %then %do;

                %if (%upcase(%substr(&catvar_across,1,1)) eq Y and &subgroup ne
%str())
                    or
                    (%upcase(%substr(&catvar_across,1,1)) ne Y )
                    %then %do;
                proc sort data=cat_count
                    out=keep(keep = &subgroup
                    %if %upcase(%substr(&catvar_across,1,1)) ne Y %then %do;
                    &catvar
                    %end;
                    ) nodupkey;
                    by &subgroup
                    %if %upcase(%substr(&catvar_across,1,1)) ne Y %then %do;
```

```
                    &catvar
                    %end;
                    ;
                    where n gt 0;
                run;

                proc sort data=cat_count;
                    by &subgroup &catvar;
                run;

                data cat_count;
                    merge cat_count keep(in=b);
                    by &subgroup
                    %if %upcase(%substr(&catvar_across,1,1)) ne Y %then %do;
                    &catvar
                    %end;
                    ;
                    if b;
                run;
                %end;

            %end;
            %else %if %upcase(%substr(&display_zero_row, 1, 1)) = N and
"&catvar_preloadfmt" ne "" and &dsin_nobs EQ 0 %then %do;
                data cat_count;
                    set cat_count;
                    n = .;
                    n_event = .;
                run;
            %end;

        %end;
    %* if subsetted data has 0 observations and displaying rows with zeros then
only keep values of CATVAR that are in the format;

    %if &dsin_nobs eq 0 and "&catvar_preloadfmt" ne "" and
%upcase(%substr(&display_zero_row, 1, 1)) eq Y %then %do;

        proc format lib=work cntlout=_formats_(keep=fmtname start label);
        run;
        proc sort data=_formats_ nodupkey;
        where upcase(compress(fmtname)) = upcase(compress("&catvar_preloadfmt",'
$.'));
        by fmtname label;
        run;

    data _null_;
        set _formats_;
        call symputx('chars_in_fmt', verify(label, '0123456789.'));
    run;
```

```
        %put chars_in_fmt = &chars_in_fmt;


        %mu_var_attributes(
         datasets=cat_count
        ,variables=&catvar
        ,debug=n
        ,help=n
        );
                %put Macro variable cat_count_&catvar._len resolves to
&&cat_count_&catvar._len  ;
                %put Macro variable cat_count_&catvar._vartyp resolves to
&&cat_count_&catvar._vartyp  ;
                %put Macro variable cat_count_&catvar._lbl resolves to
&&cat_count_&catvar._lbl  ;

        data _formats_;
            set _formats_;
            where upcase(compress(fmtname)) =
upcase(compress("&catvar_preloadfmt",' $.'));
            %if &&cat_count_&catvar._vartyp = C and &catvar_mlf_nobs gt 0 %then
%do;
              length &catvar $ &&cat_count_&catvar._len ;
              &catvar = trim(left(label));
            %end;
            %else %if &&cat_count_&catvar._vartyp = C %then %do;
              length &catvar $ &&cat_count_&catvar._len ;
              &catvar = trim(left(start));
            %end;
            %else %do;
              &catvar = input(compress(start), 8.);
            %end;
            keep &catvar;
        run;

        proc sort data=_formats_ nodupkey;
            by &catvar;
        run;

        proc sort data=cat_count;
            by &catvar;
        run;

        data check_cat_count;
        set cat_count;
        run;


        data cat_count;
            merge cat_count _formats_(in=b);
```

```
        by &catvar;
        if b;
    run;

%end;

%* order CATVAR;

%if %upcase(%substr(&catvar_sort_order, 1, 1)) = I and
    %upcase(%substr(&catvar_sort_asc_or_desc, 1, 1)) = A %then %do;

    proc sort data=cat_count;
        by &catvar;
    run;

    data cat_count;
        set cat_count;
        by &catvar;
        if first.&catvar then _order1_+1;
    run;

%end;

%if %upcase(%substr(&catvar_sort_order, 1, 1)) = I and
    %upcase(%substr(&catvar_sort_asc_or_desc, 1, 1)) = D %then %do;

    %if %upcase(%substr(&catvar_across, 1, 1)) ne Y %then %do;
        proc sort data=cat_count;
            by descending &catvar ;
        run;

        data cat_count;
            set cat_count;
            by descending &catvar;
            if first.&catvar then _order1_+1;
        run;
    %end;

    %else %do;
        data cat_count;
            set cat_count;
            _order1_ = 1;
        run;
    %end;

%end;

%if %upcase(%substr(&catvar_sort_order, 1, 1)) = F and
    %upcase(%substr(&catvar_sort_asc_or_desc, 1, 1)) = A %then %do;
```

```
    %if %upcase(%substr(&catvar_across, 1, 1)) ne Y %then %do;
        proc sort data=cat_count;
            by col1;
        run;

        data cat_count;
            set cat_count;
            by col1;
            if first.col1 then _order1_+1;
        run;
    %end;

    %else %do;
        data cat_count;
            set cat_count;
            _order1_ = 1;
        run;
    %end;

%end;

%if %upcase(%substr(&catvar_sort_order, 1, 1)) = F and
    %upcase(%substr(&catvar_sort_asc_or_desc, 1, 1)) = D %then %do;

    %if %upcase(%substr(&catvar_across, 1, 1)) ne Y %then %do;
        proc sort data=cat_count;
            by descending col1;
        run;

        data cat_count;
            set cat_count;
            by descending col1;
            if first.col1 then _order1_+1;
        run;
    %end;
    %else %do;
        data cat_count;
            set cat_count;
            _order1_ = 1;
        run;
    %end;

%end;




%if %upcase(%substr(&catvar_sort_order, 1, 1)) = N %then %do;

    data catvar_notsorted;
```

```
        set catvar_notsorted;
        length col1 $200;
        col1=label;
        _order1_+1;
        %if catvar_mlf_nobs > 0 %then %do;
            lag_label=lag(label);
            if lag_label=label then delete;
        %end;
    run;

    %if %upcase(%substr(&catvar_across, 1, 1)) ne Y %then %do;
        proc sort data=cat_count;
            by col1;
        run;

        data cat_count;
                set cat_count(rename=(col1=orig_col1));
                length col1 $200;
                col1=orig_col1;
                drop orig_col1;
        run;

        proc sort data=catvar_notsorted;
            by col1;
        run;

        data cat_count;
            merge cat_count(in=a) catvar_notsorted(keep=col1 _order1_);
            by col1;
            if a;
        run;
    %end;
    %else %do;
        data cat_count;
            set cat_count;
            _order1_ = 1;
        run;
    %end;

%end;
```

```
    %* reorder to the beginning or end, where specified in the REORDER_WHERE
parameter;

    %if %upcase(%substr(&catvar_across, 1, 1)) ne Y %then %do;

        %if %upcase(%substr(&catvar_sort_order, 1, 1)) = I or
            %upcase(%substr(&catvar_sort_order, 1, 1)) = F or
                        %upcase(%substr(&catvar_sort_order, 1, 1)) = N %then %do;

            %if "&reorder_where" ne "" %then %do;

                data cat_count;
                    set cat_count;
                    if &reorder_where then do;
                    %if %substr(%upcase(&reorder_first_last),1,1)=F %then %do;
                        _order1_=0.5;
                    %end;
                    %else %if %substr(%upcase(&reorder_first_last),1,1)=L %then
%do;
                        _order1_=999999;
                    %end;
                    end;
                run;

            %end;

        %end;

    %end;



   %md_add_byvar_order(in_data = cat_count
                      ,byvars=&byvars
                      ,out_data=cat_count
                      ,subgroup=&subgroup
                      ,byvars_preloadfmt=&byvars_preloadfmt
                      ,byvars_sort_order=&byvars_sort_order
                      ,byvars_sort_asc_or_desc=&byvars_sort_asc_or_desc
                      );


%*** STEP 5 - CREATE HEADERS;

    %* get maximum length of column1 of categorical count datasets and categorical
header;
```

```
    %if %upcase(%substr(&catvar_across, 1, 1)) ne Y and "&header" ne "" %then %do;

        data cat_count;
            set cat_count;
            if col1 = '' then col1 = '';
        run;



        %* add header rows;

        proc sort data=cat_count out=header(drop=COL1) nodupkey;
            by &subgroup &trtvar;
        run;

        data header ;
            set header;
            COL1 = "&header";
            n = . ;
            %if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;
                n_event = . ;
            %end;
            header_flag = 1;
            %if %upcase(%substr(&catvar_sort_order, 1, 1)) = I or
              %upcase(%substr(&catvar_sort_order, 1, 1)) = F or
                          %upcase(%substr(&catvar_sort_order, 1, 1)) = N %then %do;
                _order1_ = 0;
            %end;
            _indent_ = &header_indent;
        run;

        %mu_var_attributes(datasets=cat_count, variables=col1)
        %mu_var_attributes(datasets=header, variables=col1)

        data _null_;
        call symputx('max_length_col1', max(&cat_count_col1_len.,
&header_col1_len));
        run;

        data cat_count;
            length COL1 $ &max_length_col1;
            set header cat_count;
        run;


        %if "&catvar_preloadfmt" eq "" and "&subgroup" ne "" %then %do;

            proc sort data=cat_count;
                by  &subgroup;
            run;
```

```
            data cat_count;
                merge cat_count subgroups_in_data(in=b);
                by &subgroup;
                if not b and header_flag=1 then delete;
            run;

        %end;

    %end;


    %let dsid_out_data = %sysfunc(open(cat_count));
    %let bign_dummy_flag_exists = %sysfunc(varnum(&dsid_out_data,
bign_dummy_flag));
    %let close_dsid_out_data = %sysfunc(close(&dsid_out_data));

    %if &bign_dummy_flag_exists ne 0 %then %do;

        proc sort data = &bign_in_data(keep = &subgroup &trtvar bign_dummy_flag)
            out=bign_dummy;
            by &subgroup &trtvar;
        run;

        proc sort data=cat_count;
            by &subgroup &trtvar;
        run;

        data cat_count;
            merge cat_count(drop = bign_dummy_flag)  bign_dummy;
            by &subgroup &trtvar;
        run;

    %end;

    %*if subsetted data has 0 observations, then clean data;

    %if &dsin_nobs eq 0 and "&catvar_preloadfmt" eq "" %then %do;

        %if "&header" ne "" %then %do;

            data cat_count;
                set cat_count;
                if header_flag=1 then delete;
                else do;
                    col1='';
                end;
            run;

        %end;
```

```
        %else %do;

            data cat_count;
                set cat_count;
                col1='';
            run;

        %end;

    %end;

    %if %upcase(%substr(&display_zero_row, 1, 1)) = N and &dsin_nobs eq 0 %then
%do;
        data cat_count;
            set cat_count;
              col1 ='';
              n=.;
              n_event=.;
        run;

        %put ALERT_I: Since there are no records in input dataset after
subesetting;
        %put ALERT_I: and parameter DISPLAY_ZERO_ROW is set to NO, final dataset
will;
        %put ALERT_I: have blank values for COL1 and N.;
    %end;


    %* create ordering variables if CATVAR_SORT_ORDER is I(nternal) or F(ormatted)
       Otherwise user should call nested_ordering macro to derive necessary
variables
       for ordering.;

    %if %upcase(%substr(&catvar_sort_order, 1, 1)) = I or
          %upcase(%substr(&catvar_sort_order, 1, 1)) = F or
                  %upcase(%substr(&catvar_sort_order, 1, 1)) = N %then %do;

        data cat_count;
            set cat_count;
            _section_ = &_section_;
            _skipvar_ = &_section_;
        run;

        %if %upcase(%substr(&get_subj_count,1,1)) = N %then %do;

            data cat_count;
                set cat_count(drop=n);
            run;
```

```
        %end;

        proc sort data=cat_count out=&out_data(rename=(
            %if %upcase(%substr(&get_subj_count,1,1)) = Y %then %do;
                n=&count_subj_var
            %end;
            %if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;
                n_event=&count_event_var
            %end;
            ));
            by &subgroup _order1_;
        run;

    %end;

    %else %do;

        %if %upcase(%substr(&get_subj_count,1,1)) = N %then %do;

            data cat_count;
                set cat_count(drop=n);
            run;

        %end;

        proc sort data=cat_count out=&out_data(rename=(
            %if %upcase(%substr(&get_subj_count,1,1)) = Y %then %do;
                n=&count_subj_var
            %end;
            %if %upcase(%substr(&get_event_count,1,1)) = Y %then %do;
                n_event=&count_event_var
            %end;
            ));
            by &subgroup &catvar
            %if "&header" ne "" %then %do;
                descending header_flag
            %end;
            ;
        run;

    %end;

%end;


%** STEP 6 - change the starting value of _ORDER1_ as requested through
_ORDER1_START;
%if &_order1_start ne %str() and &_order1_start ne 1 %then %do;
data &out_data;
        set &out_data;
```

```
        if _n_ = 1 and "&help" = "Y" then put "ALERT_I: HELP: changing value of
_order1_ to start with &_order1_start";
        _order1_ = _order1_ + (&_order1_start - 1);
run;
%end;



%* STEP 7 - END OF PROCESS TASKS;


%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._RC = &&&sysmacroname._RC;
%PUT ------------------------------------------------ ;


%if &&&sysmacroname._RC gt 0 %then %do;
    data _null_;
        abort;
    run;
%end;


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

%exit:
%if &abort = yes %then %do;
    data _null_;
        abort;
    run;
%end;

%mend ma_count_categorical;
```