PharmaSUG 2019 - Paper BP-181

Sum Fun with Flags! Sum Any Flagged Occurrence Data with FLAGSUM and Report It with FLAGRPT or PROC REPORT

Brendan Bartley, Harvard T.H. Chan School of Public Health, Center for Biostatistics in AIDS Research (CBAR)

ABSTRACT

In 2016, CDISC created the ADaM Structure for Occurrence Data Structure Implementation Guide, which describes the data structure and content for counting any events in the CDISC world. One of the most commonly flagged SDTM domains as it moves to ADaM is the Adverse Event (AE) domain, although other domains could be flagged as they move to ADaM such as CE, CM, and MH. The concepts behind flagged records in the AE/ADAE domain (AOCCFL, AOCCSFL, AOCCPFL, AOCCIFL, AOCCSIFL, and AOCCPIFL) can apply to the creation of flags for other ADaM dataset equivalent domains (CE, CM, MH, etc.). Instead of creating several reporting macros coded specifically for individual domains, let's be flexible. We describe and demonstrate two macros, FLAGSUM and FLAGRPT, which are flexible enough to handle occurrence data in various ADaM equivalent domains. These macros have the added bonus of being capable of handling non-CDISC data, which allow users to learn and adopt CDISC ways with non-CDISC data.

INTRODUCTION

The FLAGSUM/FLAGRPT macros accomplish the following goals:

1. Count adverse events via flag variables and display them in a standard table
2. Provide the transparency and traceability that CDISC needs in this process
    a. Count events and make a dataset – FLAGSUM
    b. Report event frequencies from the dataset above - FLAGRPT

You can use these macros to report flagged occurrence data in various settings. Although this paper references CDISC terminology and assumptions, the macros are flexible enough to work with other types of data. In fact, FLAGSUM and FLAGRPT can act as a great training opportunity for those moving from non-CDISC data to CDISC data and methods.

In our organization, statisticians perform a lot of programming. Therefore, there are many well-validated macros that provide report-ready output. The aim of these macros is to reduce the need for complex programming.

DATA PREPARATION

FLAGSUM requires two datasets that match the structure of ADAE and ADSL. The ADSL dataset is straightforward. It contains one record per USUBJID and a variable containing the value for a treatment arm (ARM).

The ADAE dataset contains one record per event. Relevant variables include AESOC, AEDECOD (Preferred Term, PT, or event value), AETOXGR (grade of the event, if used), USUBJID, ARM (matching the treatment arm variable from ADSL), and the flag variables. The ADaM OCCDS IG describes 6 flags that form the basis of the rules for any flag variables to summarize in FLAGSUM.

PARAMETER TERMINOLOGY AND CONCEPTS

In Table 1, the flags are listed with their corresponding FLAGSUM parameter. As you can see, the variable entered in ALLFLG is similar to the AOCCFL or AOCCIFL variables. The first three rows in Table 1 lists the set of flags when there is no AETOXGR or equivalent event grade variable. The last three flags, ending with "IFL", are the set of flags when AETOXGR or an equivalent event grade variable is present.

Table 1. Occurrence Flag Variables

NAME	Explanation	FLAGSUM PARAMETER
AOCCFL	1st Occurrence of Any AE Flag	ALLFLG
AOCCSFL	1st Occurrence of SOC Flag	GRPFLG
AOCCPFL	1st Occurrence of PT Flag	EVTFLG
AOCCIFL	1st Max Severity/Intensity Occurrence Flag	ALLFLG
AOCCSIFL	1st Max Severity/Intensity Occurrence within a SOC Flag	GRPFLG
AOCCPIFL	1st Max Severity/Intensity Occurrence within a PT Flag	EVTFLG

Table 2 shows a listing of CDISC terms, general term(s), and brief explanations with their appropriate FLAGSUM/FLAGRPT parameter that needs this input. These items help you understand the terminology in the Parameter Map and example macro calls below.

Table 2. Parameter Terminology and Explanation

CDISC Term	General Term	Explanation	FLAGSUM/FLAGRPT Parameter and/or Table Location
ARM	TREATMENT	Any classification of a subject that comes from a dataset with one record per subject. It aids in calculating the denominator, i.e. "(N=4)", in the output table. This denominator helps calculate percentages in the table.	COLVAR
SEVERITY or INTENSITY	GRADE	Any classification of the severity/intensity of the event or term that exists in the occurrence data. This parameter and variable are optional to the user, but are common in our organization.	SUBVAR
System Organ Class or SOC	GROUP	Any group formed by combining several events or terms from the occurrence data. Counts by group originate from records flagged by AOCCSFL or AOCCSIFL depending on which set of flags are used. This flag variable is entered into the GRPFLG parameter.	GRPVAR
Preferred Term or PT	EVENT	Anything you want to count in occurrence data for a subject and show up in the table. Each of these records is flagged by AOCCPFL or AOCCPIFL depending on which flags are used. The flag variable is entered into the EVTFLG parameter.	EVTVAR

PARAMETER MAP

This map illustrates parameter names and where their values and frequencies end up in the intended output table from FLAGRPT. It also describes the various outputs that can be produced from the PCTMODE parameter. This parameter controls the display in the table, see Appendix B for more details. For example, the table below shows the default PCTMODE value display:

1. The ”Overall” row displays bolded frequencies and percentages where the frequencies come from records marked by ALLFLG.
2. The GRPVAR/GROUP rows display bolded frequencies and percentages where the frequencies come from counting records flagged by GRPFLG.
3. The EVTVAR/EVENT rows solely display frequencies from counting records flagged by EVTFLG.

Table 3. Parameter Map

KEY: **DATA PARAMETER**

PCTMODE TITLE DESCRIPTOR

	COLVAR VALUE A (N=4)			COLVAR VALUE B (N=5)			TOTITLE “TOTAL” (N=9)		
Adverse Events	**SUBVAR VALUE 3**	**SUBVAR VALUE 4**	**COLTOTITLE “All”**	**3**	**4**	**“All”**	**3**	**4**	**“All”**
ALLFLG “Overall”	**0 (0%)**	**4 (100%)**	**4 (100%)**	**1 (20%)**	**3 (60%)**	**4 (80%)**	**1 (11%)**	**7 (78%)**	**8 (89%)**
GRPVAR Gastrointestinal	**0 (0%)**	**2 (50%)**	**2 (50%)**	**1 (20%)**	**1 (20%)**	**2 (40%)**	**1 (11%)**	**3 (33%)**	**4 (44%)**
EVTVAR Nausea	1	1	2	2	0	2	3	1	4
EVTVAR Vomit	0	1	1	1	1	2	1	2	3
GRPVAR General Body	**0 (0%)**	**2 (50%)**	**2 (50%)**	**0 (0%)**	**2 (40%)**	**2 (40%)**	**0 (0%)**	**4 (44%)**	**4 (44%)**
EVTVAR Ache	0	0	0	0	1	1	0	1	1
EVTVAR Fever	0	1	1	0	1	1	0	2	2
EVTVAR Lumps	0	1	1	0	0	0	0	1	1

For data parameters in green, we list the parameter name and the value that comes from the variable entered in it. Only the columns and sub-columns under “COLVAR VALUE A” are fully annotated to describe the expected output. COLVAR and SUBVAR refer to the location of their respective value in the final table. The COLVAR parameter refers to “columns” and SUBVAR to “sub-columns”.

The items quoted in the parameter map, i.e. “TOTAL”, “All”, and “Overall”, are fixed labels and currently cannot change in FLAGSUM and FLAGRPT. If necessary, you can alter these values in the FLAGSUM output data and create your own PROC REPORT code instead of using FLAGRPT. In the case of TOTITLE:“TOTAL” and COLTITLE:“All”, these columns represent their respective overall total values for the COLVAR and SUBVAR variables.

FLAGSUM

The call to FLAGSUM below uses an ADAE dataset and produces the output table in the Parameter Map (Table 3 above). All the parameters show the default values with comments about each:

```
%flagsum(
    OUTDATA  = FLAGEND,  /* DEFAULT, BUT USER MAY CHANGE IT */
    COLDATA  = ADSL,
    COLWHERE = ,         /* CAN ENTER A WHERE STATEMENT – READ BELOW */
    COLVAR   = ARM,      /* VARIABLE LISTED REQUIRED IN ADSL AND ADAE */
```

```
    EVTDATA  = ADAE,
    GRPVAR   = AESOC,
    EVTVAR   = AEDECOD,
    SUBVAR   = AETOXGR,
    IDVAR    = USUBJID,  /* DEFAULT VALUE          */
    ALLFLG   = AOCCIFL,  /* WITHOUT SUBVAR USE AOCCFL  */
    GRPFLG   = AOCCSIFL, /* WITHOUT SUBVAR USE AOCCSFL */
    EVTFLG   = AOCCPIFL, /* WITHOUT SUBVAR USE AOCCPFL */
    PCTMODE  = 1,        /* DEFAULT VALUE SEEN IN PARAMETER MAP        */
    PCTFMT   = flgsmft4.,/* DEFAULT – DETAILS BELOW                    */
    LSTMORE  = N,        /* SET TO Y TO SHOW MORE VARIABLE PROCESSING  */
    DEBUG    = N         /* SET TO Y TO HELP ERADICATE BUGS IN PROGRAM */
);
```

There can be additional variables in both of the COLDATA and EVTDATA datasets, but those listed above are the primary variables for our macros. Again, non-CDISC studies can use this same macro.

FLAGSUM has the ability to subset data based on a WHERE statement entered in the COLWHERE parameter. This parameter intends to operate only on population flags or similar variables. Any flag variables entered in a WHERE statement must be present in both the ADSL and ADAE datasets. The ADSL dataset needs this to correctly calculate the denominator for our columns and ADAE needs these same values to count events by the same rules.

The macro notes incomplete records and prints them for review, but the macro continues to run to completion so that you can see frequencies about complete records. You can make any adjustments for uncounted records at a later time.

There are two reasons the macro uses a COLDATA dataset separate from the EVTDATA:

1. The COLDATA determines the population of legitimate IDVAR values that EVTDATA must match. Any IDVAR value from EVTDATA that is not in COLDATA will end the macro because this should never happen.
2. Denominator counts are completed without merging to the EVTDATA. PROC SQL code retrieves a list of COLVAR values. A DATA _NULL_ data step counts those records. The resulting counts are output to a macro variable whose name starts with “DENOM”. This information is added to our final dataset and used for percent calculations.

FLAGSUM creates frequencies from EVTDATA based on the occurrence flags listed in ALLFLG, GRPFLG, and EVTFLG. The macro creates an output dataset that contains frequencies for each populated flag parameter using the ODS OUTPUT statement for CrossTabFreqs with PROC FREQ code. Some special cases require additional processing to avoid problematic output. When the ALLFLG variable is without a SUBVAR, there is additional code to ensure the macro only retrieves values for the “Overall” row in your table. The GRPFLG variable needs help to capture only the GRPVAR/GROUP row frequencies. Without this processing, the final table may get double rows or incorrect results and look weird. EVTFLG has similar processing as a precaution. For more details, see Appendix E and search for ALLFLG_LOOK.

The macro sets these frequency datasets together in the same order it asks for them: ALLFLG, GRPFLG, and then EVTFLG. It then uses the internal “DENOM” macro variables to add denominator values to this data superset so it can create variables with percent values. For traceability, the macro creates a variable containing the denominator value for each calculated percent value. This value matches the denominator for the COLVAR value present in that record. Once this is complete, the macro creates reporting variables starting with “RPT”. These are important to identify the expected variables and values that will populate the final table.

The only reporting variable that does not start with “RPT” is GRPEV1COL, which contains the group and event values. The macro indents the event values in the variable so that output in the .lst file looks like the final table. The final PROC REPORT requires additional code to see this indentation in the final table. By

looking at a listing of reporting variables, you can see how a transpose of the RPTCOL and RPTSUB variables by the RPTME and GRPEV1COL values can achieve the output in the Parameter Map. You will see later how the RPTCOL and RPTSUB variables are important in the PROC REPORT code. Appendix A lists the reporting variables from FLAGSUM for the Parameter Map table. By default, the macro only prints these reporting variables along with the contents of the output dataset to the .lst file. If you want to see the PROC FREQ processing and all of the variables and values in the output dataset, set the LSTMORE parameter to Y.

FROM FLAGSUM TO FLAGRPT

Now that you have the input data for a PROC REPORT, how will you present this data in the "right" order? Our organization has a macro that creates a numeric variable using a format to allow, for example, values A, B, and C in the data to appear in order of B, C, and A in the table. There could be other data adjustments you want that change the values displayed in a table, but not the frequencies. Any of these newly created variables could be entered in the matching parameter in FLAGRPT and work as expected for its "RPT" variable sibling. You will see these "RPT" variables in a FLAGRPT call below. To create beautiful output, you must include the following statements before the PROC REPORT code from FLAGRPT or your own PROC REPORT code:

- TITLE statements
- FOOTNOTE statements
- ODS RTF or other ODS statements
  - Our workplace often utilizes the BODYTITLE and STYLE = options
- OPTIONS statement to change the orientation to landscape or control other parts of output

FLAGRPT

FLAGRPT creates PROC REPORT code with DEFINE statements that utilize the ACROSS option for the COLVAR and SUBVAR variables. This takes the long dataset from FLAGSUM and makes it wide.

Let's talk about the parameters you will recognize from FLAGSUM or are easily interpreted. Here is the FLAGRPT call with defaults listed:

```
%flagrpt(   INDATA     = flagend
          , GRPVAR     = rptgrp
          , EVTVAR     = rptevt
          , GRPEVTVAR  = grpev1col
          , COLVAR     = rptcol
          , SUBVAR     = rptsub
          , GRPEV1COLW =
          , PCTMODE    = 1
          , PCTFMT     = flgsmft4.
          , EVTTITLE   = Adverse Events
          , COLTITLE   =
          , SUBTITLE   =
          , RTFFNTSZ   =
          , DEBUG      = N
);
```

INDATA contains the name of the dataset to enter into the macro. The GRPEVTVAR parameter is new to FLAGRPT. The default variable, GRPEV1COL, comes from FLAGSUM and represents our left most column which displays occurrences as SOCs/groups and PTs/events. GRPVAR, EVTVAR, COLVAR, and SUBVAR correlate to their FLAGSUM sibling. You may wonder why we need GRPVAR and EVTVAR when we have GRPEVTVAR. They allow you to order the rows in the table. They are not displayed at all because their DEFINE statements use the NOPRINT option. RPTCOL and RPTSUB are our variables

that utilize the ACROSS option in the DEFINE statements to create the easy to read table. Below is the PROC REPORT code that resolves from the FLAGRPT macro code for the Parameter Map (the numbers, i.e. [1], in the comments correlate to explanations below the code):

```
proc report nowd data= flagend
  out= flagrptd split='$'
/* [1] MISSING VALUES DISPLAYED AS VALID VALUES FOR
   GROUP, ORDER, AND ACROSS VARIABLES */
  missing
/* [2] CREATE ALL POSSIBLE VALUE COMBINATIONS FOR ACROSS VARIABLES
   EVEN WHEN IT IS NOT IN THE INPUT DATA */
  nocompletecols
;
/* [3] DUMMY VARIABLE IS REQUIRED WHEN USING ACROSS OPTION WITH CHAR VARS */
/* ALL "RPT" VARS ARE CHARACTER */
  columns ("Adverse Events" rptgrp rptevt grpev1col)
          rptcol, rptsub, rptme dummy;
  define rptgrp    / group order= data noprint;
  define rptevt    / group            noprint;
  define grpev1col / group   " " style=[cellwidth=25 em];
/* THE IMPORTANT ACROSS VARIABLES */
  define rptcol    / across  " " order= internal;
  define rptsub    / across  " " order= internal;
  define rptme     / display " " center;
  define dummy     / noprint;

/* [4] COMPUTE BLOCK FOR dummy VARIABLE PLACES ZERO VALUES ('0' or '0 (0%)'
IN THE TABLE FOR ALL MISSING VALUES BASED ON THE PCTMODE VALUE */
  compute dummy;

/* LIST OF VARIABLES CREATED BY MACRO CODE BASED ON NUMBER OF VALUES IN
COLVAR AND SUBVAR VARIABLES */
    array zeros _c4_ _c5_ _c6_ _c7_ _c8_ _c9_ _c10_ _c11_ _c12_;
    do z= 1 to dim(zeros);
      if missing(zeros(z)) then do;

      /* PRXMATCH CODE IDS GROUP ROWS NOT INDENTED: DISPLAY '0 (0%)' */
        if prxmatch('/\s/', grpev1col) ^= 1 then do;
          zeros(z) = put(0,3. -r)||' '||'{'||strip(put(0,flgsmft4.))||'%)';
        end;
      /* SHOW '0' FOR EVENT ROWS WHICH ARE INDENTED */
      /* -r OPTION RIGHT ALIGNS VALUES IN .lst WHICH HELPS HUMAN VIEWING */
        else do;
          zeros(z) = put(0,3. -r);
        end;
      end;
    end;
  endcomp;

/* [5] PRXMATCH CODE IDS GROUP ROWS AND LEFT ALIGNS AND BOLDS THOSE ROWS */
  compute grpev1col;
    if prxmatch('/\s/', grpev1col) ^= 1 then do;
      call define(_col_,"style","style=[just=l]");
      call define(_row_,"style","style=[vjust=bottom font_weight=bold]");
    end;
/* INDENT ROWS THAT ARE NOT GROUP ROWS */
    else do;
```

```
      call define(_col_,"style","style=[leftmargin=4 em]");
    end;
  endcomp;
run;
```

Items of note from the code above:

1. The MISSING option in the PROC REPORT statement allows for missing values to be displayed as valid values for group, order, or across variables.
2. The NOCOMPLETECOLS option in the PROC REPORT statement creates all possible combinations for the values of the across variables even if not all of the combinations are in the input dataset. This is important so you see the same SUBVAR values under COLVAR values even if they do not exist in a COLVAR column.
3. The DUMMY variable is required when using the ACROSS option with character variables.
4. The compute block for the DUMMY variable and the array of zeros is to place '0' values according to the values in FLAGRPT's PCTMODE and PCTFMT parameters. The ACROSS option leaves cells blank because there was not an observation in the input dataset. These cells need a '0' value with formatting that matches the formatting from FLAGSUM. Thus, the values for PCTMODE and PCTFMT need to match those in FLAGSUM to get consistent results for the cells that equal 0.

   Because one of the goals was to keep these macros simple, it was decided that most PCTMODE values would be mapped to only four values: 0, 1, 2, or 3. It was too difficult to figure out from the input dataset how to do this correctly for other PCTMODE numbers and you can create your own PROC REPORT code if the table produced does not suit your needs. More about the compute blocks:

   a. The macro code generates the list of "_c" variables and is based on how many levels are in RPTCOL and RPTSUB.
   b. The prxmatch code looks to see if the code is indented. If not, it generates a frequency with a percentage as "0 (0%)". If it is, then just give a frequency as "0". In other words,
      i. If it is a group row or the "Overall" row then print "0 (0%)".
      ii. If it is an event row, just print "0".
      iii. This code shows when PCTMODE = 1. This part of the code changes based on the value of PCTMODE. See Appendix F to see how this happens.
5. The compute block for grpev1col creates the indent for event rows and bolds the group rows.

CONCLUSION

The FLAGSUM and FLAGRPT macros accept occurrence data structure records and count complete records for specific flag variables, regardless of whether the dataset is CDISC or non-CDISC. Other domains that have a similar data structure to AE are CE, CM, and MH. An ADSL-like dataset generates denominators and checks that all subjects are in the same population that generates the denominators. By using these rules for dataset creation, it is easy to use PROC FREQ code to count records for flag variables. The output datasets from these PROC FREQs can be concatenated to create a long output dataset. This dataset can be output with PROC REPORT code that contains code for the ACROSS option that transposes the data for expected treatment and grade values. The appendices contain further details on a few other options for FLAGRPT. The full output dataset of FLAGSUM is there as well. The full code for both macros is included and any comments and suggestions are welcome.

REFERENCES

[1] Eslinger, Jane, 2015. “An In-depth Look at PROC REPORT”, *BASUG,* Q3 Meeting on Sept. 25, 2015.

Jane Eslinger and all her papers and presentations are excellent resources for all things PROC REPORT.

[2] SAS© Help Center: Base SAS 9.4 Procedures Guide, Seventh Edition, 2017, SAS Institute, Inc.

https://documentation.sas.com/api/docsets/proc/9.4/content/proc.pdf?locale=en#nameddest=titlepage

[3] ADaM Structure for Occurrence Data (OCCDS), Version 1.0, Feb 12, 2016

ACKNOWLEDGMENTS

This work was supported by the Statistical and Data Management Center (SDMC), AIDS Clinical Trials Group (ACTG), under the National Institute of Allergy and Infectious Diseases (NIAID) grant No. UM1 AI068634. Harvard fund 114660

This work was supported by the Statistical and Data Management Center (SDMC) of the IMPAACT Leadership Group under

Eunice Kennedy Shriver National Institute Of Child Health & Human Development (NICHD) and National Institute Of Allergy And Infectious Diseases (NIAID) UMM1 AI068616, Harvard fund 114661

Thank you to wife, M, and kids, Z and B, Matt Manne, and Heather Ribaudo for allowing me the time to work on this paper and presentation. Thank you to Matt Manne and Andrew Ellingson for their help with the paper. To Leavitt Morrison and Nari Savanorke-Joyce, thank you very much for your tremendous help and support through the writing process.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Brendan Bartley
bbartley@sdac.harvard.edu

APPENDICES

APPENDIX A: FULL OUTPUT OF REPORTING VARIABLES FROM FLAGSUM

Here are a few items to note about the reporting variables below:

1. The “$” in the RPTCOL values exist as the split character in titles so that the COLVAR value is on a line above the denominator value listed.
    a. This can be seen in the Parameter Map.
2. There are no 0 values listed.
    a. These will be handled in the PROC REPORT code.
3. In RPTME, some values contain frequencies and some have frequencies and percentages.
    a. The PCTMODE parameter controls this.
4. The event values, like “Nausea”, are indented in GRPEV1COL and group values are left aligned.
5. All of these variables are character.
    a. This is important for FLAGRPT and the PROC REPORT code because it requires a DUMMY variable.

rptpct	grpev1col	rptgrp	rptevt	rptcol	rptsub	rptfrq	rptme
(100%)	Overall			A$(N=4)	4	4	4 (100%)
(100%)	Overall			A$(N=4)	All	4	4 (100%)
(20%)	Overall			B$(N=5)	3	1	1 (20%)
(60%)	Overall			B$(N=5)	4	3	3 (60%)
(80%)	Overall			B$(N=5)	All	4	4 (80%)
(11%)	Overall			TOTAL$(N=9)	3	1	1 (11%)
(78%)	Overall			TOTAL$(N=9)	4	7	7 (78%)
(89%)	Overall			TOTAL$(N=9)	All	8	8 (89%)
(20%)	Gastrointestinal	Gastrointestinal		B$(N=5)	3	1	1 (20%)
(11%)	Gastrointestinal	Gastrointestinal		TOTAL$(N=9)	3	1	1 (11%)
(50%)	Gastrointestinal	Gastrointestinal		A$(N=4)	4	2	2 (50%)
(20%)	Gastrointestinal	Gastrointestinal		B$(N=5)	4	1	1 (20%)
(33%)	Gastrointestinal	Gastrointestinal		TOTAL$(N=9)	4	3	3 (33%)
(50%)	Gastrointestinal	Gastrointestinal		A$(N=4)	All	2	2 (50%)
(40%)	Gastrointestinal	Gastrointestinal		B$(N=5)	All	2	2 (40%)
(44%)	Gastrointestinal	Gastrointestinal		TOTAL$(N=9)	All	4	4 (44%)
(25%)	Nausea	Gastrointestinal	Nausea	A$(N=4)	3	1	1
(40%)	Nausea	Gastrointestinal	Nausea	B$(N=5)	3	2	2
(33%)	Nausea	Gastrointestinal	Nausea	TOTAL$(N=9)	3	3	3
(25%)	Nausea	Gastrointestinal	Nausea	A$(N=4)	4	1	1
(11%)	Nausea	Gastrointestinal	Nausea	TOTAL$(N=9)	4	1	1
(50%)	Nausea	Gastrointestinal	Nausea	A$(N=4)	All	2	2
(40%)	Nausea	Gastrointestinal	Nausea	B$(N=5)	All	2	2
(44%)	Nausea	Gastrointestinal	Nausea	TOTAL$(N=9)	All	4	4
(20%)	Vomit	Gastrointestinal	Vomit	B$(N=5)	3	1	1
(11%)	Vomit	Gastrointestinal	Vomit	TOTAL$(N=9)	3	1	1
(25%)	Vomit	Gastrointestinal	Vomit	A$(N=4)	4	1	1
(20%)	Vomit	Gastrointestinal	Vomit	B$(N=5)	4	1	1
(22%)	Vomit	Gastrointestinal	Vomit	TOTAL$(N=9)	4	2	2
(25%)	Vomit	Gastrointestinal	Vomit	A$(N=4)	All	1	1
(40%)	Vomit	Gastrointestinal	Vomit	B$(N=5)	All	2	2
(33%)	Vomit	Gastrointestinal	Vomit	TOTAL$(N=9)	All	3	3
(50%)	General Body	General Body		A$(N=4)	4	2	2 (50%)
(40%)	General Body	General Body		B$(N=5)	4	2	2 (40%)
(44%)	General Body	General Body		TOTAL$(N=9)	4	4	4 (44%)
(50%)	General Body	General Body		A$(N=4)	All	2	2 (50%)
(40%)	General Body	General Body		B$(N=5)	All	2	2 (40%)
(44%)	General Body	General Body		TOTAL$(N=9)	All	4	4 (44%)
(20%)	Ache	General Body	Ache	B$(N=5)	4	1	1
(11%)	Ache	General Body	Ache	TOTAL$(N=9)	4	1	1
(20%)	Ache	General Body	Ache	B$(N=5)	All	1	1
(11%)	Ache	General Body	Ache	TOTAL$(N=9)	All	1	1
(25%)	Fever	General Body	Fever	A$(N=4)	4	1	1
(20%)	Fever	General Body	Fever	B$(N=5)	4	1	1

```
(22%)        Fever          General Body       Fever     TOTAL$(N=9)     4          2       2
(25%)        Fever          General Body       Fever     A$(N=4)         All        1       1
(20%)        Fever          General Body       Fever     B$(N=5)         All        1       1
(22%)        Fever          General Body       Fever     TOTAL$(N=9)     All        2       2
(25%)        Lumps          General Body       Lumps     A$(N=4)         4          1       1
(11%)        Lumps          General Body       Lumps     TOTAL$(N=9)     4          1       1
(25%)        Lumps          General Body       Lumps     A$(N=4)         All        1       1
(11%)        Lumps          General Body       Lumps     TOTAL$(N=9)     All        1       1
```

APPENDIX B: THE "PCT" FLAGSUM PARAMETERS

PCTFMT

This parameter contains the name of the format to apply to percent values generated by the macro. The default value is flgsmft4. The 4 at the end of the format name sets the width of the output. Both macros create the format via the following code:

```
proc format
   value flgsmft
   0<-<.5 = "<0.5"
   other  = [4.]
   ;
run;
```

PCTMODE

This is the most difficult parameter to describe. It accepts values 0-10 and requires the Parameter Map to understand the descriptions. This parameter primarily controls where frequencies and/or percentages appear in our final table based on column and row designations. FLAGSUM does this by how it creates the RPTME variable. PCTMODE also controls whether denominators appear in RPTCOL. This variable is the default for FLAGRPT when creating the final table. The descriptions of the various modes refer to "TOTITLE" and "COLTOTITLE". The Parameter Map identifies these columns, which is helpful for you to understand the descriptions in the table below. Please try to experiment with the values to find what you want:

Table 4. PCTMODE Values and Descriptions

PCTMODE VALUE	DESCRIPTION
0	Only frequencies, no percentages
1	Percentages and frequencies for group rows and only frequencies for event rows, See the Parameter Map as an example
2	Only percentages, "N=" in column titles can be controlled by the statistician
3	Percentages and frequencies for all rows and columns
4	Like 1, but only for TOTITLE columns
5	Like 3, but only for TOTITLE columns
6	Like 1, but only for COLTOTITLE columns
7	Like 3, but only for COLTOTITLE columns
8	Uses 2, "N=" in column titles can be controlled by the statistician

9	Uses 0, “N=” in column titles can be controlled by the statistician
10	Percentages and frequencies for COLTOTITLE columns and group rows, only frequencies elsewhere

APPENDIX C: FLAGRPT CONTROLS FONT SIZE AND GRPEV1COL WIDTH

There are some parameters to control the size of the table produced. Sometimes the left column of GRPEV1COL can get wider than you need or want in order to fit a table on one page. If that is the case, you can use GRPEV1COLW to control the width. It accepts a value as a number and a unit, like GRPEV1COLW = 10 em, or it can accept a percentage value if the %nrbquote() function is used, like GPREV1COLW = %nrbquote(30%). Another way to change the output to fit on a page is to use the RTFFNTSZ parameter. It can affect the size of the font in the table. If RTFFNTSZ = 8, then the table will contain 8 pt font throughout. When RTFFNTSZ contains a value, FLAGRPT adds the following code to the PROC REPORT statement to control the font size:

```
style(column header report summary)=[font_size=&RTFFNTSZ pt]
```

Here is a table that utilizes RTFFNTSZ = 5 and GRPEV1COLW = %nrbquote(30%):

FLAGRPT Table Applying RTFFNTSZ and GRPEV1COLW

	A (N=4)			B (N=5)			TOTAL (N=9)		
Adverse Events	3	4	All	3	4	All	3	4	All
Overall	**0 (0%)**	**4 (100%)**	**4 (100%)**	**1 (20%)**	**3 (60%)**	**4 (80%)**	**1 (11%)**	**7 (78%)**	**8 (89%)**
Gastrointestinal	**0 (0%)**	**2 (50%)**	**2 (50%)**	**1 (20%)**	**1 (20%)**	**2 (40%)**	**1 (11%)**	**3 (33%)**	**4 (44%)**
Nausea	1	1	2	2	0	2	3	1	4
Vomit	0	1	1	1	1	2	1	2	3
General Body	**0 (0%)**	**2 (50%)**	**2 (50%)**	**0 (0%)**	**2 (40%)**	**2 (40%)**	**0 (0%)**	**4 (44%)**	**4 (44%)**
Ache	0	0	0	0	1	1	0	1	1
Fever	0	1	1	0	1	1	0	2	2
Lumps	0	1	1	0	0	0	0	1	1

It is obvious that the font got VERY small and the left column takes up 30% of the total table.

APPENDIX D: FLAGRPT SUMMARY AND TITLE PARAMETERS

In some cases, you want to show just the GRPVARs/SOC values in a table. This can be done by setting SUMMARY = Y. The default is N, which shows EVTVARs/PTs and GRPVARs/SOCs.

The final parameters to discuss in FLAGRPT all affect titles within the table:

FLAGRPT Title Parameters and Descriptions

FLAGRPT Title Parameter	Description, Defaults, and Likely Values
EVTTITLE	Title over the left column of our tables. The default is “Adverse Events”.
COLTITILE	Title over COLVAR values, which could be “Treatment Arms”
SUBTITLE	Title over SUBVAR values, which could be “Grades”

The table below utilizes SUMMARY = Y and populates the quoted values from the table above into the title parameters:

FLAGRPT SUMMARY = Y Using Title Parameters

	Treatment Arms								
	A (N=4)			B (N=5)			TOTAL (N=9)		
	Grades			Grades			Grades		
Adverse Events	**3**	**4**	**All**	**3**	**4**	**All**	**3**	**4**	**All**
Overall	**0 (0%)**	**4 (100%)**	**4 (100%)**	**1 (20%)**	**3 (60%)**	**4 (80%)**	**1 (11%)**	**7 (78%)**	**8 (89%)**
Gastrointestinal	0 (0%)	2 (50%)	2 (50%)	1 (20%)	1 (20%)	2 (40%)	1 (11%)	3 (33%)	4 (44%)
General Body	0 (0%)	2 (50%)	2 (50%)	0 (0%)	2 (40%)	2 (40%)	0 (0%)	4 (44%)	4 (44%)

Notice the macro only bolds the “Overall” row. This bolding separates it from other group rows that use another flag variable, GRPFLG. You can see the “Treatment Arms” and “Grades” titles in the table.

APPENDIX E: FLAGSUM CODE

Please excuse the small font, but I believe in code aesthetics and larger font made it difficult to read and understand the code. I encourage you to utilize document magnification when viewing.

```
/*************************************************************/
/* PURPOSE:       Create a dataset with counts intended for an AE (adverse events) or similar type table.
/*                INCLUDED IN FINAL DATASET:
/*                            1 var of percents
/*                            1 var of counts and pcts together
/*                            vars for denoms of each COLVAR value (likely treatment values)
/*
/* INPUT:         2 datasets:
/*                  1. EVTDATA in occurrence data structure AND containing all variables
/*                     listed in any of the parameters, including any
/*                     population flag variables from COLDATA (likely ADSL)
/*                     listed in COLWHERE that are used for the analysis -
/*                     aka counting of flag variables.
/*                  2. COLDATA in an ADSL-like data structure and solely used
/*                     to count denominator values for calculations in the
/*                     macro.  It will NOT be merged onto EVTDAT.
/*                     It must contain all variables listed in COLWHERE.
/*
/* OUTPUT:        LONG (not WIDE) flagsum dataset that can be reported with a PROC REPORT
/*
/* ORGANIZATIONAL MACROS WITH THEIR DESCRIPTION
/* - SUGGEST MAKING SIMILAR ONES IF THEY DO NOT EXIST
/* CODE BELOW WILL NOT RUN WITHOUT THESE MACROS REPLACED
/*
/* MACROS USED:   %dataexst - check dataset exists
/*                %paramreq - check have required parameters
/*                %varexst  - check vars exist in dataset
/*                %nobs     - retrieve number of obs in a dataset
/*                %dupkey   - check for duplicate records in dataset
/*
/*************************************************************/

%macro flagsum(     OUTDATA   =,
                    COLDATA   =,
                    COLWHERE  =,
                    COLVAR    =,
                    EVTDATA   =,
                    GRPVAR    =,
                    EVTVAR    =,
                    SUBVAR    =,
                    IDVAR     = usubjid,
                    ALLFLG    =,
                    GRPFLG    =,
                    EVTFLG    =,
                    PCTMODE   = 1,
                    PCTFMT    =,
                    LSTMORE   = N,
                    DEBUG     = N
               );

/* DECLARE LOCAL MACRO VARS SO NOT TO OVERRIDE USER MACRO VARS */
%local       A B C D E F
             USERMISS NOOBIE ATLEAST1FLG
             _NEWAND CHKFMT
             DSID1 DSID2
             COLVTYPE1 COLVTYPE2 FMTEQ COLFMT1 COLFMT2 COLTOTVAL
             IDEVTYPE IDEVLEN
             IDCVTYPE IDCVLEN
             IDLEN FLGIDLEN RC RC2
             MISSNUM FLGLIST FLGLISTMAX FLGLISTWH
             CVNMTOP GRPEVFTW RPTCOLW
             CLIST CLISTMAX
```

```
            OOPSNUM MANYGRPS
            SETALL SETGRP SETEVT
            DENOMTOTAL DENOMLIST
            RPTMEW
          ;

/* STORE missing SYSTEM OPTION FOR USER */
/*   SET missing = .                    */
/* USED IN PROCs                        */

%let USERMISS = %sysfunc(GETOPTION(missing));
options missing = .;

/* CHECK THAT THERE IS A VALUE IN EVTDATA AND COLDATA */
%if %length(&EVTDATA)= 0 or %length(&COLDATA)= 0 %then %do;
   %if %length(&EVTDATA)= 0 %then %do;
       %put %upcase(error:) EVTDATA requires a value and currently there is none.;
       %put %upcase(error:) Please enter a value into EVTDATA.;
   %end;
   %if %length(&COLDATA)= 0 %then %do;
       %put %upcase(error:) COLDATA requires a value and currently there is none.;
       %put %upcase(error:) Please enter a value into COLDATA.;
   %end;
       %put %upcase(error:) Once corrections are made, please try running FLAGSUM again.;
       %put %upcase(error:) FLAGSUM will now terminate.;
       %abort abend 1;
%end;

/* CHECK THAT BOTH DATASETS EXIST */
%dataexst(&EVTDATA &COLDATA);

/* CHECK THAT THE DATASETS HAVE OBSERVATIONS */
%if %nobs(&EVTDATA)= 0 or %nobs(&COLDATA)= 0 %then %do;
   %if %nobs(&EVTDATA)= 0 %then %do;
         %put %upcase(error:) &EVTDATA does not have any observations.;
   %end;
   %if %nobs(&COLDATA)= 0 %then %do;
         %put %upcase(error:) &COLDATA does not have any observations.;
   %end;
         %put %upcase(error:) Please check datasets and submit them with observations.;
         %put %upcase(error:) FLAGSUM will now terminate.;
         %abort abend 2;
%end;

/* CHECK OF OBSERVATIONS WITH COLWHERE VALUE */
%if %length(&COLWHERE) > 0 %then %do;

/* When NOOBIE = 1 THEN IT MEANS THERE WAS AN ERROR */
   %let NOOBIE = ;

   %if %nobs(&EVTDATA,&COLWHERE) = 0 %then %do;
         %put %upcase(error:) ----EVTDATA----------------------------------------------------------------------------------------;
         %put %upcase(error:) The &EVTDATA data with the COLWHERE parameter does not have any observations.                    ;
         %put %upcase(error:) Here is the COLWHERE value for &EVTDATA:                                                         ;
         %put %upcase(error:)      &COLWHERE                                                                                   ;
         %put %upcase(error:) Please check there are still observations in &EVTDATA when the COLWHERE parameter is used.      ;
         %let NOOBIE = 1;
   %end;

/* THE LINE WITH JUST &COLWHERE IS INTENTIONALLY OFF BY 1 FROM ABOVE SO PRINTED OUT BY chk PROGRAM  */
/*       chk is a UNIX script that checks logs for NOTEs, WARNINGs, and ERRORs in a log.  It runs right after sas finishes. */
/* OTHERWISE chk DELETES ALL DUPLICATE LINES AND THIS WOULD BE DELETED IF BOTH DATASETS HAVE THE PROBLEM */
   %if %nobs(&COLDATA,&COLWHERE) = 0 %then %do;
         %put %upcase(error:) ----COLDATA----------------------------------------------------------------------------------------;
         %put %upcase(error:) The &COLDATA data with the COLWHERE parameter does not have any observations.                    ;
         %put %upcase(error:) Here is the COLWHERE value for &COLDATA:                                                         ;
         %put %upcase(error:)     &COLWHERE                                                                                    ;
         %put %upcase(error:) Please check there are still observations in &COLDATA when the COLWHERE parameter is used.      ;
         %let NOOBIE = 1;
   %end;

/* ABORT IF ONE OF THE ERRORS ABOVE OCCURRED */
   %if &NOOBIE = 1 %then %do;
         %put %upcase(error:) ----------------------------------------------------------------------------------------------;
         %put %upcase(error:) FLAGSUM will now terminate.                                                                    ;
         %abort abend 3;
   %end;

         %put %upcase(note: (cbar)) It looks like we are using a COLWHERE value.                                  ;
         %put %upcase(note: (cbar)) COLWHERE is only intended to be used with population flag variables.          ;
         %put %upcase(note: (cbar)) Problems in the COLWHERE value can cause FLAGSUM to fail painfully.           ;
         %put %upcase(note: (cbar)) BE CAREFUL.....                                                               ;
%end;

/*************************************************/
/* CHECK THAT SOME VALUE IS IN REQUIRED PARAMETERS */
/*************************************************/
%if %length(&GRPVAR)  = 0 or  %length(&EVTVAR) = 0 or
    %length(&COLVAR)  = 0 %then %do;

   %paramreq(GRPVAR EVTVAR COLVAR);

   %if %length(&SUBVAR) > 0 and %length(&COLVAR) = 0 %then %do;
      %put %upcase(error:) %str( );
      %put %upcase(error:) COLVAR is not defined and SUBVAR is.;
```

```
      %put %upcase(error:) If you want to use SUBVAR, a definition for COLVAR is required.;
      %put %upcase(error:) Perhaps consider creating a dummy variable to enter into COLVAR.;
   %end;

   %put %upcase(error:) Please define the above parameters and try again.;
   %abort abend 4;

%end;

/*************************************************/
/* CHECK THAT AT LEAST 1 FLAG PARAM HAS A VALUE */
/*************************************************/
%let ATLEAST1FLG = &ALLFLG &GRPFLG &EVTFLG;
%if %length(&ATLEAST1FLG) = 0 %then %do;
      %put %upcase(error:) At least 1 FLG parameter must have a value.;
      %put %upcase(error:) Here are the FLG parameters:  ;
      %put %upcase(error:)    ALLFLG GRPFLG EVTFLG          ;
      %put %upcase(error:) Please enter a value for at least one FLG parameter ;
      %put %upcase(error:) and rerun the program.;
      %abort abend 5;
%end;

/*******************************************************/
/* CHECK IF IDVAR ENTERED AND SET DEFAULT VALUE IF NOT  */
/* IDVAR NEEDS A VALUE BEFORE %varexst CALL             */
/*******************************************************/
%if %length(&IDVAR)= 0 %then %do;
       %let IDVAR  = usubjid;
%end;

/*************************************/
/* CHECK THAT VARIABLES EXIST           */
/*************************************/

%varexst(DATA       = &EVTDATA.,
           VARS     = &IDVAR &COLVAR &GRPVAR &EVTVAR,
           ADDMSG   = %str(for FLAGSUM.  Please include this variable in &EVTDATA and rerun the program.),
           IFERR    = endsas);

%varexst(DATA       = &COLDATA.,
           VARS     = &IDVAR &COLVAR,
           ADDMSG   = %str(when both are specified.  Please include this variable in &COLDATA and rerun the program.),
           IFERR    = endsas);

/*******************************************/
/* CHECK IF VARS EXIST IF LISTED IN PARAMETERS */
/*******************************************/
%if %length(&SUBVAR) > 0 or
    %length(&ALLFLG) > 0 or
    %length(&GRPFLG) > 0 or
    %length(&EVTFLG) > 0
%then %do;
      %local VAREXSTLIST;
      %let   VAREXSTLIST=;

      %if %length(&SUBVAR) > 0 %then %do;
           %let VAREXSTLIST= &SUBVAR;
      %end;
      %if %length(&ALLFLG) > 0 %then %do;
         %let VAREXSTLIST= &VAREXSTLIST &ALLFLG;
      %end;
      %if %length(&GRPFLG) > 0 %then %do;
         %let VAREXSTLIST= &VAREXSTLIST &GRPFLG;
      %end;
      %if %length(&EVTFLG) > 0 %then %do;
         %let VAREXSTLIST= &VAREXSTLIST &EVTFLG;
      %end;

      %varexst(DATA   = &EVTDATA.,
               VARS   = &VAREXSTLIST.,
               ADDMSG = %str(when specified.  Please include &VAREXSTLIST in &EVTDATA and rerun the program.),
               IFERR  = endsas);
%end;

/*********************************************************/
/* ADD COLVAR CHECK TO SEE IF IT MIGHT BE A GRADE VARIABLE  */
/* IF SO, REMIND USER IT SHOULD BE LISTED IN SUBVAR INSTEAD */
/*********************************************************/
%if %index(%lowcase(&COLVAR), grad) or
    %index(%lowcase(&COLVAR), grd)  %then %do;
          %put %upcase(warning:) The given COLVAR seems to represent the grade of your EVTVAR.;
          %put %upcase(warning:) This may be a mistake.  Typically, COLVAR lists a treatment or arm type variable.;
          %put %upcase(warning:) Please use the following UNIX command to see examples:  read2me -w eventsum ;
          %put %upcase(warning:) They will be at the end of the WORD document and show tables expected from flagrpt.;
%end;

/*****************************************************/
/* ADD DEBUG OPTION TO PARAMETERS                    */
/*****************************************************/
%if %length(&DEBUG)= 0 %then %let DEBUG = Y;
%else                        %let DEBUG = %substr(%upcase(&DEBUG),1,1);

%if &DEBUG = Y %then %do;
   options mprint mlogic mlogicnest source2 symbolgen;
%end;
```

```
/*******************************************************/
/* SET DEFAULT FOR OUTDATA IF BLANK                    */
/* SET DEFAULT FOR LSTMORE IF BLANK                    */
/* SET DEFAULT FOR PCTFMT  IF BLANK                    */
/*             CHECK PCTFMT  EXISTS                    */
/* SET DEFAULT FOR PCTMODE IF BLANK                    */
/*            ADJUST PCTMODE AS NEEDED                 */
/*******************************************************/
%if %length(&OUTDATA) = 0 %then %let OUTDATA = flagend;

%if %length(&LSTMORE)= 0 %then  %let LSTMORE = Y;
%else                           %let LSTMORE = %substr(%upcase(&LSTMORE),1,1);

/* FYI - CANNOT CHECK EXISTENCE OF FORMATS LIKE 4. OR 7.2 */
/*         AND THESE ARE EXPECTED TO BE USED IN PCTFMT    */
/*         SO THERE IS NO CHECK FOR EXISTENCE OF FMT      */
/*******************************************************/
%if %length(&PCTFMT) = 0 %then %do;
   proc format;
        value flgsmft
             0<-<.5 = "<0.5"
             other  = [4.]
             ;
   run;
   %let PCTFMT = flgsmft4.;
%end;
%else %do;
   %if %sysfunc(find(&PCTFMT,%str(.))) = 0 %then %do;
      %let PCTFMT = %sysfunc(strip(&PCTFMT%str(.)));
   %end;
   %put &PCTFMT;
%end;

/* DELETE PUNCTUATION AND DIGITS FROM PCTFMT VALUE  */
/* THIS GIVES NAME OF FORMAT TO SEARCH IF IT EXISTS */
%let CHKFMT = %sysfunc(compress(&PCTFMT,,dp));

/* CHECK FORMAT IF NOT SOLELY A NUMERIC FORMAT, LIKE 7. or 7.2 */
%if %length(&CHKFMT) ^= 0 %then %do;
   proc sql noprint;
      create table chkflagfmt
      as select libname, fmtname, source, fmttype
      from dictionary.formats
      where fmtname="%upcase(&CHKFMT)"
      ;
  quit;

  %if %nobs(chkflagfmt) = 0 %then %do;
     %put %upcase(error:) The following format does not exist:  &PCTFMT.;
     %put %upcase(error:) Please enter a format that exists in  PCTFMT.;
     %put %upcase(error:) Once corrections are made, please try running FLAGSUM again.;
     %put %upcase(error:) FLAGSUM will now terminate.;
     %abort abend 6;
  %end;
%end;

%if %length(&PCTMODE) = 0 %then %let PCTMODE = 1;
%if &PCTMODE^=0 and &PCTMODE^=1 and
    &PCTMODE^=2 and &PCTMODE^=3 and
    &PCTMODE^=4 and &PCTMODE^=5 and
    &PCTMODE^=6 and &PCTMODE^=7 and
    &PCTMODE^=8 and &PCTMODE^=9 and
    &PCTMODE^=10
                          %then %let PCTMODE = 1;

/*******************************************************/
/* CHANGE PCTMODE CODES ACCORDINGLY                    */
/* VALUES OF 6, 7, and 10 CHANGE IF THERE IS NO SUBVAR  */
/*******************************************************/
%if %length(&SUBVAR) = 0 %then %do;
   %if &PCTMODE = 6  %then %let PCTMODE = 1;;
   %if &PCTMODE = 7 or
       &PCTMODE = 10 %then %let PCTMODE = 3;;
%end;

/*************************************************/
/* GET CHARACTERISTICS OF VARIABLES IN DATASETS   */
/*************************************************/
/* OPEN DATASETS */
/*****************/
%let DSID1    = %sysfunc(open(&EVTDATA));
%let DSID2    = %sysfunc(open(&COLDATA));

/*************************************************************************************/
/* RETRIEVE CHARACTERISTICS OF VARIABLES NEEDED FOR CODING BELOW                     */
/* IDVAR NEEDS THIS TO AVOID "Multiple length" WARNING THAT FAIL VALIDLOG            */
/*************************************************************************************/

/*********************************************************/
/* SET COLVAR RELATED ITEMS - VARTYPE AND VARFMT         */
/* CHECK THAT EVTDATA COLVAR INFO MATCHES COLDATA COLVAR INFO */
/*********************************************************/
%let COLVTYPE1 = %sysfunc(vartype(&DSID1,%sysfunc(varnum(&DSID1,&COLVAR))));
%let COLFMT1   = %sysfunc(varfmt( &DSID1,%sysfunc(varnum(&DSID1,&COLVAR))));
```

```
%let COLVTYPE2 = %sysfunc(vartype(&DSID2,%sysfunc(varnum(&DSID2,&COLVAR))));
%let COLFMT2   = %sysfunc(varfmt( &DSID2,%sysfunc(varnum(&DSID2,&COLVAR))));

/* CHECK COLVAR VALUES IN EVTDATA AND COLDATA MATCH */
%if %sysfunc(strip(&COLVTYPE1)) ^= %sysfunc(strip(&COLVTYPE2)) %then %do;
   %put %upcase(error:) The variable type for &COLVAR in &EVTDATA and &COLDATA do not match.;
   %put %upcase(error:) EVTDATA = &COLVTYPE1 and COLDATA = &COLVTYPE2;
   %put %upcase(error:) ;
   %put %upcase(error:) Please make sure &COLVAR in &COLDATA and &EVTDATA are the same variable.;
   %put %upcase(error:) FLAGSUM will now terminate.;
   %abort abend 7;
%end;

/* CHECK FORMATS FOR COLVAR ARE THE SAME IN EVTDATA AND COLDATA           */
/*   FMTEQ = 1 INDICATES THE MACRO NEEDS TO END BECAUSE COLFMTS NOT EQUAL */
%let FMTEQ = 0;

%if %length(&COLFMT1) > 0 and %length(&COLFMT2) > 0 %then %do;
   %if %sysfunc(strip(&COLFMT1)) ^= %sysfunc(strip(&COLFMT2)) %then %do;
       %let FMTEQ = 1;
   %end;
%end;
%else %if %length(&COLFMT1) > 0 or  %length(&COLFMT2) > 0 %then %do;

/* AT THIS POINT, IF ONE HAS A VALUE THEN THE OTHER DOES NOT */
/* SO THEY ARE NOT EQUAL */
       %let FMTEQ = 1;
%end;

/* PRINT MESSAGE AND TERMINATE IF THERE IS A PROBLEM */
%if &FMTEQ = 1 %then %do;
       %put %upcase(error:) The formats for &COLVAR in &EVTDATA and &COLDATA do not match.;
       %put %upcase(error:) EVTDATA = &COLFMT1;;
       %put %upcase(error:) COLDATA = &COLFMT2;;
       %put %upcase(error:) ;
       %put %upcase(error:) Please make sure &COLVAR in &COLDATA and &EVTDATA are the same variable.;
       %put %upcase(error:) FLAGSUM will now terminate.;
       %abort abend 8;
%end;

/*************************************************************************/
/* "TOTAL" BECOMES THE VALUE OF THE TOTAL COLUMNS IN FREQ DATASETS       */
/*************************************************************************/
%let COLTOTVAL = TOTAL;

/***************************/
/* SET SUBVAR RELATED ITEMS */
/***************************/
%if %length(&SUBVAR) > 0 %then %do;

/*************************************************************************/
/* "All" BECOMES THE VALUE OF THE TOTAL COLUMNS IN FREQ DATASETS         */
/*************************************************************************/
   %let SUBTOTVAL = All;
%end;

/*********************************************/
/* SET IDVAR RELATED ITEMS - VARTYPE, VARLEN  */
/*********************************************/
/*****************************************************************/
/* SET LENGTH TO LONGEST VALUE TO AVOID "Multiple length" WARNING */
/* CHECK THAT IDVAR IS THE SAME TYPE IN EVTDATA AND COLDATA       */
/*****************************************************************/

%let IDEVTYPE  = %sysfunc(vartype(&DSID1, %sysfunc(varnum(&DSID1,&IDVAR))));
%let IDEVLEN   = %sysfunc(varlen( &DSID1, %sysfunc(varnum(&DSID1,&IDVAR))));

%let IDCVTYPE  = %sysfunc(vartype(&DSID2,%sysfunc(varnum(&DSID2,&IDVAR))));
%let IDCVLEN   = %sysfunc(varlen( &DSID2,%sysfunc(varnum(&DSID2,&IDVAR))));

%if %sysfunc(strip(&IDEVTYPE)) ^= %sysfunc(strip(&IDCVTYPE)) %then %do;
   %put %upcase(error:) The &IDVAR variable type in EVTDATA does not match that in COLDATA.;
   %put %upcase(error:) The &IDVAR variable type in EVTDATA is &IDEVTYPE..;
   %put %upcase(error:) The &IDVAR variable type in COLDATA is &IDCVTYPE..;
   %put %upcase(error:) C is character.  N is numeric.;
   %put %upcase(error:) Please correct the variable types and try again.;
   %abort abend 9;
%end;

/* GET MAX LENGTH OF IDVAR REGARDLESS IF VALUES DIFFERENT */
%let IDLEN = %sysfunc(max(&IDEVLEN,&IDCVLEN));

%if &IDEVTYPE = C %then %do;
    %let FLGIDLEN = length &IDVAR $&IDLEN..;
%end;
%else %do;
    %let FLGIDLEN = length &IDVAR &IDLEN..;
%end;

/*************************************************************************/
/* CLOSE DATASETS - END GETTING CHARACTERISTICS OF DATA AND PERFORMING CHECKS */
/*************************************************************************/
%let RC       = %sysfunc(close(&DSID1));
%let RC2      = %sysfunc(close(&DSID2));
```

```
/*************************************************************************************************
   COLDATA - NEEDS RECS TO CALCULATE N= TOTALS AKA THE DENOMINATOR FOR PCTS
   EVTDATA - ONLY AE/OCCURRENCE DATA AND SUBVAR (LIKELY GRADE) VALUES

   IF COLVAR OR SUBVAR ARE NUMERIC, ANOTHER VARIABLE NEEDS TO BE MADE THAT IS THE
         ACTUAL VALUE WITHOUT THE FORMAT TO USE THE NOBS MACRO CORRECTLY.
*************************************************************************************************/

/*****************************************/
/* CHECK FOR DUPLICATE RECORDS IN &COLDATA */
/*****************************************/
%dupkey( DATA   = &COLDATA
        ,BYVARS = &IDVAR &COLVAR
        ,IFDUP  = ENDSAS
        );

/*****************************************/
/* REMOVE RECS WITH MISSING VALUES IN &COLVAR */
/*****************************************/
data misscol;
   set &COLDATA
       %if  %length(&COLWHERE) > 0 %then %do;
            (where=(&COLWHERE))
       %end;
       ;
   if   missing(&COLVAR) or
        missing(&IDVAR)  then output;
run;

/*************************************************/
/* IF RECS IN misscol THEN PROGRAM STILL RUNS TO END */
/*************************************************/
%let MISSNUM = %nobs(misscol);
%if &MISSNUM > 0 %then %do;
   %put %upcase(note: (cbar)) -- There were records with missing &COLVAR or &IDVAR values  ;
%if %length(&COLWHERE) > 0 %then %do;
   %put %upcase(note: (cbar)) -- when COLWHERE is:  &COLWHERE..                                                    ;
%end;
   %put %upcase(note: (cbar)) -- These were omitted from the counts and denominator calculations.;
   %put %upcase(note: (cbar)) -- Please check the .lst for the omitted values.                    ;
   %put %upcase(note: (cbar)) -- FLAGSUM will continue....                                        ;
   %put %upcase(note: (cbar)) -- ++++ &COLDATA HAD RECORDS WITH MISSING VALUES ++++++++++++++++++;

   title "RECS WITH MISSING &COLVAR OR &IDVAR VALUES FROM &COLDATA";
   proc print data= misscol;
   run;
%end;

/* CHECK IDVAR/COLVAR COMBINATIONS IN EVTDATA AND COLDATA MATCH */
proc sort data = &COLDATA
          %if   %length(&COLWHERE) > 0 %then %do;
              (where = (&COLWHERE))
          %end;
            out  = colcolv (keep = &IDVAR &COLVAR)
            nodupkey;
     by &IDVAR &COLVAR;
run;

proc sort data = &EVTDATA
          %if   %length(&COLWHERE) > 0 %then %do;
              (where = (&COLWHERE))
          %end;
            out  = evtcolv ( keep = &IDVAR &COLVAR)
            nodupkey;
     by &IDVAR &COLVAR;
run;

/* WILL NEED A SPECIFIC EXAMPLE TO TEST THIS! */
data colprobs;
   &FLGIDLEN;
   merge colcolv (in = incolv)
         evtcolv (in = inevtv);
   by &IDVAR &COLVAR;
   length in_col in_evt 3.;
   if incolv and inevtv then delete;
   if incolv then in_col = 1;
   if inevtv then in_evt = 1;
run;

/* ONLY PRINT OUT IF THERE IS A RECORD THAT IS ONLY IN EVTDATA, aka in_evt = 1 */
%let BADCOLN = %nobs(colprobs,%str(in_evt = 1));
%if &BADCOLN > 0 %then %do;

   %put %upcase(error:) Please look at the end of the .lst for those records              ;
   %put %upcase(error:) in &EVTDATA that did not match one in &COLDATA.                   ;
   %put %upcase(error:) ============================================================     ;

%if %length(&COLWHERE) > 0 %then %do;
   %put %upcase(error:)                                                                   ;
   %put %upcase(error:) If known problem records are not seen, it may be because COLWHERE  ;
   %put %upcase(error:) was set to:  &COLWHERE..                                          ;
%end;

   %put %upcase(error:) -- Please check the datasets and correct the unmatching values before running again.           ;
```

```
   %put %upcase(error:) -- FLAGSUM will now terminate.                                                                    ;

   title  "WE HAVE ONE OF THREE PROBLEMS:"                                                                                ;
   title2 "1. There are recs in &EVTDATA with a value for &IDVAR that does match a record in &COLDATA."                  ;
   title3 "2. There are recs in &EVTDATA with a value for &COLVAR that does match a record in &COLDATA."                 ;
   title4 "3. Both of the 2 items above contained in one record of &EVTDATA."                                             ;
   title5 "-------------------------------------------------------------------------------------------"                  ;
   title6 "IF in_col= 1 THEN THE RECORD COMES FROM &COLDATA"                                                              ;
   title7 "IF in_evt= 1 THEN THE RECORD COMES FROM &EVTDATA"                                                              ;
   title8 "THIS INFORMATION SHOULD HELP DETERMINE THE ISSUE."                                                             ;

   proc print data= colprobs;
   run;
   %abort abend 10;

%end;

/* PRINT OUT MISSING IDVAR VALUES */
/* EVTDATA MISSING RECS FOR:  IDVAR, COLVAR, GRPVAR, EVTVAR, SUBVAR */
data evtidmiss;
   set &EVTDATA
       %if %length(&COLWHERE) > 0 %then %do;
             (where=(&COLWHERE))
       %end;
       ;
   if missing(&IDVAR)  or missing(&COLVAR) or
      missing(&GRPVAR) or missing(&EVTVAR)
      %if %length(&SUBVAR) > 0 %then %do;
           or missing(&SUBVAR)
      %end;
      then output;
run;

%let EVTIDMISSN = %nobs(evtidmiss);
%if &EVTIDMISSN > 0 %then %do;
   %put %upcase(note: (cbar)) -- In &EVTDATA, there are missing values for the following variables:           ;
   %put %upcase(note: (cbar)) -- &IDVAR. &COLVAR. &GRPVAR. &EVTVAR. &SUBVAR.                                  ;

%if %length(&COLWHERE) > 0 %then %do;
   %put %upcase(note: (cbar)) -- When COLWHERE is:  &COLWHERE..                                               ;
%end;

   %put %upcase(note: (cbar)) -- These records will not be counted at all.                                    ;
   %put %upcase(note: (cbar)) -- Please check the data and ensure these are fixed in the future.              ;
   %put %upcase(note: (cbar)) -- FLAGSUM will continue.....                                                   ;
   %put %upcase(note: (cbar)) -- ====== &EVTDATA HAD RECORDS WITH MISSING VALUES ========================      ;

   title  "RECS WITH MISSING VALUES IN &EVTDATA FOR THE FOLLOWING VARIABLES:  &IDVAR &COLVAR &GRPVAR &EVTVAR &SUBVAR";
%if %length(&COLWHERE) > 0 %then %do;
   title2 "WHEN COLWHERE IS:  &COLWHERE";
%end;
   proc print data= evtidmiss;
      var &IDVAR. &COLVAR. &GRPVAR. &EVTVAR. &SUBVAR. ;
   run;

%end;

/*****************************************/
/* GET LIST OF FORMATTED VALUES OF COLVAR    */
/* CALCULATE DENOM FOR TOTAL COLVAR COLUMN   */
/* ONLY WHERE COLVAR IS NON-MISSING          */
/*****************************************/
/* THERE CANNOT BE DUPLICATES IN COLDATA AT  */
/* THIS POINT BECAUSE %dupkey WILL endsas    */
/* IF ANY FOUND                              */
/* ******************************************/
/* AT THIS POINT, WE KNOW                    */
/* COLVAR VALUES IN COLDATA MATCH THOSE IN   */
/* IN EVTDATA, SO COLDATA COUNTS ARE         */
/* ACCURATE TO COLVAR VALUES IN EVTDATA      */
/*****************************************/
/* PROC SQL HAS feedback OPTION THAT SHOWS MORE ABOUT WHAT IT IS DOING */
/***********************************************************************/
proc sql noprint; * feedback;
   select distinct &COLVAR into : CLIST separated by '@'
   from &COLDATA
   where ^(missing(&COLVAR)) and ^(missing(&IDVAR))
      %if %length(&COLWHERE) > 0 %then %do;
            and &COLWHERE
      %end;
   ;
quit;

/* GET NUMBER OF ITEMS IN CLIST */
   %let CLISTMAX = %sysfunc(countw(%nrbquote(&CLIST),@));

/* CREATE VALID VARNAME LIST IN A MACRO VAR FOR COLVAR VALUES           */
/* ALONG WITH OTHER MACRO VAR FOR MACRO VARS WITH DENOM VALUES          */
/* NEED TOP LENGTHS FOR VVALUE AND VARNAME VALUES FOR length STATEMENT  */
/* IN A DATA STEP                                                       */
/************************************************************************/
   %let   DENOMLIST=;

/*************************************************************/
/* DEFAULT VALUE PICKED BY CONSIDERING A NUMBER WITH 3 PLACES      */
```

```
/* ADDING AN _ TO THE BEGINNING OF THE VAR NAME AND _10 TO THE END */
/*****************************************************************/
   %let   CVNMTOP  = 7;

/* PROCESS COLVAR VALUES */
%do A=1 %to &CLISTMAX;
        %local  THISCVAL&A ;
        %let    THISCVAL&A = %qscan(%nrbquote(&CLIST), &A, @);

/*************************************************************************/
/* REMOVE SPACES AND PUNCTUATION FROM VALUE SO IT CAN BE A VALID VARIABLE NAME */
/*************************************************************************/
         %local  VARCNAME&A ;
         %let    VARCNAME&A = %sysfunc(compress("&&THISCVAL&A", ,sp))_&A;

/*************************************************************************/
/* CHECK THAT VARCNAME VALUE IS NOT TOO LARGE: OVER 27 CHARS IN LENGTH          */
/* SO THAT IT CAN ACCOMMODATE ADDING "DENOM" (5 CHARS LONG) TO THE NAME         */
/* SAS MAX VARNAME LENGTH = 32                                                  */
/* IF CLISTMAX >= 10 (ASSUME IT WILL NOT BE 100 OR MORE) THEN                   */
/* LENGTH OF COLVAR VALUE MUST BE <= 24 BECAUSE ADDING "_10" ON END OF NAME     */
/* ELSE IT IS <= 25 (ADDING "_9" ON END OF NAME)                                */
/*************************************************************************/
         %if %length(&&VARCNAME&A) > 27 %then %do;
            %if &CLISTMAX >= 10 %then %do;
                  %put %upcase(error:) The following &COLVAR value is over 24 characters long.             ;
                  %put %upcase(error:)   %sysfunc(compress("&&THISCVAL&A", ,sp))                           ;
                  %put %upcase(error:)   ------------------------                                          ;
                  %put %upcase(error:) -- LINE ABOVE IS 24 CHARS FOR COMPARISON                            ;
                  %put %upcase(error:) FLAGSUM only caught the first instance in &COLVAR.                  ;
                  %put %upcase(error:) If there are others, please change them before re-running FLAGSUM.  ;
            %end;
            %else %do;
               %put %upcase(error:) The following &COLVAR value is over 25 characters long.                ;
               %put %upcase(error:)   %sysfunc(compress("&&THISCVAL&A", ,sp))                            ;
               %put %upcase(error:)   -------------------------                                          ;
               %put %upcase(error:) -- LINE ABOVE IS 25 CHARS FOR COMPARISON                             ;
               %put %upcase(error:) FLAGSUM only caught the first instance in &COLVAR.                   ;
               %put %upcase(error:) If there are others, please change them before re-running FLAGSUM.   ;
            %end;
            %abort abend 11;;
         %end;

/* IF FIRST CHARACTER IS A NUMBER ADD AN UNDERSCORE AS FIRST CHAR */
         %if %sysfunc(anydigit(%sysfunc(substr(&&VARCNAME&A,1,1)))) > 0 %then %do;
            %let VARCNAME&A = _&&VARCNAME&A;
         %end;

/* GET COUNT  OF RECS WITH COLVAR VALUE */
/* GET LENGTH OF LONGEST   COLVAR VALUE */

   data _null_;
       set &COLDATA (where=( ^(missing(&IDVAR))
            %if %length(&COLWHERE) > 0 %then %do;
                    and &COLWHERE
            %end;
                    ))
                    end = last;

       retain nobs_&A 0;

       if strip(vvalue(&COLVAR)) = "%nrbquote(&&THISCVAL&A)" then
             nobs_&A+1;
       if last then do;
             call symput("DENOM&&VARCNAME&A",nobs_&A);
       end;
   run;

/* MAKE MACRO VARS TO WRITE CODE FOR flagdenom DATASET */
   %local FLAGDENOM&A LBLDENOM&A;

   %let   FLAGDENOM&A = %str(denom&&VARCNAME&A = &&&&DENOM&&VARCNAME&A);
   %let   LBLDENOM&A  = %str(denom&&VARCNAME&A = "DENOM FOR:  &&THISCVAL&A");

   %let   DENOMLIST   = &DENOMLIST denom&&VARCNAME&A;

/* NEED A SIMILAR COMMA DELIMITED LIST FOR CALCULATING LENGTH OF SUM OF ALL VALUES */
/* VALUE WILL BE USED IN A sum() FUNCTION IN MACRO CODE */
   %if &A = 1 %then %do;
        %local MACCOMLIST;
        %let   MACCOMLIST =  &&&&DENOM&&VARCNAME&A;
   %end;
   %else %do;
        %let   MACCOMLIST = &MACCOMLIST,&&&&DENOM&&VARCNAME&A;
   %end;

   %put &MACCOMLIST;

   %if %length(&&VARCNAME&A) > &CVNMTOP %then %do;
      %let CVNMTOP    = %eval(%length(&&VARCNAME&A)+4);
   %end;

/* CHECK IT OUT */
   %put CLIST:       &CLISTMAX %superq(CLIST) &&VARCNAME&A ;
   %put FLAGDENOM&A: &&FLAGDENOM&A;
```

```
   %put LBLDENOM&A:  &&LBLDENOM&A;
%end;

/* GET TOTAL FOR denomTOTAL IN FINAL DATASET AND LENGTH OF THIS TOTAL */
%local MACCOMTOT TOTLEN;
%let MACCOMTOT = %sysfunc(sum(&MACCOMLIST));
%let TOTLEN    = %length(&MACCOMTOT);

%put MACCOMTOT: &MACCOMTOT;
%put TOTLEN:    &TOTLEN;


/* RETRIEVE FORMAT WIDTHS FOR VARIABLES IN &EVTDATA */
   data _null_;
       set &EVTDATA (
                      where=( ^(missing(&COLVAR)) and ^(missing(&IDVAR))
                      %if %length(&COLWHERE) > 0 %then %do;
                              and &COLWHERE
                      %end;
                                )
                     );

/* GET FORMAT LENGTHS TO CREATE REPORTING (rpt: AND grpev1col) VARIABLES */
       call symput("GRPVARFTW",strip(vformatw(&GRPVAR)));
       call symput("EVTVARFTW",strip(vformatw(&EVTVAR)));
       call symput("COLVARFTW",strip(vformatw(&COLVAR)));
%if %length(&SUBVAR) > 0 %then %do;
       call symput("SUBVARFTW",strip(vformatw(&SUBVAR)));
%end;
   run;

/* CREATE LENGTH OF grpev1col VARIABLE HERE, NAMED GRPEVFTW */
%let GRPEVFTW = %eval(%sysfunc(max(&GRPVARFTW,&EVTVARFTW))+27);

/*****************************************************************/
/* CHECK TO DETERMINE IF ANY EVENT HAS MULTIPLE GROUPS          */
/*****************************************************************/
/* SORT BY EVENT FIRST SO IF IT IS IN MULTIPLE GROUPS   */
/* THE COUNT WILL BE > 1 BELOW                          */
/********************************************************/
proc sort data = &EVTDATA (keep= &EVTVAR &GRPVAR )
            out  = manygrps
            nodupkey;
    by &EVTVAR &GRPVAR;
run;

%if &DEBUG = Y %then %do;
   title "ANY EVENTS WITH MULTIPLE GROUP VALUES";
   proc print data= manygrps;
   run;
%end;

data manycount (where=( count > 1));
    set manygrps;
    by &EVTVAR &GRPVAR;
    if first.&GRPVAR;
    if first.&EVTVAR then
         count= 1;
    else count+1;
run;

%let MANYGRPS = %nobs(manycount);

%if &MANYGRPS > 0 %then %do;

   %put %upcase(note: (cbar)) -- The &EVTVAR variable has multiple possible &GRPVAR values.        ;
   %put %upcase(note: (cbar)) -- Please check the .lst file to see which &EVTVAR values            ;
   %put %upcase(note: (cbar)) -- have multiple &GRPVAR values and correct any unexpected results.  ;
   %put %upcase(note: (cbar)) =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-                  ;
   %put %upcase(note: (cbar)) -- FLAGSUM will continue...                                          ;
   %put %upcase(note: (cbar)) >>>> &EVTVAR HAS AT LEAST ONE VALUE WITH MULTIPLE &GRPVAR VALUES <<<<;

   data manyprint;
      merge manycount (in= incount)
              manygrps  (in= ingrps)
      ;
      by &EVTVAR;
      if incount and ingrps;
   run;

   title "EVENTS WITH MORE THAN ONE GROUP";
   proc print data= manyprint;
   run;

%end;

/************************/
/* GET EVENT TOTALS     */
/************************/
/**************************************************************************************************/
/*  SIMPLE PROC FREQ TO GET ALL OF THE TOTALS THAT ARE NEEDED ;                                  */
/*  LATER THESE WILL BE MERGED TOGETHER AND THE PERCENTAGES OBTAINED BY A SIMPLE                 */
/*       QUOTIENT WITH NUMBERS OF RANDOMIZED SUBJECTS ;                                          */
/*  NOTE: NEED TO GET SOC AND OVERALL TOTALS SEPARATELY AS THE HIGHEST GRADE                     */
/*        BY PT/EVENT FOR AN INDIVIDUAL MAY NOT BE THE HIGHEST GRADE PT/EVENT BY SOC ;           */
```

```
/*************************************************************************************************/
/* FYI - IF THERE ARE NUMBERS COMMENTED OUT, USUALLY THESE REFER TO TOTALS vs SPECIFIC VALUES   */
/* MAY NEED THIS INFO LATER...BELOW IS WHAT I BELIEVE I UNDERSTAND:                            */
/* 1111 AND 111 ARE "EVENT" FREQS,                                                             */
/* 1110 AND 110 ARE "TOTAL" FREQS - ANY NUMBER ENDING IN 0 IS A TOTAL                          */
/*************************************************************************************************/
   ods listing exclude all;

/* SET MACRO VARIABLE VALUES IN CASE THEY ARE NOT CREATED LATER */
%let SETALL =;
%let SETGRP =;
%let SETEVT =;

/* GET COUNTS WHERE IDVAR, COLVAR, GRPVAR, EVTVAR, and SUBVAR ARE NOT MISSING */
/* TOTALS FOR THE OVERALL ROW/ALL */
/* ALLFLG_LOOK */
%if %length(&ALLFLG) > 0 %then %do;
   ods output CrossTabFreqs = ALL_N (where= (frequency > 0));
   proc freq data= &EVTDATA         (where= (&ALLFLG   = "Y"
                                             and ^(missing(&IDVAR  ))
                                             and ^(missing(&COLVAR ))
                                             and ^(missing(&GRPVAR ))
                                             and ^(missing(&EVTVAR ))
                                        %if %length(&SUBVAR) > 0 %then %do;
                                             and ^(missing(&SUBVAR))
                                        %end;
                                             )
                                    );
%if %length(&COLWHERE) > 0 %then %do;
      where &COLWHERE;
%end;

%if %length(&SUBVAR) > 0 %then %do;
      table &COLVAR*&SUBVAR   / nopercent norow nocol missprint;
   run;
%end;
%else %do;
      table &GRPVAR*&COLVAR   / nopercent norow nocol missprint;
   run;

/*******************************************************/
/* NEED TO AVOID DOUBLE VALUES FOR "Overall" ROW       */
/* SINCE ALL_N AND GRP_N ARE GRPVAR*COLVAR             */
/* WHERE missing(GRPVAR) ARE VALUES THAT OVERLAP AND   */
/* ONLY WANT VALUES FROM ALL_N FOR "Overall" ROW       */
/*******************************************************/
   data ALL_N;
     set ALL_N;
     if missing(&GRPVAR);
   run;

%end; /* ELSE DO FOR %length(&SUBVAR) > 0 */
%end; /* IF %length(&ALLFLG) > 0          */

** TOTALS FOR OVERALL GROUP/SOC ROWS;
%if %length(&GRPFLG) > 0 %then %do;
   ods output CrossTabFreqs = GRP_N (where= (frequency > 0));
   proc freq data= &EVTDATA         (where= (&GRPFLG   = "Y"
                                             and ^(missing(&IDVAR  ))
                                             and ^(missing(&COLVAR ))
                                             and ^(missing(&GRPVAR ))
                                             and ^(missing(&EVTVAR ))
                                        %if %length(&SUBVAR) > 0 %then %do;
                                             and ^(missing(&SUBVAR))
                                        %end;
                                             )
                                    );

%if %length(&COLWHERE) > 0 %then %do;
      where &COLWHERE;
%end;

%if %length(&SUBVAR) > 0 %then %do;
      table &GRPVAR*&SUBVAR*&COLVAR / nopercent norow nocol missprint;
   run;
%end;
%else %do;
      table &GRPVAR*&COLVAR         / nopercent norow nocol missprint;
   run;

/* CANNOT HAVE DOUBLE VALUES FOR "Overall" ROW          */
/* SINCE ALL_N AND GRP_N ARE GRPVAR*COLVAR              */
/* WHERE missing(GRPVAR) ARE VALUES THAT OVERLAP AND    */
/* ONLY WANT VALUES FROM ALL_N FOR "Overall" ROW        */
/*******************************************************/
   data GRP_N;
     set GRP_N;
     if ^missing(&GRPVAR);
   run;

%end; /* ELSE DO FOR %length(&SUBVAR) > 0  */
%end; /* IF %length(&GRPFLG) > 0           */

/* TOTALS FOR EVENT ROWS */
%if %length(&EVTFLG) > 0 %then %do;
```

```
   ods output CrossTabFreqs = EVT_N (where= (frequency > 0) drop= Table);
   proc freq data= &EVTDATA         (where= (&EVTFLG    = "Y"
                                             and ^(missing(&IDVAR  ))
                                             and ^(missing(&COLVAR ))
                                             and ^(missing(&GRPVAR ))
                                             and ^(missing(&EVTVAR ))
                                        %if %length(&SUBVAR) > 0 %then %do;
                                             and ^(missing(&SUBVAR))
                                        %end;
                                            )
                                    );

%if %length(&COLWHERE) > 0 %then %do;
      where &COLWHERE;
%end;

%if %length(&SUBVAR) > 0 %then %do;
      table &GRPVAR*&EVTVAR*&SUBVAR*&COLVAR / nopercent norow nocol missprint;
   run;
%end;
%else %do;
      table &GRPVAR*&EVTVAR*&COLVAR         / nopercent norow nocol missprint;
   run;

/*******************************************************/
/* NEED TO AVOID DOUBLE VALUES FOR GROUP ROWS          */
/* WHERE missing(EVTVAR) ARE VALUES THAT OVERLAP WITH  */
/* THOSE IN GRP_N AND ONLY WANT VALUES FROM GRP_N FOR  */
/* GROUP ROWS                                          */
/*******************************************************/
   data EVT_N;
     set EVT_N;
     if ^missing(&EVTVAR);
   run;

%end;  /* ELSE DO FOR %length(&SUBVAR) > 0 */
%end;  /* IF %length(&EVTFLG) > 0          */

   ods listing select all;

/*******************************************************/
/* PROC PRINT ANY DATASETS OF FREQUENCIES OBTAINED     */
/* THIS NEEDED TO BE SEPARATE FROM CODE ABOVE BECAUSE  */
/* OF THE ods listing select all STATEMENT             */
/* ONLY PRINT IF LSTMORE = Y                           */
/*******************************************************/
%if %length(&ALLFLG) > 0 %then %do;

   %if &LSTMORE = Y %then %do;
      title  "PRINT OF FREQS FOR THE ALL_N/OVERALL ROW";
      title2 "MISSING VALUES IN &COLVAR ARE FOR TOTAL COLUMN";
    %if %length(&SUBVAR) > 0 %then %do;
      title3 "MISSING VALUES IN &SUBVAR ARE TOTAL COLUMNS FOR EACH &COLVAR VALUE";
    %end;
      proc print data= All_N;
      run;

      title "CONTENTS FOR ALL_N/OVERALL ROW";
      proc contents data= All_N;
      run;
   %end;

/* CODE TO USE IN set STATEMENT BELOW THAT PUTS ALL FREQS TOGETHER */
   %let SETALL = %str(ALL_N (keep= &COLVAR &SUBVAR frequency));

%end;

%if %length(&GRPFLG) > 0 %then %do;

   %if &LSTMORE = Y %then %do;
      title  "PRINT OF FREQS FOR SOC_N/GROUP ROWS";
      title2 "MISSING VALUES IN &COLVAR ARE FOR TOTAL COLUMN";
    %if %length(&SUBVAR) > 0 %then %do;
      title3 "MISSING VALUES IN &SUBVAR ARE TOTAL COLUMNS FOR EACH &COLVAR";
    %end;

      proc print data= GRP_N; *(obs=20);
      run;

      title "CONTENTS FOR SOC_N/GROUP ROWS";
      proc contents data= GRP_N;
      run;
   %end;

/* CODE TO USE IN set STATEMENT BELOW THAT PUTS ALL FREQS TOGETHER */
   %let SETGRP = %str(GRP_N (keep= &COLVAR &SUBVAR frequency &GRPVAR));

%end;

%if %length(&EVTFLG) > 0 %then %do;

   %if &LSTMORE = Y %then %do;
      title  "PRINT OF FREQS FOR PT_N/EVENT ROWS";
      title2 "MISSING VALUES IN &COLVAR ARE FOR TOTAL COLUMN";
    %if %length(&SUBVAR) > 0 %then %do;
```

```
      title3 "MISSING VALUES IN &SUBVAR ARE TOTAL COLUMNS FOR EACH &COLVAR";
    %end;

      proc print data= EVT_N; *(obs=20);
      run;

      title "CONTENTS FOR PT_N/EVENT ROWS";
      proc contents data= EVT_N;
      run;
   %end;

/* CODE TO USE IN set STATEMENT BELOW THAT PUTS ALL FREQS TOGETHER */
   %let SETEVT =  %str(EVT_N (keep= &COLVAR &SUBVAR frequency &GRPVAR &EVTVAR));

%end;

/*********************************************************************/
/* PULL ALL OF THE VARIOUS TOTALS TOGETHER                           */
/* OVERALL COLVAR IS DEFINED AS "TOTAL" IN rptcol WHEN COLVAR missing */
/* OVERALL SUBVAR IS DEFINED AS "All"                                */
/*********************************************************************/
   data soclose;
       set &SETALL
           &SETGRP
           &SETEVT
           end = eof
       ;

/* ENTER DENOM VARIABLES */
     length &DENOMLIST 8.;
     %do D=1 %to &CLISTMAX;
        &&FLAGDENOM&D;
        label &&LBLDENOM&D;
     %end;

/* THESE RECS ARE TOTALS */
/* DENOM VARIABLE LOGIC MUST STAY WITHIN COLVTYPE CODE */
       denom&COLTOTVAL        = &MACCOMTOT;* sum(&DENOMLISTCOM);
       %let DENOMTOTAL        = denom&COLTOTVAL;
       label denom&COLTOTVAL  = "DENOM FOR ALL COLVAR VALUES";

/* ADD VARIABLES WITH VALID VARIABLE VALUES FOR COLVAR VALUES */
/* MAKE percent AND pctdenom VARS */
       length colvnm $&CVNMTOP.. percent 8. ;
   %do E=1 %to &CLISTMAX;
     %if &E = 1 %then %do;
       if      strip(vvalue(&COLVAR)) = "&&THISCVAL&E" then do;
     %end;
     %else %do;
       else if strip(vvalue(&COLVAR)) = "&&THISCVAL&E" then do;
     %end;
                    colvnm    = "%nrbquote(&&VARCNAME&E)";
                    percent   = 100*(frequency/denom&&VARCNAME&E);
                    pctdenom  = denom&&VARCNAME&E;
       end;
   %end;
/* END OF &E TO &CLISTMAX LOOP */
       else do;
                    colvnm    = strip(vvalue(&COLVAR));
                    percent   = 100*(frequency/&DENOMTOTAL);
                    pctdenom  = &DENOMTOTAL;
       end;

       label        colvnm    = "COLVAR VALUE USED IN DENOM"
                    percent   = "PERCENT = 100*(frequency/denom<colvnm>)"
                    pctdenom  = "DENOM VALUE USED FOR EACH percent CALCULATION"
                    ;

/* VAR CREATION FOR SIMPLE REPORTING */
        rptpct = '('||strip(put(percent,&PCTFMT))||'%)';
        label   rptpct    = "FORMATTED PERCENT VALUE FOR REPORT"
                ;

/* CREATE grpev1col VAR WITH INDENTATION FOR EVENT ROWS */
       length grpev1col $&GRPEVFTW;
            if missing(&GRPVAR) and
               missing(&EVTVAR) then  grpev1col = "Overall";
       else if missing(&EVTVAR) then  grpev1col = strip(vvalue(&GRPVAR));
       else                           grpev1col = cat('    ',strip(vvalue(&EVTVAR)));

       label grpev1col = "%upcase(&GRPVAR) and %upcase(&EVTVAR) values in one variable"
         ;

/* ADD rptgrp AND rptevt VARS FOR FLAGRPT AUTOMATION */
       length rptgrp $&GRPVARFTW;
              rptgrp = strip(vvalue(&GRPVAR));
       label  rptgrp = "RPTGRP: FLAGSUM &GRPVAR VARIABLE FOR AUTOMATION OF FLAGRPT";

       length rptevt $&EVTVARFTW;
              rptevt = strip(vvalue(&EVTVAR));
       label  rptevt = "RPTEVT: FLAGSUM &EVTVAR VARIABLE FOR AUTOMATION OF FLAGRPT";

/* MAKE REPORTING VARS FOR COLVAR AND SUBVAR   */
/* NEED INCREASED LENGTH                       */
/* ADDED &TOTLEN TO %eval() BELOW              */
```

```
/*        TOTLEN = LENGTH OF denomTOTAL value  */
   %let RPTCOLW = %eval(&COLVARFTW.+&TOTLEN.+10);
        length rptcol $&RPTCOLW..;
        label  rptcol = "REPORT READY &COLVAR WITH/WITHOUT (N=) VALUE";

/* IF PCTMODE = 2 OR 9, NO DENOMS IN THE TITLE    */
/* OTHERWISE DENOMS ARE IN THE TITLE AS "N="      */
   %if &PCTMODE = 2 or
       &PCTMODE = 9 %then %do;
       if missing(&COLVAR) then rptcol = "&COLTOTVAL";
       else                     rptcol = strip(vvalue(&COLVAR));
   %end;
   %else %do;
       if missing(&COLVAR) then rptcol = "&COLTOTVAL"||'$(N='||strip(pctdenom)||')';
       else                     rptcol = strip(vvalue(&COLVAR))||'$(N='||strip(pctdenom)||')';
   %end;

   %if %length(&SUBVAR) > 0 %then %do;
       length rptsub $&SUBVARFTW..;
       label  rptsub = "REPORT READY &SUBVAR";

       if missing(&SUBVAR) then rptsub = "&SUBTOTVAL";
       else                     rptsub = strip(vvalue(&SUBVAR));
   %end;

/* HELPS TO ALIGN FREQS IN .lst AND MAKE IT CHARACTER FOR REPORTING */
       label rptfrq = "REPORT READY frequency";
             rptfrq = put(frequency,3. -r);

/* GET WIDTH OF freq AND pct VARS FOR rptme VAR IN DATA STEP BELOW */
       call symput("PCTRPTW",strip(vformatw(rptpct)));
       call symput("FRQRPTW",strip(vformatw(rptfrq)));
   run;

/* CREATE MACRO VAR THAT ADDS LENGTH OF FREQ AND PCT VAR     */
/* ADDED 1 TO EQUATION FOR SPACE BETWEEN FREQ AND PCT VALUES */
%let   RPTMEW = %eval(&PCTRPTW+&FRQRPTW+1);

/* NEEDED LENGTH OF rptpct VALUES TO DETERMINE LENGTH OF rptme  */
/* AND LOGIC WAS COMPLICATED, SO USED ANOTHER DATA STEP         */
   data &OUTDATA;
      set soclose;

/* rptme CODE FOR FREQS AND PCTS TO SHOW IN REPORT */
/**************************************************/
      %if        &PCTMODE = 0 or
                 &PCTMODE = 9 %then %do;
           length rptme $&FRQRPTW..;
                  rptme     = rptfrq;
           label  rptme     = "N";
      %end;
      %else %if  &PCTMODE = 2 or
                 &PCTMODE = 8 %then %do;
           length rptme $&PCTRPTW..;
                  rptme         = rptpct;
           label  rptme     = "(%)";
      %end;
      %else %do;
           length rptme $&RPTMEW..;
           label  rptme     = "N (%)";

         %if     &PCTMODE = 3  %then %do;
                 rptme = rptfrq||' '||rptpct;
         %end;
           %else %do;
/* BELOW ARE THE CODITIONS FOR THE rptme CODE CHANGE,       */
/* BUT FINAL VALUE ASSIGNMENT CODE IS THE SAME.             */
            %if       &PCTMODE = 1  %then %do;
               if missing(&EVTVAR) then
            %end;
            %else %if &PCTMODE = 4  %then %do;
               if missing(&EVTVAR) and missing(&COLVAR) then
              %end;
            %else %if &PCTMODE = 5  %then %do;
                 if missing(&COLVAR) then
              %end;
            %else %if &PCTMODE = 6  %then %do;
               if missing(&EVTVAR) and missing(&SUBVAR) then
              %end;
            %else %if &PCTMODE = 7  %then %do;
               if missing(&SUBVAR) then
              %end;
            %else %if &PCTMODE = 10 %then %do;
               if missing(&EVTVAR) or missing(&SUBVAR) then
              %end;
/* SAME CODE TO ASSIGN VALUES */
                     rptme  = rptfrq||' '||rptpct;
               else  rptme  = rptfrq;
           %end;
      %end;

   run;

   /* KEEP TO SORT BY ALPHA ORDER FOR GRPVAR AND THEN EVTVAR */
   proc sort data = &OUTDATA;
```

```
      by &GRPVAR &EVTVAR;
   run;

/***************/
/* MAKE TITLES */
/***************/
      title  "THE DATASET PRINTED BELOW IS NAMED &OUTDATA.";
      title2 "THIS DATASET IS SORTED BY &GRPVAR AND &EVTVAR FOR EASY VIEWING.";
%if &LSTMORE = Y %then %do;

      title3 "SET LSTMORE = N, FOR _LESS_ INFORMATION IN THE .lst.";
      title4 "VARIABLES STARTING WITH rpt ARE INTENDED FOR REPORTING.";
      title5 "ALONG WITH EITHER grpev1col OR THE COMBINATION OF &GRPVAR AND &EVTVAR..";
      title6 "PRINT SHOWS AE TOTALS FOR FLAGS:  &ALLFLG &GRPFLG &EVTFLG";
   %if %length(&COLWHERE) > 0 %then %do;
      title7 "WITH COLWHERE SET TO:  &COLWHERE";
   %end;
%end;
%else %do;
      title3 "THE VARIABLES &GRPVAR AND &EVTVAR ARE NOT SHOWN.";
      title4 "SET LSTMORE = Y, FOR _MORE_ INFORMATION IN THE .lst";
      title5 "ONLY VARIABLES INTENDED FOR REPORTING ARE SHOWN.";
      title6 "SHOWS AE TOTALS FOR FLAGS:  &ALLFLG &GRPFLG &EVTFLG";
   %if %length(&COLWHERE) > 0 %then %do;
      title7 "WITH COLWHERE SET TO:  &COLWHERE";
   %end;
%end;

/* OUTPUT */
   proc print    data= &OUTDATA;
      %if &LSTMORE ^= Y %then %do;
         var grpev1col rptcol
             %if %length(&SUBVAR) %then %do;
                 rptsub
               %end;
              rptfrq pctdenom rptme;
      %end;
   run;

   title  "SHOWS ALL VARIALBLES IN &OUTDATA!!";
%if &LSTMORE ^= Y %then %do;
   title2 "NOT ALL VARIABLES ARE IN PROC PRINT ABOVE.";
%end;

   proc contents data= &OUTDATA;
   run;

/* SET missing OPTION BACK TO USER SETTING BEFORE flagsum MACRO */
options missing="&USERMISS";

%mend  flagsum;
```

APPENDIX F: FLAGRPT CODE

```
/*************************************************************************/
/* PURPOSE:       Create a report table for counts of AE (adverse events) or
/*                       a similar type table from FLAGSUM-like output.
/*
/* INPUT:         Dataset listed in INDATA
/*                REQUIRED VARIABLES:
/*                      &GRPEVTVAR      - name of variable holding group and event
/*                                        info is entered,
/*                      &COLVAR         - name of variable holding column/treatment
/*                                        info is entered,
/*                      rptme           - name of variable likely created in FLAGSUM
/*                                        that contains numbers/pcts to print in table
/*
/* OUTPUT:        Produce PROC REPORT code to be placed with ODS statements.
/*
/* ORGANIZATIONAL MACROS WITH THEIR DESCRIPTION
/* - SUGGEST MAKING SIMILAR ONES IF THEY DO NOT EXIST
/* CODE BELOW WILL NOT RUN WITHOUT THESE MACROS REPLACED
/*
/* MACROS USED:   %dataexst - check dataset exists
/*                %varexst  - check vars exist in dataset
/*                %nobs     - retrieve number of obs in a dataset
/*
/*************************************************************************/

%macro flagrpt( INDATA     =
              , GRPVAR     = rptgrp
              , EVTVAR     = rptevt
              , GRPEVTVAR  = grpev1col
              , COLVAR     = rptcol
              , SUBVAR     =

              , GRPEV1COLW =
              , PCTMODE    = 1
              , PCTFMT     =
```

```
              , SUMMARY    = N

              , EVTTITLE   = Adverse Events
              , COLTITLE   =
              , SUBTITLE   =

              , RTFFNTSZ   =

              , DEBUG  = N
              );

%local  USERLS
        CLIST _C_MAX
        A B C D E F
        ZEROLIST NEXT
        GRPEVFTW
        ;


/**************/
/* BEGIN CODE */
/**************/
/* CHECK THAT THERE IS A VALUE IN INDATA */
%if %length(&INDATA)= 0 %then %do;
    %put %upcase(error:) INDATA requires a value.;
    %put %upcase(error:) FLAGRPT will now terminate.;
    %abort abend 1;
%end;

/* CHECK INDATA DATASET EXISTS */
%dataexst(&INDATA);

/* CHECK THAT THE DATASET HAS OBSERVATIONS */
%if %nobs(&INDATA)= 0 %then %do;
    %put %upcase(note: (cbar)) &INDATA does not contain any observations.;
    %put %upcase(note: (cbar)) Please check dataset and submit it with observations.;
    %put %upcase(note: (cbar)) FLAGRPT will now return to the SAS program.;
    %return;
%end;

/*******************************************************/
/* ADD DEBUG OPTION TO PARAMETERS                      */
/*******************************************************/
%if %length(&DEBUG)= 0 %then %let DEBUG = Y;
%else                        %let DEBUG = %substr(%upcase(&DEBUG),1,1);

%if &DEBUG = Y %then %do;
   options mprint mlogic mlogicnest source2 symbolgen;
%end;

/*************************************************/
/* CHECK THAT SOME VALUE IS IN REQUIRED MACRO VARS */
/*************************************************/

%if %length(&GRPEVTVAR) = 0 or  %length(&COLVAR) = 0  %then %do;

   %put %upcase(note: (cbar)) Either GRPEVTVAR or COLVAR is missing.;
   %put %upcase(note: (cbar)) The defaults are grpev1col and rptcol, respectively.;
   %put %upcase(note: (cbar)) FLAGRPT will continue with those values.;
   %put %upcase(note: (cbar)) If this is not correct, please change these parameters and rerun.;
   %put %upcase(note: (cbar)) If this is correct and want to avoid this message, ;
   %put %upcase(note: (cbar))    please delete these parameters and rerun.;

%end;

/* SINCE USING DEFAULTS FOR GRPEVTVAR AND COLVAR IF MISSING */
/* THEN NEED TO SET THESE VALUES BEFORE USING %varexst()    */
/*******************************************************/
/* SET DEFAULT FOR GRPVAR    IF BLANK                  */
/* SET DEFAULT FOR EVTVAR    IF BLANK                  */
/* SET DEFAULT FOR GPREVTVAR IF BLANK                  */
/* SET DEFAULT FOR COLVAR    IF BLANK                  */
/* SET DEFAULT FOR EVTTITLE  IF BLANK                  */
/* SET DEFAULT FOR COLTOTVAL AND SUBTOTVAL IF BLANK    */
/*      BOTH VARS ARE GLOBAL FROM FLAGSUM              */
/*      NEED TO CHECK IF THEY EXIST FROM FLAGSUM       */
/* SET DEFAULT FOR PCTMODE   IF BLANK                  */
/*          ADJUST PCTMODE AS NEEDED                   */
/* SET DEFAULT FOR PCTFMT  IF BLANK                    */
/*           CHECK PCTFMT  EXISTS                      */
/*******************************************************/
%if %length(&GRPVAR)    = 0 %then %let GRPVAR    = rptgrp;
%if %length(&EVTVAR)    = 0 %then %let EVTVAR    = rptevt;
%if %length(&GRPEVTVAR) = 0 %then %let GRPEVTVAR = grpev1col;
%if %length(&COLVAR)    = 0 %then %let COLVAR    = rptcol;
%if %length(&EVTTITLE)  = 0 %then %let EVTTTILE  = Adverse Events;
%if %length(&SUMMARY)   = 0 %then %let SUMMARY   = N;
%else                             %let SUMMARY   = %substr(%upcase(&SUMMARY),1,1);

/* NEEDED FOR CODING 0 VALUES WHEN PCTMODE = 4-7 and 10 */
%if %symexist(COLTOTVAL) = 0 %then %do;
      %local COLTOTVAL;
      %let   COLTOTVAL = TOTAL;
   %if %length(&SUBVAR) > 0 %then %do;
      %local SUBTOTVAL;
```

```
      %let   SUBTOTVAL = All;
   %end;
%end;

options minoperator;
%if %length(&PCTMODE) = 0 %then %let PCTMODE = 1;
%if ^(&PCTMODE in 0 1 2 3 4 5 6 7 8 9 10)
                          %then %let PCTMODE = 1;

/* FYI - CANNOT CHECK EXISTENCE OF FORMATS LIKE 4. OR 7.2 */
/*       AND THESE ARE EXPECTED TO BE USED IN PCTFMT      */
/*       SO THERE IS NO CHECK FOR EXISTENCE OF FMT        */
/* flgsmft IS THE DEFAULT IN FLAGSUM AS WELL              */
/*********************************************************/
%if %length(&PCTFMT) = 0 %then %do;
   proc format;
        value flgsmft
             0<-<.5 = "<0.5"
             other  = [4.]
             ;
   run;
   %let PCTFMT = flgsmft4.;
%end;
%else %do;
   %if %sysfunc(find(&PCTFMT,%str(.))) = 0 %then %do;
      %let PCTFMT = %sysfunc(strip(&PCTFMT%str(.)));
   %end;
   %put PCTFMT: &PCTFMT;
%end;

/* DELETE PUNCTUATION AND DIGITS FROM PCTFMT VALUE  */
/* THIS GIVES NAME OF FORMAT TO SEARCH IF IT EXISTS */
%let CHKFMT = %sysfunc(compress(&PCTFMT,,dp));

/* CHECK FORMAT IF NOT SOLELY A NUMERIC FORMAT, LIKE 7. or 7.2 */
%if %length(&CHKFMT) ^= 0 %then %do;
   proc sql noprint;
      create table chkflagfmt
      as select libname, fmtname, source, fmttype
      from dictionary.formats
      where fmtname="%upcase(&CHKFMT)"
      ;
  quit;

  %if %nobs(chkflagfmt) = 0 %then %do;
     %put %upcase(error:) The following format does not exist:  &PCTFMT.;
     %put %upcase(error:) Please enter a format that exists in  PCTFMT.;
     %put %upcase(error:) Once corrections are made, please try running FLAGRPT again.;
     %put %upcase(error:) FLAGRPT will now terminate.;
     %abort abend 2;
  %end;
%end;

/***************************************/
/* CHECK THAT VARIABLES EXIST          */
/***************************************/
%varexst(DATA   = &INDATA.,
         VARS   = &GRPVAR &EVTVAR &GRPEVTVAR &COLVAR &SUBVAR,
         ADDMSG = %str(for FLAGRPT.  Please include this variable in &INDATA and rerun the program.),
         IFERR  = endsas);

/*********************************/
/* ENSURE LINESIZE IS LONG ENOUGH */
/* FOR PRINTING BY SETTING ls=max */
/*********************************/
proc sql noprint;
     select setting into: USERLS
     from sashelp.voption
     where upcase(optname)='LINESIZE';
quit;

options ls = max;

/*********************************************************/
/* CHANGE MOST PCTMODE CODES TO ONLY 4 values: 0, 1, 2, 3 */
/* VALUES > 3 NEED TO CHANGE                              */
/* FOR THE OUTPUT INTENDED BY THESE VALUES THE USER       */
/* LIKELY NEEDS TO CREATE A DATASET FIRST AND THEN        */
/* PROC REPORT IT,                                        */
/* THESE CHANGED VALUES JUST HELP US GET THRU FLAGRPT     */
/* THESE PCTMODE VALUES CANNOT BE CODED CORRECTLY B/C     */
/* DO NOT KNOW WHICH _c?_ VARS CONTAIN VALUES TO CODE     */
/* THESE OUTPUTS                                          */
/*********************************************************/
%if %length(&SUBVAR) = 0 %then %do;
   %if       &PCTMODE =  6    %then %let PCTMODE = 1;
   %else %if &PCTMODE =  7 or
             &PCTMODE = 10    %then %let PCTMODE = 3;;
%end;

   %if       &PCTMODE =  8    %then %let PCTMODE = 2;
   %else %if &PCTMODE =  9    %then %let PCTMODE = 0;;


/***********************************/
```

```
/* CALCULATE NUMBER OF _C?_ COLUMNS */
/* FOR 0 VALUES IN TABLE            */
/************************************/
/******************************************/
/* GET LIST OF FORMATTED VALUES OF COLVAR    */
/* FOR NOW: ONLY NON-MISSING VALUES INCLUDED */
/******************************************/
/* PROC SQL HAS FEEDBACK OPTION THAT SHOWS MORE ABOUT WHAT IT IS DOING  */
/* LEFT IT IN BUT COMMENTED OUT AS A REMINDER IF NEEDED FOR DEBUGGING   */
/************************************************************************/
proc sql noprint; * feedback;
   select distinct &COLVAR into : CLIST separated by '@'
   from &INDATA
   where ^(missing(&COLVAR))
   ;
quit;

/* SET _C_MAX FROM NUMBER OF ITEMS IN CLIST */
%let _C_MAX = %sysfunc(countw(%nrbquote(&CLIST),@));

/* GET VALUE WITH COLTOTVAL */
/* ONLY NEED TOTAL INFO IF PCTMODE > 3 */
%if &PCTMODE > 3 %then %do;
   %do A=1 %to &_C_MAX;
      %if %index(%qscan(%nrbquote(&CLIST), &A, @), &COLTOTVAL) > 0 %then %do;
         %local COLTOTNUM;
         %let   COLTOTNUM = &A;
         %local CTOTLIST CTOTUP;
/* Adjusted CTOTUP from +1 TO +3 FOR ADDITION OF rptgrp AND rptevt */
         %let CTOTUP = %eval(&A+3);
         %let CTOTLIST = _c&CTOTUP._;
      %end;
   %end;
   %if %symexist(CTOTLIST) = 0 %then %do;
/* CHANGE PCTMODE THAT DEAL WITH TOTITLE WHEN NO TOTITLE FOUND */
       %if &PCTMODE = 4 %then %let PCTMODE = 1;;
       %if &PCTMODE = 5 %then %let PCTMODE = 3;;
   %end;
%end;


/******************************************/
/* GET LIST OF FORMATTED VALUES OF COLVAR    */
/* FOR NOW: ONLY NON-MISSING VALUES INCLUDED */
/******************************************/
/* PROC SQL HAS FEEDBACK OPTION THAT SHOWS MORE ABOUT WHAT IT IS DOING */
/************************************************************************/
%if %length(&SUBVAR) > 0 %then %do;
   %local SLIST;
   proc sql noprint; * feedback;
      select distinct &SUBVAR into : SLIST separated by '@'
      from &INDATA
      where ^(missing(&SUBVAR))
      ;
   quit;

/*********************************************************/
/* SET _C_MAX FROM NUMBER OF ITEMS IN CLIST              */
/* MULTIPLY PREV _C_MAX VALUE BY NUMBER OF SUBVAR LEVELS */
/* THIS GETS NUMBER OF _C?_ COLUMNS FOR ZEROS ARRAY      */
/*********************************************************/
   %let _C_MAX = %eval(%eval(%sysfunc(countw(%nrbquote(&SLIST),@)))*&_C_MAX);

/* GET VALUE WITH SUBTOTVAL */
/* ONLY NEED TOTAL INFO IF PCTMODE > 3 */
   %if &PCTMODE > 3 %then %do;
      %local _S_MAX;
      %let   _S_MAX = %sysfunc(countw(%nrbquote(&SLIST),@));
      %local B;
      %do B=1 %to &_S_MAX;
         %if %index(%qscan(%nrbquote(&SLIST), &B, @), &SUBTOTVAL) > 0 %then %do;
            %local SUBTOTNUM STOTLIST SUPONE;;
            %let STOTLIST  =;
/* GET LIST OF COLTOTITLE COLUMNS */
/* Adjusted SUPONE from +1 TO +3 FOR ADDITION OF RPTGRP AND RPTEVT */
            %do C= &B %to &_C_MAX %by &_S_MAX;
                %let SUPONE = %eval(&C+3);
                %let STOTLIST = &STOTLIST _c&SUPONE._;
            %end;
         %end;
      %end;
/* GET LIST OF TOTITLE COLUMNS */
      %if &PCTMODE = 4 or &PCTMODE = 5 %then %do;
          %local NOW END;

/* CALCULATIONS BELOW FOR rptgrp AND rptevt */
/*      ADDED TO PROC REPORT FROM INDATA    */
/* +4 BECAUSE FIRST COLUMN IS _c4_          */
/* +3 TO ACCOUNT FOR grpev1col, rptgrp, AND rptevt COLUMNS      */
          %let NOW = %eval(((&COLTOTNUM-1)*&_S_MAX)+4);
          %let END = %eval((&COLTOTNUM*&_S_MAX)+3);

/* CLEAR CTOTLIST OF PREVIOUS VALUE */
          %let CTOTLIST =;
/* MAKE LIST FOR TOTITLE COLUMNS */
```

```
          %do D = &NOW %to &END;
             %let CTOTLIST = &CTOTLIST _c&D._;
          %end;
      %end;
      %if %symexist(STOTLIST) = 0 %then %do;
/* CHANGE PCTMODE THAT DEAL WITH COLTOTITLE WHEN NO COLTOTITLE FOUND */
           %if &PCTMODE = 6  or
               &PCTMODE = 10 %then %let PCTMODE = 1;;
           %if &PCTMODE = 7  %then %let PCTMODE = 3;;
      %end;
   %end;
%end;

/*************************************************************/
/* MAKE LIST OF _c?_ VARS FOR ZEROS ARRAY IN COMPUTE BLOCK BELOW */
/*************************************************************/
/* NEED TO CHANGE +1 TO +3 BECAUSE NOW INCLUDE               */
/*      RPTGRP AND RPTEVT VARIABLES INTO PROC REPORT         */
/*************************************************************/
%let    ZEROLIST = ;
%do E=1 %to &_C_MAX;
   %let NEXT     = %eval(&E+3);
   %let ZEROLIST = &ZEROLIST _c&NEXT._;
%end;

/* FIND WIDTH TO USE FOR grpev1col      */
/* IF SUMMARY = Y, DELETE INDENTED RECS */
/* AKA EVENT RECORDS                    */
/****************************************/
data _null_;
   set &INDATA;
   call symput("GRPEVFTW",strip(vformatw(&GRPEVTVAR)));
%if &SUMMARY = Y %then %do;
   if prxmatch('/\s/', grpev1col) = 1 then delete;
%end;

run;

/*************************/
/* FINAL TABLE PROC REPORT */
/*************************/
/***********************************************/
/* nocompletecols CREATES CONSISTENT COLUMNS    */
/* WHETHER VALUES WERE FOUND OR NOT FOR THOSE   */
/* VALUES                                       */
/***********************************************/

proc report nowd data= &INDATA
%if &SUMMARY = Y %then %do;
   (where=(prxmatch('/\s/', grpev1col) ^= 1))
%end;
                out = flagrptd
            split='$' missing nocompletecols

/* IF PROBLEMS, CAN ADD style(calldef lines) TO BELOW */
/******************************************************/
    %if %length(&RTFFNTSZ) > 0 %then %do;
       style(column header report summary)=[font_size=&RTFFNTSZ pt]
    %end;
    ;
   columns ("&EVTTITLE" &GRPVAR &EVTVAR &GRPEVTVAR) &COLVAR,
   %if %length(&SUBVAR) > 0 %then %do;
               &SUBVAR,
   %end;
            rptme dummy;

      define  &GRPVAR    / group order= data noprint;
      define  &EVTVAR    / group             noprint;
      define  &GRPEVTVAR / group " "
   %if %length(&GRPEV1COLW)>0 %then %do;
           style=[cellwidth=&GRPEV1COLW];
   %end;
   %else %do;
           style=[cellwidth=%eval(7*(&GRPEVFTW)/12) em];
   %end;
    ;

      define  &COLVAR    / across  "&COLTITLE " order= internal;
   %if %length(&SUBVAR) > 0 %then %do;
      define  &SUBVAR    / across  "&SUBTITLE " order= internal;
   %end;
      define  rptme      / display " " center;
      define  dummy      / noprint;
      compute dummy;

/* REMINDER */
/* PCTMODE = 8 CHANGED TO 2 */
/* PCTMODE = 9 CHANGED TO 0 */

      %if &PCTMODE < 4 %then %do;
        array zeros &ZEROLIST;
        do z=1 to dim(zeros);
           if missing(zeros(z)) then do;

        %if &PCTMODE = 1 %then %do;
```

```
               if prxmatch('/\s/', &GRPEVTVAR) ^= 1 then do;
                  zeros(z) = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
               end;
               else do;
                  zeros(z) = put(0,3. -r);
               end;
         %end;
         %else %if &PCTMODE = 0 %then %do;
                  zeros(z) = put(0,3. -r);
         %end;
         %else %if &PCTMODE = 2 %then %do;
                  zeros(z) = '('||strip(put(0,&PCTFMT))||'%)';
         %end;
         %else %if &PCTMODE = 3 %then %do;
                  zeros(z) = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
         %end;
            end;
         end;
      %end;

/* IF PCTMODE >= 4 */
/* REMINDER */
/* PCTMODE = 8 CHANGED TO 2 */
/* PCTMODE = 9 CHANGED TO 0 */

      %else %do;
         %do F=1 %to &_C_MAX;
            %local THIS;
            %let   THIS = %sysfunc(strip(%scan(&ZEROLIST, &F,' ')));

/* USE CTOTLIST FOR INDEX WHEN 4 OR 5*/
            %if &PCTMODE = 4 %then %do;
               %if %index(&CTOTLIST, &THIS) > 0 %then %do;
                     if missing(&THIS) then do;
                         if prxmatch('/\s/', &GRPEVTVAR) ^= 1 then do;
                             &THIS = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
                         end;
                         else do;
                             &THIS = put(0,3. -r);
                         end;
                     end;
               %end;
               %else %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r);
                     end;
               %end;
            %end;
            %if &PCTMODE = 5 %then %do;
               %if %index(&CTOTLIST, &THIS) > 0 %then %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
                     end;
               %end;
               %else %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r);
                     end;
               %end;
            %end;
/* USE STOTLIST FOR INDEX WHEN 6, 7, OR 10 */
            %if &PCTMODE = 6 %then %do;
               %if %index(&STOTLIST, &THIS) > 0 %then %do;
                     if missing(&THIS) then do;
                         if prxmatch('/\s/', &GRPEVTVAR) ^= 1 then do;
                             &THIS = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
                         end;
                         else do;
                             &THIS = put(0,3. -r);
                         end;
                     end;
               %end;
               %else %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r);
                     end;
               %end;
            %end;
            %if &PCTMODE = 7 %then %do;
               %if %index(&STOTLIST, &THIS) > 0 %then %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
                     end;
               %end;
               %else %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r);
                     end;
               %end;
            %end;
            %if &PCTMODE = 10 %then %do;
               %if %index(&STOTLIST, &THIS) > 0 %then %do;
                     if missing(&THIS) then do;
                             &THIS = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
                     end;
```

```
               %end;
               %else %do;
                    if missing(&THIS) then do;
                        if prxmatch('/\s/', &GRPEVTVAR) ^= 1 then do;
                            &THIS = put(0,3. -r)||' '||'('||strip(put(0,&PCTFMT))||'%)';
                        end;
                        else do;
                            &THIS = put(0,3. -r);
                        end;
                    end;
               %end;
            %end;
         %end; /* FOR F DO LOOP */
      %end;    /* FOR ELSE DO OF PCTMODE >= 4 */

         endcomp;
/* GET INDENT OF &GRPEVTVAR EVENT ROWS */
         compute &GRPEVTVAR;
         %if &SUMMARY = Y %then %do;
              call define(_col_,"style","style=[just=l]");
              if strip(&GRPEVTVAR)= "Overall" then do;
                 call define(_row_,"style","style=[vjust=bottom font_weight=bold]");
              end;
         %end;
         %else %do;
              if prxmatch('/\s/', &GRPEVTVAR) ^= 1 then do;
                 call define(_col_,"style","style=[just=l]");
                 call define(_row_,"style","style=[vjust=bottom font_weight=bold]");
              end;
              else do;
                 call define(_col_,"style","style=[leftmargin=4 em]");
              end;
         %end;
         endcomp;
run;

/* RETURN LINESIZE TO USER SPECIFICATION */
options ls = &USERLS;

%mend  flagrpt;
```