

Paper CC-145

PROC DOC III: Self-generating Codebooks Using SAS®

Louise Hadden, Abt Associates Inc.

ABSTRACT

This paper will demonstrate how to use good documentation practices and SAS® to easily produce attractive, camera-ready data codebooks (and accompanying materials such as label statements, format assignment statements, etc.) Four primary steps in the codebook production process will be explored: use of SAS metadata to produce a master documentation spreadsheet for a file; review and modification of the master documentation spreadsheet; import and manipulation of the metadata in the master documentation spreadsheet to self-generate code to be included to generate a codebook; and use of the documentation metadata to self-generate other helpful code such as label statements. Full code for the example shown (using the SASHELP.HEART data base) will be provided.

INTRODUCTION

The most onerous task any SAS programming professional faces is to accurately document files and processes. The truth is that there are no easy answers to the documentation quandary. It takes hard, painstaking work! By setting careful standards at the outset of a programming task, documenting your processes, labelling your data files and variables, providing value labels (formats) for your variables when appropriate, and using the many tools the SAS® system provides to assist in the documentation process, producing codebooks can be a piece of cake.

ON THE OTHER SIDE OF THE MOUNTAIN: CODEBOOK GENERATION

You've done a lot of hard work documenting every aspect of your programming project, and now it is time to reap your rewards. There are a number of ways that you can present information from PROC CONTENTS and PROC DATASETS covered in many other papers, including some of my own. We are going to focus on the use of an intermediate spreadsheet to drive creation of a robust codebook with self-generating code.

STEP 1

It is important to review and evaluate the metadata associated with the data set to be documented. Data sets should be labeled accurately. Variables should be labelled accurately. If variables have informats or formats, that information should be available and accurate. There should be a program available to create a permanent format library with a two level catalog name, if applicable – and those formats should be accurate. For our example, we create an age category variable that we wish to format, and write a program to generate a format in a permanent, two-level format catalog.

Code snippet from 1gen_formats_SESUG_CC145.sas – supplied in the appendix:

```
TITLE1 "SESUG 2016 CC145";
FOOTNOTE1 "%SYSFUNC(GETOPTION(SYSIN)) - &SYSDATE - &SYSTIME - run by
&SYSUSERID in &SYSPROCESSMODE";
RUN;

LIBNAME dd '.';
LIBNAME library '.';
FILENAME odsout '.';
RUN;

PROC FORMAT LIBRARY=LIBRARY.HEART;
VALUE startage 25 - 34='25 to 34 years'
               35 - 44='35 to 44 years'
               45 - 54='45 to 55 years'
```

```

                    55 - 64='55 to 64 years';
VALUE agefmt 1='25 to 34 years'
              2='35 to 44 years'
              3='45 to 54 years'
              4='55 to 64 years';

RUN;
```

STEP 2

In the example shown below, a Microsoft Excel spreadsheet with selected variables from PROC CONTENTS output is generated using PROC EXPORT in program 2gen_metadata_SESUG_CC145.sas. I am using a modified copy of SASHELP.HEART as the sample data set for several reasons, one of which is that not all variables are labelled, requiring some changes. Another reason is that this data set is available to all SAS users.

Code snippet from gen_metadata_SESUG_CC145.sas – supplied in the appendix:

```

DATA dd.heart (LABEL="Copy of SASHELP.HEART for SESUG 2016 CC145 - created
by %SYSFUNC(GETOPTION(SYSIN))
- &SYSDATE - &SYSTIME - run by &SYSUSERID in &SYSPROCESSMODE");
LENGTH dslabel $ 200 source $ 32;
SET sashelp.heart;

/* put in some missing labels */

dslabel="Copy of SASHELP.HEART for SESUG 2016 CC145 - created by
%SYSFUNC(GETOPTION(SYSIN))
- &SYSDATE - &SYSTIME - RUN by &SYSUSERID in &SYSPROCESSMODE";

source="&dsname";

IF 25 LE ageatstart LE 34 THEN age=1;
IF 35 LE ageatstart LE 44 THEN age=2;
IF 45 LE ageatstart LE 54 THEN age=3;
IF 55 LE ageatstart LE 64 THEN age=4;

IF ageatstart ge 85 THEN age=7;
FORMAT age agefmt.;

LABEL choLEsterol='Cholesterol level'
      diastolic='Diastolic blood pressure'
      height='Height'
      sex='Gender'
      smoking='Cigarettes per day'
      status='Wanted, dead or alive'
      systolic='Systolic blood pressure'
      weight='Weight'
      source='Data set name'
      dslabel='Data set information'
      age='Age at Start Category'
      ;

RUN;
```

```

. . . PROC EXPORT DATA = dd.heart_cb DBMS = excel
OUTFILE = ".\heart_db.xlsx" REPLACE;
RUN;

```

Of course, you want to review the results of your spreadsheet creation in Excel and maybe modify a label or format assignment. Note that I have created a variable indicating a specialized variable type (VARTYPE), as I want to treat formatted variables differently from unformatted variables.

Figure 2: Screenshot of Microsoft Excel® worksheet created by program 2gen_metadata_SESUG_CC145.sas

	A	B	C	D	E	F	G	H	I	J	K
	varnum	vartype	name	label	format	length	npos	type	source	dsinfo	
1											
2	1	2	dslabel	Data set information		200	88	2	HEART	Copy of SASHELP	
3	2	2	source	Data set name		32	288	2	HEART	Copy of SASHELP	
4	3	2	Status	Wanted, dead or alive		5	320	2	HEART	Copy of SASHELP	
5	4	2	DeathCau	Cause of Death		26	325	2	HEART	Copy of SASHELP	
6	5	3	AgeCHDdi	Age CHD Diagnosed		8	0	1	HEART	Copy of SASHELP	
7	6	2	Sex	Gender		6	351	2	HEART	Copy of SASHELP	
8	7	3	AgeAtStar	Age at Start		8	8	1	HEART	Copy of SASHELP	
9	8	3	Height	Height		8	16	1	HEART	Copy of SASHELP	
10	9	3	Weight	Weight		8	24	1	HEART	Copy of SASHELP	
11	10	3	Diastolic	Diastolic blood pressure		8	32	1	HEART	Copy of SASHELP	
12	11	3	Systolic	Systolic blood pressure		8	40	1	HEART	Copy of SASHELP	
13	12	3	MRW	Metropolitan Relative Weight		8	48	1	HEART	Copy of SASHELP	
14	13	3	Smoking	Cigarettes per day		8	56	1	HEART	Copy of SASHELP	
15	14	3	AgeAtDea	Age at Death		8	64	1	HEART	Copy of SASHELP	
16	15	3	Cholester	Cholesterol level		8	72	1	HEART	Copy of SASHELP	
17	16	2	Chol_Stat	Cholesterol Status		10	357	2	HEART	Copy of SASHELP	
18	17	2	BP_Status	Blood Pressure Status		7	367	2	HEART	Copy of SASHELP	
19	18	2	Weight_S	Weight Status		11	374	2	HEART	Copy of SASHELP	
20	19	2	Smoking_	Smoking Status		17	385	2	HEART	Copy of SASHELP	
21	20	1	age	Age at Start Category	AGEFMT	8	80	1	HEART	Copy of SASHELP	
22											
23											
24											
25											

You can then reimport the modified spreadsheet for use in the next step to: (a) write code to be included to generate a codebook with output varying by variable type; (b) write code to generate a label statement; and (c) write code to generate a format assignment statement, among other normally onerous tasks.

STEP 3

The codebook generation program, 3gen_codebook_SESUG_CC145.sas, starts with reimporting the edited version of the metadata spreadsheet, shown above. A number of macros are then constructed: to report on "header information" (i.e. variable name, label, etc.), missing values, and then details on non-missing values, differential by variable type (character, continuous, categorical). Additionally, the

program accesses the metadata and outputs text files with macro calls to the macros created above conditional upon the variable type in the metadata and reporting macros, that are then reused in the program as include files.

Code snippet from 3gen_codebook_SESUG_CC145.sas – supplied in the appendix:

```
DATA _null_;
  FILE out1 LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (KEEP=varnum name vartype);

  include_string=CATS('%header(',name,',',varnum,');');
  PUT include_string;
RUN;

. . .

DATA _null_;
  FILE out4 LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (KEEP=varnum name vartype);

  IF vartype=1 THEN include_string=CATS('%printtable(',varnum,');');
  IF vartype=2 THEN include_string=CATS('%printtablec(',varnum,');');
  IF vartype=3 THEN include_string=CATS('%printblurb(',varnum,');');

  PUT include_string;
RUN;
```

Macros are written to report on each variable, creating an RTF codebook. These printing macros are utilized in the %include files written by the program inside a TAGSETS.RTF sandwich.

Code snippet from 3gen_codebook_SESUG_CC145.sas – supplied in the appendix

```
%MACRO printblurb(order);

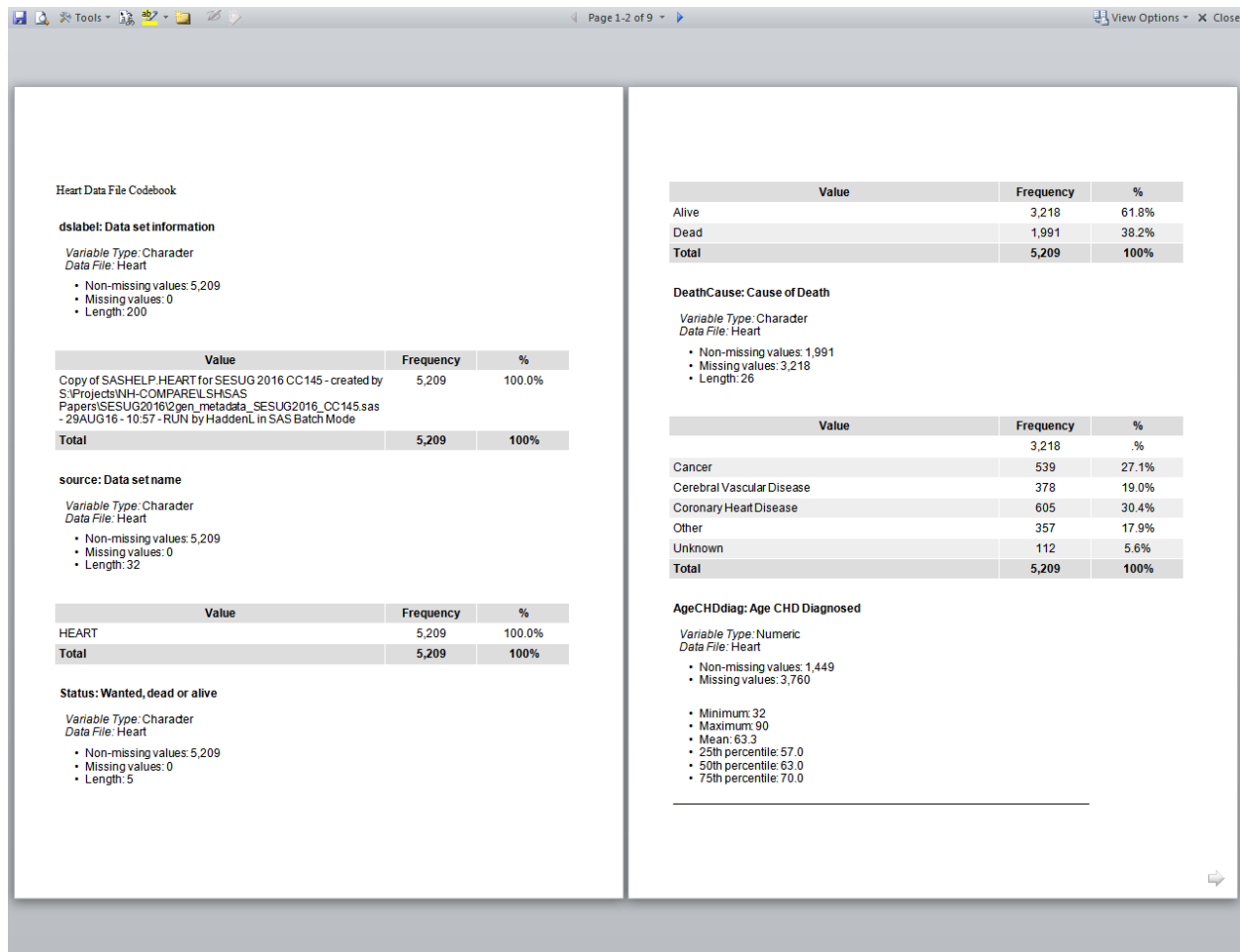
ODS TAGSETS.RTF STYLE=styles.noborder;
ODS STARTPAGE=no;

PROC REPORT NOWD DATA=print&order
  STYLE(report)=[cellpadding=3pt vjust=b]
  STYLE(header)=[just=center font_face=Helvetica font_weight=bold
                  font_size=10pt]
  STYLE(lines)=[just=left font_face=Helvetica] ;
  COLUMNS blurb ;
  DEFINE blurb / style(COLUMN)={just=1 font_face=Helvetica
                              font_size=10pt cellwidth=988 }
                style(HEADER)={just=1 font_face=Helvetica
                              font_size=10pt };
RUN;

ODS STARTPAGE=no;

%MEND;
```

Figure 3: Screenshot of two pages from RTF codebook created by program gen_codebook_SESUG_CC145.sas



The codebook construction can take some time. Arrange to send yourself a text message with the condition code of your job when it finishes, and get a cup of coffee.

Code snippet from 3gen_codebook_SESUG_CC145.sas – supplied in the appendix:

```
FILENAME msg EMAIL TO="0000000000@txt.att.net"
FROM = "Big Nerd <louise_hadden@abtassoc.com>"
SUBJECT="All Systems Go (or not)?" ;

DATA _null_;
FILE msg;
PUT "Program Path and Name: %SYSFUNC(GETOPTION(SYSIN))";
PUT "RUN &SYSDATE - &SYSTIME - by &SYSUSERID in &SYSPROCESSMODE";
PUT "Condition Code is &SYSCC.";
RUN;
```

STEP 4

Similarly, metadata can be accessed to create label, format, and length, etc. statements.

Code snippet from 4gen_label_fmt_stmnt_SESUG_CC145.sas – supplied in the appendix:

```
DATA temp1;
  LENGTH include_string $ 180;
  SET dd.heart_cb;

  label=COMPRESS(label, '');

  qlabel=CATS(' ', label, '');
  include_string=CATX(' ', name, '=', qlabel);
RUN;

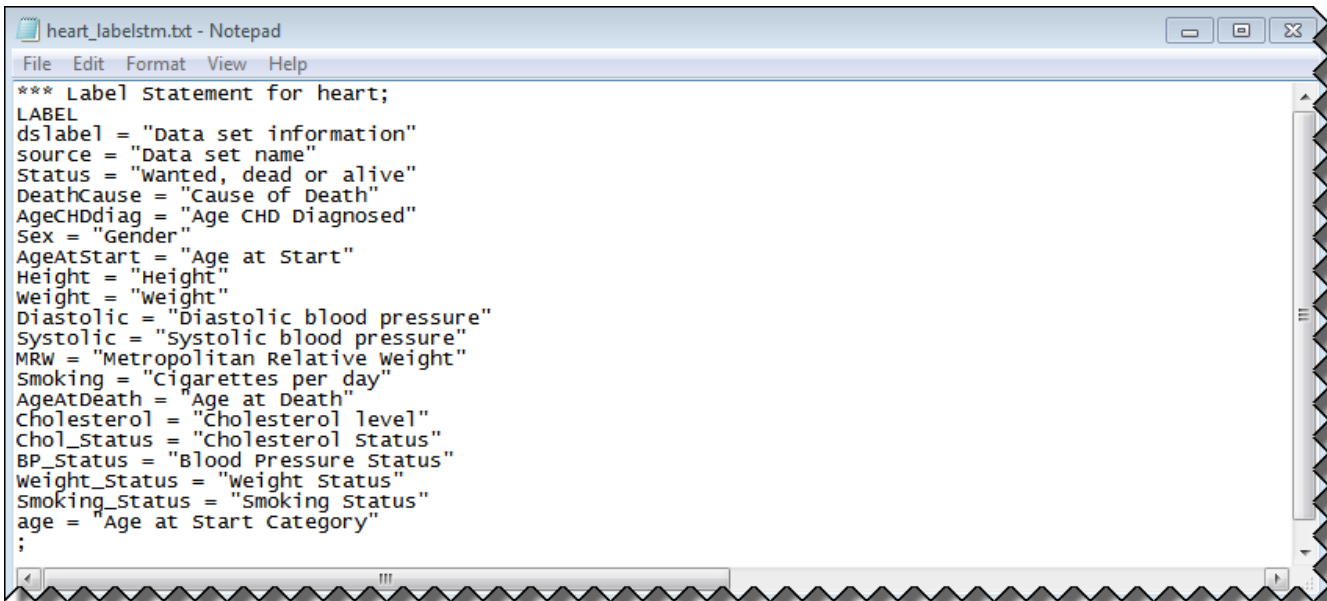
DATA templabel (KEEP=include_string);
  FILE out1 LRECL=180 PAD;
  LENGTH include_string $ 180;
  SET runlabel temp1 runrun;
  PUT include_string;
RUN;

DATA temp2;
  LENGTH include_string $ 180;
  SET dd.heart_cb (WHERE=(format NE ''));
  qformat=CATS(format, '.');
  include_string=CATX(' ', name, qformat);
RUN;

DATA tempfmt (KEEP=include_string);
  FILE out2 LRECL=180 PAD;
  LENGTH include_string $ 180;
  SET runformat temp2 runrun;
  PUT include_string;
RUN;
```

The resulting statement, example shown below, can be included in other programs seamlessly.

Figure 4: Screenshot of label statement created by program 4gen_label_fmt_stmnt_SESUG_CC145.sas



```
heart_labelstm.txt - Notepad
File Edit Format View Help
*** Label Statement for heart;
LABEL
dslabel = "Data set information"
source = "Data set name"
Status = "wanted, dead or alive"
DeathCause = "Cause of Death"
AgeCHDdiag = "Age CHD Diagnosed"
Sex = "Gender"
AgeAtStart = "Age at Start"
Height = "Height"
weight = "weight"
Diastolic = "Diastolic blood pressure"
Systolic = "systolic blood pressure"
MRW = "Metropolitan Relative Weight"
Smoking = "Cigarettes per day"
AgeAtDeath = "Age at Death"
Cholesterol = "Cholesterol level"
chol_status = "Cholesterol Status"
BP_status = "Blood Pressure Status"
weight_status = "Weight Status"
smoking_status = "Smoking Status"
age = "Age at Start Category"
;
```

CONCLUSION

Only code snippets are shown here: full code is available in the appendix and from the authors upon request.

The SAS system provides numerous opportunities for creating self-documenting data sets. With care at the onset of a project, SAS programmers can utilize the power of the SAS system to ensure quality data and accurate documentation. Simple documentation is not enough to ensure quality data and analyses. SAS can show us the way to create user-friendly documentation, and generate components of your SAS programs without typing a word.

REFERENCES

- Carey, Helen and Carey, Ginger, 2011. "Tips and Techniques for the SAS Programmer!" Proceedings of SAS Global Forum 2011.
- Crawford, Peter, 2013. "A Day in the Life of Data – Part 3." Proceedings of SAS Global Forum 2013.
- Fraeman, Kathy Hardis, 2008. "Get into the Groove with %SYSFUNC: Generalizing SAS® Macros with Conditionally Executed Code." Proceedings of NESUG 2008.
- Hadden, Louise, 2014. "Build your Metadata with PROC CONTENTS and ODS OUTPUT", Proceedings of SAS Global Forum 2014.
- Huang, Chao, 2014. "Top 10 SQL Tricks in SAS®." Proceedings of SAS Global Forum 2014.
- Karafa, Matthew T., 2012. "Macro Coding Tips and Tricks to Avoid "PEBCAK" Errors." Proceedings of SAS Global Forum 2012.
- Kuligowski, Andrew T. and Shankar, Charu, 2013. "Know Thy Data: Techniques for Data Exploration." Proceedings of SAS Global Forum 2013.
- Lafler, Kirk Paul, 2014. "Powerful and Hard-to-find PROC SQL Features." Proceedings of SAS Global

Forum 2014.

Murphy, William C., 2013. “What’s in a SAS® Variable? Get Answers with a V!” Proceedings of SAS Global Forum 2013.

Raithel, Michael A., 2011. “PROC DATASETS: the Swiss Army Knife of SAS® Procedures.” Proceedings of SAS Global Forum 2011.

Thornton, Patrick, 2011. “SAS® DICTIONARY: Step by Step.” Proceedings of SAS Global Forum 2011.

Zhang, Jingxian, 2012. “Techniques for Generating Dynamic Code from SAS® Dictionary Tables.” Proceedings of SAS Global Forum 2012.

ACKNOWLEDGMENTS

The author gratefully acknowledges the helpful work of Kathy Fraeman, Michael Raithel, Patrick Thornton, Roberta Glass and Kirk Paul Lafler, among others.

CONTACT INFORMATION

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

Louise Hadden: Louise_Hadden@abtassoc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Appendix

README for included code for SESUG 2016 – CC145

Codebook generation programs:

Designed to be run in order. Uses a SAS-provided data set as input (SASHELP.HEART). Written to be used in batch mode processing – if using with interactive modes of SAS, some features may not work and libname and filenames need to be given full paths. Remove or comment out the ENDSAS; command if running interactively. There is a workaround for %SYSFUNC code while running interactively, so if you have problems with the codebook generation programs, contact louise_hadden@abtassoc.com.

1. 1gen_formats_SESUG2016_CC145.sas – produces user-defined formats for use in the suite of codebook programs
2. 2gen_metadata_SESUG2016_CC145.sas – produces metadata on SASHELP.HEART database for use in the suite of codebook programs
3. 3gen_codebook_SESUG2016_CC145.sas – produces a codebook for an updated copy of the SASHELP.HEART data set
4. 4gen_label_fmt_stmnt_SESUG2016_CC145.sas – produces label and format assignment statements for the updated copy of SASHELP.HEART data set

```
*Program 1: lgen_formats_SESUG2016_CC145.sas;

OPTIONS PS=55 LS=175 ERRORABEND NOFMterr ERRORS=1 MPRINT SYMBOLGEN NOCENTER NODATE
NONUMBER;
TITLE1 "SESUG 2016 CC145";
FOOTNOTE1 "%SYSFUNC(GETOPTION(SYSIN)) - &SYSDATE - &SYSTIME - run by &SYSUSERID in
&SYSPROCESSMODE";
RUN;

LIBNAME dd '.';
LIBNAME library '.';
FILENAME odsout '.';
run;

PROC FORMAT LIBRARY=LIBRARY.HEART;

    VALUE startage 25 - 34='25 to 34 years'
                  35 - 44='35 to 44 years'
                  45 - 54='45 to 55 years'
                  55 - 64='55 to 64 years';

    VALUE agefmt 1='25 to 34 years'
                2='35 to 44 years'
                3='45 to 54 years'
                4='55 to 64 years';

RUN;

ENDSAS;
```

```

*Program 2: lgen_metadata_SESUG2016_CC145.sas;

OPTIONS PS=55 LS=175 ERRORABEND NOFMterr ERRORS=1 MPRINT SYMBOLGEN NOCENTER;

TITLE1 "SESUG 2016 CC145";
FOOTNOTE1 "%SYSFUNC(GETOPTION(SYSIN)) - &SYSDATE - &SYSTIME - run by &SYSUSERID in
&SYSPROCESSMODE";
RUN;

LIBNAME dd '.';
LIBNAME library '.';
FILENAME odsout '.';
RUN;

%LET dsname=HEART;

ODS RTF FILE='sashelpheart_CONTENTS.RTF' PATH=odsout STYLE=styles.journal2;

PROC CONTENTS DATA=sashelp.heart;
RUN;

PROC PRINT DATA=sashelp.heart (OBS=5) NOOBS;
RUN;

ODS RTF CLOSE;

PROC FORMAT ;
    VALUE startage 25 - 34='25 to 34 years'
                  35 - 44='35 to 44 years'
                  45 - 54='45 to 55 years'
                  55 - 64='55 to 64 years';

    VALUE agefmt 1='25 to 34 years'
                2='35 to 44 years'
                3='45 to 54 years'
                4='55 to 64 years';
RUN;

DATA dd.heart (LABEL="Copy of SASHELP.HEART for SESUG 2016 CC145 - created by
%SYSFUNC(GETOPTION(SYSIN))
- &SYSDATE - &SYSTIME - run by &SYSUSERID in &SYSPROCESSMODE");
    LENGTH dslabel $ 200 source $ 32;
    SET sashelp.heart;

    /* put in some missing labels */

    dslabel="Copy of SASHELP.HEART for SESUG 2016 CC145 - created by
%SYSFUNC(GETOPTION(SYSIN))
- &SYSDATE - &SYSTIME - RUN by &SYSUSERID in &SYSPROCESSMODE";
    source="&dsname";

    IF 25 LE ageatstart LE 34 THEN age=1;
    IF 35 LE ageatstart LE 44 THEN age=2;
    IF 45 LE ageatstart LE 54 THEN age=3;
    IF 55 LE ageatstart LE 64 THEN age=4;

    IF ageatstart ge 85 THEN age=7;
    FORMAT age agefmt.;

    LABEL choLEsterol='Cholesterol level'
           diastolic='Diastolic blood pressure'
           height='Height'
           sex='Gender'

```

```

        smoking='Cigarettes per day'
        status='Wanted, dead or alive'
        systolic='Systolic blood pressure'
        weight='Weight'
        source='Data set name'
        dslabel='Data set information'
        age='Age at Start Category'
    ;

RUN;

PROC CONTENTS DATA=dd.heart OUT=heartconts;
RUN;

PROC PRINT DATA=heartconts (OBS=5) NOOBS;
RUN;

PROC FREQ DATA=dd.heart;
    TABLES age*ageatstart / MISSING LIST;
RUN;

DATA heartcontsa;
    SET heartconts (KEEP=name format label length name npos type varnum memname
memlabel);

    IF type=1 AND format NE '' THEN vartype=1;
    IF type=1 AND format EQ '' THEN vartype=3;
    IF type=2 THEN vartype=2;

    source=memname;
    dsinfo=memlabel;
    DROP mem: ;

RUN;

DATA dd.heart_cb;
    RETAIN varnum vartype name label FORMAT Length npos type source dsinfo;
    SET heartcontsa;
RUN;

PROC SORT DATA=dd.heart_cb;
    BY varnum;
RUN;

PROC EXPORT DATA = dd.heart_cb DBMS = excel
    OUTFILE = ".\heart_db.xlsx" REPLACE;
RUN;

/* if running interactively comment endsas out */
ENDSAS;
*Program 3: 3gen_codebook_SESUG2016_CC145.sas;

OPTIONS PS=55 LS=175 ERRORABEND NOFMTERR ERRORS=1 MPRINT SYMBOLGEN NOCENTER NODATE
NONUMBER
;

TITLE1 "SGF2016 - Session 8300 - Heart Codebook";
FOOTNOTE1 "%SYSFUNC(GETOPTION(SYSIN)) - &SYSDATE - &SYSTIME - run by &SYSUSERID in
&SYSPROCESSMODE";
RUN;

OPTIONS FMTSEARCH=(library.heart work);

```

```

TITLE2 "HEART";
RUN;

LIBNAME dd '.';
FILENAME odsout ".";
FILENAME out1 ".\heart_include1.txt"; /* for header macro */
FILENAME out2 ".\heart_include2.txt"; /* for missval macro */
FILENAME out2a ".\heart_include2a.txt"; /* for cmissval macro */
FILENAME out3 ".\heart_include3.txt"; /* for detail macros */
FILENAME out4 ".\heart_include4.txt"; /* for printing macros */
LIBNAME library '.';
RUN;

/* step 1 - import the metadata spreadsheet for the file to be documented */
/* although it was created in SAS, it may have been modified to correct identify
variable types, etc. */

PROC IMPORT DBMS=EXCEL OUT = dd.heart_cb
  DATAFILE = ".\heart_db.xlsx" REPLACE;
RUN;

/* step 2 - create some reusable work data sets for codebook */

DATA bottomline;

  blurb='_____';

  LABEL blurb = " "; /* this is NOT blank - it is a non-printing character */

RUN;

/* step 3 - create macro to generate header information */

%MACRO header(varname,order);

DATA temp&order.a;
  LENGTH sentence1 sentence2 sentence3 $ 500 blurb $ 25000 namecolon labelfmt typec
vartypec dsc $ 250;
  SET dd.heart_cb (WHERE=(name="&varname"));

  /* we want to make a long sentence with long breaks */

  /* the first sentence is variable name and variable label */

  namecolon=CATS('^ {STYLE [FONTSIZE=10pt FONTWEIGHT=bold] ',name,"}");
  labelfmt=CATS('^ {STYLE [FONTSIZE=10pt FONTWEIGHT=bold] ',label,"}");
  sentence1=CATX(' ',namecolon,labelfmt);

  /* the second sentence is variable type */

  IF type=1 THEN typec='Numeric';
  ELSE IF type=2 THEN typec='Character';

  vartypec='^ {NEWLINE 2} ^ {NBSpace 2} ^ {style [FONTSIZE=10pt FONTSTYLE=italic] Variable
Type:}';
  sentence2=CATX(' ',vartypec,typec);

  /* the third sentence is Data File */

  dsc='^ {NEWLINE 1} ^ {NBSpace 2} ^ {STYLE [FONTSIZE=10pt FONTSTYLE=italic] Data File:}';

```

```

sentence3=CATX(' ',dsc,"Heart");;

/* now we want to string into a paragraph for printing; */

blurb=CATT(sentence1,sentence2,sentence3);

LABEL blurb =" "; /* this is NOT blank - it is a non-printing character */

RUN;

DATA header&order;
  SET temp&order.a (KEEP=blurb);
  order=&order;
RUN;

PROC PRINT DATA=header&order;
RUN;

%MEND;

/* step 4 - create macro to generate missing value information (numeric variables)
*/

%MACRO missval(varname,order);

DATA miss;
  SET dd.heart (WHERE=(&varname in(.)));
  counter=1;
RUN;

PROC MEANS DATA=miss NOPRINT;
  VAR counter;
  OUTPUT OUT=sum SUM=nmiss;
RUN;

PROC MEANS DATA=dd.heart NOPRINT;
  VAR &varname;
  OUTPUT OUT=temp&order.z N=n ;
RUN;

DATA missval&order (KEEP=blurb order);
  LENGTH sentence1 sentence2 $ 500
  blurb $ 25000 ncolon nmisscolon $ 200;
  MERGE temp&order.z sum (KEEP=nmiss);

  order=&order;

  IF nmiss=. then nmiss=0;

  ncolon='^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}Non-missing values: ';
  nmisscolon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}Missing values: ';

  putn=PUT(n,comma9.);
  putnmiss=PUT(nmiss,comma9.);

  sentence1=CATX(' ',ncolon,putn);
  sentence2=CATX(' ',nmisscolon,putnmiss);

```

```

/* now we want to string into a paragraph for printing; */

blurb=CATT(sentence1,sentence2);

label BLURB =" "; /* this is NOT blank - it is a non-printing character */

RUN;

%MEND;

/* step 4a - create macro to generate missing value information (character
variables) */

%MACRO cmissval(varname,order);

DATA appmiss;
    SET dd.heart (WHERE=(&varname IN(' ')));
    counter=1;
RUN;

DATA present;
    SET dd.heart (WHERE=(&varname NOT IN(' ')));
    counter=1;
RUN;

DATA temp&order.gc (KEEP=length);
    SET dd.heart_cb (WHERE=(name="&varname"));
RUN;

PROC MEANS DATA=appmiss NOPRINT;
    VAR counter;
    OUTPUT OUT=appsum SUM=nmiss;
RUN;

PROC MEANS DATA=present NOPRINT;
    VAR counter;
    OUTPUT OUT=temp&order.z SUM=n ;
RUN;

DATA missval&order (keep=blurb order);
    LENGTH sentence1 sentence2 sentence3 sentence4 $ 500
    blurb $ 25000 ncolon nmisscolon inappmisscolon lengthcolon $ 200;
    MERGE temp&order.z appsum (KEEP=nmiss) temp&order.gc (KEEP=length);

    order=&order;

    IF nmiss=. THEN nmiss=0;
    IF n=. THEN n=0;

    ncolon='^{NBSPACE 5}^{Unicode 2022}^{NBSPACE 2}Non-missing values: ';
    nmisscolon='^{NEWLINE 1}^{NBSPACE 5}^{Unicode 2022}^{NBSPACE 2}Missing values: ';
    lengthcolon='^{NEWLINE 1}^{NBSPACE 5}^{Unicode 2022}^{NBSPACE 2}Length: ';

    putn=PUT(n,comma9.);
    putnmiss=PUT(nmiss,comma9.);
    putlength=PUT(length,comma9.);

```

```

sentence1=CATX(' ',ncolon,putn);
sentence2=CATX(' ',nmisscolon,putnmiss);
sentence3=CATX(' ',lengthcolon,putlength);

blurb=CATT(sentence1,sentence2,sentence3);

LABEL blurb =" "; /* this is NOT blank - it is a non-printing character */

RUN;

%MEND;

/* step 5 - create a series of macros for 3 different variable types to generate
detail information */

%MACRO detailcont(varname,order);

PROC MEANS DATA=dd.heart NOPRINT;
  VAR &varname;
  OUTPUT OUT=temp&order.f MIN=min MAX=max MEAN=mean Q1=q1 MEDIAN=median Q3=q3;
RUN;

DATA detail&order (KEEP=blurb order);
  LENGTH sentence4 sentence5 sentence6 sentence7 sentence8 sentence9 $ 500
  blurb $ 25000 mincolon maxcolon meancolon qlcolon mediancolon q3colon $ 200;
  SET temp&order.f ;

  order=&order;

  mincolon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}Minimum: ';
  maxcolon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}Maximum: ';
  meancolon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}Mean: ';
  qlcolon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}25th percentile: ';
  mediancolon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}50th percentile: ';
  q3colon='^{\NEWLINE 1}^{\NBSpace 5}^{\Unicode 2022}^{\NBSpace 2}75th percentile: ';

  putmin=PUT(min,9.);
  putmax=PUT(max,9.);
  putmean=PUT(mean,comma9.1);
  putq1=PUT(q1,comma9.1);
  putmedian=PUT(median,comma9.1);
  putq3=PUT(q3,comma9.1);

  sentence4=CATX(' ',mincolon,putmin);
  sentence5=CATX(' ',maxcolon,putmax);
  sentence6=CATX(' ',meancolon,putmean);
  sentence7=CATX(' ',qlcolon,putq1);
  sentence8=CATX(' ',mediancolon,putmedian);
  sentence9=CATX(' ',q3colon,putq3);

  /* now we want to string into a paragraph for printing; */

  blurb=CATT(sentence4,sentence5,sentence6,sentence7,sentence8,sentence9);

  label BLURB =" "; /* this is NOT blank - it is a non-printing character */

RUN;

DATA bottomline;

```

```

    LENGTH blurb $ 25000;

    /* now we want to string into a paragraph for printing; */

    blurb='
    LABEL blurb =" "; /* this is NOT blank - it is a non-printing character */
RUN;

DATA print&order;
    LENGTH blurb $ 25000;
    SET header&order
        missval&order
        detail&order
        bottomline;

    LABEL blurb =" "; /* this is NOT blank - it is a non-printing character */
RUN;

%MEND;

%macro detailcat(varname,order);

DATA print&order;
    LENGTH blurb $ 25000;
    SET header&order
        missval&order
        ;

    LABEL blurb =" "; /* this is NOT blank - it is a non-printing character */
RUN;

/* get format to be used from contents db and turn into macro variable */

DATA _null_;
    length formata $ 50;
    SET dd.heart_cb (KEEP=format name WHERE=(name="&varname"));
    formata=STRIP(format);
    CALL SYMPUT('fmt',STRIP(formata));
RUN;

/* strip formats so we can get unformatted values */

DATA temp&order.c;
    SET dd.heart ;
    FORMAT _all_;
    INFORMAT _all_;
    IF &varname NOT IN(.,.D,.S,.R,.Z,.U,.N,.C) THEN counter=1;
RUN;

PROC FREQ DATA=temp&order.c (WHERE=(&varname NOT IN(.,.D,.S,.R,.Z,.U,.N,.C)));
    TABLES &varname / NOPRINT OUT=temp&order.d;
RUN;

PROC MEANS DATA=temp&order.c NOPRINT;
    VAR counter;
    OUTPUT OUT=temp&order.ca N=total;
RUN;

```

```

DATA temp&order.e (KEEP=unfmtval fmtval frequency pct altshade);
  LENGTH unfmtval frequency pct $ 15 fmtval $ 200;
  SET temp&order.d;

  IF MOD(_n_,2)=0 THEN altshade=1;

  unfmtval=PUT(&varname,comma7.);
  fmtval=PUT(&varname,&fmt.);
  frequency=PUT(count,comma9.);
  pct=CATS(PUT(percent,comma7.1),'%');

  LABEL unfmtval='Value'
        fmtval='Label'
        frequency='Frequency'
        pct='%';
RUN;

DATA temp&order.ea (KEEP=unfmtval fmtval frequency pct boldit);
  LENGTH unfmtval frequency pct $ 15 fmtval $ 200;
  SET temp&order.ca;

  unfmtval='Total';
  frequency=put(total,comma9.);
  pct='100%';

  boldit=1;
RUN;

DATA detailtab&order.;
  SET temp&order.e temp&order.ea;
  order=&order;
RUN;

%MEND;

%MACRO detailcharcat(varname,order);

DATA print&order;
  LENGTH blurb $ 25000;
  SET header&order
      missval&order
      ;

  LABEL blurb =" "; /* this is NOT blank - it is a non-printing character */
RUN;

/* First remove any missing values */

DATA temp&order.c;
  SET dd.heart (WHERE=(&varname NOT IN('Logical Skip','Refused','Don't
Know','Question Not Answered',
                                     'Item Non-Response','Not Specified')));
  counter=1;
RUN;

PROC FREQ DATA=temp&order.c ;
  TABLES &varname / NOPRINT out=temp&order.d;

```

```

RUN;

PROC MEANS data=temp&order.c NOPRINT;
  VAR counter;
  OUTPUT OUT=temp&order.ca N=total;
RUN;

DATA temp&order.e (KEEP=fmtval frequency pct altshade);
  LENGTH frequency pct $ 15 fmtval $ 200;
  SET temp&order.d;

  IF MOD(_n_,2)=0 THEN altshade=1;

  fmtval=&varname.;
  frequency=PUT(count,comma9.);
  pct=CATS(PUT(percent,comma7.1),'%');

  LABEL fmtval='Value'
         frequency='Frequency'
         pct='%';
RUN;

DATA temp&order.ea (KEEP=fmtval frequency pct boldit);
  LENGTH frequency pct $ 15 fmtval $ 200;
  SET temp&order.ca;

  fmtval='Total';
  frequency=PUT(total,comma9.);
  pct='100%';

  boldit=1;
RUN;

DATA detailtab&order.;
  SET temp&order.e temp&order.ea;
  order=&order;
RUN;

%MEND;

/* step 6 - write out files to run macros */

DATA _null_;
  FILE out1 LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (KEEP=varnum name vartype);

  include_string=CATS('%header(',name,",",varnum,");");
  PUT include_string;
RUN;

DATA _null_;
  FILE out2 LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (KEEP=varnum name type WHERE=(type NOT IN(2)));

  include_string=CATS('%missval(',name,",",varnum,");");
  PUT include_string;
RUN;

```

```

DATA _null_;
  FILE out2a LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (keep=varnum name type WHERE=(type IN(2)));

  include_string=CATS('%cmissval(',name,"",varnum,"");");
  PUT include_string;
RUN;

data _null_;
  FILE out3 LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (keep=varnum name vartype);

  IF vartype=1 THEN include_string=CATS('%detailcat(',name,"",varnum,"");");
  IF vartype=2 THEN include_string=CATS('%detailcharcat(',name,"",varnum,"");");
  IF vartype=3 THEN include_string=CATS('%detailcont(',name,"",varnum,"");");

  PUT include_string;
RUN;

DATA _null_;
  FILE out4 LRECL=80 PAD;
  LENGTH include_string $ 80;
  SET dd.heart_cb (KEEP=varnum name vartype);

  IF vartype=1 THEN include_string=CATS('%printtable(',varnum,"");");
  IF vartype=2 THEN include_string=CATS('%printtablec(',varnum,"");");
  IF vartype=3 THEN include_string=CATS('%printblurb(',varnum,"");");

  PUT include_string;
RUN;

/* step 8 - run the codebook macros in order (1,2,3) */

%INCLUDE ".\heart_include1.txt";
%INCLUDE ".\heart_include2.txt";
%INCLUDE ".\heart_include2a.txt";
%INCLUDE ".\heart_include3.txt";

/* step 9 - create printing macros */

%MACRO printblurb(order);

ODS TAGSETS.RTF STYLE=styles.noborder;
ODS STARTPAGE=no;

PROC REPORT NOWD DATA=print&order
  STYLE(report)=[cellpadding=3pt vjust=b]
  STYLE(header)=[just=center font_face=Helvetica font_weight=bold font_size=10pt]
  STYLE(lines)=[just=left font_face=Helvetica] ;
COLUMNS blurb ;
DEFINE blurb / style(COLUMN)={just=1 font_face=Helvetica
  font_size=10pt cellwidth=988 }
  style(HEADER)={just=1 font_face=Helvetica
  font_size=10pt };
RUN;

ODS STARTPAGE=no;

%MEND;

```

```

%MACRO printtable(order);

ODS TAGSETS.RTF STYLE=styles.noborder;
ODS STARTPAGE=no;

proc report nowd data=print&order noheader
  STYLE(report)=[cellpadding=3pt vjust=b]
  STYLE(header)=[just=center font_face=Helvetica font_weight=bold font_size=10pt]
  STYLE(lines)=[just=left font_face=Helvetica] ;
  COLUMNS blurb ;
  DEFINE blurb / style(COLUMN)={just=l font_face=Helvetica
    font_size=10pt cellwidth=988 }
    style(HEADER)={just=l font_face=Helvetica
    font_size=10pt };
RUN;

ODS TEXT ='^n ';

ODS TAGSETS.RTF STYLE=styles.minimal;
ODS STARTPAGE=no;

proc report nowd data=detailtab&order /* spacing=8 */
  STYLE(report)=[cellpadding=3pt vjust=b]
  STYLE(header)=[just=center font_face=Helvetica font_weight=bold font_size=10pt
background=graydd
    borderrightcolor=white borderleftcolor=white bordertopcolor=white
borderbottomcolor=white]
  STYLE(lines)=[just=left font_face=Helvetica] split='|';
  COLUMNS unfmtval fmtval frequency pct boldit altshade;
  DEFINE boldit / display ' ' noprint;
  DEFINE altshade / display ' ' noprint;
  DEFINE unfmtval / style(COLUMN)={just=c font_face=Helvetica foreground=black
    font_size=10pt cellwidth=170 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} CENTER
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
  DEFINE fmtval / style(COLUMN)={just=l font_face=Helvetica foreground=black
    font_size=10pt cellwidth=440 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} LEFT
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
  DEFINE frequency / style(COLUMN)={just=c font_face=Helvetica foreground=black
    font_size=10pt cellwidth=170 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} CENTER
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
  DEFINE pct / style(COLUMN)={just=c font_face=Helvetica foreground=black
    font_size=10pt cellwidth=170 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} CENTER
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
COMPUTE boldit;
  IF (boldit EQ 1) THEN CALL DEFINE(_row_,"STYLE","STYLE=[FONT_WEIGHT=BOLD
BACKGROUND=GRAYDD]");
  ENDCOMP;

COMPUTE altshade;
  IF (altshade EQ 1) THEN CALL DEFINE(_row_,"STYLE","STYLE=[BACKGROUND=GRAYEE]");
  ENDCOMP;

RUN;

```

```

ODS STARTPAGE=no;

%MEND;

%macro printtablec(order);

ODS TAGSETS.RTF STYLE=styles.noborder;
ods STARTPAGE=no;

proc report nowd data=print&order noheader
  STYLE(report)=[cellpadding=3pt vjust=b]
  STYLE(header)=[just=center font_face=Helvetica font_weight=bold font_size=10pt]
  STYLE(lines)=[just=left font_face=Helvetica] ;
  COLUMNS blurb ;
  DEFINE blurb / style(COLUMN)={just=1 font_face=Helvetica
    font_size=10pt cellwidth=988 }
    style(HEADER)={just=1 font_face=Helvetica
    font_size=10pt };
RUN;

ODS TEXT='^n ';

ODS TAGSETS.RTF STYLE=styles.minimal;
ods STARTPAGE=no;

proc report nowd data=detailtab&order /* spacing=8 */
  STYLE(report)=[cellpadding=3pt vjust=b]
  STYLE(header)=[just=center font_face=Helvetica font_weight=bold font_size=10pt
background=graydd
  borderrightcolor=white borderleftcolor=white bordertopcolor=white
borderbottomcolor=white]
  STYLE(lines)=[just=left font_face=Helvetica] split='|';
  COLUMNS fmtval frequency pct boldit altshade;
  DEFINE boldit / display ' ' noprint;
  DEFINE altshade / display ' ' noprint;
  DEFINE fmtval / style(COLUMN)={just=1 font_face=Helvetica foreground=black
    font_size=10pt cellwidth=610 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} LEFT
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
  DEFINE frequency / style(COLUMN)={just=c font_face=Helvetica foreground=black
    font_size=10pt cellwidth=170 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} CENTER
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
  DEFINE pct / style(COLUMN)={just=c font_face=Helvetica foreground=black
    font_size=10pt cellwidth=170 borderrightcolor=white borderleftcolor=white
bordertopcolor=white borderbottomcolor=white} CENTER
    style(HEADER)={just=c font_face=Helvetica font_weight=bold
    font_size=10pt };
  COMPUTE boldit;
  IF (boldit EQ 1) THEN CALL DEFINE(_row_,"STYLE","STYLE=[FONT_WEIGHT=BOLD
BACKGROUND=GRAYDD]");
  ENDCOMP;

  COMPUTE altshade;
  IF (altshade EQ 1) THEN CALL DEFINE(_row_,"STYLE","STYLE=[BACKGROUND=GRAYEE]");
  ENDCOMP;

RUN;

```

```
ODS STARTPAGE=no;

%MEND;

/* step 10 - run the printing macro INSIDE the TAGSETS RTF call */

ODS LISTING CLOSE;
TITLE1;
TITLE2;
FOOTNOTE1;

OPTIONS TOPMARGIN=1in LEFTMARGIN=1in RIGHTMARGIN=1in BOTTOMMARGIN=1in
PAPERSIZE=letter;

ODS TAGSETS.RTF FILE="heart_codebook.rtf" OPTIONS(vspace='no') PATH=odsout
STYLE=styles.noborder;

ODS TEXT="Heart Data File Codebook";

ODS ESCAPECHAR='^';

%INCLUDE ".\heart_include4.txt";

ODS TAGSETS.RTF CLOSE;

FILENAME msg EMAIL TO="9990000000@txt.att.net"
FROM = "Big Nerd <youremail@yourcompany.com>"
SUBJECT="All Systems Go (or not)?";

DATA _null_;
FILE msg;
PUT "Program Name is %SYSFUNC(GETOPTION(SYSIN))";
PUT "Run &SYSDATE - &SYSTIME - by &SYSUSERID in &SYSPROCESSMODE";
PUT "Condition Code is &SYSCC.";
RUN;

ENDSAS;
```

```

*Program 4: 4gen_label_fmt_stmnt_SESUG2016_CC145.sas;

OPTIONS PS=55 LS=175 ERRORABEND NOFMterr ERRORS=1 MPRINT SYMBOLGEN NOCENTER NODATE
NONUMBER
;

TITLE1 "SESUG 2016 CC145";
FOOTNOTE1 "%SYSFUNC(GETOPTION(SYSIN)) - &SYSDATE - &SYSTIME - run by &SYSUSERID in
&SYSPROCESSMODE";
RUN;

TITLE2 "Heart";
RUN;

LIBNAME dd '.';

FILENAME odsout ".";
FILENAME out1 ".\heart_labelstm.txt"; /* for labels */
FILENAME out2 ".\heart_formatstm.txt"; /* for format assignment */

LIBNAME library '.';
RUN;

/* set up some mini data sets with top and bottom lines */

DATA runlabel;
    LENGTH include_string $ 180;
    include_string="*** Label Statement for heart;";
    OUTPUT;
    include_string='LABEL';
    OUTPUT;
RUN;

DATA runformat;
    LENGTH include_string $ 180;
    include_string="*** Format Assignments for heart;";
    OUTPUT;
    include_string='FORMAT';
    OUTPUT;
RUN;

DATA runrun;
    LENGTH include_string $ 180;
    include_string='';
RUN;

DATA temp1;
    LENGTH include_string $ 180;
    SET dd.heart_cb;

    label=COMPRESS(label, '');

    qlabel=CATS(' ', label, '');
    include_string=CATX(' ', name, '=', qlabel);
RUN;

DATA templabel (KEEP=include_string);

```

```
FILE out1 LRECL=180 PAD;
LENGTH include_string $ 180;
SET runlabel temp1 runrun;
PUT include_string;
RUN;

DATA temp2;
LENGTH include_string $ 180;
SET dd.heart_cb (WHERE=(format NE ''));
qformat=CATS(format, '.');
include_string=CATX(' ', name, qformat);
RUN;

DATA tempfmt (KEEP=include_string);
FILE out2 LRECL=180 PAD;
LENGTH include_string $ 180;
SET runformat temp2 runrun;
PUT include_string;
RUN;

ENDSAS;
```