```
************************************************************************************************;
**  Program Name    :  admh.sas                                                                  **;
**  Date Created    :  17Nov2021                                                                 **;
**  Programmer Name :  (b) (4), (b) (6)                                                          **;
**  Purpose         :  Create admh dataset                                                       **;
**  Input data      :  mh suppmh adsl                                                            **;
**  Output data     :  admh.sas7bdat                                                             **;
************************************************************************************************;
options mprint mlogic symbolgen mprint symbolgen mlogic nocenter missing=" ";

proc datasets library=WORK kill nolist nodetails;
quit;

**Setup the environment**;
%let
oprot=/Volumes/app/cdars/prod/sites/cdars4/prjC459/nda2_unblinded_esub/sbla1215_esub_sdtm/saseng/cdisc3_0/data/
sdtm;
%let
protori=/Volumes/app/cdars/prod/sites/cdars4/prjC459/nda2_unblinded_esub/sbla1215_esub_adam/saseng/cdisc3_0;
%let
prot=/Volumes/app/cdars/prod/sites/cdars4/prjC459/nda2_unblinded_esub/sbla1215_esub_adam/saseng/cdisc3_0/analy
sis/eSUB;
*Path for external files;
%let
expath=/Volumes/app/cdars/prod/sites/cdars4/prjC459/nda2_unblinded_esub/sbla1215_esub_adam/saseng/cdisc3_0/out
put/xpt;

libname dataprot "&oprot." access=readonly;
libname datvprot "&protori./data_vai" access=readonly;
libname datvout "&prot./data_vai";
libname viewpx "/Volumes/app/saseng/prod/cdisc3_0/view/" access=readonly;


proc printto print="&prot./output/admh.rpt"
      log="&prot./logs/admh.log" new;
run;

************************************************************************************************;
* Specification 1 *;
* Merge MH dataset with SUPPMH dataset. *;
************************************************************************************************;

proc sql noprint;
   select distinct QNAM into :_suppmh_keep_vars separated by " " from
      dataprot.suppmh;
quit;

************************************************************************************************;
*Specification 1: Reading INPUT SDTM and Supplemental Datasets *;
*Subsetting Supplemental Dataset based on _supp_subset parameter*;
************************************************************************************************;

data _spmdel_supp_dsin_subset;
```

```
   set dataprot.suppmh;
   where;
run;

data _spmdel_sdtm_ds;
   set dataprot.mh;
run;

*************************************************************************;
*Specification 2: Supplemental Dataset will be merged with SDTM for all values*;
*of IDVAR including missing values. *;
* a. Find whether IDVAR has a missing a value *;
* b. Calculate number of non-missing values for IDVAR *;
* c. Checking whether non-missing value of IDVAR is character or Numeric *;
*************************************************************************;

proc sql noprint;
   select NMISS(distinct idvar) into :_cntvar from _spmdel_supp_dsin_subset;
   select N(distinct idvar) into :_cntvar1 from _spmdel_supp_dsin_subset;
quit;

proc sql noprint;
   select distinct idvar into :_idvar1 - :_idvar1 from _spmdel_supp_dsin_subset
      where idvar is not missing;
quit;

data _spmdel_supp_dsin_subset_idvar1;
   set _spmdel_supp_dsin_subset;
   where idvar="MHSEQ";
run;

*********************************************************;
*Specification 3:Tranposing Supplemental Dataset *;
*********************************************************;

proc sort data=_spmdel_supp_dsin_subset_idvar1;
   by studyid usubjid idvar idvarval;
   quit;

proc transpose data=_spmdel_supp_dsin_subset_idvar1
      out=_spmdel_supp_dsin_idvar1_h;
   by studyid usubjid idvar idvarval;
   id qnam;
   idlabel qlabel;
   var qval;
   quit;
   *********************************************************;
   *Specification 4:Creating IDVAR from IDVARVAL *;
   *********************************************************;

data _spmdel_temp(keep=MHSEQ);
   set _spmdel_sdtm_ds;
run;
```

```
data _spmdel_suppds1 (drop=idvar idvarval _NAME_ _LABEL_);
   set _spmdel_supp_dsin_idvar1_h;

   if idvar="MHSEQ";
   MHSEQ=input(idvarval, best12.);
run;

*******************************************************************************;
*Specification 5: If IDVAR is character or Numeric then Merge condition is *;
*STUDYID USUBJID IDVAR else when IDVAR is missing the Merge condition will be*;
*STUDYID USUBJID *;
*******************************************************************************;
*****************************************************************;
*Specification 6: Merging SDTM and intermediate Supplemental Datasets*;
*****************************************************************;
*******************************************************************************;
* SPECIFICATION 1 - Check the existence of keep or drop variables in the *;
* keep or drop dataset option and quit if necessary. *;
* 1. Call util_parm_valid to check for the valid values of macro variable *;
* _join. Valid values are F R L I. Default value is F for full join. *;
* 2. Call util_var_exist. *;
* 3. If any of the keep or drop variables specified in the keep or drop *;
* dataset option do not exist then display an error message in the log *;
* and stop further execution of this macro. *;
*******************************************************************************;
;
*******************************************************************************;
* SPECIFICATION 2 - Merge the datasets. *;
* 1.Sort the datasets by key merge-by variables. *;
* 2.Merge the datasets using full join, inner join,left join or right join. *;
* Default is full join. *;
* 3.Generate a list of subjects not having any matching observations based on *;
* key-by variables. *;
*******************************************************************************;

proc sort data=_spmdel_sdtm_ds out=_ds1;
   by STUDYID USUBJID MHSEQ;
run;

proc sort data=_spmdel_suppds1 out=_ds2;
   by STUDYID USUBJID MHSEQ;
run;

data _spmdel_sdtm_temp_out1;
   merge _ds1(in=d1) _ds2(in=d2);
   by STUDYID USUBJID MHSEQ;

   if d1;
run;

;
*****************************************************************;
*Specification 7: Final Merged output dataset *;
*****************************************************************;
```

```
data _mh;
   set _spmdel_sdtm_temp_out1;
run;

*******************************************************************;
*Specification 8: Deleting all temporary local datasets *;
*******************************************************************;

proc datasets library=work;
   delete _spmdel:;
quit;

;
*****************************************************************************************;
* Specification 2 *;
* Check for required variables in MH dataset. If missing, then bail out. *;
*****************************************************************************************;
*****************************************************************************************;
* Specification 1 *;
* Call util_chkvars to check for the existence of required and expected variables. *;
* Req: STUDYID USUBJID MHSEQ MHTERM MHCAT DICTVER *;
* Exp: MHBDSYCD MHSOC *;
*****************************************************************************************;
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;
*****************************************************************************************;
* Specification 2 *;
* If MHCAT or MHTERM have missing values, then drop record and display *;
* the dropped record in a supplemental listing. *;
*****************************************************************************************;

data _mh _droprecs;
   set _mh;

   if MHCAT eq ' ' or MHTERM eq ' ' then
      do;
         output _droprecs;
      end;
   else
      do;
         output _mh;
      end;
run;

proc sort data=_droprecs out=_droprecs(keep=STUDYID USUBJID MHSEQ MHTERM MHCAT
      DICTVER MHBDSYCD MHSOC);
   by usubjid;
run;

*****************************************************************************************;
* Specification 3 *;
* Abort the program if required and expected variables do not exist. *;
```

```
*******************************************************************************************;
;
*******************************************************************************************;
* Specification 3 *;
* Check for the existence of optional and conditional variables before using them. *;
*******************************************************************************************;
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;
*******************************************************************************************;
* Specification 4 *;
* Impute partial date from G_ADMH_DIAGNOSIS_DATE (if exists) into ASTDT with flag ASTDTF.*;
* Impute partial date from MHENDTC (if exists) into AENDT with flag AENDTF. *;
* Impute partial date from MHDTC (if exists) into ADT with flag ADTF. *;
*******************************************************************************************;

data _mh;
   set _mh;
   *******************************************************************************************;
   * SPECIFICATION 1 *;
   * Call parsem macro for each variable in macro call to split the supplied values into *;
   * separate words by / delimiter. *;
   *******************************************************************************************;
   ;
   ;
   ;
   ;
   *******************************************************************************************;
   * SPECIFICATION 2 *;
   * -Process Date (if requested in macro call) *;
   * Substring isodate into year, month and day. *;
   * Based upon _imputation_rule_date (start or stop), impute missing values in date. *;
   * If _impdateflag is supplied in macro call, then the highest date imputed will *;
   * be populated. *;
   * If only year exists, then month and day will be imputed and flag will contain 'M'. *;
   * if year and month exist, then day will be imputed and flag will contain 'D'. *;
   *******************************************************************************************;

   if ^missing(MHSTDTC) then
      do;
         length yr $4 mm dd $2;
         yr=substr(MHSTDTC, 1, 4);
         mm=substr(MHSTDTC, 6, 2);
         dd=substr(MHSTDTC, 9, 2);
         ;

         if yr ne ' ' then
            do;
               dflag=' ';

               if (dd eq "  " or dd eq "-T") and mm ne " " then
                  do;
                     dd='01';
                     dflag='D';
```

```
            end;

          if mm eq "  " or mm eq "--" then
            do;
              mm='01';
              dd='01';
              dflag='M';
            end;
          newdate=(trim(left(yr))||'-'||trim(left(mm))||'-'||trim(left(dd)));
          ASTDT=input(newdate, ??is8601da.);
          format ASTDT date9.;
          ASTDTF=dflag;
        end;
      drop yr mm dd dflag newdate;
  end;
*******************************************************************************;
* SPECIFICATION 3 *;
* -Process Time (if requested in macro call) *;
* Substring isodate into hour, minutes and seconds. *;
* Based upon _imputation_rule_time (start or stop), impute missing values in time. *;
* If _imptimeflag is supplied in macro call, then the highest time imputed will *;
* be populated. *;
* If no time exists,then hour, minutes and seconds are imputed and flag will contain 'H'*;
* if only hour exists,then minutes and seconds will be imputed and flag will contain 'M'*;
* if hour and minutes exist, then seconds will be imputed and flag will contain 'S' *;
*******************************************************************************;
;
*******************************************************************************;
* SPECIFICATION 1 *;
* Call parsem macro for each variable in macro call to split the supplied values into *;
* separate words by / delimiter. *;
*******************************************************************************;
;
;
;
;
*******************************************************************************;
* SPECIFICATION 2 *;
* -Process Date (if requested in macro call) *;
* Substring isodate into year, month and day. *;
* Based upon _imputation_rule_date (start or stop), impute missing values in date. *;
* If _impdateflag is supplied in macro call, then the highest date imputed will *;
* be populated. *;
* If only year exists, then month and day will be imputed and flag will contain 'M'. *;
* if year and month exist, then day will be imputed and flag will contain 'D'. *;
*******************************************************************************;

if ^missing(MHENDTC) then
  do;
    length yr $4 mm dd $2;
    yr=substr(MHENDTC, 1, 4);
    mm=substr(MHENDTC, 6, 2);
    dd=substr(MHENDTC, 9, 2);
    ;
```

```
      if yr ne ' ' then
         do;
            dflag=' ';

            if (dd eq "  " or dd eq "-T") and mm ne "  " then
               do;
                  fakedate=input(((trim(left(yr))||'-'||trim(left(mm))||'-'||'01')),
                     ??is8601da.);
                  format fakedate date9.;
                  tempdate=intnx('month', fakedate, 1)-1;
                  dd=strip(put(day(tempdate), best.));
                  dflag='D';
               end;

            if (dd eq "  " or dd eq "-T") and mm eq "  " or mm eq "--" then
               do;
                  mm='12';
                  dd='31';
                  dflag='M';
               end;
            newdate=(trim(left(yr))||'-'||trim(left(mm))||'-'||trim(left(dd)));
            AENDT=input(newdate, ??is8601da.);
            format AENDT date9.;
            AENDTF=dflag;
            drop fakedate tempdate;
         end;
      drop yr mm dd dflag newdate;
   end;
*******************************************************************************************;
* SPECIFICATION 3 *;
* -Process Time (if requested in macro call) *;
* Substring isodate into hour, minutes and seconds. *;
* Based upon _imputation_rule_time (start or stop), impute missing values in time. *;
* If _imptimeflag is supplied in macro call, then the highest time imputed will *;
* be populated. *;
* If no time exists,then hour, minutes and seconds are imputed and flag will contain 'H'*;
* if only hour exists,then minutes and seconds will be imputed and flag will contain 'M'*;
* if hour and minutes exist, then seconds will be imputed and flag will contain 'S' *;
*******************************************************************************************;
;
*******************************************************************************************;
* SPECIFICATION 1 *;
* Call parsem macro for each variable in macro call to split the supplied values into *;
* separate words by / delimiter. *;
*******************************************************************************************;
;
;
;
;
*******************************************************************************************;
* SPECIFICATION 2 *;
* -Process Date (if requested in macro call) *;
* Substring isodate into year, month and day. *;
```

```
   * Based upon _imputation_rule_date (start or stop), impute missing values in date. *;
   * If _impdateflag is supplied in macro call, then the highest date imputed will *;
   * be populated. *;
   * If only year exists, then month and day will be imputed and flag will contain 'M'. *;
   * if year and month exist, then day will be imputed and flag will contain 'D'. *;
   ********************************************************************************************;

   if ^missing(MHDTC) then
      do;
         length yr $4 mm dd $2;
         yr=substr(MHDTC, 1, 4);
         mm=substr(MHDTC, 6, 2);
         dd=substr(MHDTC, 9, 2);
         ;

         if yr ne ' ' then
            do;
               dflag=' ';

               if (dd eq "  " or dd eq "-T") and mm ne " " then
                  do;
                     dd='01';
                     dflag='D';
                  end;

               if mm eq "  " or mm eq "--" then
                  do;
                     mm='01';
                     dd='01';
                     dflag='M';
                  end;
               newdate=(trim(left(yr))||'-'||trim(left(mm))||'-'||trim(left(dd)));
               ADT=input(newdate, ??is8601da.);
               format ADT date9.;
               ADTF=dflag;
            end;
         drop yr mm dd dflag newdate;
      end;
   ********************************************************************************************;
   * SPECIFICATION 3 *;
   * -Process Time (if requested in macro call) *;
   * Substring isodate into hour, minutes and seconds. *;
   * Based upon _imputation_rule_time (start or stop), impute missing values in time. *;
   * If _imptimeflag is supplied in macro call, then the highest time imputed will *;
   * be populated. *;
   * If no time exists,then hour, minutes and seconds are imputed and flag will contain 'H'*;
   * if only hour exists,then minutes and seconds will be imputed and flag will contain 'M'*;
   * if hour and minutes exist, then seconds will be imputed and flag will contain 'S' *;
   ********************************************************************************************;
   ;
run;

********************************************************************************************;
* SPECIFICATION 1: *;
```

```
* Protocol Specific Imputation Rules. *;
*******************************************************************************;

data _mh;
   set _mh;
run;

;
*******************************************************************************;
* Specification 5 *;
* Join ADSL variables(identified in ADMH.txt as G_ADSL_VARS) onto MH dataset. *;
* Check the existence of reference date variable. *;
*******************************************************************************;
*******************************************************************************;
* Specification 1 *;
* Define local and global macro variables. *;
*******************************************************************************;
*******************************************************************************;
* Specification 2 *;
* Call utility util_num_periods to get number of periods in ADSL *;
* Parse G_ADSL_VARS to get all variables with Period XX *;
* Replace Period XX with actual period number *;
* Parse APXX variables if not part of G_ADSL_VARS and add them to process. Purpose is to *;
* handle them separatly if user does not intend to keep them in final ADAM. *;
*******************************************************************************;
Options NoMprint NoMlogic MLOGIC;
;
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;
*******************************************************************************;
*Specification 3 *;
*Create a list of all Treatment Time related column names from user input of G_ADSL_VARS *;
*Create a Master List of all Treatment Time related column names from datvprot.adsl *;
*Compare User list with Master List *;
*Bailout if any of the columns of Master List are not specified by USER *;
*******************************************************************************;

data _tmpcol2(keep=_usrlst);
   length _usrlst $20;
   drop _string;
   _string="SUBJID SITEID AGE AGEU AGEGR1 AGEGR1N SEX SEXN RACE RACEN ARACE ARACEN
RANDFL SAFFL COMPLFL ARM ARMCD ACTARM ACTARMCD TRTSDT TRTSTM TRTSDTM TRTEDT
TRTETM TRTEDTM TRT01A TRT01AN TRT02A TRT02AN TRT01P TRT01PN TRT02P TRT02PN TR01SDT
TR01STM TR01SDTM TR01EDT TR01ETM TR01EDTM TR02SDT TR02STM TR02SDTM TR02EDT TR02ETM
TR02EDTM VAX101 VAX102 VAX10U VAX201 VAX202 VAX20U VAX101DT VAX102DT VAX10UDT
VAX201DT VAX202DT VAX20UDT UNBLNDDT RANDDT COHORT COHORTN DOSALVL DOSALVLN
DOSPLVL DOSPLVLN DS30KFL PHASE PHASEN AGEGR4 AGEGR4N HIVFL PEDIMMFL PEDREAFL
DS3KFL AGETR01 AGETRU01 AP01SDT AP01STM AP01SDTM AP01EDT AP01ETM AP01EDTM AP02SDT
AP02STM AP02SDTM AP02EDT AP02ETM AP02EDTM";

   do until(_usrlst=' ');
      _count+1;
      _usrlst=scan(_string, _count);
```

```
      output;
   end;
run;

proc sql noprint;
   create table _tmpcol4 as select distinct upcase(_usrlst) as _usrlst from
      _tmpcol2 where upcase(_usrlst) like 'TR%' or upcase(_usrlst) like 'AP%';
quit;

proc contents data=datvprot.adsl out=_tmpcol(keep=NAME) noprint;
proc sql noprint;
   select distinct upcase(NAME) into : _masterlist separated by " " from _tmpcol
      where (upcase(name) like 'TR%' or upcase(name) like 'AP%')
      and (prxmatch('/TR\d{2}STM\b/', Name) or prxmatch('/TR\d{2}ETM\b/', Name) or
      prxmatch('/TR\d{2}SDT\b/', Name) or prxmatch('/TR\d{2}EDT\b/', Name) or
      upcase(Name) in ('TRTEDT', 'TRTSDT', 'TRTETM', 'TRTSTM') or
      prxmatch('/AP\d{2}[EDT\b|SDT\b]/', Name) or
      prxmatch('/AP\d{2}[STM\b|ETM\b]/', Name) or
      prxmatch('/AP\d{2}[SDTM\b|EDTM\b]/', Name) );
quit;

data _tmpcol3(keep=_mstrlst);
   length _mstrlst $20;
   drop _string;
   _string="TR01EDT TR01ETM TR01SDT TR01STM TR02EDT TR02ETM TR02SDT TR02STM TRTEDT
TRTETM TRTSDT TRTSTM";

   do until(_mstrlst=' ');
      _count+1;
      _mstrlst=scan(_string, _count);
      output;
   end;
run;

proc sql noprint;
   select distinct(a._mstrlst) into :_usrlst_missing separated by ' ' from
      _tmpcol3 as a where a._mstrlst not in (select _usrlst from _tmpcol4);
quit;

*******************************************************************************;
* Specification 4 *;
* Call util_chkvars to check for existence of variables defined in G_ADSL_VARS *;
*******************************************************************************;
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;
*******************************************************************************;
* Specification 5 *;
* Call gen_merge_ds to merge the input dataset with adsl dataset by usubjid *;
*******************************************************************************;
*******************************************************************************;
```

```
* SPECIFICATION 1 - Check the existence of keep or drop variables in the *;
* keep or drop dataset option and quit if necessary. *;
* 1. Call util_parm_valid to check for the valid values of macro variable *;
* _join. Valid values are F R L I. Default value is F for full join. *;
* 2. Call util_var_exist. *;
* 3. If any of the keep or drop variables specified in the keep or drop *;
* dataset option do not exist then display an error message in the log *;
* and stop further execution of this macro. *;
*********************************************************************************;
;
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT MLOGIC SYMBOLGEN;
option nonotes;

proc sql noprint;
   create table _list_ (name char(32));
   insert into _list_ values("G_VEXIST") values("");
   select name into:G_NOMATCH separated by ' ' from _list_ where name not
      in (select name from dictionary.macros);
   drop table _list_;
quit;

option NOTES;
options MPRINT SYMBOLGEN MLOGIC;
;

data _null_;
   length _retlist _retlst2 $2000 _column $40;
   dsid=open(upcase("DATVPROT.ADSL"));
   i=1;

   do while (scan("USUBJID SUBJID SITEID AGE AGEU AGEGR1 AGEGR1N SEX SEXN RACE RACEN
ARACE ARACEN RANDFL SAFFL ARM ARMCD ACTARM ACTARMCD TRTSDT TRTSTM TRTSDTM
TRTEDT TRTETM TRTEDTM TRT01A TRT01AN TRT02A TRT02AN TRT01P TRT01PN TRT02P TRT02PN
TR01SDT TR01STM TR01SDTM TR01EDT TR01ETM TR01EDTM TR02SDT TR02STM TR02SDTM TR02EDT
TR02ETM TR02EDTM VAX101 VAX102 VAX10U VAX201 VAX202 VAX20U VAX101DT VAX102DT
VAX10UDT VAX201DT VAX202DT VAX20UDT UNBLNDDT RANDDT COHORT COHORTN DOSALVL
DOSALVLN DOSPLVL DOSPLVLN DS30KFL PHASE PHASEN AGEGR4 AGEGR4N HIVFL PEDIMMFL
PEDREAFL DS3KFL AGETR01 AGETRU01 TR01SDT TR01STM TR01SDTM TR01EDT TR01ETM TR01EDTM
TR02SDT TR02STM TR02SDTM TR02EDT TR02ETM TR02EDTM",
         i, ") > ");
      _column=upcase(scan("USUBJID SUBJID SITEID AGE AGEU AGEGR1 AGEGR1N SEX SEXN RACE
RACEN ARACE ARACEN RANDFL SAFFL ARM ARMCD ACTARM ACTARMCD TRTSDT TRTSTM
TRTSDTM TRTEDT TRTETM TRTEDTM TRT01A TRT01AN TRT02A TRT02AN TRT01P TRT01PN TRT02P
TRT02PN TR01SDT TR01STM TR01SDTM TR01EDT TR01ETM TR01EDTM TR02SDT TR02STM TR02SDTM
TR02EDT TR02ETM TR02EDTM VAX101 VAX102 VAX10U VAX201 VAX202 VAX20U VAX101DT
VAX102DT VAX10UDT VAX201DT VAX202DT VAX20UDT UNBLNDDT RANDDT COHORT COHORTN
DOSALVL DOSALVLN DOSPLVL DOSPLVLN DS30KFL PHASE PHASEN AGEGR4 AGEGR4N HIVFL
PEDIMMFL PEDREAFL DS3KFL AGETR01 AGETRU01 TR01SDT TR01STM TR01SDTM TR01EDT TR01ETM
TR01EDTM TR02SDT TR02STM TR02SDTM TR02EDT TR02ETM TR02EDTM",
         i, "));

      if varnum(dsid, _column) then
         do;
```

```
            _retlist=trim(left(_retlist))||' '||_column;
            _retlst2=trim(left(_retlst2))||'/'||_column;
          end;
        i=i+1;
    end;
    dsid=close(dsid);
    call symput('g_vexist', trim(left(compbl(_retlist))));
    call symput('g_vexist2', trim(left(_retlst2))||'/');
run;

options MPRINT SYMBOLGEN MLOGIC;
;
*******************************************************************************;
* SPECIFICATION 2 - Merge the datasets. *;
* 1.Sort the datasets by key merge-by variables. *;
* 2.Merge the datasets using full join, inner join,left join or right join. *;
* Default is full join. *;
* 3.Generate a list of subjects not having any matching observations based on *;
* key-by variables. *;
*******************************************************************************;

proc sort data=_mh out=_ds1;
    by USUBJID;
run;

proc sort data=datvprot.adsl out=_ds2;
    by USUBJID;
run;

data _mh;
    merge _ds1(in=d1) _ds2(in=d2 keep=Usubjid SUBJID SITEID AGE AGEU AGEGR1
        AGEGR1N SEX SEXN RACE RACEN ARACE ARACEN RANDFL SAFFL ARM ARMCD ACTARM
        ACTARMCD TRTSDT TRTSTM TRTSDTM TRTEDT TRTETM TRTEDTM TRT01A TRT01AN TRT02A
        TRT02AN TRT01P TRT01PN TRT02P TRT02PN TR01SDT TR01STM TR01SDTM TR01EDT
        TR01ETM TR01EDTM TR02SDT TR02STM TR02SDTM TR02EDT TR02ETM TR02EDTM VAX101
        VAX102 VAX10U VAX201 VAX202 VAX20U VAX101DT VAX102DT VAX10UDT VAX201DT
        VAX202DT VAX20UDT UNBLNDDT RANDDT COHORT COHORTN DOSALVL DOSALVLN DOSPLVL
        DOSPLVLN DS30KFL PHASE PHASEN AGEGR4 AGEGR4N HIVFL PEDIMMFL PEDREAFL DS3KFL
        AGETR01 AGETRU01 TR01SDT TR01STM TR01SDTM TR01EDT TR01ETM TR01EDTM TR02SDT
        TR02STM TR02SDTM TR02EDT TR02ETM TR02EDTM);
    by USUBJID;

    if d1;
run;

;
;

data _mh;
    merge _mh (in=a) datvprot.adsl (in=b keep=usubjid);
    by usubjid;

    if a and b;
run;
```

```
options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;
*******************************************************************************************;
* Specification 6 *;
* Calculate Study Day *;
* If MHCAT is Primary Diagnosis or Secondary Diagnosis then derive ADURN *;
* sg4: add GENERAL MEDICAL HISTORY for covid study *;
*******************************************************************************************;

data _mh;
   length ADURU $10.;
   set _mh;
   ********************************************************************************************;
   * Specification 1 *;
   * Calculate the study day using the dates supplied by the user. *;
   ********************************************************************************************;

   If ^Missing(ASTDT) and ^Missing(TRTSDT) then
      do;

         If ASTDT lt TRTSDT then
            ASTDY=ASTDT - TRTSDT;
         Else If ASTDT ge TRTSDT then
            ASTDY=ASTDT - TRTSDT + 1;
      end;
   ;
   ********************************************************************************************;
   * Specification 1 *;
   * Calculate the study day using the dates supplied by the user. *;
   ********************************************************************************************;

   If ^Missing(AENDT) and ^Missing(TRTSDT) then
      do;

         If AENDT lt TRTSDT then
            AENDY=AENDT - TRTSDT;
         Else If AENDT ge TRTSDT then
            AENDY=AENDT - TRTSDT + 1;
      end;
   ;

   if upcase(MHCAT) in ('PRIMARY DIAGNOSIS', 'SECONDARY DIAGNOSIS',
      'GENERAL MEDICAL HISTORY') then
         do;
         _utilflg='N';

         if ((_utilflg eq 'Y') and (MHDUR ne ' ')) then
            do;

*******************************************************************************************;
               * SPECIFICATION 1 *;
               * -Call SAS Function 'call is8601_convert' to convert _isodur duration variable to a SAS*;
```

```
            * time variable. *;
            * -By using conversion factor: *;
            * -if _conv_to_unit = DAYS, then convert default hours to number of days *;
            * -if _conv_to_unit = WEEKS, then convert default hours to number of weeks *;
            * -if _conv_to_unit = MONTHS, then convert default hours to number of months *;
            * -if _conv_to_unit = YEARS, then convert default hours to number of years *;

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

            if MHDUR ne ' ' then
               do;
                  call is8601_convert('du', 'du', MHDUR, dur1);
                  ;
                  ;
                  ADURN=round((dur1/(24*60*60*365.25)), 0.01);
                  drop dur1;
               end;
            else
               do;
                  ADURN=.;
               end;
            ;

            if ADURN ne . then
               do;
                  ADURU="YEARS";
               end;
         end;
      else
         do;
            ADURN=round(((ADT - ASTDT + 1)/365.25), 0.01);

            if ADURN ne . then
               do;
                  ADURU="YEARS";
               end;
         end;
      drop _utilflg;
   end;
run;

*******************************************************************************************;
* Specification 7 *;
* Attach attributes to all variables as per ADaM Spec *;
*******************************************************************************************;

proc import file="&expath./comorbidity-categories.xlsx" out=_comobi dbms=xlsx
      replace;
   RXLX;
   getnames=yes;
run;

proc import file="&expath./report-cci-aids-hiv-30oct2020.xlsx" out=_hiv
      dbms=xlsx replace;
```

```
    RXLX;
    datarow=17;
    getnames=no;
run;

data _hiv;
    set _hiv;
    CAT='AIDS/HIV';
run;

proc import file="&expath./report-cci-any-malignancy-06aug2021.xlsx" out=_mali
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _mali;
    set _mali;
    CAT='Any malignancy';
run;

proc import file="&expath./report-cci-cerebrovascular-30oct2020.xlsx" out=_cere
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _cere;
    set _cere;
    CAT='Cerebrovascular disease';
run;

proc import file="&expath./report-cci-chf-30oct2020.xlsx" out=_chf dbms=xlsx
        replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _chf;
    set _chf;
    CAT='Congestive heart failure';
run;

proc import file="&expath./report-cci-dementia-30oct2020.xlsx" out=_deme
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _deme;
```

```
    set _deme;
    CAT='Dementia';
run;

proc import file="&expath./report-cci-diabetes-with-comp-06aug2021.xlsx"
        out=_diabe_c dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _diabe_c;
    set _diabe_c;
    CAT='Diabetes with chronic complication';
run;

proc import file="&expath./report-cci-diabetes-without-comp-30oct2020.xlsx"
        out=_diabe dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _diabe;
    set _diabe;
    CAT='Diabetes without chronic complication';
run;

proc import file="&expath./report-cci-hemiplegia-30oct2020.xlsx" out=_hemip
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _hemip;
    set _hemip;
    CAT='Hemiplegia or paraplegia';
run;

proc import file="&expath./report-cci-leukemia-30oct2020.xlsx" out=_leuk
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _leuk;
    set _leuk;
    CAT='Leukemia';
run;

proc import file="&expath./report-cci-lymphoma-30oct2020.xlsx" out=_lymph
        dbms=xlsx replace;
```

```
    RXLX;
    datarow=17;
    getnames=no;
run;

data _lymph;
    set _lymph;
    CAT='Lymphoma';
run;

proc import file="&expath./report-cci-metastatic-tumour-30oct2020.xlsx"
        out=_metas dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _metas;
    set _metas;
    CAT='Metastatic solid tumor';
run;

proc import file="&expath./report-cci-mi-30oct2020.xlsx" out=_mi dbms=xlsx
        replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _mi;
    set _mi;
    CAT='Myocardial infarction';
run;

proc import file="&expath./report-cci-mild-liver-30oct2020.xlsx" out=_mild
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _mild;
    set _mild;
    CAT='Mild liver disease';
run;

proc import file="&expath./report-cci-mod-sev-liver-06aug2021.xlsx" out=_modsev
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _modsev;
```

```
    set _modsev;
    CAT='Moderate or severe liver disease';
run;

proc import file="&expath./report-cci-peptic-ulcer-30oct2020.xlsx" out=_peptic
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _peptic;
    set _peptic;
    CAT='Peptic ulcer disease';
run;

proc import file="&expath./report-cci-periph-vasc-30oct2020.xlsx" out=_peri
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _peri;
    set _peri;
    CAT='Peripheral vascular disease';
run;

proc import file="&expath./report-cci-pulmonary-30oct2020.xlsx" out=_pulm
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _pulm;
    set _pulm;
    CAT='Chronic pulmonary disease';
run;

proc import file="&expath./report-cci-renal-30oct2020.xlsx" out=_renal
        dbms=xlsx replace;
    RXLX;
    datarow=17;
    getnames=no;
run;

data _renal;
    set _renal;
    CAT='Renal disease';
run;

proc import file="&expath./report-cci-rheumatic-06aug2021.xlsx" out=_rheuma
        dbms=xlsx replace;
```

```
    RXLX;
    datarow=17;
    getnames=no;
run;

data _rheuma;
    set _rheuma;
    CAT='Rheumatic disease';
run;

data _pt;
    length a b $ 200 c CAT $ 100;
    set _hiv _mali _cere _chf _deme _diabe_c _diabe _hemip _leuk _lymph _metas _mi
        _mild _modsev _peptic _peri _pulm _renal _rheuma;
    drop d e f;
run;

data report_cci;
    set _pt;
    mhptcd=input(b, best.);
    rename a=term;

proc sort;
    by mhptcd;
run;

proc transpose data=report_cci out=t_cci prefix=CAT;
    by mhptcd term;
    var CAT;
run;

proc sort data=_mh;
    by mhptcd;
run;

data _mh;
    merge _mh (in=a) t_cci(in=b);
    by mhptcd;
    if agegr4n=1;
    if a & b then
        COMORBFL='Y';
    else
        COMORBFL='N';

    if a;
    drop _NAME_ term;
run;

options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;

data admh;
    retain STUDYID USUBJID SUBJID SITEID MHSEQ MHTERM MHDECOD MHPTCD MHBODSYS
```

```
      MHBDSYCD MHLLT MHLLTCD MHPTCD MHHLT MHHLTCD MHHLGT MHHLGTCD MHSOC
MHSOCCD
      MHCAT MHSTDTC MHENDTC MHENRTPT MHENTPT DICTVER MHSPID ASTDT ASTDTF ASTDY
      AENDT AENDTF AENDY COMORBFL CAT1 CAT2 ADT ADTF ADURN ADURU DICTVER USUBJID
      SUBJID SITEID AGE AGEU AGEGR1 AGEGR1N SEX SEXN RACE RACEN ARACE ARACEN RANDFL
      SAFFL ARM ARMCD ACTARM ACTARMCD TRTSDT TRTSTM TRTSDTM TRTEDT TRTETM
TRTEDTM
      TRT01A TRT01AN TRT02A TRT02AN TRT01P TRT01PN TRT02P TRT02PN TR01SDT TR01STM
      TR01SDTM TR01EDT TR01ETM TR01EDTM TR02SDT TR02STM TR02SDTM TR02EDT TR02ETM
      TR02EDTM VAX101 VAX102 VAX10U VAX201 VAX202 VAX20U VAX101DT VAX102DT VAX10UDT
      VAX201DT VAX202DT VAX20UDT UNBLNDDT RANDDT COHORT COHORTN DOSALVL DOSALVLN
      DOSPLVL DOSPLVLN DS30KFL PHASE PHASEN AGEGR4 AGEGR4N HIVFL PEDIMMFL PEDREAFL
      DS3KFL AGETR01 AGETRU01;
   attrib ASTDT length=8. label="Analysis Start Date" COMORBFL length=$1.
      label="Comorbidity Flag" CAT1 length=$100.
      label="Charlson Comorbidity Index Category 1" CAT2 length=$100.
      label="Charlson Comorbidity Index Category 2" ASTDTF length=$1.
      label="Analysis Start Date Imputation Flag" ASTDY
      label="Analysis Start Relative Day" AENDT length=8. label="Analysis End Date"
      AENDTF length=$1. label="Analysis End Date Imputation Flag" AENDY
      label="Analysis End Relative Day" ADT length=8. label="Analysis Date" ADTF
      length=$1. label="Analysis Date Imputation Flag" ADURN length=8.
      label="Analysis Duration (N)" ADURU length=$10.
      label="Analysis Duration Units";
   set _mh(keep=STUDYID USUBJID SUBJID SITEID MHSEQ MHTERM MHDECOD MHPTCD
      MHBODSYS MHBDSYCD MHLLT MHLLTCD MHPTCD MHHLT MHHLTCD MHHLGT MHHLGTCD
MHSOC
      MHSOCCD MHCAT MHSTDTC MHENDTC MHENRTPT MHENTPT DICTVER MHSPID ASTDT ASTDTF
      ASTDY AENDT AENDTF AENDY COMORBFL CAT1 CAT2 ADT ADTF ADURN ADURU DICTVER
      USUBJID SUBJID SITEID AGE AGEU AGEGR1 AGEGR1N SEX SEXN RACE RACEN ARACE
      ARACEN RANDFL SAFFL ARM ARMCD ACTARM ACTARMCD TRTSDT TRTSTM TRTSDTM TRTEDT
      TRTETM TRTEDTM TRT01A TRT01AN TRT02A TRT02AN TRT01P TRT01PN TRT02P TRT02PN
      TR01SDT TR01STM TR01SDTM TR01EDT TR01ETM TR01EDTM TR02SDT TR02STM TR02SDTM
      TR02EDT TR02ETM TR02EDTM VAX101 VAX102 VAX10U VAX201 VAX202 VAX20U VAX101DT
      VAX102DT VAX10UDT VAX201DT VAX202DT VAX20UDT UNBLNDDT RANDDT COHORT COHORTN
      DOSALVL DOSALVLN DOSPLVL DOSPLVLN DS30KFL PHASE PHASEN AGEGR4 AGEGR4N HIVFL
      PEDIMMFL PEDREAFL DS3KFL AGETR01 AGETRU01);
run;

options MPRINT MLOGIC SYMBOLGEN;
options MPRINT SYMBOLGEN MLOGIC;
;

proc sort data=admh out=datvout.admh(label='Medical History Analysis Dataset');
   by MHCAT USUBJID MHSPID MHTERM;
run;

proc printto;
run;
```