```
*************************************************************************************
**************
*              CLIENT:   BigPharma
*            PROTOCOL:   CDISCV2
*       PROGRAM NAME:   REVSUPP.SAS
*        SAS VERSION:   SAS V9 (Windows) 9.2, 9.3
*            PURPOSE:   Merges SUPPQUAL back onto the original datasets.
*
*        USAGE NOTES:
*         PARAMETERS:   LIBIN: Input libname
*                       LIBOUT: Output libname, defaults to WORK
*                       DS: Base SDTM domain
*                       SUPP: Name of suppqual file
*                       OUTDS: Output dataset name, defaults to input dataset name
*                       MAPSPEC = Name of the mapping spec
*                       MAPLOC = Directory of the mapping spec
*                       DELSUPP:  DEPRECATED - Delete supplemental records merged
to DS from SUPP__.  Defaulted to N (Y,N)
*                       MOVSUPP:  DEPRECATED - Move supplemental dataset to libout
if libout <> libin.  Defaulted to N (Y,N)
*              AUTHOR:    (b) (6)
*       DATE CREATED:   03/05/2009
*
*       DEPENDENCIES:
*   MODIFICATION LOG:
*************************************************************************************
**************
* DATE         BY          DESCRIPTION
*************************************************************************************
**************
*  19AUG2009  (b) (6)       Add logic to handle when the same variable (QNAME) qname
is merged onto
*                           the domain multiple times because it is associated with
more than one IDVAR
*  11NOV2009  (b) (6)       Add code to retain dataset label and sortedby
information
*                           Add parameter DELSUPP
*                           Add logic to report data no match when supp records not
found in domain dataset
*                           Add code to move supplemental table to libout using
parameter MOVSUPP
*                           Add code to remove supplemental records that have been
merged back to domain
*                           Add code to delete supplemental table if all records are
merged back on to domains
*                           Add code to delete output datasets if ERR occurs
*  30NOV2009 (b) (6)        Add code to handle when no records for domain exist in
supplemental table
*  24FEB2010  (b) (6)       Fixed the code to update a variable with information
from suppqual (for multiple IDVARs)
```

```
*  15JUL2016 (b) (6)       Handle where main DS name length is greater than 2.
Need to adjust it to just the first
*                          2 characters to process.
*  15JAN2017 (b) (6)       See SVN history for further modification log changes.
********************************************************************************
**************
* Copyright 2019 Pharmaceutical Product Development, Inc.
* All Rights Reserved.
********************************************************************************
**************;

%macro
revsupp(libin=,libout=WORK,ds=,supp=,outds=,maploc=,mapspec=,delsupp=,movsupp=) /
STORE  des="V1.3.0.0";
%let id = &sysmacroname v1.3.0.0;
%put &id begin;

/**** ERROR TRAPPING ****/
%let err = E%str(RROR: );

/* Ensure libraries and datasets exist */
%if %sysfunc(exist(&libin..&ds,DATA)) ne 1 %then %do;
        %put &err. LIBIN or DS parameter is invalid. Make sure both exist and are
correct.;
        %goto ERR;
%end;

%if %sysfunc(libref(&libout)) ne 0 %then %do;
        %put &err. LIBOUT parameter is invalid.;
        %goto ERR;
%end;

/* If Maploc and/or Mapspec is populated, make sure they're both */
/* there and point to a valid file                               */
%if "&maploc" ne "" and "&mapspec" eq "" %then %do;
        %put &err. MAPLOC specified but MAPSPEC is missing;
        %goto ERR;
%end;
%if "&maploc" eq "" and "&mapspec" ne "" %then %do;
    %put &err. MAPSPEC specified but MAPLOC is missing;
    %goto ERR;
%end;
%if "&maploc" ne "" and "&mapspec" ne "" %then %do;
        %if %sysfunc(fileexist(&maploc.\&mapspec.)) = 0 %then %do;
                %put &err. The mapping spec passed can not be found, please update
MAPLOC and MAPSPEC;
                %goto err;
        %end;
%end;
/**** END ERROR TRAPPING ****/
```

```
%if "&delsupp" ne "" or "&movsupp" ne "" %then %do;
        %put ALERT%str(_I): Parameters DELSUPP and MOVSUPP are no longer used.
Start planning to remove them from macro calls.;
%end;

%if %sysfunc(exist(&libout..&outds,DATA)) ne 0 %then %do;
        %put ALERT%str(_I): Output dataset already exists and will be replaced;
%end;

%local initial_DS_param;

/* HOLD SINCE DS PARAM CAN GET CHANGED DEPENDING UPON RDOMAIN VALUE, BUT MAY BE
NEEDED LATER IF MAPSPEC PROVIDED */
%let initial_DS_param=&DS;

%if &outds eq  %then %let outds = &ds;

/* SETUP A TEMP SUBFOLDER IN THE SAS ASSIGNED WORK FOLDER FOR INTERMEDIATE
PROCESSING */
%local work_subdir;
%let work_subdir=%sysfunc(getoption(work));

%local _xwait _xsync;
%let _xwait=%sysfunc(getoption(XWAIT));
%let _sync=%sysfunc(getoption(XSYNC));

options noxwait xsync;
x "mkdir &work_subdir\tmp";

/* RESET BACK TO WHAT IT WAS PRIOR TO MACRO CALL */
options &_xwait &_xsync;

libname _tmp "&work_subdir\tmp";

/* CLEAN UP JUST IN CASE THIS IS THE SAME SESSION WITH MULTIPLE CALLS TO REVSUPP */
proc datasets lib=_tmp nolist kill;
quit;
run;

%let libsupp = %str();
%let libsupp1 = %str();
%let supp1 = %str();
%let idlen = %str();
%let idtype = %str();

%* Check to see if the supplemental libref and dataset exist and is valid;
%let SUPPFND = 1;
%if %index(&supp,.) = 0 %then %do;
        %put ALERT_I: Libref not stated for %left(&supp) assuming same as libin.;
```

```
        %if %sysfunc(exist(&libin..&supp,DATA)) ne 1 %then %do;
                %if "&supp" gt "" %then
                        %put ALERT_C: The %left(&supp) supplemental qualifier
dataset provided is not currently valid.;
                %else
                        %put ALERT_I: No supplemental qualifier dataset was
provided.;
                %let SUPPFND = 0;
                %let SUPP=;
        %end; %else %do;
                %let supp1 = %upcase(&supp);
                %let libsupp = &libin;
        %end;
%end;  %else %do;
        %if %sysfunc(exist(&supp,DATA)) ne 1 %then %do;
                %if "&supp" gt "" %then
                        %put ALERT_C: The %left(&supp) supplemental qualifier
dataset provided is not currently valid.;
                %else
                        %put ALERT_I: No supplemental qualifier dataset was
provided.;
                %let SUPPFND = 0;
                %let SUPP=;
        %end; %else %do;
                %let supp1 = %scan(%upcase(&supp),-1,'.');
                %let libsupp = %scan(%upcase(&supp),1,'.');
        %end;
%end;

/* BRING IN THE DATASETS INVOLVED WITH THE REVSUPP CALL */
proc copy in=&libin out=_tmp noclone;
        select  &ds;
run ;
%if &suppfnd ne 0 %then %do;
        proc copy in=&libsupp out=_tmp noclone;
                select  &supp1;
        run ;
%end;
%let libsupp=_tmp;

/* Ensure libraries and datasets exist after the proc copy from above */
%if %sysfunc(exist(&libin..&ds,DATA)) ne 1 %then %do;
        %put &err. Either the LIBIN or DS value is invalid or not accessible. Make
sure both exist and are correct.;
        %goto ERR;
%end;

/*
** PER AN ENHANCEMENT REQUEST IN 2016/2017, IF THE MAIN DOMAIN DATASET NAME HAS A
LENGTH
```

```
** GREATER THAN 2, THEN ITS ASSOCIATED RDOMAIN IN THE SUPP DATASET MAY BE DIFFERENT
** (MORE THAN LIKELY JUST THE FIRST 2 LETTERS).  HERE WE MAKE THE PROCESS GENERIC
BY
** JUST CAPTURING THE RDOMAIN VALUE UPFRONT AND USE THAT.  IF RDOMAIN~=DS, THEN THE
** THE DS DATASET WILL GET RENAMED TO THE VALUE OF RDOMAIN FOR MERGING PROCESSING.
*/
%if %sysfunc(exist(_tmp.&supp1.)) %then %do;
        data _null_;
                set _tmp.&supp1.;
                /* USE THE RDOMAIN VALUE TO PROCESS THE REST OF THE WAY */
                call symputx('DS', strip(rdomain));
                stop;
        run;
%end;

/* RENAME DS DATASET TO THE VALUE OF RDOMAIN.  PROC DATASETS JUST PROVIDES A NOTE
IF THEY'RE BOTH THE SAME */
proc datasets library=_tmp nolist;
        change &initial_DS_param.=&DS.;
quit;

%let libin = %upcase(_tmp);
%let ds = %upcase(&ds);
%let libout= %upcase(&libout);
%let supp = %upcase(&supp1);
%let outds =%upcase(&outds);
%let cntqnam = 0;

%If %Length(&maploc)=0 %Then %Do;
        proc sql noprint;
                select * from dictionary.macros
                where upcase(name) = "G_MAPLOC";
        quit;
        %If &sqlobs > 0 %Then %Let maploc=&g_maploc.;;
%End;
%If %Length(&mapspec)=0 %Then %Do;
        proc sql noprint;
                select * from dictionary.macros
                where upcase(name) = "G_MAPSPEC";
        quit;
        %If &sqlobs > 0 %Then %Let mapspec=&g_mapspec.;;
%End;

%let libsupp1 = &libsupp;
%put supp1 = &supp1;
%put libsupp = &libsupp;
%put supp = &supp;

/* CAPTURE SOURCE PARENT DOMAIN SORTEDBY AND LABEL INFO */
%local ds_label ds_sortby;
```

```
%let dsid = %sysfunc(open(&libin..&ds,is));
%if &DSID = 0 %then %put %sysfunc(sysmsg());
%let ds_sortby =%sysfunc(attrc(&dsid,SORTEDBY));
%let ds_label =%sysfunc(attrc(&dsid,LABEL));
%if "%left(%trim(&ds_label))" = "" %then %do;
        %let ds_label =%str( );
%end;
%let rc = %sysfunc(close(&dsid));

/* SUPP DATASET EXISTS */
%if &SUPPFND ne 0 %then %do;
        proc sql noprint;
                select count(*) into: cntDomain
                from &libsupp..&supp where upcase(rdomain) = upcase("&ds");
        quit;
        %put cntDomain=&cntDomain;
        %if &cntDomain ne 0 %then %do;
                proc sql noprint;
                        select count(distinct idvar) into: idpres
                        from &libsupp..&supp where idvar ne '' and
                        upcase(rdomain) = upcase("&ds") and idvarval ne '';
                quit;
                %put idpres=&idpres;

                %if &idpres = 0 %then %do;
                        %let idcnt = 1; %let id1 = ;
                %end;
                %else %do;
                        /* Determine how many ID vars the suppqual file has */
                        proc sql noprint;
                                select count(distinct upcase(idvar||'~')) into:
idcnt
                                from &libsupp1..&supp where upcase(rdomain) =
upcase("&ds");
                                %let idcnt = %trim(%left(&idcnt));

                                select distinct compress(upcase(idvar||'~'),'~')
into: id1 -: id&idcnt
                                from &libsupp1..&supp where upcase(rdomain) =
upcase("&ds");

                                /* get a count of the distinct qnam's across idvars
this will be used lower in program */
                                select count(distinct qnam) into :cntqnam
                                from &libsupp1..&supp where upcase(rdomain) =
upcase("&ds")
                                and idvar is not null
                                group by qnam having count(distinct idvar) > 1;
                        quit;
                %end;
```

```
                %do i = 1 %to &idcnt;
                        %put *************idval=&&id&i..******************;
                        %put *************id=&&id&i     ******************;
                        %if &&id&i.. ne  %then %do;

                                proc contents data=&libin.._all_ out=__cont
noprint;
                                run;
                                data __cont(drop=oldtype);
                                        set __cont(rename=(type=oldtype));
                                        length type $4;
                                        if oldtype = 1 then type = 'num';
                                        else if oldtype = 2 then type = 'char';
                                run;

                                /* Find the length and type of the ID variable in
the original DS */
                                proc sql noprint;
                                        select length, type into: idlen, :idtype
                                        from __cont
                                        where upcase(memname) = upcase("&ds") and
                                        upcase(name) = upcase("&&id&i");

                                        select length, type into: usubjlen,
:usubjtype  /*make sure usubjid var has same attributes as */
                                        from __cont
    /*domain dataset */
                                        where upcase(memname) = upcase("&ds") and
                                        upcase(name) = upcase("usubjid");
                                quit;
                                /* Create a dataset that contains a single ID value
*/

                                data  __TOTRAN;
                                        %if %upcase(&usubjtype) ~= NUM %then %do;
                                                length usubjid
$%trim(%left(&usubjlen.));
                                        %end;

                                        %if %upcase(&idtype) ~= NUM %then %do;
                                                length &&id&i..
$%trim(%left(&idlen.)) ;
                                        %end;
                                        set &libsupp1..&supp;

                                        %if %upcase(&usubjtype) = NUM %then %do;
                                                usubjid =input(usubjid,best.);
                                        %end;

                                        %if %upcase(&idtype) = NUM %then %do;
```

```
                                        &&id&i.. = input(idvarval,best.) ;

                                    %end;
                                    %else %do;
                                        &&id&i..= idvarval ;
                                    %end;

                                    where upcase(rdomain) = upcase("&ds") and
upcase(idvar) = "&&id&i";

                                    keep usubjid &&id&i.. qnam  qlabel qval;
                                run;

                                proc sort data= __TOTRAN tagsort nodup;
                                    by usubjid &&id&i..;
                                run;

                                /* Now we tranpose to get ready for the merge */
                                proc transpose data=__TOTRAN
out=__POSTTRAN(drop=_NAME_ _LABEL_);
                                    by usubjid &&id&i..;
                                    id qnam;
                                    var qval;
                                quit;
                            %end;
                            %else %do; /*This is when the idvar is missing */
                                proc sql;
                                    create table __TOTRAN as
                                    select distinct usubjid, qnam, qlabel, qval
                                    from &libsupp1..&supp
                                    where upcase(rdomain) = upcase("&ds")
                                        and idvar is null
                                    order by usubjid;
                                quit;
                                proc transpose data=__TOTRAN
out=__POSTTRAN(drop=_NAME_ _LABEL_);
                                    by usubjid;
                                    id qnam;
                                    var qval;
                                quit;
                            %end;
                            /* Fix the lengths of the variables */
                            %if "&maploc" ne "" and "&mapspec" ne "" %then %do;
                                PROC IMPORT OUT= WORK.__spec(keep=outname outlabel
outlength flag)
                                    DATAFILE= "&maploc.\&mapspec" DBMS=EXCEL
REPLACE;
                                    /* THE "ZZ" COLUMN IS NOTHING SPECIFIC.
JUST ENSURING THAT IF ADDITIONAL COLUMNS ARE TO THE SPEC THAT ITS COVERED ENOUGH TO
INCLUDE FLAG COLUMN */
```

```
                                        RANGE="&initial_DS_param.$H6:ZZ600";
                                        GETNAMES=YES;
                                RUN;

                                proc contents data=__posttran
out=__posttran_attributes noprint;
                                run;

                                proc sql noprint;
                                        create table __Torun as
                                        select *
                                        from
                                        (
                                        select outname, outlabel, outlength
                                        from (select * from __spec where outname ne
'' and flag in ("**",'S'))
                                        where outname in (select distinct qnam from
&libsupp1..&supp)
                                        ) a
                                        inner join __posttran_attributes b
                                        on upcase(a.outname)=upcase(b.name)
                                        ;
                                quit;

                                %if &sqlobs > 0 %then %do;
                                        /* TEMPORARILY RENAME THE VARIABLES GETTING
ATTRIBUTE CHANGES IN ORDER TO HANDLE VAR LENGTH CHANGES WITHOUT WARNINGS */
                                        data _null_;
                                                set __torun end=eof;
                                                if _n_ = 1 then do;
                                                        call execute("proc datasets
library=work nolist;");
                                                        call execute("modify
__POSTTRAN;  rename ");
                                                end;
                                                call execute(strip(outname) || '=
___' ||strip(outname));
                                                if eof then call execute("; run;");
                                        run;

                                        /* SET ATTRIBUTES FOR THE VARIABLES BEING
REVISED */
                                        data _null_;
                                                set __torun end=eof;
                                                if _n_ = 1 then
                                                        call execute("Data
__POSTTRAN;");

                                                call execute("attrib " ||
left(trim(outname)) || ' label="' ||
```

```
                                                left(trim(outlabel)) || '"
length=$' || left(trim(outlength)) || ';');
                                                call execute("if " ||
left(trim(outname)) || "= '' then " || left(trim(outname)) || "='';");
                                                if eof then
                                                        call execute("set
__POSTTRAN; run;");
                                        run;

                                        /* SET ATTRIBUTES FOR THE VARIABLES BEING
REVISED */
                                        data _null_;
                                                set __torun end=eof;
                                                if _n_ = 1 then
                                                        call execute("Data
__POSTTRAN(drop=___: ) __trunc(keep=___trunc_var); length ___trunc_var $8.; set
__POSTTRAN;");
                                                call execute(strip(outname) ||
'=strip(___'||strip(outname)||');');
                                                call execute('if
'||strip(outlength)||' < length(strip(___'||strip(outname)||')) then do;');
                                                call execute('  ___trunc_var =
"'||strip(outname)||'";' );
                                                call execute('end;');
                                                if eof then
                                                        call execute("run;");
                                        run;
                                        proc sort data=__trunc(where=(not
missing(___trunc_var))) nodupkey;
                                                by ___trunc_var;
                                        run;

                                        /* PROVIDE ALERTS FOR TRUNC-ATION CASES
FROM THE ABOVE PROCESSING */
                                        data _null_;
                                                set __trunc;
                                                put 'ALERT' '_R: Trunc' 'ation
detected upon applying mapping spec length for variable: ' ___trunc_var;
                                        run;
                                %end;
                        %end; /* %if "&maploc" ne "" and "&mapspec" ne "" %then */

                        /* Get the labels so we can add them later */
                        proc sql noprint;
                                select distinct(qnam || '="' || trim(qlabel) ||
'"') into: label
                                separated by ' '
                                from &libsupp1..&supp
                                where upcase(rdomain) = upcase("&ds")
                                %if &&id&i.. ne  %then %do;
```

```
                                        and upcase(idvar) = "&&id&i";
                                %end;
                                %else %do;
                                        and idvar is null
                                %end;
                                ;
                        quit;
                        /* Determine where the input comes from based on run
iterations */
                        %if &i = 1 %then %do;
                                %let input = &libin..&ds;
                        %end;
                        %if &i ne 1 %then %do;
                                %let input = &libout..&outds.;
                        %end;
                        /* Sort SDTM dataset by ID variable */
                        proc sort data=&input out=&libout..&outds. tagsort;
                                by usubjid  %if &&id&i.. ne  %then &&id&i..;;
                        run;

                        /*multiple qnam's found */
                        %if &idcnt > 1 %then %do;
                                proc contents data=&libout.._all_ out=__cont1
noprint;
                                run;
                                proc contents data=work._all_ out=__cont2 noprint;
                                run;
                                data __cont(drop=oldtype);
                                        set __cont1(rename=(type=oldtype))
__cont2(rename=(type=oldtype));
                                        length type $4;
                                        if oldtype = 1 then type = 'num';
                                        else if oldtype = 2 then type = 'char';
                                run;
                                proc sql noprint;
                                        /*get a list of vars not on domain ds*/
                                        %let mergecols=%str();
                                        select distinct name into :mergecols
SEPARATED by ' '
                                        from __cont a
                                        where upcase(libname) = upcase("work")
                                                and upcase(memname) =
upcase("__posttran")
                                                and not exists
                                                        (select 1 from __cont b
                                                        where upcase(libname) =
upcase("&libout")
                                                                and upcase(memname) =
upcase("&outds")
                                                                and
```

```
upcase(a.name)=upcase(b.name))
                                        ;
                                        /*get a count of vars on __posttran that
are already on domain dataset*/
                                        select count(distinct name) into
:cntupdatecols
                                        from __cont a
                                        where upcase(libname) = upcase("work")
                                                and upcase(memname) =
upcase("__posttran")
                                                and upcase(name) not in ('USUBJID',
"&&id&i")
                                                and exists
                                                        (select 1 from __cont b
                                                        where upcase(libname) =
upcase("&libout")
                                                        and upcase(memname) =
upcase("&outds")
                                                        and
upcase(a.name)=upcase(b.name))
                                        ;
                                        %let cntupdatecols =
%trim(%left(&cntupdatecols));
                                        /*set macro vars = list of columns that are
already found on domain dataset*/
                                        %if &cntupdatecols ^= 0 %then %do;
                                                select distinct name into
:updatecols1-:updatecols&cntupdatecols
                                                from __cont a
                                                where upcase(libname) =
upcase("work")
                                                        and upcase(memname) =
upcase("__posttran")
                                                        and upcase(name) not in
('USUBJID', "&&id&i")
                                                        and exists
                                                                (select 1 from
__cont b
                                                                where
upcase(libname) = upcase("&libout")
                                                                and upcase(memname)
= upcase("&outds")
                                                                and
upcase(a.name)=upcase(b.name))
                                                ;
                                        %end;
                                quit;
                                %put &mergecols;
                                %put &cntupdatecols;
                                %put
```

```
************mergecols=&mergecols******************;
                                /*Here is where the qnam's not found on domain are
merged to domain*/
                                %if &mergecols ^= %str() %then %do;
                                        /* Now we merge together the base dataset
and SUPPQUAL */
                                        data &libout..&outds.;
                                                merge &libout..&outds.(in=a)
__POSTTRAN(in=b keep=usubjid &&id&i.. &mergecols);
                                                by usubjid %if &&id&i.. ne  %then
&&id&i..;;
                                                label &label.;
                                                if b and not a then put "ALERT"
"_R: There is data in the %left(&supp) supplemental dataset that had no row to
merge on!";
                                                if a;
                                        run;
                                %end;
                                %if &cntupdatecols ^= 0 %then %do;
                                        /*rare but it happens we could have
multiple duplicate qnam's to update on domain for different idvars*/
                                        /*whether one or many it should be handled
here*/
                                        %put NOTE: updating values on dataset
%left(&libout..&outds.);

                                        data &libout..&outds.;
                                                merge &libout..&outds.(in=a)
__POSTTRAN(in=b rename=(
                                                %do x = 1 %to &cntupdatecols;

&&updatecols&x..=__&&updatecols&x..
                                                %end;)
                                                %if &mergecols ne %then
drop=&mergecols;);
                                                by usubjid %if &&id&i.. ne  %then
&&id&i..;;
                                                if b and not a then put 'ALERT'
"_R: There is data in the %left(&supp) supplemental dataset that had no row to
merge on!";
                                                if a;

                                                %do x = 1 %to &cntupdatecols;
                                                        if &&updatecols&x.. eq ''
and __&&updatecols&x.. ne '' then
                                                                &&updatecols&x.. =
__&&updatecols&x..;
                                                        drop __&&updatecols&x..;
                                                %end;
                                        run;
```

```
                                        %put NOTE: %left(&sqlobs) records on
dataset %left(&libout..&outds.) have been updated;
                                %end;
                        %end; /* %if &idcnt > 1 %then */
                        /*unique qname across idvar's or first idvar iteration just
merge*/
                        %else %do;
                                /* Now we merge together the base dataset and
SUPPQUAL */
                                data &libout..&outds.(label="");
                                        merge &libout..&outds.(in=a)
__POSTTRAN(in=b);
                                        by usubjid %if &&id&i.. ne  %then
&&id&i..;;
                                        label &label.;
                                        if b and not a then put "ALERT" "_R: There
is data in the %left(&supp) supplemental dataset that had no row to merge on!";
                                        if a;
                                run;
                        %end;
                %end; /* %do i = 1 %to &idcnt; */
        %end; /* %if &cntDomain ne 0 %then */
        %else %do;
                %put ALERT_R: supplemental dataset does not have any records for
the &DS domain dataset;
        %end;
%end; /* End merging together suppqual, SUPPFND ne 0 */
/* COPY OVER THE DOMAIN DATASET AS IS */
%else %do;
        data &libout..&outds.;
                set &libin..&ds.;
        run;
%end;


%if "&maploc" ne "" and "&mapspec" ne "" %then %do;
        PROC IMPORT OUT= WORK.__spec(keep=outname outlabel outlength flag)
                DATAFILE= "&maploc.\&mapspec"
                DBMS=EXCEL REPLACE;
                /* THE "ZZ" COLUMN IS NOTHING SPECIFIC.  JUST ENSURING THAT IF
ADDITIONAL COLUMNS ARE TO THE SPEC THAT ITS COVERED ENOUGH TO INCLUDE FLAG COLUMN
*/
                RANGE="&initial_DS_param.$H6:ZZ600";
                GETNAMES=YES;
        RUN;

        proc sql noprint;
                create table __Torun as
                select outname, outlabel, outlength
```

```
                from (select * from __spec where outname ne '' and flag in
("**",'S'))
                %if &SUPPFND ne 0 %then where outname not in (select distinct qnam
from &libsupp..&supp);
                ;
        quit;

        %if &sqlobs > 0 %then %do;
                data _null_;
                        set __torun end=eof;
                        if _n_ = 1 then
                        call execute("Data &libout..&outds.;  set
&libout..&outds.;");
                        call execute("attrib " || left(trim(outname)) || ' label="'
||
                        left(trim(outlabel)) || '" length=$' ||
left(trim(outlength)) || ';');
                        call execute(outname || '="";');
                        if eof then call execute("run;");
                run;
        %end;
%end;


/* KEEP ORIGINAL LABEL AND SORT */
%if "&ds_sortby" eq "" %then %do;
        data &libout..&outds(label="&ds_label");
                set &libout..&outds;
        run;
%end;
%else %do;
        proc sort data=&libout..&outds out=&libout..&outds(label="&ds_label")
tagsort;
        by &ds_sortby.; ;
        run;
%end;

/* Cleanup the work directory */

proc datasets library=work nodetails nolist;
        delete __posttran __totran __cont __cont1 __cont2
        %if "&maploc" ne "" and "&mapspec" ne "" %then __spec __torun;
        ;
run; quit;

%goto fin;

%err:

%if &libin ne &libout and "&outds" gt " " and %sysfunc(libref(&libout)) eq 0 %then
```

```
%do;
        proc datasets library=&libout. nodetails nolist;
                delete &outds. &supp.;
        run; quit;
%end;
%goto fin;

%fin:
%put &id: End;

%mend REVSUPP;
```