

```

*****
** Program Name   : adsl-demo-7d-peds-eval-eff.sas           **
** Date Created  : 15Nov2021                                **
** Programmer Name : (b) (4), (b) (6)                       **
** Purpose       : Create adsl-demo-7d-peds-eval-eff        **
** Input data    : adsl, adc19ef                             **
** Output file   : adsl-demo-7d-peds-eval-eff.html          **
*****
options mprint mlogic symbolgen nocenter missing=" ";
ods escapechar="~";

proc datasets library=WORK kill nolist nodetails;
quit;

**Setup the environment**
%let
bprot=/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/analysis/eSUB;
%let codename=adsl-demo-7d-peds-eval-eff;
libname datvprot "&bprot.data_vai" access=readonly;
%let outlog=&prot./logs/&codename..log;
%let outtable=&prot./output/&codename..html;

proc printto log="&outlog." new;
run;

*****
* Clean *;
*****

proc delete data=work._all_;
run;

proc format;
value sex 1='Male' 2='Female';
value ethnic 1='Hispanic/Latino' 2='Non-Hispanic/non-Latino' 3='Not reported'
4='Unknown';
value crace 1="White" 2="Black or African American" 2.5="All others"
3="(*ESC*){nbspspace 5}American Indian or Alaska Native"
4="(*ESC*){nbspspace 5}Asian"
5="(*ESC*){nbspspace 5}Native Hawaiian or other Pacific Islander"
6="(*ESC*){nbspspace 5}Multiracial" 7="(*ESC*){nbspspace 5}Not reported"
8="(*ESC*){nbspspace 5}Unknown";
run;

proc sort data=datvprot.adsl out=adsl;
by usubjid;
run;

proc sort data=datvprot.adc19ef out=adc19ef(keep=usubjid PDSDMFL VRBLNGFL
CRD1NGFL PDP17FL PDP27FL) nodupkey;

```

```

    by usubjid;
run;

data adsl;
    merge adsl(in=a) adc19ef;
    by usubjid;

    if a;
run;

data adsl;
    set adsl(in=a);
    by usubjid;

    if a;
length covblstx COMBODFLX OBSCATFL $50.;

if covblst="POS" then
    do;
        covblstx="Positive(*ESC*){super e}";
        covblstnx=1;
    end;
else if covblst="NEG" then
    do;
        covblstx="Negative(*ESC*){super f}";
        covblstnx=2;
    end;
else if covblst not in ("POS" "NEG") then
    do;
        covblstx="Unknown";
        covblstnx=3;
    end;

if COMBODFL="Y" or /*(bmicatn = 4 and age >=16) or*/
OBESEFL="Y" then
    do;
        COMBODFLNX=1;
        COMBODFLX="Yes";
    end;
else

    /* if COMBODFL='N' then */
    do;
        COMBODFLNX=2;
        COMBODFLX="No";
    end;

if /*(bmicatn = 4 and AGETR01 >=16) or*/
OBESEFL="Y" then
    do;
        OBSCATFN=1;
        OBSCATFL="Yes";
    end;
else

```

```
do;
  OBSCATFN=2;
  OBSCATFL="No";
end;

if racialdn=999 then
  racialdn=.;
run;
```

```
data adsl;
  set adsl;
  length countryx $50;

  if country='ARG' then
    countryx='Argentina';
  else if country='BRA' then
    countryx='Brazil';
  else if country='DEU' then
    countryx='Germany';
  else if country='TUR' then
    countryx='Turkey';
  else if country='USA' then
    countryx='USA';
  else if country='ZAF' then
    countryx='South Africa';
  else
    countryx='Others';
run;
```

```
data race_ao;
  set adsl;

  if aracen not in (1, 2);
  aracen=2.5;
  arace="ALL OTHERS";
  AGETR01=.;
run;
```

```
data adsl;
  set adsl race_ao;
run;
```

```
data adsl;
  set adsl;
  length agegr1x $50.;

  if 12<=age<=15 then
    do;
      agegr1nx=1;
      agegr1x="12 to 15 ";
    end;

  if 16<=age<=55 then
    do;
```

```
    agegr1nx=2;
    agegr1x="16 to 55 ";
end;
```

```
if age>55 then
  do;
    agegr1nx=3;
    agegr1x="(*ESC*){Unicode 003E}55 ";
  end;
```

```
run;
```

```
data agp1;
  set adsl;
```

```
  if 16<=age<=17;
  agegr1nx=5;
  agegr1x="16 to 17 ";
  AGETR01=.;
```

```
run;
```

```
data agp2;
  set adsl;
```

```
  if 16<=age<=25;
  agegr1nx=6;
  agegr1x="16 to 25 ";
  AGETR01=.;
```

```
run;
```

```
data agp3;
  set adsl;
```

```
  if 16<=age<=64;
  agegr1nx=7;
  agegr1x="16 to 64 ";
  AGETR01=.;
```

```
run;
```

```
data agp4;
  set adsl;
```

```
  if 18<=age<=64;
  agegr1nx=8;
  agegr1x="18 to 64 ";
  AGETR01=.;
```

```
run;
```

```
data agp5;
  set adsl;
```

```
  if 55<=age<=64;
  agegr1nx=9;
  agegr1x="55 to 64 ";
  AGETR01=.;
```

```

run;

data agp7;
  set adsl;

  if age>=65;
  agegr1nx=4;
  agegr1x="(*ESC*){unicode 2265}65 ";
  AGETR01=.;
run;

data agp8;
  set adsl;

  if 65<=age<=74;
  agegr1nx=10;
  agegr1x="65 to 74 ";
  AGETR01=.;
run;

data agp9;
  set adsl;

  if age>=75;
  agegr1nx=11;
  agegr1x="(*ESC*){unicode 2265}75 ";
  AGETR01=.;
run;

data agp10;
  set adsl;

  if 75<=age<=85;
  agegr1nx=12;
  agegr1x="75 to 85 ";
  AGETR01=.;
run;

data agp11;
  set adsl;

  if age>85;
  agegr1nx=13;
  agegr1x="(*ESC*){Unicode 003E}85 ";
  AGETR01=.;
run;

data adsl;
  set adsl agp1 agp2 agp3 agp4 agp5 agp7 agp8 agp9 agp10 agp11;
run;

data g_adsl_dsin;
  set adsl;
  where EVALEFFL="Y" and PDP27FL='Y' and agegr1n=1 and phasen ne 1;

```

FDA-CBER-2022-5812-0072461

```

run;

data _trtmap;
  length trtcode trtdec $100;

  if 0 then
    set g_adsl_dsin(keep=TRT01PN);
  trtval=1;

  if vtype(TRT01PN)='C' then
    trtcode=tranwrd(compbl(quote("8")), ' ', "" );
  else
    trtcode="8";
  trtdec="BNT162b2 (30 (*ESC*){unicode 03BC}g)";
  trtvar="TRT01PN";
  trtlbl="TRT01P";
  output;
  trtval=2;

  if vtype(TRT01PN)='C' then
    trtcode=tranwrd(compbl(quote("9")), ' ', "" );
  else
    trtcode="9";
  trtdec="Placebo";
  trtvar="TRT01PN";
  trtlbl="TRT01P";
  output;
  trtval=3;

  if vtype(TRT01PN)='C' then
    trtcode=tranwrd(compbl(quote("8 9")), ' ', "" );
  else
    trtcode="8 9";
  trtdec="Total";
  trtvar="TRT01PN";
  trtlbl="TRT01P";
  output;
  stop;
run;

data g_adsl_dsin;
  set g_adsl_dsin;

  if TRT01PN in (8) then
    do;
      newtrtn=1;
      newtrt=coalescec("BNT162b2 (30 (*ESC*){unicode 03BC}g)", TRT01P);
      output;
    end;

  if TRT01PN in (9) then
    do;
      newtrtn=2;
      newtrt=coalescec("Placebo", TRT01P);

```

```

        output;
    end;

    if TRT01PN in (8 9) then
        do;
            newtrtn=3;
            newtrt=coalescec("Total", TRT01P);
            output;
        end;
    run;

*-----;
* Initialize dataset for non-pvalue footnote queue. ;
*-----;

data _stdft1(compress=no);
    length model $200 mark $5;
    index=0;
    model=' ';
    mark=' ';
run;

*-----;
* Initialize dataset for pvalue related footnote queue.;
*-----;

data _stdft2(compress=no);
    length model $200 mark $5;
    index=0;
    model=' ';
    mark=' ';
run;

*-----;
* Initialize structure for _BASETEMPLATE dataset. ;
*-----;

data _basetemplate(compress=no);
    length _varname $8 _cvalue $35 _direct $20 _vrlabel $200 _rwlabel
        _colabel $800 _datatyp $5 _module $8 _pr_lbl $ 200;
    array _c _character_;
    delete;
run;

*-----;
* Create next _DATAn dataset ;
*-----;

data _data1;
    set g_adsl_dsin;
    where (NEWTRTN is not missing);
run;

proc sql noprint;

```

```

select put(nobs - delobs, 12.) into :_nobs from dictionary.tables
      where (libname="WORK" and memname="_DATA1");
select setting into :miss from dictionary.options where
      upcase(optname)="MISSING";
quit;

*-----;
* Count number of treatment groups ;
*-----;

proc sql noprint;
      select count(unique NEWTRTN) into :_trtn from _data1 where NEWTRTN is not
            missing;
quit;

*-----;
* Generate variable _TRT. Use assigned order if applicable ;
*-----;

proc sort data=_data1;
      by NEWTRTN USUBJID;
run;

data _data1;
      retain _trt 0;
      length _str $200;
      _datasrt=1;
      set _data1 end=eof;
      by NEWTRTN USUBJID;
      drop _str;
      _str=' ';
      _lastby=1;
      _dummyby=0;

      if first.NEWTRTN then
            do;

                  if not missing(NEWTRTN) then
                        do;
                              _trt=_trt + 1;
                        end;

                  *-----;
                  * Generate _STR as the treatment label ;
                  *-----;
                  _str=NEWTRT;
                  *-----;
                  * Update _TRTLB&n with generated treatment label ;
                  *-----;

                  if _trt > 0 then
                        call symput('_trtlb'||compress(put(_trt, 4.)), trim(left(_str)));
            end;
run;

```



```

*-----;
* Count number of patients in each treatment. ;
*-----;

proc sql noprint;
  select compress(put(count(*), 5.)) into :_trt1 - :_trt3 from (select distinct
    USUBJID, _trt from _data1 where NEWTRTN is not missing) group by _trt;
  select compress(put(count(*), 5.)) into :_trt4 from (select distinct USUBJID
    from _data1 where NEWTRTN is not missing);
quit;

*-----;
* Generate a dataset containing all by-variables ;
*-----;

proc sort data=_data1 out=_bydat1(keep=_datasrt _dummyby) nodupkey;
  by _datasrt;
run;

data _bydat1;
  set _bydat1 end=eof;
  by _datasrt;
  retain _preby 0;
  drop _preby;
  _byvar1=0;

  if eof then
    do;
      call symput("_preby1", compress(put(_byvar1, 4.)));

      if 0=0 then
        output;
    end;
run;

data _bydat1;
  set _bydat1;
  by _datasrt;
  length _bycol _byindnt $50 _bylast $10;
  _bycol=" ";
  _byindnt=" ";
  _bylast=" ";
run;

proc sort data=_bydat1;
  by _datasrt;
run;

proc sort data=_data1 out=_data1;
  by _datasrt;
run;

data _null_ ;
  set _data1 end=eof;

```

```

    if eof then
        call symput('dptlab', vlabel(SEXN));
run;

data _anall;
    length SEXN 8;
    set _data1;

    if SEXN=. then
        SEXN=9998;
    _blcksrt=1;
    _cnt=1;
    _cat=1;

    if _trt <=0 then
        delete;
    output;
run;

proc sort data=_anall;
    by _datasrt _blcksrt SEXN _trt _cat;
run;

*--- Counts for each by-sequence, dependant var, and treatment combination ---*;

data _temp1;
    set _anall;
    output;
run;

proc sort data=_temp1 out=_temp91 nodupkey;
    by _datasrt _blcksrt _cat SEXN _trt USUBJID;
run;

proc freq data=_temp91;
    format SEXN;
    tables _datasrt*_blcksrt*_cat * SEXN * _trt / sparse norow nocol nopercnt
        out=_pct1(drop=percent);
run;

proc sort data=_anall out=_denom1(keep=_datasrt _cat) nodupkey;
    by _datasrt _cat;
run;

data _denom1;
    set _denom1;
    by _datasrt _cat;
    label count='count';
    _trt=1;
    count=&_trt1;
    output;
    _trt=2;
    count=&_trt2;

```

```

output;
  _trt=3;
count=&_trt3;
output;
run;

*-----;
* Create _DENOMF a frame dataset for the denominators ;
*-----;

data _denomf1;
  _datasrt=1;
set _bydat1(keep=);
  * All treatment groups ;
  _trt1=0;
  _trt2=0;
  _trt3=0;
  * _CAT is the subgroup variable ;
  _cat=1;
output;
run;

*-----;
* Transpose _DENOM into _DENOMIN to get COUNT as _TRTn columns ;
*-----;

proc sql noprint;
  select put(nobs - delobs, 12.) into :_nobs from dictionary.tables
    where (libname="WORK" and memname="_DENOM1");
  select setting into :miss from dictionary.options where
    upcase(optname)="MISSING";
quit;

proc transpose data=_denom1 out=_denomin1(drop=_name__label_) prefix=_trt;
  by _datasrt _cat;
  var count;
  id _trt;
run;

*-----;
* VALRANGE=FULL. Create full rank categories WITHOUT using where. ;
*-----;

proc sql noprint;
  select count(distinct SEXN) into :totexpv from _anal1;
  select distinct SEXN into :expv1 - :expv2 from _anal1 order by SEXN;
quit;

*-----;
* Create _FRAME dataset using all combinations of category variable ;
*-----;

data _frame1;
  _datasrt=1;

```

```

set _bydat1(keep=);
  _blcksrt=1;
length SEXN 8;
  _catLbl=" ";
  _trt=1;
SEXN=1;
  _catord=1;
  _cat=1;
output;
  _trt=2;
SEXN=1;
  _catord=1;
  _cat=1;
output;
  _trt=3;
SEXN=1;
  _catord=1;
  _cat=1;
output;
  _catLbl=" ";
  _trt=1;
SEXN=2;
  _catord=2;
  _cat=1;
output;
  _trt=2;
SEXN=2;
  _catord=2;
  _cat=1;
output;
  _trt=3;
SEXN=2;
  _catord=2;
  _cat=1;
output;
run;

*-----;
* Merge the _PCT dataset with its frameup dataset(_FRAME) ;
*-----;

proc sort data=_frame1;
  by _datasrt _blcksrt _cat SEXN _trt;
run;

proc sort data=_pct1;
  by _datasrt _blcksrt _cat SEXN _trt;
run;

data _pct1;
  merge _frame1(in=_inframe) _pct1;
  by _datasrt _blcksrt _cat SEXN _trt;

  if _inframe;

```

```

    if count=. then
        count=0;
run;

*-----;
* Delete Zero filled MISSING category rows for each combination of;
* _datasrt & _byvar _blcksrt;
*-----;

proc sort data=_pct1;
    by _datasrt _blcksrt SEXN;
run;

data _miss1(keep=_datasrt _blcksrt SEXN totcount);
    set _pct1;
    where SEXN=9998;
    retain totcount;
    by _datasrt _blcksrt SEXN;

    if first.SEXN then
        totcount=0;
    totcount=totcount+count;

    if last.SEXN;
run;

data _pct1(drop=totcount);
    merge _pct1 _miss1;
    by _datasrt _blcksrt SEXN;

    if totcount=0 then
        delete;
run;

*****;
*IF PCTDISP=CAT/DPTVAR then add dptvar into denominator frame dataset;
*****;
*-----;
* Merge the _DENOMIN with its frame up dataset (_denomf) ;
*-----;

proc sort data=_denomf1;
    by _datasrt _cat;
run;

proc sort data=_denomin1;
    by _datasrt _cat;
run;

data _denomin1;
    merge _denomf1(in=_inframe) _denomin1;
    by _datasrt _cat;

```

```

    if _inframe;
    _blcksrt=1;
run;

*-----;
* Merge in _PCT(counts) with the _DENOMIN(denominator for percents) ;
*-----;

proc sort data=_pct1;
    by _datasrt _cat;
run;

*-----;
* Create _VARNAME variable to hold depend variable name. ;
* Create _VRLABEL variable to display Group label. ;
* Create _RWLABEL variable to display &dptvar categories. ;
*-----;

data _pct1;
    if 0 then
        set _basetemplate;
    merge _denomin1(in=_a) _pct1;
    by _datasrt _cat;

    if _a;
    _varname="SEXN ";
    _vrlabel="Sex ";
    _rwlable=put(SEXN, sex.);

    if SEXN=9998 then
        do;
            _rwlable="Unknown ";
            _catord=9998;
        end;
    else if SEXN=9999 then
        do;
            _rwlable="Total ";
            _catord=9999;
        end;

    if _catord=. then
        _catord=9997;
run;

proc sort data=_pct1;
    by _datasrt _blcksrt _catord SEXN _trt _cat;
run;

*-----;
* Create _CVALUE variable to display results. ;
* Create _ROWSRT variable to order results. ;
*-----;

data _base1;

```

```

length _catlabl $200;
set _pct1 end=eof;
by _datasrt _blcksrt _catord SEXN _trt _cat;
retain _rowsrt 0 _rowmax 0;
array _trctcnt(*) _trt1- _trt4;
drop _rowmax _cpct;
length _cpct $100;
_cpct=' ';
_module='mcatstat';

if count > . then
    _cvalue=put(count, 5.);
else
    _cvalue=put(0, 5.);
*-----;
* Format percent to append to display value in _CVALUE ;
*-----;

if _trt ne . then
    do;

        if _trctcnt(_trt) > 0 then
            do;
                percent=count / _trctcnt(_trt) * 100;

                if percent > 0 then
                    do;

                        if round(percent, 0.1) GE 0.1 then
                            _cpct="(*ESC*){nbspspace 1}("||strip(put(percent, 5.1))||")";
                        else
                            _cpct="(*ESC*){nbspspace 1}(0.0)";
                        _cvalue=trim(_cvalue)||_cpct;
                    end;
                end;
            end;
        end;

    end;

if length(_cvalue) < 13 then
    do;
        *-----;
        * Put character A0x at right most character to pad text;
        *-----;
        substr(_cvalue, 13, 1)='A0'x;
    end;

if first.SEXN then
    do;
        _rowsrt=_rowsrt + 1;
        _rowmax=max(_rowsrt, _rowmax);
    end;
_datatyp='data';
_indent=0;
_dptindt=0;
_vorder=1;

```

```

_rowjump=1;

if upcase(_rwlabel)='_NONE_' then
    _rwlabel=' ';
_indent=3;
_dptindt=0;

if _trt=3 +1 then
    _trt=9999;

if eof then
    call symput('_rowsrt', compress(put(_rowmax, 4.)));
_direct="TOP ";
_p=2;
run;

data _null_;
set _data1 end=eof;

if eof then
    call symput('dptlab', vlabel(ARACEN));
run;

data _anal2;
length ARACEN 8;
set _data1;
where same and ARACEN is not missing;
_blcksrt=2;
_cnt=1;
_cat=1;

if _trt <=0 then
    delete;
output;
run;

proc sort data=_anal2;
by _datasrt _blcksrt ARACEN _trt _cat;
run;

*--- Counts for each by-sequence, dependant var, and treatment combination ---*;

data _temp2;
set _anal2;
output;
run;

proc sort data=_temp2 out=_temp92 nodupkey;
by _datasrt _blcksrt _cat ARACEN _trt USUBJID;
run;

proc freq data=_temp92;
format ARACEN;
tables _datasrt*_blcksrt*_cat * ARACEN * _trt / sparse norow nocol nopercnt

```

FDA-CBER-2022-5812-0072472


```

        out=_pct2(drop=percent);
run;

proc sort data=_anal2 out=_denom2(keep=_datasrt _cat) nodupkey;
    by _datasrt _cat;
run;

data _denom2;
    set _denom2;
    by _datasrt _cat;
    label count='count';
    _trt=1;
    count=&_trt1;
    output;
    _trt=2;
    count=&_trt2;
    output;
    _trt=3;
    count=&_trt3;
    output;
run;

*-----;
* Create _DENOMF a frame dataset for the denominators ;
*-----;

data _denomf2;
    _datasrt=1;
    set _bydat1(keep=);
    * All treatment groups ;
    _trt1=0;
    _trt2=0;
    _trt3=0;
    * _CAT is the subgroup variable ;
    _cat=1;
    output;
run;

*-----;
* Transpose _DENOM into _DENOMIN to get COUNT as _TRTn columns ;
*-----;

proc sql noprint;
    select put(nobs - delobs, 12.) into :_nobs from dictionary.tables
        where (libname="WORK" and memname="_DENOM2");
    select setting into :miss from dictionary.options where
        upcase(optname)="MISSING";
quit;

proc transpose data=_denom2 out=_denomin2(drop=_name _label_) prefix=_trt;
    by _datasrt _cat;
    var count;
    id _trt;
run;

```

```
*-----;  
* Create _FRAME dataset using all combinations of category variable ;  
*-----;
```

```
data _frame2;  
  _datasrt=1;  
  set _bydat1(keep=);  
  _blcksrt=2;  
  length ARACEN 8;  
  _catLabl=" ";  
  _trt=1;  
  ARACEN=1;  
  _catord=1;  
  _cat=1;  
  output;  
  _trt=2;  
  ARACEN=1;  
  _catord=1;  
  _cat=1;  
  output;  
  _trt=3;  
  ARACEN=1;  
  _catord=1;  
  _cat=1;  
  output;  
  _catLabl=" ";  
  _trt=1;  
  ARACEN=2;  
  _catord=2;  
  _cat=1;  
  output;  
  _trt=2;  
  ARACEN=2;  
  _catord=2;  
  _cat=1;  
  output;  
  _trt=3;  
  ARACEN=2;  
  _catord=2;  
  _cat=1;  
  output;  
  _catLabl=" ";  
  _trt=1;  
  ARACEN=2.5;  
  _catord=3;  
  _cat=1;  
  output;  
  _trt=2;  
  ARACEN=2.5;  
  _catord=3;  
  _cat=1;  
  output;  
  _trt=3;
```

```
ARACEN=2.5;
_catord=3;
_cat=1;
output;
_catLbl=" ";
_trt=1;
ARACEN=3;
_catord=4;
_cat=1;
output;
_trt=2;
ARACEN=3;
_catord=4;
_cat=1;
output;
_trt=3;
ARACEN=3;
_catord=4;
_cat=1;
output;
_catLbl=" ";
_trt=1;
ARACEN=4;
_catord=5;
_cat=1;
output;
_trt=2;
ARACEN=4;
_catord=5;
_cat=1;
output;
_trt=3;
ARACEN=4;
_catord=5;
_cat=1;
output;
_catLbl=" ";
_trt=1;
ARACEN=5;
_catord=6;
_cat=1;
output;
_trt=2;
ARACEN=5;
_catord=6;
_cat=1;
output;
_trt=3;
ARACEN=5;
_catord=6;
_cat=1;
output;
_catLbl=" ";
_trt=1;
```

```

ARACEN=6;
_catord=7;
_cat=1;
output;
_trt=2;
ARACEN=6;
_catord=7;
_cat=1;
output;
_trt=3;
ARACEN=6;
_catord=7;
_cat=1;
output;
_catLabl=" ";
_trt=1;
ARACEN=7;
_catord=8;
_cat=1;
output;
_trt=2;
ARACEN=7;
_catord=8;
_cat=1;
output;
_trt=3;
ARACEN=7;
_catord=8;
_cat=1;
output;
run;

*-----;
* Merge the _PCT dataset with its frameup dataset(_FRAME) ;
*-----;

proc sort data=_frame2;
  by _datasrt _blcksrt _cat ARACEN _trt;
run;

proc sort data=_pct2;
  by _datasrt _blcksrt _cat ARACEN _trt;
run;

data _pct2;
  merge _frame2(in=_inframe) _pct2;
  by _datasrt _blcksrt _cat ARACEN _trt;

  if _inframe;

  if count=. then
    count=0;
run;

```

```

*-----;
* Delete Zero filled MISSING category rows for each combination of;
* _datasrt & _blcksrt;
*-----;

proc sort data=_pct2;
  by _datasrt _blcksrt ARACEN;
run;

data _miss2(keep=_datasrt _blcksrt ARACEN totcount);
  set _pct2;
  where ARACEN=9998;
  retain totcount;
  by _datasrt _blcksrt ARACEN;

  if first.ARACEN then
    totcount=0;
  totcount=totcount+count;

  if last.ARACEN;
run;

data _pct2(drop=totcount);
  merge _pct2 _miss2;
  by _datasrt _blcksrt ARACEN;

  if totcount=0 then
    delete;
run;

proc sort data=_denomf2;
  by _datasrt _cat;
run;

proc sort data=_denomin2;
  by _datasrt _cat;
run;

data _denomin2;
  merge _denomf2(in=_inframe) _denomin2;
  by _datasrt _cat;

  if _inframe;
  _blcksrt=2;
run;

*-----;
* Merge in _PCT(counts) with the _DENOMIN(denominator for percents) ;
*-----;

proc sort data=_pct2;
  by _datasrt _cat;
run;

```

```

data _pct2;
  if 0 then
    set _basetemplate;
  merge _denomin2(in=_a) _pct2;
  by _datasrt _cat;

  if _a;
  _varname="ARACEN ";
  _vrlabel="Race ";
  _rwlabel=put(ARACEN, crace.);

  if ARACEN=9998 then
    do;
      _rwlabel="Missing ";
      _catord=9998;
    end;
  else if ARACEN=9999 then
    do;
      _rwlabel="Total ";
      _catord=9999;
    end;

  if _catord=. then
    _catord=9997;
run;

proc sort data=_pct2;
  by _datasrt _blcksrt _catord ARACEN _trt _cat;
run;

*-----;
* Create _CVALUE variable to display results. ;
* Create _ROWSRT variable to order results. ;
*-----;

data _base2;
  length _catlabl $200;
  set _pct2 end=eof;
  by _datasrt _blcksrt _catord ARACEN _trt _cat;
  retain _rowsrt 0 _rowmax 0;
  array _trcnt(*) _trt1- _trt4;
  drop _rowmax _cpct;
  length _cpct $100;
  _cpct=' ';
  _module='mcatstat';

  if count > . then
    _cvalue=put(count, 5.);
  else
    _cvalue=put(0, 5.);
  *-----;
  * Format percent to append to display value in _CVALUE ;
  *-----;

```

```

if _trt ne . then
  do;
    if _trtcnt(_trt) > 0 then
      do;
        percent=count / _trtcnt(_trt) * 100;
        if percent > 0 then
          do;
            if round(percent, 0.1) GE 0.1 then
              _cpct="(*ESC*){nbspspace 1}("||strip(put(percent, 5.1))||")";
            else
              _cpct="(*ESC*){nbspspace 1}(0.0)";
            _cvalue=trim(_cvalue)||_cpct;
          end;
        end;
      end;
    end;
  end;

```

```

if length(_cvalue) < 13 then
  do;
    *-----;
    * Put character A0x at right most character to pad text;
    *-----;
    substr(_cvalue, 13, 1)='A0'x;
  end;

```

```

if first.ARACEN then
  do;
    _rowsrt=_rowsrt + 1;
    _rowmax=max(_rowsrt, _rowmax);
  end;

```

```

_datatyp='data';
_indent=0;
_dptindt=0;
_vorder=1;
_rowjump=1;

```

```

if upcase(_rwlabel)='_NONE_' then
  _rwlabel=' ';
_indent=3;
_dptindt=0;

```

```

if _trt=3 +1 then
  _trt=9999;

```

```

if eof then
  call symput('_rowsrt', compress(put(_rowmax, 4)));
_direct="TOP ";
_p=2;

```

```
run;
```

```

*****
*SPECIFICATION 5 -1) Ethnicity - n and percent *;

```

*****,

```
data _anal3;
  length ETHNICN 8;
  set _data1;
  where same and ETHNICN is not missing;
  _blcksrt=3;
  _cnt=1;
  _cat=1;

  if _trt <=0 then
    delete;
  output;
run;

proc sort data=_anal3;
  by _datasrt _blcksrt ETHNICN _trt _cat;
run;

*--- Counts for each by-sequence, dependant var, and treatment combination ---*;

data _temp3;
  set _anal3;
  output;
run;

proc sort data=_temp3 out=_temp93 nodupkey;
  by _datasrt _blcksrt _cat ETHNICN _trt USUBJID;
run;

proc freq data=_temp93;
  format ETHNICN;
  tables _datasrt*_blcksrt*_cat * ETHNICN * _trt / sparse norow nocol nopercnt
    out=_pct3(drop=percent);
run;

proc sort data=_anal3 out=_denom3(keep=_datasrt _cat) nodupkey;
  by _datasrt _cat;
run;

data _denom3;
  set _denom3;
  by _datasrt _cat;
  label count='count';
  _trt=1;
  count=&_trt1;
  output;
  _trt=2;
  count=&_trt2;
  ;
  output;
  _trt=3;
  count=&_trt3;
  output;
```

FDA-CBER-2022-5812-0072480


```

run;

*-----;
* Create _DENOMF a frame dataset for the denominators ;
*-----;

data _denomf3;
  _datasrt=1;
  set _bydat1(keep=);
  * All treatment groups ;
  _trt1=0;
  _trt2=0;
  _trt3=0;
  * _CAT is the subgroup variable ;
  _cat=1;
  output;
run;

*-----;
* Transpose _DENOM into _DENOMIN to get COUNT as _TRTn columns ;
*-----;

proc transpose data=_denom3 out=_denomin3(drop=_name __label_) prefix=_trt;
  by _datasrt _cat;
  var count;
  id _trt;
run;

*-----;
* VALRANGE=FULL. Create full rank categories WITHOUT using where. ;
*-----;

proc sql noprint;
  select count(distinct ETHNICN) into :totexpv from _anal3;
  select distinct ETHNICN into :expv1 - :expv3 from _anal3 order by ETHNICN;
quit;

*-----;
* Create _FRAME dataset using all combinations of category variable ;
*-----;

data _frame3;
  _datasrt=1;
  set _bydat1(keep=);
  _blcksrt=3;
  length ETHNICN 8;
  _catLbl=" ";
  _trt=1;
  ETHNICN=1;
  _catord=1;
  _cat=1;
  output;
  _trt=2;
  ETHNICN=1;

```

```

    _catord=1;
    _cat=1;
output;
    _trt=3;
ETHNICN=1;
    _catord=1;
    _cat=1;
output;
    _catLbl=" ";
    _trt=1;
ETHNICN=2;
    _catord=2;
    _cat=1;
output;
    _trt=2;
ETHNICN=2;
    _catord=2;
    _cat=1;
output;
    _trt=3;
ETHNICN=2;
    _catord=2;
    _cat=1;
output;
    _catLbl=" ";
    _trt=1;
ETHNICN=3;
    _catord=3;
    _cat=1;
output;
    _trt=2;
ETHNICN=3;
    _catord=3;
    _cat=1;
output;
    _trt=3;
ETHNICN=3;
    _catord=3;
    _cat=1;
output;
run;

*-----;
* Merge the _PCT dataset with its frameup dataset(_FRAME) ;
*-----;

proc sort data=_frame3;
    by _datasrt _blcksrt _cat ETHNICN _trt;
run;

proc sort data=_pct3;
    by _datasrt _blcksrt _cat ETHNICN _trt;
run;

```

```

data _pct3;
  merge _frame3(in=_inframe) _pct3;
  by _datasrt _blcksrt _cat ETHNICN _trt;

  if _inframe;

  if count=. then
    count=0;
run;

proc sort data=_pct3;
  by _datasrt _blcksrt ETHNICN;
run;

data _miss3(keep=_datasrt _blcksrt ETHNICN totcount);
  set _pct3;
  where ETHNICN=9998;
  retain totcount;
  by _datasrt _blcksrt ETHNICN;

  if first.ETHNICN then
    totcount=0;
  totcount=totcount+count;

  if last.ETHNICN;
run;

data _pct3(drop=totcount);
  merge _pct3 _miss3;
  by _datasrt _blcksrt ETHNICN;

  if totcount=0 then
    delete;
run;

proc sort data=_denomf3;
  by _datasrt _cat;
run;

proc sort data=_denomin3;
  by _datasrt _cat;
run;

data _denomin3;
  merge _denomf3(in=_inframe) _denomin3;
  by _datasrt _cat;

  if _inframe;
  _blcksrt=3;
run;

proc sort data=_pct3;
  by _datasrt _cat;
run;

```

```

data _pct3;
  if 0 then
    set _basetemplate;
  merge _denomin3(in=_a) _pct3;
  by _datasrt _cat;

  if _a;
  _varname="ETHNICN ";
  _vrlabel="Ethnicity ";
  _rwlabel=put(ETHNICN, ethnic.);

  if ETHNICN=9998 then
    do;
      _rwlabel="Missing ";
      _catord=9998;
    end;
  else if ETHNICN=9999 then
    do;
      _rwlabel="Total ";
      _catord=9999;
    end;

  if _catord=. then
    _catord=9997;
run;

proc sort data=_pct3;
  by _datasrt _blcksrt _catord ETHNICN _trt _cat;
run;

*-----;
* Create _CVALUE variable to display results. ;
* Create _ROWSRT variable to order results. ;
*-----;

data _base3;
  length _catlabl $200;
  set _pct3 end=eof;
  by _datasrt _blcksrt _catord ETHNICN _trt _cat;
  retain _rowsrt 0 _rowmax 0;
  array _trcnt(*) _trt1- _trt4;
  drop _rowmax _cpct;
  length _cpct $100;
  _cpct='';
  _module='mcatstat';

  if count > . then
    _cvalue=put(count, 5.);
  else
    _cvalue=put(0, 5.);
  *-----;
  * Format percent to append to display value in _CVALUE ;
  *-----;

```

```

if _trt ne . then
  do;

      if _trtcnt(_trt) > 0 then
        do;
          percent=count / _trtcnt(_trt) * 100;

          if percent > 0 then
            do;

              if round(percent, 0.1) GE 0.1 then
                _cpct="(*ESC*){nbspspace 1}("||strip(put(percent, 5.1))||")";
              else
                _cpct="(*ESC*){nbspspace 1}(0.0)";
              _cvalue=trim(_cvalue)||_cpct;
            end;
          end;
        end;

      end;

/* if length(_cvalue) < 13 then do; */
/* *-----; */
/* * Put character A0x at right most character to pad text; */
/* *-----; */
/* substr(_cvalue,13,1)= 'A0'x ; */
/* end; */
if first.ETHNICN then
  do;
    _rowsrt=_rowsrt + 1;
    _rowmax=max(_rowsrt, _rowmax);
  end;
  _datatyp='data';
  _indent=0;
  _dptindt=0;
  _vorder=1;
  _rowjump=1;

if upcase(_rlabel)='_NONE_' then
  _rlabel=' ';
  _indent=3;
  _dptindt=0;

if _trt=3 +1 then
  _trt=9999;

if eof then
  call symput('_rowsrt', compress(put(_rowmax, 4.)));
  _direct="TOP ";
  _p=2;
run;

data _anal4;
  length COUNTRYX $50;
  set _data1;

```

```

where same and COUNTRYX is not missing;
  _blcksrt=4;
  _cnt=1;
  _cat=1;

if _trt <=0 then
  delete;
output;
run;

proc sort data=_anal4;
  by _datasrt _blcksrt COUNTRYX _trt _cat;
run;

*--- Counts for each by-sequence, dependant var, and treatment combination ---*;

data _temp4;
  set _anal4;
  output;
run;

proc sort data=_temp4 out=_temp94 nodupkey;
  by _datasrt _blcksrt _cat COUNTRYX _trt USUBJID;
run;

proc freq data=_temp94;
  format COUNTRYX;
  tables _datasrt*_blcksrt*_cat * COUNTRYX * _trt / sparse norow nocol nopercnt
  out=_pct4(drop=percent);
run;

proc sort data=_anal4 out=_denom4(keep=_datasrt _cat) nodupkey;
  by _datasrt _cat;
run;

data _denom4;
  set _denom4;
  by _datasrt _cat;
  label count='count';
  _trt=1;
  count=&_trt1;
  output;
  _trt=2;
  count=&_trt2;
  output;
  _trt=3;
  count=&_trt3;
  output;
run;

*-----;
* Create _DENOMF a frame dataset for the denominators ;
*-----;

```

```

data _denomf4;
  _datasrt=1;
  set _bydat1(keep=);
  * All treatment groups ;
  _trt1=0;
  _trt2=0;
  _trt3=0;
  * _CAT is the subgroup variable ;
  _cat=1;
  output;
run;

*-----;
* Transpose _DENOM into _DENOMIN to get COUNT as _TRTn columns ;
*-----;

proc transpose data=_denomf4 out=_denomin4(drop=_name __label_) prefix=_trt;
  by _datasrt _cat;
  var count;
  id _trt;
run;

proc sort data=_pct4 out=_expv4 (keep=_datasrt _blcksrt COUNTRYX) nodupkey;
  by _datasrt _blcksrt COUNTRYX;
run;

proc sql noprint;
  select put(nobs - delobs, 12.) into :_nobs from dictionary.tables
         where (libname="WORK" and memname="_PCT4");
  select setting into :miss from dictionary.options where
         upcase(optname)="MISSING";
quit;

proc sort data=_expv4;
  by _datasrt _blcksrt COUNTRYX;
run;

data _frame4;
  set _expv4;
  by _datasrt _blcksrt COUNTRYX;

  if first._blcksrt then
    _catord=0;
  _catord + 1;
  _trt=1;
  _cat=1;
  output;
  _trt=2;
  _cat=1;
  output;
  _trt=3;
  _cat=1;
  output;
run;

```

```
*-----;  
* Merge the _PCT dataset with its frameup dataset(_FRAME) ;  
*-----;
```

```
proc sort data=_frame4;  
  by _datasrt _blcksrt _cat COUNTRYX _trt;  
run;
```

```
proc sort data=_pct4;  
  by _datasrt _blcksrt _cat COUNTRYX _trt;  
run;
```

```
data _pct4;  
  merge _frame4(in=_inframe) _pct4;  
  by _datasrt _blcksrt _cat COUNTRYX _trt;
```

```
  if _inframe;
```

```
  if count=. then  
    count=0;
```

```
run;
```

```
proc sort data=_pct4;  
  by _datasrt _blcksrt COUNTRYX;  
run;
```

```
data _miss4(keep=_datasrt _blcksrt COUNTRYX totcount);  
  set _pct4;  
  where COUNTRYX='ZZZY';  
  retain totcount;  
  by _datasrt _blcksrt COUNTRYX;
```

```
  if first.COUNTRYX then  
    totcount=0;  
  totcount=totcount+count;
```

```
  if last.COUNTRYX;
```

```
run;
```

```
data _pct4(drop=totcount);  
  merge _pct4 _miss4;  
  by _datasrt _blcksrt COUNTRYX;
```

```
  if totcount=0 then  
    delete;
```

```
run;
```

```
proc sort data=_denomf4;  
  by _datasrt _cat;  
run;
```

```
proc sort data=_denomin4;  
  by _datasrt _cat;
```



```

run;

data _denomin4;
  merge _denomf4(in=_inframe) _denomin4;
  by _datasrt _cat;

  if _inframe;
  _blcksrt=4;
run;

proc sort data=_pct4;
  by _datasrt _cat;
run;

data _pct4;
  if 0 then
    set _basetemplate;
  merge _denomin4(in=_a) _pct4;
  by _datasrt _cat;

  if _a;
  _varname="COUNTRYX ";
  _vrlabel="Country ";
  _rwlabel=COUNTRYX;

  if COUNTRYX='ZZZY' then
    do;
      _rwlabel="Missing ";
      _catord=9998;
    end;
  else if COUNTRYX='ZZZZ' then
    do;
      _rwlabel="Total ";
      _catord=9999;
    end;

  if _catord=. then
    _catord=9997;
run;

proc sort data=_pct4;
  by _datasrt _blcksrt _catord COUNTRYX _trt _cat;
run;

*-----;
* Create _CVALUE variable to display results. ;
* Create _ROWSRT variable to order results. ;
*-----;

data _base4;
  length _catlabl $200;
  set _pct4 end=eof;
  by _datasrt _blcksrt _catord COUNTRYX _trt _cat;
  retain _rowsrt 0 _rowmax 0;

```

```

array _trcnt(*) _tr1-_trt4;
drop _rowmax _cpct;
length _cpct $100;
_cpct='';
_module='mcatstat';

if count > . then
    _cvalue=put(count, 5.);
else
    _cvalue=put(0, 5.);
*-----;
* Format percent to append to display value in _CVALUE ;
*-----;

if _trt ne . then
    do;

        if _trcnt(_trt) > 0 then
            do;
                percent=count / _trcnt(_trt) * 100;

                if percent > 0 then
                    do;

                        if round(percent, 0.1) GE 0.1 then
                            _cpct="(*ESC*){nbspspace 1}("||strip(put(percent, 5.1))||")";
                        else
                            _cpct="(*ESC*){nbspspace 1}(0.0)";
                        _cvalue=trim(_cvalue)||_cpct;
                    end;
                end;
            end;
        end;

if length(_cvalue) < 13 then
    do;
        *-----;
        * Put character A0x at right most character to pad text;
        *-----;
        substr(_cvalue, 13, 1)='A0'x;
    end;

if first.COUNTRYX then
    do;
        _rowsrt=_rowsrt + 1;
        _rowmax=max(_rowsrt, _rowmax);
    end;
_datatyp='data';
_indent=0;
_dptindt=0;
_vorder=1;
_rowjump=1;

if upcase(_rwlabel)='_NONE_' then
    _rwlabel='';

```

```

    _indent=3;
    _dptindt=0;

    if _trt=3 +1 then
        _trt=9999;

    if eof then
        call symput('_rowsrt', compress(put(_rowmax, 4.)));
        _direct="TOP ";
        _p=2;
run;

data _anal5;
    length COMBODFLNX 8;
    set _data1;
    where same and COMBODFLNX is not missing;
    _blcksrt=5;
    _cnt=1;
    _cat=1;

    if _trt <=0 then
        delete;
    output;
run;

proc sort data=_anal5;
    by _datasrt _blcksrt COMBODFLNX _trt _cat;
run;

*--- Counts for each by-sequence, dependant var, and treatment combination ---*;

data _temp5;
    set _anal5;
    output;
run;

proc sort data=_temp5 out=_temp95 nodupkey;
    by _datasrt _blcksrt _cat COMBODFLNX _trt USUBJID;
run;

proc freq data=_temp95;
    format COMBODFLNX;
    tables _datasrt*_blcksrt*_cat * COMBODFLNX * _trt / sparse norow nocol
        nopercnt out=_pct5(drop=percent);
run;

proc sort data=_anal5 out=_denom5(keep=_datasrt _cat) nodupkey;
    ;
    by _datasrt _cat;
run;

data _denom5;
    set _denom5;
    by _datasrt _cat;

```

```

label count='count';
  _trt=1;
count=&_trt1;
output;
  _trt=2;
count=&_trt2;
output;
  _trt=3;
count=&_trt3;
output;
run;

data _denomf5;
  _datasrt=1;
set _bydat1(keep=);
* All treatment groups ;
  _trt1=0;
  _trt2=0;
  _trt3=0;
* _CAT is the subgroup variable ;
  _cat=1;
output;
run;

proc transpose data=_denom5 out=_denomin5(drop=_name __label_) prefix=_trt;
  by _datasrt _cat;
  var count;
  id _trt;
run;

proc sql noprint;
  select count(distinct COMBODFLNX) into :totexpv from _anal5;
  select distinct COMBODFLNX , COMBODFLX into :expv1 - :expv2 ,
         :catlab1 - :catlab2 from _anal5 order by COMBODFLNX;
quit;

data _frame5;
  _datasrt=1;
set _bydat1(keep=);
  _blcksrt=5;
length COMBODFLNX 8;
length _catLabl $50;
  _catLabl=' ';
  _catLabl="Yes ";
  _trt=1;
COMBODFLNX=1;
  _catord=1;
  _cat=1;
output;
  _trt=2;
COMBODFLNX=1;
  _catord=1;
  _cat=1;
output;

```

```

    _trt=3;
    COMBODFLNX=1;
    _catord=1;
    _cat=1;
    output;
    _catLbl="No ";
    _trt=1;
    COMBODFLNX=2;
    _catord=2;
    _cat=1;
    output;
    _trt=2;
    COMBODFLNX=2;
    _catord=2;
    _cat=1;
    output;
    _trt=3;
    COMBODFLNX=2;
    _catord=2;
    _cat=1;
    output;
run;

proc sort data=_frame5;
    by _datasrt _blcksrt _cat COMBODFLNX _trt;
run;

proc sort data=_pct5;
    by _datasrt _blcksrt _cat COMBODFLNX _trt;
run;

data _pct5;
    merge _frame5(in=_inframe) _pct5;
    by _datasrt _blcksrt _cat COMBODFLNX _trt;

    if _inframe;

    if count=. then
        count=0;
run;

proc sort data=_pct5;
    by _datasrt _blcksrt COMBODFLNX;
run;

data _miss5(keep=_datasrt _blcksrt COMBODFLNX totcount);
    set _pct5;
    where COMBODFLNX=9998;
    retain totcount;
    by _datasrt _blcksrt COMBODFLNX;

    if first.COMBODFLNX then
        totcount=0;
    totcount=totcount+count;

```

```

    if last.COMBODFLNX;
run;

data _pct5(drop=totcount);
    merge _pct5 _miss5;
    by _datasrt _blcksrt COMBODFLNX;

    if totcount=0 then
        delete;
run;

proc sort data=_denomf5;
    by _datasrt _cat;
run;

proc sort data=_denomin5;
    by _datasrt _cat;
run;

data _denomin5;
    merge _denomf5(in=_inframe) _denomin5;
    by _datasrt _cat;

    if _inframe;
        _blcksrt=5;
run;

proc sort data=_pct5;
    by _datasrt _cat;
run;

data _pct5;
    if 0 then
        set _basetemplate;
    merge _denomin5(in=_a) _pct5;
    by _datasrt _cat;

    if _a;
        _varname="COMBODFLNX ";
        _vrlabel="Comorbidities(*ESC*){super c} ";
        _rwlable=_catLabl;

    if COMBODFLNX=9998 then
        do;
            _rwlable="Missing ";
            _catord=9998;
        end;
    else if COMBODFLNX=9999 then
        do;
            _rwlable="Total ";
            _catord=9999;
        end;
end;

```

```

if _catord=. then
    _catord=9997;
run;

proc sort data=_pct5;
    by _datasrt _blcksrt _catord COMBODFLNX _trt _cat;
run;

data _base5;
    length _catlabl $200;
    set _pct5 end=eof;
    by _datasrt _blcksrt _catord COMBODFLNX _trt _cat;
    retain _rowsrt 0 _rowmax 0;
    array _trtcnt(*) _trt1- _trt4;
    drop _rowmax _cpct;
    length _cpct $100;
    _cpct=' ';
    _module='mcatstat';

    if count > . then
        _cvalue=put(count, 5.);
    else
        _cvalue=put(0, 5.);

    if _trt ne . then
        do;

            if _trtcnt(_trt) > 0 then
                do;
                    percent=count / _trtcnt(_trt) * 100;

                    if percent > 0 then
                        do;

                            if round(percent, 0.1) GE 0.1 then
                                _cpct="(*ESC*){nbspspace 1}("||strip(put(percent, 5.1))||")";
                            else
                                _cpct="(*ESC*){nbspspace 1}(0.0)";
                            _cvalue=trim(_cvalue)||_cpct;
                        end;
                    end;
                end;

            end;

        do;

            if length(_cvalue) < 13 then
                do;
                    *-----;
                    * Put character A0x at right most character to pad text;
                    *-----;
                    substr(_cvalue, 13, 1)='A0'x;
                end;
            end;

        do;

            if first.COMBODFLNX then
                do;
                    _rowsrt=_rowsrt + 1;

```

```

        _rowmax=max(_rowsrt, _rowmax);
    end;
    _datatyp='data';
    _indent=0;
    _dptindt=0;
    _vorder=1;
    _rowjump=1;

    if upcase(_rwlabel)='_NONE_' then
        _rwlabel=' ';
        _indent=3;
        _dptindt=0;

    if _trt=3 +1 then
        _trt=9999;

    if eof then
        call symput('_rowsrt', compress(put(_rowmax, 4)));
        _direct="TOP ";
        _p=2;
run;

data _anal6;
    length OBSCATFN 8;
    set _data1;
    where same and OBSCATFN is not missing;
    _blcksrt=6;
    _cnt=1;
    _cat=1;

    if _trt <=0 then
        delete;
    output;
run;

proc sort data=_anal6;
    by _datasrt _blcksrt OBSCATFN _trt _cat;
run;

*--- Counts for each by-sequence, dependant var, and treatment combination ---*;

data _temp6;
    set _anal6;
    output;
run;

proc sort data=_temp6 out=_temp96 nodupkey;
    by _datasrt _blcksrt _cat OBSCATFN _trt USUBJID;
run;

proc freq data=_temp96;
    format OBSCATFN;
    tables _datasrt*_blcksrt*_cat * OBSCATFN * _trt / sparse norow nocol nopercnt
        out=_pct6(drop=percent);

```



```

run;

proc sort data=_anal6 out=_denom6(keep=_datasrt _cat) nodupkey;
  by _datasrt _cat;
run;

data _denom6;
  set _denom6;
  by _datasrt _cat;
  label count='count';
  _trt=1;
  count=&_trt1;
  output;
  _trt=2;
  count=&_trt2;
  output;
  _trt=3;
  count=&_trt3;
  output;
run;

data _denomf6;
  _datasrt=1;
  set _bydat1(keep=);
  * All treatment groups ;
  _trt1=0;
  _trt2=0;
  _trt3=0;
  * _CAT is the subgroup variable ;
  _cat=1;
  output;
run;

proc transpose data=_denom6 out=_denomin6(drop=_name __label_) prefix=_trt;
  by _datasrt _cat;
  var count;
  id _trt;
run;

proc sql noprint;
  select count(distinct OBSCATFN) into :totexpv from _anal6;
  select distinct OBSCATFN , OBSCATFL into :expv1 - :expv2 , :catlab1 - :catlab2
    from _anal6 order by OBSCATFN;
quit;

data _frame6;
  _datasrt=1;
  set _bydat1(keep=);
  _blcksrt=6;
  length OBSCATFN 8;
  length _catLabl $50;
  _catLabl=' ';
  _catLabl="Yes ";
  _trt=1;

```

```

OBSCATFN=1;
_catord=1;
_cat=1;
output;
_trt=2;
OBSCATFN=1;
_catord=1;
_cat=1;
output;
_trt=3;
OBSCATFN=1;
_catord=1;
_cat=1;
output;
_catLabl="No ";
_trt=1;
OBSCATFN=2;
_catord=2;
_cat=1;
output;
_trt=2;
OBSCATFN=2;
_catord=2;
_cat=1;
output;
_trt=3;
OBSCATFN=2;
_catord=2;
_cat=1;
output;
run;

proc sort data=_frame6;
  by _datasrt _blcksrt _cat OBSCATFN _trt;
run;

proc sort data=_pct6;
  by _datasrt _blcksrt _cat OBSCATFN _trt;
run;

data _pct6;
  merge _frame6(in=_inframe) _pct6;
  by _datasrt _blcksrt _cat OBSCATFN _trt;

  if _inframe;

  if count=. then
    count=0;
run;

proc sort data=_pct6;
  by _datasrt _blcksrt OBSCATFN;
run;

```

```

data _miss6(keep=_datasrt _blcksrt OBSCATFN totcount);
  set _pct6;
  where OBSCATFN=9998;
  retain totcount;
  by _datasrt _blcksrt OBSCATFN;

  if first.OBSCATFN then
    totcount=0;
  totcount=totcount+count;

  if last.OBSCATFN;
run;

data _pct6(drop=totcount);
  merge _pct6 _miss6;
  by _datasrt _blcksrt OBSCATFN;

  if totcount=0 then
    delete;
run;

proc sort data=_denomf6;
  by _datasrt _cat;
run;

proc sort data=_denomin6;
  by _datasrt _cat;
run;

data _denomin6;
  merge _denomf6(in=_inframe) _denomin6;
  by _datasrt _cat;

  if _inframe;
  _blcksrt=6;
run;

proc sort data=_pct6;
  by _datasrt _cat;
run;

data _pct6;
  if 0 then
    set _basetemplate;
  merge _denomin6(in=_a) _pct6;
  by _datasrt _cat;

  if _a;
  _varname="OBSCATFN ";
  _vrlabel="Obese(*ESC*){super d} ";
  _rwlabel=_catLbl;

  if OBSCATFN=9998 then
    do;

```

```

        _rwlabel="Missing ";
        _catord=9998;
    end;
else if OBSCATFN=9999 then
    do;
        _rwlabel="Total ";
        _catord=9999;
    end;

    if _catord=. then
        _catord=9997;
run;

proc sort data=_pct6;
    by _datasrt _blcksrt _catord OBSCATFN _trt _cat;
run;

data _base6;
    length _catlabl $200;
    set _pct6 end=eof;
    by _datasrt _blcksrt _catord OBSCATFN _trt _cat;
    retain _rowsrt 0 _rowmax 0;
    array _trtcnt(*) _trt1-_trt4;
    drop _rowmax _cpct;
    length _cpct $100;
    _cpct='';
    _module='mcatstat';

    if count > . then
        _cvalue=put(count, 5.);
    else
        _cvalue=put(0, 5.);

    if _trt ne . then
        do;

            if _trtcnt(_trt) > 0 then
                do;
                    percent=count / _trtcnt(_trt) * 100;

                    if percent > 0 then
                        do;

                            if round(percent, 0.1) GE 0.1 then
                                _cpct="(*ESC*){nbspspace 1}{||strip(put(percent, 5.1))||}";
                            else
                                _cpct="(*ESC*){nbspspace 1}{(0.0)";
                            _cvalue=trim(_cvalue)||_cpct;
                        end;
                    end;
                end;

            end;
        end;

    if length(_cvalue) < 13 then
        do;

```

```

*-----;
* Put character A0x at right most character to pad text;
*-----;
substr(_cvalue, 13, 1)='A0'x;
end;

if first.OBSCATFN then
  do;
    _rowsrt=_rowsrt + 1;
    _rowmax=max(_rowsrt, _rowmax);
  end;
  _datatype='data';
  _indent=0;
  _dptindt=0;
  _vorder=1;
  _rowjump=1;

  if upcase(_rwlabel)='_NONE_' then
    _rwlabel=' ';
    _indent=3;
    _dptindt=0;

  if _trt=3 +1 then
    _trt=9999;

  if eof then
    call symput('_rowsrt', compress(put(_rowmax, 4.)));
    _direct="TOP ";
    _p=2;
run;

*****
*SPECIFICATION 7 -1) Age - descriptive statistics *;
*****

data _anal7;
  set _data1;
  where _trt > 0;
  _blksrt=7;
  output;
run;

*-----;
* Make sure data is sorted by groups ;
*-----;

proc sort data=_anal7;
  by _datasrt _blksrt _trt;
run;

*-----;
* Call PROC UNIVARIATE to generate all possible statistics plus any ;
* Percentiles or Confidence Intervals. ;
*-----;

```

```

proc univariate data=_anal7 noprint;
  by _datasrt _blcksrt _trt;
  var AGETR01;
  output out=_msum7 CSS=CSS CV=CV KURTOSIS=KURTOSIS MAX=MAX MEAN=MEAN N=N
    MIN=MIN MODE=MODE RANGE=RANGE NMISS=NMISS NOBS=NOBS STDMEAN=STDMEAN
    SKEWNESS=SKEWNESS STD=STD USS=USS SUM=SUM VAR=VAR MEDIAN=MEDIAN P1=P1
P5=P5
    P10=P10 P90=P90 P95=P95 P99=P99 Q1=Q1 Q3=Q3 QRANGE=QRANGE GINI=GINI MAD=MAD
    QN=QN SN=SN STD_GINI=STD_GINI STD_MAD=STD_MAD STD_QN=STD_QN
    STD_QRANGE=STD_QRANGE STD_SN=STD_SN NORMAL=NORMAL PROBN=PROBN
MSIGN=MSIGN
    PROBM=PROBM SIGNRANK=SIGNRANK PROBS=PROBS T=T PROBT=PROBT;

```

```

run;

*-----;
*Create Frame dataset when user requested Subgrouping as well as set;
*sparsesgrpyn to Y to sparse subgrp categories of a format.;
*-----;

```

```

data _frame7;
  set _bydat1(keep=);
  _datasrt=1;
  _blcksrt=7;
  _catord=1;
  _trt=1;
  _cat=1;
  output;
  _trt=2;
  _cat=1;
  output;
  _trt=3;
  _cat=1;
  output;
run;

```

```

proc sort data=_frame7;
  by _datasrt _blcksrt _trt;
run;

```

```

data _msum7;
  merge _msum7 _frame7;
  by _datasrt _blcksrt _trt;
run;

```

```

*-----;
* Generate _result1 from OUT= dataset of PROC UNIVARIATE ;
*-----;

```

```

data _result1_7;
  if 0 then
    set _basetemplate;
  set _msum7 end=eof;
  _rowsrt=0 + 1;

```

```

    _rlabel="Mean (SD) ";
    _cvalue=' ';
    _nvalue=.;
    *-----;
    * MEAN(STD) ;
    *-----;

if mean ne . and std ne . then
    do;
        _cValue=strip(put(mean, 5.1) ) || ' (' || strip(put(std, 5.2) ) || ')';
    end;
else if mean eq . then
    _cValue="-" || ' (' || "-" || ')';
else if std eq . then
    do;
        _cValue=strip(put(mean, 5.1) ) || ' (' || "NE" || ')';
    end;
output;
    _rowsrt=0 + 2;
    _rlabel="Median ";
    _cvalue=' ';
    _nvalue=.;
    _nvalue=MEDIAN;

if MEDIAN ne . then
    _cValue=strip(put(MEDIAN, 5.1) );
else
    _cValue="-";
output;
    _rowsrt=0 + 3;
    _rlabel="Min, max ";
    _cvalue=' ';
    _nvalue=.;
    *-----;
    * MINMAX MINMAXC MEDIAN(MINMAX) MEDIAN(MINMAXC) ;
    *-----;
    _cValue=' ';

if min ^=. & max ^=. then
    do;
        _cValue=trim(_cvalue) || ' (' || strip(put(min, 5.0)
            ) || ', ' || strip(put(max, 5.0) ) || ')';
    end;
else if min=. & max=. then
    do;
        _cValue=trim(_cvalue) || ' (' || "-" || ', ' || "-" || ')';
    end;
    _cValue=compbl(_cValue);
output;

run;

*-----;
* Generate _logresult1 from OUT= dataset of PROC UNIVARIATE for log stats;
*-----;

```

```

data _logresult1_7;
  if 0 then
    set _basetemplate;
  stop;
run;

*-----;
* Generate _result2 from confidence interval output dataset ;
*-----;

data _result2_7;
  if 0 then
    set _basetemplate;
  stop;
run;

*-----;
* Generate _logresult2 from confidence interval output dataset for log stats;
*-----;

data _logresult2_7;
  if 0 then
    set _basetemplate;
  stop;
run;

*-----;
* Combine to form one result dataset. Set variables that do not depend ;
* on the statistic. Sort the result. ;
*-----;

data _base7;
  set _result1_7 _result2_7 _logresult1_7 _logresult2_7;

  if _trt=4 then
    _trt=9999;
    _varname="AGETR01";
    _vrlabel="Age at vaccination (years) ";
    _datatyp='data';
    _module='msumstat';
    _indent=3;
    _rowjump=1;
    _dptindt=0;
run;

*-----;
* merge ISAM subgroup variables _SUBCAT _COLABEL ;
*-----;

proc sort data=_base7;
  by _datasrt _blcksrt _rowsrt;
run;

```



```
data _final;
  set _base1 _base2 _base3 _base4 _base5 _base6 _base7;
run;
```

```
proc sort data=_final;
  by _datasrt _blcksrt _rowsrt;
run;
```

```
*-----;
* At least one of TRT and STAT is vertical;
*-----;
```

```
data _final;
  set _final;
  drop __trt;

  if __trt=9999 then
    __trt=3 + 1;
  else
    __trt= __trt;

  if __trt=. then
    __trt=1;
    _column=__trt;

  if _column=9999 then
    _column=3 + 1;
run;
```

```
proc sort data=_final out=_final;
  by _datasrt _blcksrt _rowsrt _column;
run;
```

```
data _linecnt;
  set _final end=eof;
  by _datasrt _blcksrt _rowsrt _column;
  retain _totline _maxval _maxrow _rwlbttag _vrlbttag 0 _maxline _linecnt;
  keep _datasrt _blcksrt _totline _linecnt _maxrow;

  if _rowjump=. then
    _rowjump=1;

  if first._blcksrt then
    do;
      *-----;
      * Count words inside DATA step ;
      *-----;
      _token=repeat(' ', 99);
      _count=1;
      _token=scan(_vrlabel, _count, "|");

      if _token=: '_' then
        _tag=1;
      else
```

```

        _tag=0;

do while(_token ^=' ');
    _count=_count + 1;
    _token=scan(_vrlabel, _count, "|");
end;
_linecnt=_count - 1 + _tag;
_totline=_linecnt;

if _vrlabel ne '' and _vrlabel ne '^' & _datatype='data' then
    _vrlbtag=1;
end;

if first._rowsrt then
do;
    *-----;
    * Count words inside DATA step ;
    *-----;
    _token=repeat(' ', 99);
    _count=1;
    _token=scan(_rwlabel, _count, "|");

    if _token=: ' ' then
        _tag=1;
    else
        _tag=0;

    do while(_token ^=' ');
        _maxrow=max(_maxrow, length(_token) + _indent);
        _count=_count + 1;
        _token=scan(_rwlabel, _count, "|");
    end;
    _maxline=_count - 1 + _tag;

    if _rwlabel ne '' then
        _rwlbtag=1;
        _totline + _rowjump - 1;
end;
*-----;
* Count words inside DATA step ;
*-----;
_token=repeat(' ', 99);
_count=1;
_token=scan(_cvalue, _count, "|");

if _token=: ' ' then
    _tag=1;
else
    _tag=0;

do while(_token ^=' ');
    _maxval=max(_maxval, length(_token));
    _count=_count + 1;
    _token=scan(_cvalue, _count, "|");

```

```

end;
_ccnt=_count - 1 + _tag;
_maxline=max(_maxline, _ccnt);

if last._rowsrt then
  _totline=_maxline + _totline;

if last._blcksrt then
  do;
    _totline=_totline - _rowjump + 1;
  output;
  end;

if eof then
  do;
    call symput('_valwid', compress(put(_maxval, 3.)));
    call symput('_rwlbttag', put(_rwlbttag, 1.));
    call symput('_vrlbttag', put(_vrlbttag, 1.));
  end;

run;

data _final;
length _direct $20;
_direct=' ';
merge _final_linecnt;
by _datasrt _blcksrt;

run;

proc sql noprint;
create table rson as select distinct _trt, _column , _vrlabel as _rwlabel ,
  _datasrt, _blcksrt, (min(_rowsrt)-0.5) as _rowsrt , _dptindt as _indent , 0
as _dptindt from _final(where=( _vrlabel^=' ')) group by _trt, _column ,
  _datasrt, _blcksrt, _vrlabel;

quit;

data ADSL_DEMO_7D_PEDS_EVAL_EFF;
length _rvalue $800;
set _final rson end=eof;
_rwindt=sum(_indent, _dptindt);

if _rwindt <=0 then
  _rvalue=_rwlabel;

/* else _rvalue=repeat(byte(160),_rwindt-1)||_rwlabel; */
else
  _rvalue=repeat("~{nbspace 1}", _rwindt-1)||_rwlabel;
_dummy=1;

if _trt=. then
  _trt=1;

run;

proc sort data=ADSL_DEMO_7D_PEDS_EVAL_EFF;
by _datasrt _trt _blcksrt _rowsrt;

```

```

run;

data treat;
  length FMTNAME $8 start 8 label $200;
  fmtname='TREAT';

  do start=1 to 3 + ("N"="Y");
    label=symget('_TRTLB' || compress(put(start, 4.)));
    label=trim(label)
      || " | (N~{super a}=" || compress(symget("_TRT" || compress(put(start,
      4.)))) || ")"
  || "|n~{super b} (%)";
    output;
  end;
run;

proc format cntlin=treat;
run;

options orientation=LANDSCAPE papersize="LETTER";
ods escapechar="~";
title1 "Demographic Characteristics (*ESC*){unicode 2013} Blinded Placebo-Controlled Follow-up Period (*ESC*)
{unicode 2013} ";
title2 "Subjects 12 Through 15 Years of Age and Without Evidence of Infection Prior to 7 Days After Dose 2 (*ESC*)
{unicode 2013} ";
title3 "Evaluable Efficacy (7 Days) Population ";
footnote1 "a.(*ESC*){nbspspace 5}N = number of subjects in the specified group, or the total sample. This value is the
denominator for the percentage calculations. ";
footnote2 "b.(*ESC*){nbspspace 5}n = Number of subjects with the specified characteristic. ";
footnote3 "c.(*ESC*){nbspspace 5}Number of subjects who have 1 or more comorbidities that increase the risk of severe
COVID-19 disease: defined as subjects who had at least one of the Charlson comorbidity index category or BMI
(*ESC*){unicode 2265}95(*ESC*){super th} percentile. ";
footnote4 "d.(*ESC*){nbspspace 5}Obese is defined as BMI (*ESC*){unicode 2265}95(*ESC*){super th} percentile
from the growth chart. Refer to the CDC growth charts at
https://www.cdc.gov/growthcharts/html\_charts/bmiagerev.htm. ";

data outdata1;
  set ADSL_DEMO_7D_PEDS_EVAL_EFF;

  if upcase(_module)='MCATSTAT' then
    _cvalue=transtrn(compress(_cvalue), '(', ' ');
  _fixvar=1;
  _fix2var=1;
run;

option nobyline;

proc sort data=outdata1;
  by _datasrt _trt _blcksrt _rowsrt;
run;

proc sql noprint;
  select distinct start, label into :start1, :_trlb11 - :_trlb199 from treat
  order by start;

```

```

quit;

proc sort data=outdata1 out=_pre_transposed;
  by _fixvar _fix2var _datasrt _blcksrt _rowsrt _rvalue _trt;
run;

data _pre_transposed;
  set _pre_transposed;

  if _trt=9999 then
    _trt=3 +1;
run;

proc transpose data=_pre_transposed out=_column_transposed (drop=_name_)
  prefix=TRT;
  by _fixvar _fix2var _datasrt _blcksrt _rowsrt _rvalue;
  var _cvalue;
  id _trt;
run;

ods html file="&outtable.";

data REPORT;
  set _column_transposed;
  _dummy=1;
run;

proc sort data=report;
  by _datasrt _blcksrt _rowsrt _dummy;
run;

proc report data=report nowd list missing contents="" split="|" spanrows
  style(report)={} style(header)={} style(column)={};
  column _fixvar _fix2var _datasrt _blcksrt _rowsrt (" _rvalue)
    ("Vaccine Group (as Randomized)~{line}" TRT1 TRT2) TRT3 _dummy);
  define _fixvar / group noprint;
  define _fix2var / group noprint;
  define _datasrt / group order=internal noprint;
  define _blcksrt / group order=internal noprint;
  define _rowsrt / group order=internal noprint;
  define _rvalue / group id " " order=data style(column)={just=left width=60mm
    rightmargin=18px} style(header)={just=left} left;
  define _dummy / sum noprint;
  define TRT1 / group nozero "&_trlbl1." spacing=2 style(column)={width=35mm
    leftmargin=12px} style(header)={just=center} center;
  define TRT2 / group nozero "&_trlbl2." spacing=2 style(column)={width=35mm
    leftmargin=12px} style(header)={just=center} center;
  define TRT3 / group nozero "&_trlbl3." spacing=2 style(column)={width=35mm
    leftmargin=12px} style(header)={just=center} center;
  break before _fixvar / contents="" page;
  compute before _fix2var;
    line @1 " ~n ";
  endcomp;
  compute after _blcksrt;

```

```
        line " ~n ";
    endcomp;
run;

ods HTML close;

proc printto;
run;
```