ABSTRACT
Reviewing data is one of the most important tasks that a SAS programmer in the Pharmaceutical industry does. SAS software provides the power for records to be reviewed and exceptions reported. Along came a problem that had to be solved – a dataset with 500,000 observations was suspected to have duplicates. Presented is how a simple solution turned into a macro that could be used on any dataset.

INTRODUCTION
There exists a SAS® dataset called AE with the following variables:

CNO
PATNO
ASEQ
EVENT
START_DT
STOP_DT
ONGOING
A REL
A SEV
A ACTION
A GRADE

There are 500,000 observations but it is suspected that there are some duplicate observations, including having the same ASEQ value. What code could be written to select and put the duplicate records into a dataset called DUPOBS?

A SOLUTION
Let's start with a very simple solution:

```sas
proc sort data=ae;
  by cno patno aseq event start_dt stop_dt ongoing a_rel a_sev a_action a_grade;
data dupobs;
set ae;
  by cno patno aseq event start_dt stop_dt ongoing a_rel a_sev a_action a_grade;
  if ^first.ae_grade;
runk;
```

This solution works and does the job. But now let's look and see if this code can be made better and maybe make it a general macro. First, the variables that we are processing are the same so the code can be changed to read:

```sas
%let varlst=cno patno aseq event start_dt stop_dt ongoing a_rel a_sev a_action a_grade;
proc sort data=ae;
  by &varlst;
runk;
data dupobs;
set ae;
  by &varlst;
  if ^first.ae_grade;
runk;
```
The problem with this is that the last variable in the list needs to be known – this can be done using the following code inside a macro:

```sas
%macro dups;
    %let varlst=cno patno aseq event start_dt
            stop_dt ongoing ae_rel ae_sev ae_action
            ae_grade;
    %let i=1;
    %do %while(%scan(&varlst,&i) ne );
            %let lastvar=%scan(&varlst,&i);
            %let i=%eval(&i+1);
    %end;
    proc sort data=ae;
            by &varlst;
    run;
    data dupobs;
            set ae;
            by &varlst;
            if ^first.&lastvar;
    run;
%mend dups;
%dups(inds=ae,outds=dupobs);
run;
```

This works, but the variables still need to be put into the macro. What if the macro could get the variables from the dataset and make a macro out of it? Now the code would look something like:

```sas
%macro dups(inds=,outds=);
    proc contents data=&inds
            out=dsvars (keep=name) noprint;
    run;
    data _null_;
            length varlst $32000;
            retain varlst '';
            set dsvars end=eof;
            varlst=trim(left(varlst))||' '||trim(name);
            if eof then do;
                call symput('varlst',trim(varlst));
                call symput('lastvar',trim(name));
            end;
    run;
    proc sort data=&inds out=_mdups;
            by &varlst;
    run;
    data &outds;
            set _mdups;
            by &varlst;
            if ^first.&lastvar;
    run;
%mend dups;
%dups(inds=ae,outds=dupobs);
run;
```

This is a very good solution using SAS datasteps and procedure call. It does have the problem of having to create an internal dataset called _mdups but this should be considered a minor issue. Can SQL help? In this case yes, with the following code as an example -
%macro dups(inds=,outds=);
    proc contents data=&inds
        out=dsvars (keep=name) noprint;
    run;
    data _null_
      length varlst $32000;
      retain varlst '';
      set dsvars end=eof;
      if _n_=1 then varlst=trim(name);
      else varlst=trim(left(varlst))||'
        ',||trim(name);
      if eof then call symput('varlst',trim(varlst));
    run;
    proc sql;
      create table &outds as
        select &varlst from &inds
        group by &varlst
        having count(*)>1;
    quit;
run;
%Mend dups;
%dups(inds=ae,outds=dupobs);
run;

This is a very radical change from the one above as the data step only requires one macro variable to be now made, separated by a ',' for the SQL, and not SORT procedure call. It does use the SQL HAVING statement to subset placing only the records that are duplicates in the output dataset. Now how about getting the list of variables using SQL instead of a datastep:

%macro dups(inds=,outds=);
    %if %index(&inds,%str(.))=0 %then
        %let inds=WORK.&inds;;
    %let indslib=%scan(&inds,1,%str(.));
    %let indsdsn=%scan(&inds,2,%str(.));
    proc sql noprint;
        select name into :varlst separated by ','
        from dictionary.columns
        where upcase(libname)="%upcase(&indslib)" and
        upcase(memname)="%upcase(&indsdsn)";
    create table &outds as
        select &varlst from &inds
        group by &varlst
        having count(*)>1;
    quit;
run;
%Mend dups;
%dups(inds=ae,outds=dupobs);
run;

Note that in the code we have had to separate the incoming dataset name into two parts so that the correct names can be got from the COLUMNS table in the SAS DICTIONARY. Now lets finish this off adding some error trapping:
%macro dups(inds=, outds=);
  %if (z&inds=z) or (z&outds=z) %then %do;
    %put INDS and OUTDS parameters must be given for SELDUPS macro – macro
    SELDUPS aborted abnormally;
    %goto lvmac;
  %end;
  %if %index(&inds,%str(.))=0 %then
    %let inds=WORK.&inds;;
  %let indslib=%scan(&inds,1,%str(.));
  %let indsdsn=%scan(&inds,2,%str(.));
  %if %SYSFUNC(EXIST(&inds)) %then %do;
    proc sql noprint;
    select name into :varlst separated by ',' from dictionary.columns
    where upcase(libname)="%upcase(&indslib)"
    and upcase(memname)="%upcase(&indsdsn)";
    create table &outds as
      select &varlst from &inds group by &varlst having count(*)>1;
    quit;
  run;
  %else %put Dataset &inds does not exist – macro SELDUPS aborted abnormally;;
  %lvmac;
%mend dups(inds=,outds=);
%dups(inds=AE,outds=DUPOBS);
run;

The code above checks that INDS and OUTDS parameters are given and that the input dataset actually exists.

From SAS version 9.1 there is an option called DUPOUT=<>SASdatasetname> in the SORT procedure that will create a SAS dataset with duplicate records in it.

    proc sort data=ae dupout=dupobs;
      by cno patno aeseq event start_dt stop_dt
      ongoing ae_rel ae_sev ae_action ae_grade;
    run;

Although the new parameter in SAS version 9.1 exists so that duplicates can be found the macro developed above can be modified to suit different purposes, example finding records where there are three or more instances of the record in the dataset.

CONCLUSION
There was a dataset with suspected duplicate observations. What started out as a simple solution to find the duplicates went through a number of transformations where finally there was a solution in the form of a macro that could be used on any dataset.

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

    David Franklin
    16 Roberts Road
    Litchfield, NH 03052
    Tel/Fax 603/216-2232
    Email 100316.3451@compuserve.com
    http://ourworld.compuserve.com/homepages/dfranklinu

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