PROC SQL

Is it possible to generate a shorthand SELECT Clause in PROC SQL with every column 'except' those listed?

No. Although there is the facility to return all columns using the SELECT * syntax, there is no equivalent for the DROP = dataset option on the output data / report.

The attached utility macro can be used however to generate a SELECT clause and store this in a macro variable. The macro is self-documenting; to understand the parameters call the macro:

```
%all_except(?)
```

which returns the following in the LOG:

```
NOTE: There are five parameters passed to this macro:
      ds        = [Specify a one- or a two-
level dataset name from which variables are SELECTed]
      tbl_alias = [Specify a table alias.  If none is specified the ta
ble name will be used.]
      not_these = [Specify the variables to be excluded from the SELEC
T list.]
      keepmac   = [Specify a macro variable name for the SELECT list.
 If not specified
                  the default name will be KEEP_THESE.]
      case      = [Specify U | L | I to return the SELECT clause in Up
case | Lowcase | Internal]
```

The ds parameter accepts a one- or two-level dataset name. The macro issues an ERRROR if the dataset specified does not exist. If the parameter is not specified, the macro will process the last-created dataset as specified in SYSLAST.

The tbl_alias parameter allows a table alias to be specified, otherwise the table name will be used.

The not_these parameter allows a list of columns to be excluded from the SELECT clause. Columns may be specified with or without a table alias. If the parameter is not specified the macro issues a NOTE advising that all variables will be included on the SELECT clause.

The keepmac parameter allows the variable to be used to store the SELECT clause to be named. If not specified, the results will be stored in a Global Macro variable called keep_these.

The case parameter allows the SELECT clause to be built to match a 'house' style: returning in

(U)pper case, (L)ower case or (I)nternal (i.e. as stored).

A macro call such as:

```
%all_except(ds = sashelp.cars, tbl_alias = c, not_these = "c.invoice,
MPG_Highway")

%put &keep_these ;
```

generates a macro variable keep_these contaning the following:

```
c.make ,c.model ,c.type ,c.origin ,c.drivetrain ,c.msrp ,c.enginesize
,c.cylinders ,c.horsepower ,c.mpg_city ,c.weight ,c.wheelbase ,c.lengt
h
```

The code:

```
%macro all_except( helpme
                  ,ds        = %nrstr(&syslast)
                  ,tbl_alias =
                  ,not_these =
    ,keepmac   = keep_these
    ,case      = l
                 ) ;
  %local i var excl_list drop_some set_case ;
  %let keepmac = %upcase(&keepmac) ;
  %if %verify(&keepmac,%sysfunc(collate(65,90))%sysfunc(collate(97,122
))%sysfunc(collate(48,57))_) %then
  %do ;
    %put WARNING: The macro variable name &keepmac is invalid. The nam
e KEEP_THESE will be used instead. ;
  %end ;
  %global &keepmac ;
  %if &helpme = ? or %upcase(&helpme) = HELP %then
  %do ;
    %put NOTE: There are five parameters passed to this macro: ;
 %put NOTE- ds        = [Specify a one- or a two-
level dataset name from which variables are SELECTed] ;
    %put NOTE- tbl_alias = [Specify a table alias.  If none is specifi
ed the table name will be used.] ;
 %put NOTE- not_these = [Specify the variables to be excluded from the
 SELECT list.] ;
 %put NOTE- keepmac   = [Specify a macro variable name for the SELECT
```

```
list.  If not specified ;
 %put NOTE-              the default name will be KEEP_THESE.] ;
 %put NOTE- case      = [Specify U | L | I to return the SELECT clause
 in Upcase | Lowcase | Internal] ;
 %let &keepmac = ;
    %goto endmac ;
  %end ;
  options nonotes ;
  %let ds        = %lowcase(&ds)         ;
  %let tbl_alias = %lowcase(&tbl_alias) ;
  %let not_these = %lowcase(&not_these) ;
  %if not %sysfunc(exist(&ds)) %then
  %do ;
    %put ERROR: The dataset &ds does not exist. The macro programme wi
ll not execute. ;
 %let &keepmac = ;
    %goto endmac ;
  %end ;
  %*** Check if two-level dsname *** ;
  %if %index(&ds,.) %then
  %do ;
    %*** Split the dsname *** ;
 %let lib = %scan(&ds,1,.) ;
 %let dsn = %scan(&ds,2,.) ;
  %end ;
  %else
  %do ;
    %*** If USER not assigned, assume WORK *** ;
 %if %sysfunc(libref(user)) = 0 %then %let lib = user ;
 %else %let lib = work ;
 %let dsn = &ds ;
  %end ;
  %*** Check for Table Alias *** ;
  %if &tbl_alias = %then
  %do ;
    %*** If missing use Table Name *** ;
 %let tbl_alias = &dsn ;
  %end ;
  %*** Check if Exception List specified *** ;
  %if &not_these ne  %then
  %do ;
    %*** Cleanse, Split and Quote the exception list *** ;
 %*** Remove quotes and commas *** ;
 %let not_these = %sysfunc(compress(&not_these,%bquote(%sysfunc(byte(3
4))%sysfunc(byte(39))%sysfunc(byte(44))))) ;
    %do i = 1 %to %sysfunc(countw(&not_these)) ;
      %let var = %scan(&not_these,&i,%str( )) ;
   %*** Remove Table aliases if present *** ;
   %if %index(&var,.) %then %let var = %scan(&var,2,.) ;
```

```
   %let excl_list = &excl_list "&var" ;
 %end ;
 %*** Exclusion clause for KEEP List PROC SQL *** ;
 %let drop_some = %nrbquote(and lowcase(name) not in (&excl_list)) ;
  %end ;
  %else
  %do ;
    options notes ;
    %put NOTE: No variables listed to be excluded.  All variables will
 be selected. ;
 options nonotes ;
  %end ;
 %*** Build the KEEP List *** ;
 %if %upcase(&case) = U %then
      %do ;
        %let set_case = upcase  ;
  %let tbl_alias = %upcase(&tbl_alias) ;
   %end ;
 %else %if %upcase(&case) = L %then %let set_case = lowcase ;
    proc sql noprint ;
   select &set_case(name) into :&keepmac separated by " ,&tbl_alias.."
   from dictionary.columns
   where lowcase(libname) = "&lib"
     and lowcase(memname) = "&dsn"
        %unquote(&drop_some)
  ;
 quit ;
  %let keep_these = &tbl_alias..&&&keepmac ;
  %endmac:
%mend all_except ;


%all_except(?)

%all_except(ds = sashelp.cars, tbl_alias = c, not_these = "c.invoice,
mpg_highway")

%put &keep_these ;
```

Unique solution ID: #1038
Author: Alan D Rudland
Last update: 2017-10-27 15:58