Macros

Can I use the colon modifier to create a 'begins with' list of variables on a SELECT clause in PROC SQL?

No.

Shame though - it's a really handy way to save typing long lists of repetitive variables!
Submitting the code:

```
proc sql ;
  select unit:
  from sashelp.applianc
  ;
quit ;
```

results in the following notification in the LOG:

```
23         proc sql ;
24           select unit:
                        _
                        22
                        200
ERROR 22-322: Syntax error, expecting one of the following: a quoted s
tring, !, !!, &, (, *, **, +, ',', -, '.', /, <, <=, <>, =, >, >=, ?,
AND, AS, BETWEEN, CONTAINS, EQ, EQT, FORMAT, FROM, GE, GET, GT, GTT, I
N, INFORMAT, INTO, IS, LABEL, LE, LEN, LENGTH, LET, LIKE, LT, LTT, NE,
 NET, NOT, NOTIN, OR, TRANSCODE, ^, ^=, |, ||, ~, ~=.

ERROR 200-322: The symbol is not recognized and will be ignored.

25           from sashelp.applianc
26           ;
NOTE: PROC SQL set option NOEXEC and will continue to check the syntax
 of statements.
27         quit ;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

It is possible to build a list of the desired variables using a utility macro with a PERL Regular Expression to match the desired text string.

```
%macro prefix( helpme
             ,dsn = &syslast
```

```
     ,pfx =
     ,sep = %str( )
    ) ;
  %local varlist lib dsid i var ;

  %*** Check for HELPME *** ;
  %if &helpme = ? or &helpme = HELP %then
  %do ;
    %put NOTE: There are three keyword parameters passed to this macro
: ;
 %put NOTE- dsn = [Specify a one- or a two-level dataset name from whi
ch variables are identified.  If not specified, the last-
created dataset will be used.] ;
    %put NOTE- pfx = [Specify the text string to identify variable nam
e suffix.  The parameter is required.  If not specified ALL variables
will be returned.] ;
 %put NOTE- sep = [Specify the separator to be inserted between the va
riable names in the list.  If not specified a SPACE will be used.  Val
id options are SPACE and COMMA.] ;
 %put NOTE- Sample Call: %nrstr(%%suffix%(sashelp.applianc,unit,COMMA%
)) to return all variables in 'sashelp.applianc' which begin with 'uni
t' in a comma-separated list. ;
    %goto endmac ;
  %end ;

  %if not %sysfunc(exist(&dsn)) %then
  %do ;
    %put ERROR: The dataset &dsn does not exist. The programme will no
t execute. ;
    %goto endmac ;
  %end ;

  %if &sep ne %str( ) %then
    %do ;
   %if %upcase(&sep) = COMMA %then %let sep = %str(, ) ;
      %else %let sep = %str( ) ;
 %end ;

  %*** Open the dataset *** ;
  %let dsid = %sysfunc(open(&dsn,i));
  %if &dsid = 0 %then
    %do ;
      %put %sysfunc(sysmsg()) ;
      %goto endmac ;
    %end ;

  %let numvars = %sysfunc(attrn(&dsid,nvars)) ;

  %*** Read all of the variables - keep matching instances, or all if
```

```
no prefix specified *** ;
  %if &pfx ne %then
    %do i = 1 %to &numvars ;
      %let var = %sysfunc(varname(&dsid,&i)) ;
      %if %sysfunc(prxmatch(/^%qupcase(&pfx)\w*/,%qupcase(&var))) %the
n
        %if &i = 1 %then %let varlist = &var             ;
        %else            %let varlist = &varlist&sep&var ;
    %end;
  %else
    %do i = 1 %to &numvars ;
      %if &i = 1 %then %let varlist = &var             ;
      %else            %let varlist = &varlist&sep%sysfunc(varname(&ds
id,&i));
    %end;

  %if %sysfunc(length(&varlist)) = 0 %then %put WARNING: No variables
in the dataset &dsn. begin with the text string "&pfx". ;
  %*** Return resolved list into the Input Stack *** ;
  &varlist

  %endmac :
%mend prefix ;
```

Using the macro achieves the desired effect:

```
proc sql ;
  select %prefix(dsn = sashelp.applianc
                ,pfx = unit
                ,sep = COMMA
                )
  from sashelp.applianc
  ;
quit ;
```

This is a 'clever' solution, however it is not the most 'efficient' solution. Using a DATA step KEEP= option (which does support the colon modifier) on the source table also achieves the desired effect, but is also efficient.

```
proc sql ;
  select *
  from sashelp.applianc (keep = unit:)
  ;
```

Macros

```
quit ;
```

Unique solution ID: #1042
Author: Alan D Rudland
Last update: 2019-01-16 09:31