m_utl_varlist.sas File Reference

Utilities

Utility macro to obtain a column name list from an input table

---

Description

The macro can be used to obtain a list of columns from an input SAS dataset or database table. The result list can be filtered by column type, column name, column prefix or suffix or contain string. The output column list can be blank-, comma- or any custom character separated. Furthermore the column list can be single or double quoted and columns names can be extended with a custom prefix. The m_utl_varlist.sas macro recognizes almost all date and time variable format types including the ISO 8601 basic and extended notations. The ISO 8601 standard is an international standard for representing dates and time, including many variations for representing dates, times, and intervals. The two main representations of date, time, and datetime values within ISO 8601 standards are the basic and extended notations. This macro is originally based on the ut_varlist.sas macro program by Dave Prinsloo (dave.prinsloo@yahoo.com)

Note

In case of encrypted SAS datasets, the ENCRYPTKEY= parameter must be provided as part of the CREDS credentials string.

Autors

Paul Alexander Canals y Trocha (paul.canals@gmail.com)

Date

2021-01-21 00:00:00

Version

21.1.01

Link

https://github.com/paul-canals/toolbox

Parameters

Input	help	Parameter, if set (Help or ?) to print the Help information in the log. In all other cases this parameter should be left out from the macro call.
Input	table	Full LIBNAME.TABLENAME name of the SAS Dataset or database table to get the column list from. The parameter can contain a combination of SAS data step style statement between brackets like where, keep, drop or rename. The default value for TABLE is: _NONE_.
Input	dataset	Alias of the TABLE parameter.
Input	data	Alias of the TABLE parameter.
Input	creds	Optional. Specifies the ENCRYPTKEY= parameter value if there is an encrypted table involved.
Input	type	Indicator [CHAR|DATE|DATETIME|NUM|TIME] to filter the result varlist selection by vartype and format. The default value is: _ALL_.
Input	contain	Optional character search string to reduce columns selected in the result varlist to those having the contain parameter value string in the column name.
Input	position	Optional indicator [START|END] or left blank to determine where the parameter _contain_ value is to be found in the column name.
Input	include	Optional blank separated list of columns to limit the result varlist to only those columns mentioned in the include list parameter.
Input	limit_to	Alias of the INCLUDE parameter.
Input	exclude	Optional blank separated list of columns to limit the result varlist by excluding those columns mentioned in the exclude list parameter.
Input	prefix	Optional character string to be added as to each output varname.
Input	quotes	Indicator [DOUBLE|NAMED|SINGLE] to select wether the output list is quoted. If set to DOUBLE, double quotes are added around each varname. If QUOTES is set to NAMED, each varname is surrounded by quote-n to allow variable names with special characters or blanks in the name. If set to SINGLE, single quotes are added around every varname. The parameter value is omitted per default (e.g. QUOTES=).
Input	out_dlm	Indicator [BLANK|COMMA] to select the output delimiter type. The default value is: BLANK.
Input	newdlm	Alias of the OUT_DLM parameter.
Input	debug	Boolean [Y|N] parameter to provide verbose mode information. The default value is: N.

Returns

- Result column name list.

Calls

- m_utl_print_message.sas
- m_utl_print_mtrace.sas

Usage

Example 1: Show help information:

```
%m_utl_varlist(?)
```

Example 2: Select only CHAR columns from SASHELP.class and add a new column:

```
data WORK.class;
   set SASHELP.class (keep=%m_utl_varlist(table=SASHELP.class,type=CHAR,debug=Y));
   attrib Load_DTTM length=8 format=datetime.;
   Load_DTTM=datetime();
run;

proc print data=WORK.class;
run;
```

Example 3: Select columns from encrypted WORK.class and rename column Name into Scholar:

```
data WORK.class(encrypt=aes encryptkey=aespasskey);
   set SASHELP.class;
run;

data WORK.class (keep=%m_utl_varlist(table=WORK.class(rename=(Name=Scholar)),creds=%str(en
cryptkey=aespasskey)));
   set WORK.class(encryptkey=aespasskey);
   Scholar=Name;
run;

proc print data=WORK.class;
run;
```

Example 4: Select only CHAR columns from SASHELP.class into a quoted list:

```
%let list=
   %m_utl_varlist(
      table  = SASHELP.class
    , type   = CHAR
    , quotes = DOUBLE
    , debug  = Y
      );

%put &=list.;
```

Example 5: Select columns from SASHELP.class into a comma separated list:

```
%let list=
   %m_utl_varlist(
      table   = SASHELP.class
    , out_dlm = COMMA
    , debug   = Y
      );

%put &=list.;
```

Example 6: Select columns from SASHELP.class into a custom separated list:

```
%let list=
   %m_utl_varlist(
      table   = SASHELP.class
    , out_dlm = %str($)
    , debug   = Y
      );

%put &=list.;
```

Example 7: Select only NUM columns from SASHELP.workers (including Date):

```
data WORK.workers;
   set SASHELP.workers;
   keep %m_utl_varlist(table=SASHELP.workers,type=NUM,debug=Y);
run;

proc print data=WORK.workers;
run;
```

Example 8: Select only DATE columns from SASHELP.workers:

```
data WORK.workers;
   set SASHELP.workers;
   keep %m_utl_varlist(table=SASHELP.workers,type=DATE,debug=Y);
run;

proc print data=WORK.workers;
run;
```

Example 9: Select columns from SASHELP.class and add a prefix to the names:

```
%let list=
   %m_utl_varlist(
      table  = SASHELP.class
    , prefix = CLASS_
    , debug  = Y
      );

%put &=list.;
```

Example 10: Select only columns from SASHELP.class into the output table:

```
data WORK.class;
   set SASHELP.class;
   keep %m_utl_varlist(table=SASHELP.class,debug=Y);
   dummy='this is just a temporary column';
run;

proc print data=WORK.class;
run;
```

Example 11: To check if a column exists in a given table:

```
%macro findColumn(table=,colnm=);

   %if %m_utl_varlist(table=&table.,include=&colnm.) ne %str()
      %then %put Column &colnm. exists in table &table.!;

%mend findColumn;

%findColumn(table=SASHELP.class,colnm=Sex);
```

Example 12: To search for common columns between two tables:

```
data WORK.class;
   set SASHELP.class;
   keep Name Age;
run;

%let list=
   %m_utl_varlist(
      table    = SASHELP.class
    , limit_to = %m_utl_varlist(table=WORK.class)
    , debug    = Y
   );

%put Common columns are: &list.;
```

Example 13: Generic Join on common columns with two tables:

```
data WORK.class;
   set SASHELP.class;
   keep %m_utl_varlist(table=SASHELP.class,type=NUM) Name;
run;

proc sql noprint;
   create table WORK.join as
   select %m_utl_varlist(
             table=SASHELP.class
           , prefix=a.
           , newdlm=COMMA
           , exclude=
                %m_utl_varlist(
                   table=WORK.class
                   )
             )
         , %m_utl_varlist(
             table=WORK.class
           , prefix=b.
           , newdlm=COMMA
             )
     from SASHELP.class a
        , WORK.class b
    where a.Name eq b.Name
    order by Sex, Name
   ;
quit;

proc print data=WORK.join;
run;
```

Example 14: Select columns from WORK.class into a named quoted list:

```
data WORK.class;
   set SASHELP.class;
   rename
      name = '/bic/Name'n
      sex  = '/bic/Sex'n
      ;
run;

%let list=
   %m_utl_varlist(
      table  = WORK.class
    , quotes = NAMED
    , debug  = Y
   );

%put Result column list: &list.;
```

Copyright