m_val_run_validation.sas File Reference

Validation

Validation macro to run checks on data using a control table

Description

The macro reads a list of validation rules from a SAS dataset or database table to perform validation checks on data listed on that table (CTL_TBL). The following validation rules can be used by this macro to perform the validation data checks:
[C]ustom : data validation by using a custom validation rule
[D]uplicate : data validation by checking for duplicate values
[I]nvalid : data validation by checking for invalid values
[M]issing : data validation by checking for missing values
The following operators are allowed when using the custom validation rule:
** : exponentiation
* : multiplication
/ : division
+ : addition
- : substraction
= : equal to
^= : not equal to
< : lesser than
> : greater than
<= : lesser than or equal to
>= : greater than or equal to
<> : max operator
>< : min operator
and : and operator
eq : equal to (=)
ge : greater than or equal to (>=)
gt : greater than (>)
in : equal to one of a list
le : lesser than or equal to (<=)
lt : lesser than (<)
ne : not equal to (^=)
not : not operator
or : or operator
If the given expression syntax is invalid, the rule status value is set to 0, exceptions to -1 and action set to ERROR.

Note

If the PRINT parameter value is set to Y, a SAS proc report step is used to print the validation summary status on the result tab of SAS Enterprise Guide or Stored Process Server.

Autors

Paul Alexander Canals y Trocha (paul.canals@gmail.com)
Dr. Simone Kossmann (simone.kossmann@web.de)

Date

2021-03-27 00:00:00

Version

21.1.03

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	ctl_tbl	Specifies the LIBNAME.TABLENAME of the parameter control SAS dataset or database table containing the list of validation rules. The default value for CTL_TBL is: _NONE_.
Input	exc_tbl	Specifies the LIBNAME.TABLENAME of the target SAS dataset or database table in which the exceptions are stored. The default value for EXC_TBL is: _NONE_.
Input	replace	Boolean [Y|N] parameter value to specify if the found exceptions (and errors) are to be appended into an existing exception table or if REPLACE is set to Y into a new exception table. The default value for REPLACE is: Y.
Input	g_value	Optional. Parameter to specify the high band value for which the validation result status color is set to green. The parameter depends on the PRINT value. The default value for G_VALUE is: 1.00.
Input	y_value	Optional. Parameter to specify the high band value for which the validation result status color is set to yellow. The parameter depends on the PRINT value. The default value for Y_VALUE is: 0.99.
Input	r_value	Optional. Parameter to specify the high band value for which the validation result status color is set to red. The parameter depends on the PRINT value. The default value for G_VALUE is: 0.50.
Input	print	Boolean [Y|N] parameter to generate the output by a SAS proc report step with style HtmlBlue. The default value for PRINT is: N.
Input	debug	Boolean [Y|N] parameter to provide verbose mode information. The default value is: N.

Returns

- Validation result target tables.

Calls

- m_utl_chk_table_exist.sas
- m_utl_list_operation.sas
- m_utl_print_message.sas
- m_utl_print_mtrace.sas
- m_utl_nlobs.sas
- m_val_chk_custom.sas
- m_val_chk_duplicates.sas
- m_val_chk_invalid.sas
- m_val_chk_missing.sas

Usage

Example 1: Show help information:

```
%m_val_run_validation(?)
```

Example 2: Step 1 - Create a validation control table:

```
data WORK.rules;
   length
      rule_id    8
      rule_type  $20
      library    $10
      table      $32
      columns    $1024
      range_min  $32
      range_max  $32
      value_list $1024
      expression $1024
      ;
   infile datalines4 dlm=';'
   missover;
   input
      rule_id
      rule_type  $
      library    $
      table      $
      columns    $
      range_min  $
      range_max  $
      value_list $
      expression $
      ;
datalines4;
1;Missing;WORK;CARS;Make; ; ; ; ;
2;Missing;WORK;CARS;Model Type; ; ; ; ;
3;Duplicate;WORK;CARS;Make Model Type; ; ; ; ;
4;Invalid;WORK;CARS;Cylinders; ; ;4 6 8 10 12; ;
5;Invalid;WORK;CARS;Invoice;30000;50000; ; ;
6;Invalid;WORK;CARS;Invoice;30000; ; ; ;
7;Invalid;WORK;CARS;Invoice; ;50000; ; ;
8;Custom;WORK;CARS; ; ; ; ;Invoice lt 50000; ; ;
9;Custom;WORK;CARS; ; ; ; ;(substr(Make,1,1) ne "") and (Type ^= '');
10;Missing;WORK;CLASS;Name Sex; ; ; ; ;
11;Invalid;WORK;CLASS;Sex; ; ;F M; ;
12;Invalid;WORK;CLASS;Age; ;18; ; ;
13;Custom;WORK;CLASS; ; ; ; ;Name in (select Name from SASHELP.classfit where Name ne 'Joh
n'); ; ;
14;Custom;WORK;CLASS; ; ; ; ;Weight in (select Weight from SASHELP.classfit where Weight >
 70); ; ;
15;Custom;WORK;CLASS; ; ; ; ;Weight in (select Weight from SASHELP.classfit where Weight
gt 70); ; ;
;;;;
run;
```

Example 2: Step 2 - Prepare the SASHELP.cars table for missing values:

```
data WORK.cars;
   set SASHELP.cars;
   if mod(_N_,10) eq 0 then do;
      type = '';
      length = .;
   end;
run;
```

Example 2: Step 3 - Prepare the SASHELP.class table for invalid values:

```
data WORK.class;
   set SASHELP.class;
   if Name eq 'John' then Sex = '';
run;
```

Example 2: Step 4 - Run the valdiation checks on the control table:

```
%m_val_run_validation(
   ctl_tbl = WORK.rules
 , exc_tbl = WORK.exceptions
 , print   = Y
 , debug   = Y
   );

proc print data=WORK.exceptions label;
run;
```

Copyright