m_val_chk_custom.sas File Reference

Validation

Validation macro to check data by using a custom condition rule

---

Description

The macro can be used to validate data by using a SAS data step or SQL style where statement which represents the condition to specify the validity state of a record in a given SAS dataset or database table. The validation expression may contain one or more columns from the given source table, use functions and all kinds of operators including the _IN_ operator. The columns used in the expression are detected automatically by the macro routine and stored together with their value in the exception table. If the target, error or exception tables are not set by their parameter values accordingly, output tables are defined by the macro routine and created in the WORK library. If an output exception table is given, the exceptions found by the validation will be appended to the given exception table. The same procedure is used for the error table, just as long as the ACTION parameter value is not equal to: "Check".

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)

Date

2022-10-18 00:00:00

Version

22.1.10

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	src_tbl	Specifies the LIBNAME.TABLENAME of the source SAS dataset or database table on which the validation is performed. The default value is: _NONE_.
Input	tgt_tbl	Specifies the LIBNAME.TABLENAME of the target SAS dataset or database table in which the valid data is stored. If this target table is not provided, this target table is created automatically in the WORK library. The default value is: _NONE_.
Input	err_tbl	Specifies the LIBNAME.TABLENAME of the target SAS dataset or database table in which the error data is stored. If this target table is not provided, this target table is created automatically in the WORK library. The default value is: _NONE_.
Input	exc_tbl	Specifies the LIBNAME.TABLENAME of the target SAS dataset or database table in which the exceptions are stored. If this target table is not provided, this target table is created automatically in the WORK library. The default value is: _NONE_.
Input	val_rule	Specifies the SAS data step style rule condition. The rule may contain one or more conditions and use multiple columns from the source table. Also functions and all kinds of operators are allowed including the data step _IN_ operator.
Input	sql_type	Optional. Boolean [Y|N] parameter value to specify the type of the rule expression. If the VAL_RULE parameter value contains a SQL type rule expression, the SQL_TYPE parameter value needs to be set to Y. The default value for SQL_TYPE is: N.
Input	chk_rule	Boolean [Y|N] parameter value to specify if the VAL_RULE expression is to be checked for syntax errors. The default value for CHK_RULE is: Y.
Input	show_err	Boolean [Y|N] parameter to show or hide warnings or errors in the log. The default value is: Y.
Input	action	Indicator [CHECK|MOVE|ABORT] to specify the to be taken by the validation routine. When the ACTION value is set to "Check", the source table records are validated and found exceptions written to the exception table, but also all records will be written to the output target table. When set to "Move", the error records are written to the error table and only valid records to the target table. When set to "Abort" only the found exceptions are written to the exception table, but not into any other target tables. The default value is: Check.
Input	append	Boolean [Y|N] parameter value to specify if the found exceptions (and errors) are to be appended into a given exception table (and error table). The default value for APPEND is: Y.
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_chg_delimiter.sas
- m_utl_chk_table_exist.sas
- m_utl_delete_file.sas
- m_utl_get_col_code.sas
- m_utl_list_operation.sas
- m_utl_nlobs.sas
- m_utl_print_file.sas
- m_utl_print_message.sas
- m_utl_print_mtrace.sas
- m_utl_printto.sas
- m_utl_varlist.sas

Usage

Example 1: Show help information:

```
%m_val_chk_custom(?)
```

Example 2: Validate SASHELP.class to check that all male students are minor:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.class
 , val_rule = %str(Sex='M' and Age<18)
 , action   = Check
 , print    = Y
 , debug    = Y
   );
```

Example 3: Validate SASHELP.class to check that all student names start with a 'J':

```
%m_val_chk_custom(
   src_tbl  = SASHELP.class
 , val_rule = %str(substr(Name,1,1)='J')
 , action   = Check
 , print    = Y
 , debug    = Y
   );
```

Example 4: Validate SASHELP.class to ensure that all students are female:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.class
 , val_rule = %str(Sex eq 'F')
 , action   = Move
 , print    = Y
 , debug    = Y
   );
```

Example 5: Validate SASHELP.class to abort when not all students are female:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.class
 , val_rule = %str(Sex eq 'F')
 , action   = Abort
 , print    = Y
 , debug    = Y
   );
```

Example 6: Validate SASHELP.class to ensure the students BMI is lesser than 22:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.class
 , val_rule = %str(((Weight/2)/(((Height*2.54)/100)**2) < 22))
 , action   = Move
 , print    = Y
 , debug    = Y
   );
```

Example 7: Validate SASHELP.baseball to check if all players salaries are listed:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.baseball
 , val_rule = %str(Salary ne . and logSalary ^= .)
 , action   = Move
 , print    = Y
 , debug    = N
   );

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

Example 8: Validate SASHELP.baseball to check if all players salaries are listed:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.baseball
 , exc_tbl  = WORK.baseball_custom
 , val_rule = %str(Salary <> .)
 , action   = Check
 , print    = Y
 , debug    = N
   );

%m_val_chk_custom(
   src_tbl  = SASHELP.baseball
 , exc_tbl  = WORK.baseball_custom
 , val_rule = %str(logSalary ^= .)
 , action   = Check
 , print    = Y
 , debug    = N
   );

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

Example 9: Validate SASHELP.class to check Name against SASHELP.classfit:

```
%m_val_chk_custom(
   src_tbl  = SASHELP.class
 , exc_tbl  = WORK.class_custom
 , val_rule = %str(Name in (select Name from SASHELP.classfit where Name ne 'John'))
 , sql_type = Y
 , action   = Check
 , print    = Y
 , debug    = N
   );

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

Copyright