1. HPCC String Pattern Finding

Usage: *hpcc.string.find(out.dataframe,nameOfString,patternToFind,constant=TRUE,instance,output=20,submit=TRUE)*

Arguments

out.dataframe= Name of the output data set

nameOfString = Name of the string on which the searching has to be carried out

patternToFind = Pattern you are looking to find in the string

constant = Determines whether the pattern provided in the argument is a variable name in ECL or a

constant to search for

instance = which instance of the pattern to return, takes an integer value

output = no of records required to be return in R for viewing purpose. By default its 20

submit = Boolean. Indicates whether this is a part of the program. Else its returns the code. By default its TRUE.

Usage:

1. Suppose we need to find the index when a pattern ‘XYZ’ occurring for the second time in a string variable ABC which already exists in the ECL workspace and store it in a variable PQR.

*hpcc.string.find(out.dataframe=‘PQR’, nameOfString=’ABC’, patternToFind=’XYZ’, instance=2)*

1. Suppose we need to find the index when a pattern stored in a variable XYZ occurring for the second time in a string variable ABC which is already existing in the ECL workspace and store it in a variable PQR.

*hpcc.string.find(out.dataframe=‘PQR’, nameOfString=’ABC’, patternToFind=’XYZ’,constant=FALSE, instance=2)*

1. HPCC String Pattern Replace

*hpcc.string.replace(out.dataframe,nameOfString,patternToFind,patternToReplace,constant=TRUE,instance,output=20,submit=TRUE)*

Arguments

out.dataframe= Name of the output data set

nameOfString = Name of the string on which the searching has to be carried out

patternToFind = Pattern you are looking to find in the string

patternToReplace = pattern to replace in the string

constant = Determines whether the patterns provided in the argument is a variable name in ECL or

a constant to search for

instance = which instance of the pattern to return, takes an integer value

output = no of records required to be return in R for viewing purpose. By default its 20

submit = Boolean. TRUE indicate this line is a part of the main code. Else it just returns the code without affecting the main code. By default it is TRUE.

Usage:

1. Suppose we need to replace the pattern ‘XYZ’ with the pattern ‘UVW’ in a string variable ABC which already exists in the ECL workspace and store it in a variable PQR.

*hpcc.string.find(out.dataframe=‘PQR’, nameOfString=’ABC’, patternToFind=’XYZ’, patternToReplace=’UVW’)*

1. Suppose we need to replace the pattern stored in variable XYZ with the pattern stored in variable UVW in a string variable ABC which is already existing in the ECL workspace and store it in a variable PQR.

*hpcc.string.find(out.dataframe=‘PQR’, nameOfString=’ABC’, patternToFind=’XYZ’, patternToReplace=’UVW’,constant=FALSE*

1. TRIM

*hpcc.trim(out.dataframe, nameOfString,LEFT=FALSE,RIGHT=FALSE,ALL=FALSE,submit=TRUE,output=20)*

Arguments

out.dataframe= Name of the output data set

nameOfString = Name of the string on which the searching has to be performed

LEFT = Boolean Value. IF the beginning part has to be trimmed

RIGHT = Boolean Value. IF the ending part has to be trimmed

ALL = Boolean Value. IF the whole string has to be trimmed

output = no of records required to be return in R for viewing purpose. By default its 20

submit = Boolean. TRUE indicate this line is a part of the main code. Else it just returns the code without affecting the main code. By default it is TRUE.

Usage:

1. Suppose we need to trim the dataset XYZ in the beginning part of the string and store it in a variable ‘OutString’.

*hpcc.trim(‘OutString’,‘XYZ’,LEFT=TRUE)*

for ex: ‘ ASD GHJ ’ changes to ‘ASD GHJ ’

1. *hpcc.trim(‘OutString’,‘XYZ’,ALL=TRUE)*

‘ ASD GHJ ’ changes to ‘ASDGHJ’

1. Creating Record

Record is a structure definition. It defines the columns involved in the dataset. Every data set in ECL has a record.

*hpcc.define.record(recordName,argsTypes,argNames,submit=TRUE)*

Arguments:

recordName = Name of the record

argTypes = It defines the TYPES of the arguments in a vector format.

argNames= It defines the names of the arguments in a vector format. Its length should be matching with

the argsTypes

submit = Boolean. TRUE indicate this line is a part of the main code. Else it just returns the code without affecting the main code. By default it is TRUE.

Usage:

1. Suppose we want to create a record structure of a ID which is numeric, name which is a string and Area which is also a String, the code looks as follows

*hpcc.define.record(recordName=’SomeRecord’,argsTypes=c(‘INTEGER’,’STRING’,’STRING’),argNames=c(‘ID’,’Name’,’Area’))*