**ETL Testing/Large Datasets Testing**

# **How to perform large dataset comparison?**

## Pre-requisites:

1. Along side Katalon Studio/Katalon Runtime Engine install python.
2. Download the CsvCompare Katalon plugin from [here](https://drive.google.com/file/d/1mno8Iw3gpATy6PB8nqOn5Uj24tuKhldS/view?usp=drive_link).
3. Refer [here](https://drive.google.com/file/d/1m_-_ZPc7SnTTdz4yECoG52Q-CoYaw-f5/view?usp=drive_link) on how to use plugins in Katalon.
4. Data sets available in csv format.

## Introduction on the plugin:

CsvCompare allows you to modify and compare large data sources exported as CSVs. With modify you can transform a csv dataset for comparison. You can also perform aggregation operations for the comparison on numeric columns.

## Read Data with CSV compare:

**getRowCountInCSV:** This keyword helps in getting the row counts from a CSV file.

String filePath – Provide absolute or relative path of the csv dataset.

boolean hasHeader – Leave it as false if the CSV has no header, else change it to true.

String charSet – Provide character set as string eg.. (ISO-8859-1, UTF-8, UTF-16, Windows-1252, US-ASCII). If you leave it blank default is UTF-8

## Modify Data with CSV compare:

There are few key words the plugin provides to modify csv datasets

**addColumnsToCSV:** This keyword allows you to add columns to existing data sets and allows you to specify values or operations for the columns to be added. The keyword takes two arguments:

String filePath – Provide absolute or relative path of the csv dataset.

Map columnAndOperations – Provide as Map where the key should be the name or index of columns to be added and the value can be a constant or an operation involving other columns. For example operation can be “C0 + C1” or “C3 \* C5” or “C2 + “ ” + C3” etc where 3 in C3 represents column index, so C3 is column 4 in the csv dataset.

**deleteColumnFromCSV:** This keyword would allow you to delete columns from an existing data set. The keyword takes two arguments:

String filePath – Provide absolute or relative path of the csv dataset.

List columnsToDelete – Provide as List the columns or column indexes to delete from the dataset.

**moveColumnInCSV:** This keyword will allow you to move a column in the csv data set from one place to another. The keyword takes three arguments:

String filePath – Provide absolute or relative path of the csv dataset.

Int fromIndex – Provide index of where column to be moved is currently present.

Int toIndex – Provide index of where the column has to be moved to.

**addColumnHeaders:** You can add new column names to your csv data set if it doesn’t have one.

**renameColumnHeaders:** You can rename current column header with desired header name

## Get Aggregated value on Arithematic CSV columns:

**aggregateColumnInCSV:** The keyword aggregates csv data numeric columns for various validations. Below are the arguments the keyword takes:

String filePath – Provide absolute or relative path of the csv dataset.

Int columnIndexOrName – Provide either the column name or index of the column to be aggregated

Stromg aggregationType – Specify one of the following : “sum”, “mean”, “min”, “max”, “std” (standard deviation), “var” (variance)

## Compare CSVs:

**compareCSVs:** Compare CSVs allows you to compare large csv files with millions of records.

String file1Path – Absolute/Relative path of csv file1

String file2Path – Absolute/Relative path of csv file2

String statusFolderPath - Absolute path of the folder where comparison status files will be present. By default this will be automatically created under "Reports/Status"

String compositeKeys - Provide the composite key headers in your csv files as comma separated values as originally in the csv header. When compositeKeys is provided hasHeader should be set to True

Boolean hasHeader - If your CSV files have a header row then specify this as True

Boolean headerOrdered - If headers of two files are in two different order (same names with different order) then set this to False

Int startRow - Start row for comparison

Int endRow - End row for comparison

Boolean recursive - Mode in which each record/composite key in file 1 will be compared with all the records in file 2. Setting recursive to True for large files can increase comparison time

Boolean sorted - Not implemented yet as of now

String printKeys - Keys/Columns that will be printed as row reference when there is no composite keys. Specify this field as comma separated with same header names as in the csvs.

## Excel Operations:

**convertExcelToCSV:** Allows you to conver .xlsx file to CSV file

String excelFilePath – Absolute/Relative path of .xlsx file

String sheetName – Sheet name in the excel that has to be converted to CSV file

String csvFilePath – Provide absolute/relative path of a folder where the output CSV file will be generated

**filterExcelData:** This keyword will create a new tab called “filtered” with filtered data from the original .xlsx sheet.

String excelFilePath – Absolute/Relative path of .xlsx file

String sheetName – Sheet name in the excel that has to be converted to CSV file

Boolean noHeader – Provide false where there are no headers in the excel file else true

Map<String, List> filters – You can provide a map of filter criteria here are some examples

If there are headers you can specify the header name like this {“Name”: [“Bravo”,”Charlie”], “Date of Birth”: [1-May-2000]}

If there are no headers then you can specify the map keys either as excel columns such as AA, A, Z etc or you can specify column index starting from 0 like this {“AA”:[“Charlie”,”Devon”]} or {0:[“Charlie”,”Clara”]}

## Demonstration:

<https://drive.google.com/file/d/1eXkSsuc9K_-vjX0gSlVigZGvI9_19SEH/view?usp=drive_link>