Package 'DataQualityDashboard'

July 26, 2023

Type Package

Title Execute and View Data Quality Checks on OMOP CDM Database

Version 2.4.0

Date 2023-07-26

Maintainer Katy Sadowski <sadowski@ohdsi.org>

Description An R package for assessing data quality in standardized OMOP Common Data Model data sources.

License Apache License (>= 2)

Config/build/clean-inst-doc FALSE

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/OHDSI/DataQualityDashboard

BugReports https://github.com/OHDSI/DataQualityDashboard/issues

Depends R (>= 3.2.2),
DatabaseConnector (>= 2.0.2)

Imports magrittr,
ParallelLogger,
dplyr,
jsonlite,
rJava,
SqlRender (>= 1.10.1),
plyr,
stringr,
rlang,
tidyselect,
readr

Suggests testthat,
knitr,
rmarkdown,
markdown,
shiny,
ggplot2,
Eunomia,
R.utils

Remotes ohdsi/Eunomia

RoxygenNote 7.2.2

Encoding UTF-8

R topics documented:

`convertJsonResultsFileCase` *Convert JSON results file case*

Description

Convert a DQD JSON results file between camelcase and (all-caps) snakecase. Enables viewing of pre-v.2.1.0 results files in later DQD versions, and vice versa

Usage

```
convertJsonResultsFileCase(
  jsonFilePath,
  writeToFile,
  outputFolder = NA,
  outputFile = "",
  targetCase
)
```

Arguments

`jsonFilePath`	Path to the JSON results file to be converted
`writeToFile`	Whether or not to write the converted results back to a file (must be either TRUE or FALSE)
`outputFolder`	The folder to output the converted JSON results file to
`outputFile`	(OPTIONAL) File to write converted results JSON object to. Default is name of input file with a "_camel" or "_snake" postfix
`targetCase`	Case into which the results file parameters should be converted (must be either "camel" or "snake")

Value

DQD results object (a named list)

executeDqChecks *Execute DQ checks*

Description

This function will connect to the database, generate the sql scripts, and run the data quality checks against the database.

Usage

```
executeDqChecks(
  connectionDetails,
  cdmDatabaseSchema,
  resultsDatabaseSchema,
  vocabDatabaseSchema = cdmDatabaseSchema,
  cdmSourceName,
  numThreads = 1,
  sqlOnly = FALSE,
  sqlOnlyUnionCount = 1,
  sqlOnlyIncrementalInsert = FALSE,
  outputFolder,
  outputFile = "",
  verboseMode = FALSE,
  writeToTable = TRUE,
  writeTableName = "dqdashboard_results",
  writeToCsv = FALSE,
  csvFile = "",
  checkLevels = c("TABLE", "FIELD", "CONCEPT"),
  checkNames = c(),
  cohortDefinitionId = c(),
  cohortDatabaseSchema = resultsDatabaseSchema,
  cohortTableName = "cohort",
  tablesToExclude = c("CONCEPT", "VOCABULARY", "CONCEPT_ANCESTOR",
   "CONCEPT_RELATIONSHIP", "CONCEPT_CLASS", "CONCEPT_SYNONYM", "RELATIONSHIP", "DOMAIN"),
  cdmVersion = "5.3",
  tableCheckThresholdLoc = "default",
  fieldCheckThresholdLoc = "default",
  conceptCheckThresholdLoc = "default"
)
```

Arguments

`connectionDetails`
: A connectionDetails object for connecting to the CDM database

`cdmDatabaseSchema`
: The fully qualified database name of the CDM schema

`resultsDatabaseSchema`
: The fully qualified database name of the results schema

`vocabDatabaseSchema`
: The fully qualified database name of the vocabulary schema (default is to set it as the cdmDatabaseSchema)

`cdmSourceName` The name of the CDM data source

`numThreads` The number of concurrent threads to use to execute the queries

`sqlOnly` Should the SQLs be executed (FALSE) or just returned (TRUE)?

`sqlOnlyUnionCount`
(OPTIONAL) In sqlOnlyIncrementalInsert mode, how many SQL commands to union in each query to insert check results into results table (can speed processing when queries done in parallel). Default is 1.

`sqlOnlyIncrementalInsert`
(OPTIONAL) In sqlOnly mode, boolean to determine whether to generate SQL queries that insert check results and associated metadata into results table. Default is FALSE (for backwards compatibility to <= v2.2.0)

`outputFolder` The folder to output logs, SQL files, and JSON results file to

`outputFile` (OPTIONAL) File to write results JSON object

`verboseMode` Boolean to determine if the console will show all execution steps. Default is FALSE

`writeToTable` Boolean to indicate if the check results will be written to the dqdashboard_results table in the resultsDatabaseSchema. Default is TRUE

`writeTableName` The name of the results table. Defaults to 'dqdashboard_results'. Used when sqlOnly or writeToTable is True.

`writeToCsv` Boolean to indicate if the check results will be written to a csv file. Default is FALSE

`csvFile` (OPTIONAL) CSV file to write results

`checkLevels` Choose which DQ check levels to execute. Default is all 3 (TABLE, FIELD, CONCEPT)

`checkNames` (OPTIONAL) Choose which check names to execute. Names can be found in inst/csv/OMOP_CDM_v[cdmVersion]_Check_Descriptions.csv. Note that "cdmTable", "cdmField" and "measureValueCompleteness" are always executed.

`cohortDefinitionId`
The cohort definition id for the cohort you wish to run the DQD on. The package assumes a standard OHDSI cohort table with the fields cohort_definition_id and subject_id.

`cohortDatabaseSchema`
The schema where the cohort table is located.

`cohortTableName`
The name of the cohort table. Defaults to 'cohort'.

`tablesToExclude`
(OPTIONAL) Choose which CDM tables to exclude from the execution.

`cdmVersion` The CDM version to target for the data source. Options are "5.2", "5.3", or "5.4". By default, "5.3" is used.

`tableCheckThresholdLoc`
The location of the threshold file for evaluating the table checks. If not specified the default thresholds will be applied.

`fieldCheckThresholdLoc`
The location of the threshold file for evaluating the field checks. If not specified the default thresholds will be applied.

`conceptCheckThresholdLoc`
The location of the threshold file for evaluating the concept checks. If not specified the default thresholds will be applied.

Value

If sqlOnly = FALSE, a list object of results

`listDqChecks` *List DQ checks*

Description

Details on all checks defined by the DataQualityDashboard Package.

Usage

```
listDqChecks(
  cdmVersion = "5.3",
  tableCheckThresholdLoc = "default",
  fieldCheckThresholdLoc = "default",
  conceptCheckThresholdLoc = "default"
)
```

Arguments

`cdmVersion`	The CDM version to target for the data source. By default, 5.3 is used.
`tableCheckThresholdLoc`	The location of the threshold file for evaluating the table checks. If not specified the default thresholds will be applied.
`fieldCheckThresholdLoc`	The location of the threshold file for evaluating the field checks. If not specified the default thresholds will be applied.
`conceptCheckThresholdLoc`	The location of the threshold file for evaluating the concept checks. If not specified the default thresholds will be applied.

`reEvaluateThresholds` *Re-evaluate Thresholds*

Description

Re-evaluate an existing DQD result against an updated thresholds file.

Usage

```
reEvaluateThresholds(
  jsonFilePath,
  outputFolder,
  outputFile,
  tableCheckThresholdLoc = "default",
  fieldCheckThresholdLoc = "default",
  conceptCheckThresholdLoc = "default",
  cdmVersion = "5.3"
)
```

Arguments

jsonFilePath	Path to the JSON results file generated using the execute function
outputFolder	The folder to output new JSON result file to
outputFile	File to write results JSON object to
tableCheckThresholdLoc	The location of the threshold file for evaluating the table checks. If not specified the default thresholds will be applied.
fieldCheckThresholdLoc	The location of the threshold file for evaluating the field checks. If not specified the default thresholds will be applied.
conceptCheckThresholdLoc	The location of the threshold file for evaluating the concept checks. If not specified the default thresholds will be applied.
cdmVersion	The CDM version to target for the data source. By default, 5.3 is used.

---

viewDqDashboard *View DQ Dashboard*

Description

View DQ Dashboard

Usage

```
viewDqDashboard(jsonPath, launch.browser = NULL, display.mode = NULL, ...)
```

Arguments

jsonPath	The path to the JSON file produced by executeDqChecks
launch.browser	Passed on to shiny::runApp
display.mode	Passed on to shiny::runApp
...	Extra parameters for shiny::runApp() like "port" or "host"

---

writeJsonResultsToCsv *Write JSON Results to CSV file*

Description

Write JSON Results to CSV file

Usage

```
writeJsonResultsToCsv(
  jsonPath,
  csvPath,
 columns = c("checkId", "failed", "passed", "isError", "notApplicable", "checkName",
   "checkDescription", "thresholdValue", "notesValue", "checkLevel", "category",
  "subcategory", "context", "checkLevel", "cdmTableName", "cdmFieldName", "conceptId",
   "unitConceptId", "numViolatedRows", "pctViolatedRows", "numDenominatorRows",
    "executionTime", "notApplicableReason", "error", "queryText"),
  delimiter = ","
)
```

Arguments

jsonPath	Path to the JSON results file generated using the execute function
csvPath	Path to the CSV output file
columns	(OPTIONAL) List of desired columns
delimiter	(OPTIONAL) CSV delimiter

---

writeJsonResultsToTable

Write JSON Results to SQL Table

Description

Write JSON Results to SQL Table

Usage

```
writeJsonResultsToTable(
  connectionDetails,
  resultsDatabaseSchema,
  jsonFilePath,
  writeTableName = "dqdashboard_results",
  cohortDefinitionId = c()
)
```

Arguments

connectionDetails	A connectionDetails object for connecting to the CDM database
resultsDatabaseSchema	The fully qualified database name of the results schema
jsonFilePath	Path to the JSON results file generated using the execute function
writeTableName	Name of table in the database to write results to
cohortDefinitionId	If writing results for a single cohort this is the ID that will be appended to the table name