# Getting Started

Clair Blacketer

2026-01-08

## Contents



## 1 R Installation

```
install.packages("remotes")
remotes::install_github("OHDSI/DataQualityDashboard")
```

## 2 Note

To view the JSON results in the shiny application the package requires that the CDM_SOURCE table has at least one row with some details about the database. This is to ensure that some metadata is delivered along with the JSON, should it be shared. As a best practice it is recommended to always fill in this table during ETL or at least prior to running the DQD.

## 3 Executing Data Quality Checks

```
# fill out the connection details ---------------------------------------------------------------------
connectionDetails <- DatabaseConnector::createConnectionDetails(
dbms = "",
user = "",
password = "",
server = "",
port = "",
extraSettings = "",
pathToDriver = ""
)

cdmDatabaseSchema <- "yourCdmSchema" # the fully qualified database schema name of the CDM
resultsDatabaseSchema <- "yourResultsSchema" # the fully qualified database schema name of the results
```

```
cdmSourceName <- "Your CDM Source" # a human readable name for your CDM source
cdmVersion <- "5.4" # the CDM version you are targetting. Currently supports 5.2, 5.3, and 5.4

# determine how many threads (concurrent SQL sessions) to use ----------------------------------------
numThreads <- 1 # on Redshift, 3 seems to work well

# specify if you want to execute the queries or inspect them ------------------------------------------
sqlOnly <- FALSE # set to TRUE if you just want to get the SQL scripts and not actually run the queries
sqlOnlyIncrementalInsert <- FALSE # set to TRUE if you want the generated SQL queries to calculate DQD
sqlOnlyUnionCount <- 1  # in sqlOnlyIncrementalInsert mode, the number of check sqls to union in a sing

# NOTES specific to sqlOnly <- TRUE option ------------------------------------------------------------
# 1. You do not need a live database connection.  Instead, connectionDetails only needs these parameter
#      connectionDetails <- DatabaseConnector::createConnectionDetails(
#        dbms = "", # specify your dbms
#        pathToDriver = "/"
#      )
# 2. Since these are fully functional queries, this can help with debugging.
# 3. In the results output by the sqlOnlyIncrementalInsert queries, placeholders are populated for exec
# 4. In order to use the generated SQL to insert metadata and check results into output table, you must


# where should the results and logs go? ---------------------------------------------------------------
outputFolder <- "output"
outputFile <- "results.json"


# logging type ----------------------------------------------------------------------------------------
verboseMode <- TRUE # set to FALSE if you don't want the logs to be printed to the console

# write results to table? -----------------------------------------------------------------------------
writeToTable <- TRUE # set to FALSE if you want to skip writing to a SQL table in the results schema

# specify the name of the results table (used when writeToTable = TRUE and when sqlOnlyIncrementalInser
writeTableName <- "dqdashboard_results"

# write results to a csv file? ------------------------------------------------------------------------
writeToCsv <- FALSE # set to FALSE if you want to skip writing to csv file
csvFile <- "" # only needed if writeToCsv is set to TRUE

# if writing to table and using Redshift, bulk loading can be initialized -----------------------------

# Sys.setenv("AWS_ACCESS_KEY_ID" = "",
#            "AWS_SECRET_ACCESS_KEY" = "",
#            "AWS_DEFAULT_REGION" = "",
#            "AWS_BUCKET_NAME" = "",
#            "AWS_OBJECT_KEY" = "",
#            "AWS_SSE_TYPE" = "AES256",
#            "USE_MPP_BULK_LOAD" = TRUE)

# which DQ check levels to run ------------------------------------------------------------------------
checkLevels <- c("TABLE", "FIELD", "CONCEPT")
```

```r
# which DQ checks to run? -----------------------------------
checkNames <- c() # Names can be found in inst/csv/OMOP_CDM_v5.3_Check_Descriptions.csv

# which DQ severity levels to run? ----------------------------
checkSeverity <- c("fatal", "convention", "characterization")

# want to EXCLUDE a pre-specified list of checks? run the following code:
#
# checksToExclude <- c() # Names of check types to exclude from your DQD run
# allChecks <- DataQualityDashboard::listDqChecks()
# checkNames <- allChecks$checkDescriptions %>%
#   subset(!(checkName %in% checksToExclude)) %>%
#   select(checkName)

# which CDM tables to exclude? ------------------------------------
tablesToExclude <- c("CONCEPT", "VOCABULARY", "CONCEPT_ANCESTOR", "CONCEPT_RELATIONSHIP", "CONCEPT_CLASS

# run the job -----------------------------------------------------------------------------------
DataQualityDashboard::executeDqChecks(connectionDetails = connectionDetails,
                            cdmDatabaseSchema = cdmDatabaseSchema,
                            resultsDatabaseSchema = resultsDatabaseSchema,
                            cdmSourceName = cdmSourceName,
                            cdmVersion = cdmVersion,
                            numThreads = numThreads,
                            sqlOnly = sqlOnly,
                            sqlOnlyUnionCount = sqlOnlyUnionCount,
                            sqlOnlyIncrementalInsert = sqlOnlyIncrementalInsert,
                            outputFolder = outputFolder,
                            outputFile = outputFile,
                            verboseMode = verboseMode,
                            writeToTable = writeToTable,
                            writeToCsv = writeToCsv,
                            csvFile = csvFile,
                            checkLevels = checkLevels,
                  checkSeverity = checkSeverity,
                            tablesToExclude = tablesToExclude,
                            checkNames = checkNames)

# inspect logs ----------------------------------------------------------------------------
ParallelLogger::launchLogViewer(logFileName = file.path(outputFolder, cdmSourceName,
                                                        sprintf("log_DqDashboard_%s.txt", cdmSourceName)))

# (OPTIONAL) if you want to write the JSON file to the results table separately -----------------------
jsonFilePath <- ""
DataQualityDashboard::writeJsonResultsToTable(connectionDetails = connectionDetails,
                                              resultsDatabaseSchema = resultsDatabaseSchema,
                                              jsonFilePath = jsonFilePath)
```

# 4 Viewing Results

**Launching Dashboard as Shiny App**

```r
# Use the fully-qualified path to the JSON results file
DataQualityDashboard::viewDqDashboard(jsonFilePath)
```

**Launching on a web server**

If you have npm installed:

1. Install http-server:

```
npm install -g http-server
```

2. Name the output file *results.json* and place it in inst/shinyApps/www
3. Go to inst/shinyApps/www, then run:

```
http-server
```

# 5 View checks

To see description of checks using R, execute the command below:

```
checks <- DataQualityDashboard::listDqChecks(cdmVersion = "5.3") # Put the version of the CDM you are us
```