Package ‘condusco’

October 12, 2022

Type Package
Title Query-Driven Pipeline Execution and Query Templates
Version 0.1.0
Author Roland Stevenson
Maintainer Roland Stevenson <roland@rmg-services.com>
Description Runs a function iteratively over each row of either a dataframe or the results of a query. Use the 'BigQuery' and 'DBI' wrappers to iteratively pass each row of query results to a function. If a field contains a 'JSON' string, it will be converted to an object. This is helpful for queries that return 'JSON' strings that represent objects. These fields can then be treated as objects by the pipeline.
License GPL-3
URL https://github.com/ras44/condusco
BugReports https://github.com/ras44/condusco/issues
Encoding UTF-8
LazyData true
Suggests knitr, rmarkdown, whisker, testthat, RSQLite
VignetteBuilder knitr
Depends R (>= 3.3.2), jsonlite, assertthat, bigrquery, DBI
RoxygenNote 6.0.1.9000
NeedsCompilation no
Repository CRAN
Date/Publication 2017-11-08 19:30:17 UTC

R topics documented:

run_pipeline .......................................................... 2
run_pipeline_dbig ..................................................... 2
run_pipeline_gbq ...................................................... 4

Index 6
run_pipeline_dbi

A wrapper for running pipelines with a DBI connection invocation query

Description
A wrapper for running pipelines with a DBI connection invocation query

Usage
run_pipeline_dbi(pipeline, query, con, ...)

---

run_pipeline

Runs user-provided pipeline for each row of arguments in parameters, converting any JSON strings to objects

Description
Runs user-provided pipeline for each row of arguments in parameters, converting any JSON strings to objects

Usage
run_pipeline(pipeline, parameters)

Arguments
pipeline User-provided function with one argument, a dataframe
parameters An dataframe of fields to convert to json

Examples

library(whisker)

run_pipeline(
  function(params){
    query <- "SELECT result FROM {{table_prefix}}_results;"
    whisker.render(query,params)
  },
  data.frame(
    table_prefix = c('batman', 'robin')
  )
)
run_pipeline_dbi

Arguments

- **pipeline**: User-provided function with one argument, one row of query results
- **query**: A query to execute via the DBI connection
- **con**: The DBI connection
- **...**: Additional arguments passed to dbSendQuery() and dbFetch()

Examples

```r
## Not run:
library(whisker)
library(RSQLite)

con <- dbConnect(RSQLite::SQLite(), " mem: ")
dbWriteTable(con, " mtcars", mtcars)

#for each cylinder count, count the number of top 5 hps it has
pipeline <- function(params){
  query <- "SELECT
  {{#list}}
  SUM(CASE WHEN hp='{{val}}' THEN 1 ELSE 0 END )as n_hp_{{val}},
  {{/list}}
cyl
FROM mtcars
GROUP BY cyl
;"

dbGetQuery(
  con,
  whisker.render(query, params)
)
}

#pass the top 5 most common hps as val params
run_pipeline_dbi(
  pipeline,
  SELECT "[" || GROUP_CONCAT("{{"val": " "} | | hp | | ""}") || "]" AS list
  FROM (SELECT
    CAST(hp as INTEGER) as HP,
    count(hp) as cnt
  FROM mtcars
  GROUP BY hp
  ORDER BY cnt DESC
  LIMIT 5
)"
)```

run_pipeline_gbq

A wrapper for running pipelines with a BigQuery invocation query

Description

A wrapper for running pipelines with a BigQuery invocation query

Usage

run_pipeline_gbq(pipeline, query, project, ...)

Arguments

pipeline User-provided function with one argument, one row of query results
query A query to execute in Google BigQuery
project The Google BigQuery project to bill
... Additional arguments passed to query_exec()

Examples

## Not run:
library(whisker)

#Set GBQ project
project <- ''

#Set the following options for GBQ authentication on a cloud instance
options("httr_oauth_cache" = "/.httr-oauth")
options(httr_oob_default=TRUE)

#Run the below query to authenticate and write credentials to .httr-oauth file
query_exec("SELECT 'foo' as bar",project=project);

pipeline <- function(params){

  query <- "
  SELECT
    {{#list}}
    SUM(CASE WHEN author.name ='{name}' THEN 1 ELSE 0 END) as n{name_clean},
  '''

  dbDisconnect(con)

  ## End(Not run)
run_pipeline_gbq

```r
{{/list}}
repo_name
FROM 'bigquery-public-data.github_repos.sample_commits'
GROUP BY repo_name
;

res <- query_exec(
  whisker.render(query, params),
  project = project,
  use_legacy_sql = FALSE
);

print(res)
}

run_pipeline_gbq(pipeline, 
  SELECT CONCAT('
',
    STRING_AGG(
      CONCAT('{{"name":"', name, '","name_clean":"', REGEXP_REPLACE(name, r'[^[:alpha:]]', ''), '\"'}
    ),
    '
  ) as list
FROM ( 
  SELECT author.name,
    COUNT(commit) n_commits
  FROM 'bigquery-public-data.github_repos.sample_commits'
  GROUP BY 1
  ORDER BY 2 DESC
  LIMIT 10
  )
  ,
  project,
  use_legacy_sql = FALSE
)

## End(Not run)
```
Index

run_pipeline, 2
run_pipeline_db, 2
run_pipeline_gbq, 4