Package 'sqlcaser'

September 9, 2025

Title 'SQL' Case Statement Generator

Version 0.2.1

Date 2025-9-8

Author Leoson Hoay [aut, cre] (ORCID: <https://orcid.org/0000-0003-2079-5579>)

Maintainer Leoson Hoay <leoson.public@gmail.com>

Description Includes built-in methods for generating long 'SQL' CASE statements, and other 'SQL' statements that may otherwise be arduous to construct by hand.The generated statement can easily be concatenated to string literals to form queries to 'SQL'-like databases, such as when using the 'RODBC' package. The current methods include casewhen() for building CASE statements, inlist() for building IN statements, and updatetable() for building UPDATE statements.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2025-09-09 07:20:02 UTC

Contents

casewhen *Generate a SQL CASE statement from a mapping file*

Description

This function constructs a CASE..WHEN..THEN statement from a mapping file or dataframe. By default, it uses the first column for WHEN values and second column for THEN values, but you can specify different columns.

Usage

```
casewhen(
  inputfile = NULL,
  header = FALSE,
  when_col = 1,
  then_col = 2,
  else_value = NULL,
  quote_type = "single",
  handle_nulls = "skip"
)
```

Arguments

inputfile	Mapping dataframe OR path to the mapping file
header	If reading a csv file, TRUE if the file includes a header row, FALSE if it does not include a header row.
when_col	Column name or index for WHEN values (default: 1)
then_col	Column name or index for THEN values (default: 2)
else_value	Optional ELSE value for the CASE statement
quote_type	Type of quotes to use: "single", "double", or "auto" (default: "single")
handle_nulls	How to handle NULL/NA values: "skip", "null", or "error" (default: "skip")

Value

A string that represents the constructed CASE statement

Examples

```
input <- data.frame(Training = c("Strength", "Stamina", "Other"),
                    Duration = c(60, 30, 45))
result <- casewhen(inputfile = input, when_col = "Training",
                   then_col = "Duration", else_value = "Unknown")
```

`inlist` *Generate a SQL IN statement from a mapping file*

Description

This function constructs an IN statement from a mapping file or dataframe. By default it uses the first column, but you can specify a different column by name or index.

Usage

```
inlist(
  inputfile = NULL,
  header = FALSE,
  value_col = 1,
  quote_type = "single",
  handle_nulls = "skip",
  distinct = TRUE
)
```

Arguments

`inputfile`	Dataframe OR path to the mapping file
`header`	If reading a csv file, TRUE if the file includes a header row, FALSE if it does not include a header row.
`value_col`	Column name or index for IN values (default: 1)
`quote_type`	Type of quotes to use: "single", "double", or "auto" (default: "single")
`handle_nulls`	How to handle NULL/NA values: "skip", "null", or "error" (default: "skip")
`distinct`	Remove duplicate values if TRUE (default: TRUE)

Value

A string that represents the constructed IN statement

Examples

```
input <- data.frame(Training = c("Strength", "Stamina", "Other"))
result <- inlist(inputfile = input, value_col = "Training")
```

updatetable *Generate a SQL UPDATE statement from a mapping file*

Description

This function constructs UPDATE statements from a mapping file or dataframe. By default, it uses the first column as the key column for WHERE clauses, and updates all other columns. You can specify which columns to use.

Usage

```
updatetable(
  inputfile = NULL,
  tablename = NULL,
  key_col = 1,
  update_cols = NULL,
  quote_type = "auto",
  handle_nulls = "skip",
  batch_updates = TRUE
)
```

Arguments

inputfile	Dataframe OR path to the mapping file
tablename	Name of the SQL table
key_col	Column name or index for WHERE clause key (default: 1)
update_cols	Vector of column names/indices to update (default: all except key_col)
quote_type	Type of quotes to use: "single", "double", or "auto" (default: "auto")
handle_nulls	How to handle NULL/NA values: "skip", "null", or "error" (default: "skip")
batch_updates	If TRUE, create one UPDATE per row; if FALSE, create one per column (default: TRUE)

Value

A string that represents the constructed UPDATE statement(s)

Examples

```
input <- data.frame(id = c(1, 2, 3), name = c("John", "Jane", "Bob"),
                    age = c(25, 30, 35))
result <- updatetable(input, "users", key_col = "id", update_cols = c("name", "age"))
```

Index