Package ‘queryparser’

May 10, 2020

Type Package
Title Translate 'SQL' Queries into 'R' Expressions
Version 0.3.0
Maintainer Ian Cook <ian@cloudera.com>
Description Translate 'SQL' 'SELECT' statements into lists of 'R' expressions.
URL https://github.com/ianmcook/queryparser
BugReports https://github.com/ianmcook/queryparser/issues
NeedsCompilation no
License Apache License 2.0
Encoding UTF-8
RoxygenNote 7.0.2

Collate 'compat.R' 'agg_scalar.R' 'check_expressions.R'
    'column_references.R' 'common.R' 'extract_alias.R'
    'parse_clauses.R' 'parse_expression.R' 'parse_join.R'
    'translations.R' 'process_translations.R' 'parse_query.R'
    'parse_table_reference.R' 'replace.R' 'secure.R'
    'split_query.R' 'squish_sql.R' 'translate.R' 'unpipe.R'
    'unqualify.R' 'wrap_bangs.R'

Suggests testthat (>= 2.1.0), covr (>= 3.2.0)
Author Ian Cook [aut, cre],
    Cloudera [cph]
Repository CRAN
Date/Publication 2020-05-10 06:30:02 UTC

R topics documented:

    column_references ............................................. 2
    extract_alias .................................................. 3
    parse_expression .............................................. 3
    parse_query ................................................... 4
Description

Returns a character vector containing all the column references in the clauses of a parsed SQL SELECT statement.

Usage

column_references(tree, from = TRUE)

Arguments

tree  
a list returned by `parse_query` containing named elements representing the clauses of a SQL SELECT statement

from  
a logical value indicating whether to include the column references from the join conditions in the FROM clause

Details

The returned character vector includes only column references, not table references. Column aliases assigned in the SELECT list are not included unless they are used in other clauses.

Value

A character vector containing all the unique column references found in the SELECT, FROM (if `from` = TRUE), WHERE, GROUP BY, HAVING, and ORDER BY clauses of the SELECT statement.

See Also

`parse_query`

Examples

```r
my_query <- "SELECT f.flight, manufacturer, p.model"
FROM flights f
JOIN planes p USING (tailnum);

column_references(parse_query(my_query), from = FALSE)
```
**extract_alias**  
Extract the column alias from a SQL expression

**Description**
Extracts the column alias assignment from an expression used in the SELECT list of a SQL query

**Usage**
```r
extract_alias(expr)
```

**Arguments**
- `expr`: a character string containing a SQL expression which might have a column alias assignment at the end

**Details**
The expression must not contain any unquoted whitespace characters except spaces, and there must be no unquoted runs or two or more spaces. Use `squish_sql` to satisfy this whitespace requirement. `queryparser` also uses this function internally to extract table aliases used in the FROM clause.

**Value**
a character string containing the inputted SQL expression with the column alias assignment removed (if it existed) and with the assigned alias as its name

**Examples**
```r
eexpr <- "round(AVG(arr_delay)) AS avg_delay"
extract_alias(eexpr)
```

---

**parse_expression**  
Parse a SQL expression

**Description**
Parses a SQL expression into an R expression

**Usage**
```r
parse_expression(expr, tidyverse = FALSE, secure = TRUE)
```
Arguments

expr  a character string containing a SQL expression
tidyverse  set to TRUE to use functions from tidyverse packages including dplyr, stringr, and lubridate in the returned R expression
secure  set to FALSE to allow potentially dangerous functions in the returned R expression

Details

The expression must not end with a column alias assignment. Use extract_alias to extract and remove column alias assignments.

The expression must not contain any unquoted whitespace characters except spaces, and there must be no unquoted runs or two or more spaces. The expression must not contain line comments (--) or block comments (/* */). Use squish_sql to satisfy these whitespace requirements and remove any comments.

Value

an unevaluated R expression (a call)

See Also

parse_query

Examples

expr <- "round(AVG(arr_delay))"
parse_expression(expr)

Description

Parse a SQL query

Usage

parse_query(query, tidyverse = FALSE, secure = TRUE)

Arguments

query  a character string containing a SQL SELECT statement
 tidyverse  set to TRUE to use functions from tidyverse packages including dplyr, stringr, and lubridate in the R expressions
 secure  set to FALSE to allow potentially dangerous functions in the returned R expressions
Details

See the current limitations section of the README for information about what types of queries are supported.

Value

A list object with named elements representing the clauses of the query, containing sublists of unevaluated R expressions translated from the SQL expressions in the query.

Depending on the arguments, the returned list and its sublists will have attributes named distinct and aggregate with logical values that can aid in the evaluation of the R expressions. If query contains one or more joins, then the sublist named from will have attributes named join_types and join_conditions specifying the types of join and the join conditions.

See Also

parse_expression

Examples

my_query <- "SELECT origin, dest,
  COUNT(flight) AS num_flts,
  round(AVG(distance)) AS dist,
  round(AVG(arr_delay)) AS avg_delay
FROM flights
WHERE distance BETWEEN 200 AND 300
  AND air_time IS NOT NULL
GROUP BY origin, dest
HAVING num_flts > 3000
ORDER BY num_flts DESC, avg_delay DESC
LIMIT 100;"

parse_query(my_query)

parse_query(my_query, tidyverse = TRUE)
Arguments

query a character string containing a SQL SELECT statement
tidyverse for queryparser internal use only

Value

A list object with named elements representing the clauses of the query

See Also

parse_query

Examples

my_query <- "SELECT origin, dest,
    COUNT(flight) AS num_flts,
    round(AVG(distance)) AS dist,
    round(AVG(arr_delay)) AS avg_delay
FROM flights
WHERE distance BETWEEN 200 AND 300
    AND air_time IS NOT NULL
GROUP BY origin, dest
HAVING num_flts > 3000
ORDER BY num_flts DESC, avg_delay DESC
LIMIT 100;"

split_query(my_query)

---

**squish_sql**

Squish a SQL query or SQL expression

**Description**

Replaces every unquoted run of whitespace characters with a single space and removes all line comments (--) and block comments (/\* \*/). Whitespace and comment marks within quotes are not modified.

**Usage**

squish_sql(x)

**Arguments**

x a character string containing a SQL query or expression

**Value**

a character string containing the squished query or expression with comments removed
unqualify_query  

Remove prefixes from column references in a parsed SQL query

Description

Unqualifies column references in the clauses of a parsed SQL SELECT statement that begin with any of the specified prefixes followed by a dot.

Usage

unqualify_query(tree, prefixes, except = character(0))

Arguments

tree
a list returned by parse_query containing named elements representing the clauses of a SQL SELECT statement

prefixes
a character vector containing one or more table names or table aliases

except
a character vector containing column references to leave as is (optional)

Details

In the returned list, the FROM clause is unmodified and column alias assignments made in the SELECT clause are unmodified.

Value

A list the same as tree but with all column references in the SELECT, WHERE, GROUP BY, HAVING, and ORDER BY clauses unqualified, except those in except.

See Also

parse_query

Examples

my_query <- "SELECT f.flight,  
    manufacturer, p.model  
FROM flights f  
JOIN planes p USING (tailnum);"

unqualify_query(  
    parse_query(my_query),  
    prefixes = c("p", "f")
  )
Index

call, 4
column_references, 2

extract_alias, 3, 4

parse_expression, 3, 5
parse_query, 2, 4, 6, 7

split_query, 5
squish_sql, 3, 4, 6

unqualify_query, 7