Package: sqlfluffr (via r-universe)

May 31, 2026

Type Package

Title Wrapper to the 'SQL' Linter and Formatter 'sqlfluff'

Version 0.1.0

Description An R interface to the 'Python' 'sqlfluff' 'SQL' linter and formatter via the 'reticulate' package. Enables linting, fixing, and parsing of 'SQL' queries with support for multiple dialects. Includes special handling for 'glue' 'SQL' syntax with curly-brace placeholders.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0)

Imports reticulate

Suggests testthat (>= 3.2.0), glue, rstudioapi, withr

Config/testthat/edition 3

RoxygenNote 7.3.2

URL https://github.com/brendensm/sqlfluffr

BugReports https://github.com/brendensm/sqlfluffr/issues

Config/pak/sysreqs libpng-dev python3

Repository https://brendensm.r-universe.dev

Date/Publication 2026-03-02 17:03:32 UTC

RemoteUrl https://github.com/brendensm/sqlfluffr

RemoteRef HEAD

RemoteSha 782d30e6c8fe4c8bfd2f6f986f91945d2d15455f

Contents

`sqlf_config` *Write a project-level sqlfluff configuration file*

Description

Writes a ‘.sqlfluff‘ configuration file that sqlfluff automatically discovers. Once written, [sqlf_lint()], [sqlf_fix()], and [sqlf_parse()] will use these settings without needing explicit arguments.

Usage

```
sqlf_config(
  dialect = NULL,
  rules = NULL,
  exclude_rules = NULL,
  max_line_length = NULL,
  glue = NULL,
  path = ".sqlfluff",
  overwrite = FALSE,
  ...
)
```

Arguments

`dialect`	SQL dialect name (e.g. ‘"ansi"‘, ‘"bigquery"‘, ‘"postgres"‘, ‘"teradata"‘). See [sqlf_dialects()] for available options.
`rules`	Character vector of rule codes to enable.
`exclude_rules`	Character vector of rule codes to exclude.
`max_line_length`	Maximum allowed line length.
`glue`	If ‘TRUE‘, enables [glue::glue_sql()] placeholder handling by default for all lint, fix, and parse calls.
`path`	File path to write. Defaults to ‘".sqlfluff"‘ in the current working directory.
`overwrite`	If ‘FALSE‘ (default), refuses to overwrite an existing file. Set to ‘TRUE‘ to replace it.
`...`	Additional settings as named arguments. These are added to the ‘[sqlfluff]‘ section of the configuration.

Value

The file path, invisibly.

Examples

```
path <- tempfile(fileext = ".sqlfluff")
sqlf_config(dialect = "postgres", path = path)
readLines(path)
unlink(path)
```

`sqlf_config_edit` *Open the sqlfluff configuration file for editing*

Description

Opens the '.sqlfluff' configuration file in the default editor. In RStudio, this opens the file in the source pane.

Usage

```
sqlf_config_edit(path = ".sqlfluff")
```

Arguments

`path`	Path to the configuration file. Defaults to '".sqlfluff"'.

Value

The file path, invisibly.

Examples

```
## Not run:  # interactive option to edit config file
sqlf_config_edit()

## End(Not run)
```

sqlf_dialects *List available SQL dialects*

Description

Returns the dialects supported by the installed version of sqlfluff.

Usage

```
sqlf_dialects()
```

Value

A data.frame with columns ‘label‘, ‘name‘, and ‘inherits_from‘.

Examples

```
sqlf_dialects()
```

sqlf_fix *Fix a SQL query*

Description

Automatically fixes style and syntax violations in a SQL string or file using sqlfluff.

Usage

```
sqlf_fix(
  sql = NULL,
  file = NULL,
  dialect = NULL,
  rules = NULL,
  exclude_rules = NULL,
  config = NULL,
  glue = NULL,
  overwrite = FALSE,
  cat = !overwrite
)
```

Arguments

sql	A SQL string to fix.
file	Path to a SQL file to fix.
dialect	SQL dialect (e.g. ‘"ansi"‘, ‘"bigquery"‘, ‘"postgres"‘).
rules	Character vector of rule codes to apply.
exclude_rules	Character vector of rule codes to skip.
config	A [sqlf_config()] object.
glue	If ‘TRUE‘, treat ‘{var}‘ placeholders as ‘glue::glue_sql‘ variables and preserve them in the fixed output. ‘NULL‘ (the default) reads the ‘glue‘ setting from the project ‘.sqlfluff‘ config file; ‘FALSE‘ explicitly disables glue handling regardless of project config.
overwrite	If ‘TRUE‘ and ‘file‘ was provided, overwrite the file with fixed SQL. If ‘FALSE‘ (default), the fixed SQL is returned without modifying the file.
cat	If ‘TRUE‘ (the default when ‘overwrite‘ is ‘FALSE‘), print the fixed SQL to the console with [cat()] for easy copy-paste. The fixed string is still returned invisibly.

Value

The fixed SQL string (invisibly when printed via ‘cat‘).

Common parsing issues

sqlfluff cannot fix SQL that fails to parse. When this happens, the original SQL is returned unchanged and a warning is issued. Common causes include:

- Missing parentheses after IN: use ‘WHERE x IN (1)‘ not ‘WHERE x IN 1‘.
- "IS NOT IN" syntax: use ‘WHERE x NOT IN (...)‘ instead of ‘WHERE x IS NOT IN (...)‘.
- Glue placeholders without ‘glue = TRUE‘: if your SQL contains ‘{var}‘ placeholders from [glue::glue_sql()], pass ‘glue = TRUE‘ so sqlfluff treats them as template variables rather than syntax errors.

Examples

```
sqlf_fix(sql = "SELECT  a,b from t where x=1\n")
```

sqlf_install *Install sqlfluff Python package*

Description

Sets up a dedicated Python virtual environment and installs the 'sqlfluff' package into it. This is a one-time setup step that must be run before using any sqlfluffr functions.

Usage

```
sqlf_install(ask = interactive(), force = FALSE)
```

Arguments

ask	If 'TRUE' (the default in interactive sessions), prompt for confirmation before downloading. Set to 'FALSE' to skip the prompt.
force	If 'TRUE', reinstall even if 'sqlfluff' is already present.

Value

'NULL', invisibly.

Examples

```
## Not run:
sqlf_install()

## End(Not run)
```

sqlf_lint *Lint a SQL query*

Description

Checks a SQL string or file for style and syntax violations using sqlfluff.

Usage

```
sqlf_lint(
  sql = NULL,
  file = NULL,
  dialect = NULL,
  rules = NULL,
  exclude_rules = NULL,
  config = NULL,
  glue = NULL
)
```

Arguments

sql	A SQL string to lint.
file	Path to a SQL file to lint.
dialect	SQL dialect (e.g. ‘"ansi"‘, ‘"bigquery"‘, ‘"postgres"‘).
rules	Character vector of rule codes to check.
exclude_rules	Character vector of rule codes to skip.
config	A [sqlf_config()] object.
glue	If ‘TRUE‘, treat ‘{var}‘ placeholders as ‘glue::glue_sql‘ variables and use the placeholder templater so linting works correctly. ‘NULL‘ (the default) reads the ‘glue‘ setting from the project ‘.sqlfluff‘ config file; ‘FALSE‘ explicitly disables glue handling regardless of project config.

Value

A data.frame of class ‘"sqlf_lint_results"‘ with columns ‘line_no‘, ‘line_pos‘, ‘code‘, ‘description‘, and ‘name‘. Returns a zero-row data.frame if there are no violations.

Examples

```
sqlf_lint(sql = "SELECT  a,b from t where x=1\n")
sqlf_lint(sql = "SELECT TOP 10 * FROM t\n", dialect = "tsql")
```

---

sqlf_parse *Parse a SQL query*

Description

Parses a SQL string or file into a syntax tree using sqlfluff.

Usage

```
sqlf_parse(sql = NULL, file = NULL, dialect = NULL, config = NULL, glue = NULL)
```

Arguments

sql	A SQL string to parse.
file	Path to a SQL file to parse.
dialect	SQL dialect (e.g. ‘"ansi"‘, ‘"bigquery"‘, ‘"postgres"‘).
config	A [sqlf_config()] object.
glue	If ‘TRUE‘, treat ‘{var}‘ placeholders as ‘glue::glue_sql‘ variables before parsing. ‘NULL‘ (the default) reads the ‘glue‘ setting from the project ‘.sqlfluff‘ config file; ‘FALSE‘ explicitly disables glue handling regardless of project config.

Value

A nested list representing the parse tree.

Examples

```
sqlf_parse(sql = "SELECT 1\n")
```

---

sqlf_rules *List available linting rules*

Description

Returns the rules available in the installed version of sqlfluff.

Usage

```
sqlf_rules()
```

Value

A data.frame with columns ‘code‘ and ‘description‘.

Examples

```
sqlf_rules()
```

Index