Package 'pointblank'

August 22, 2017

Type Package

Title Validation of Local and Remote Data Tables

Version 0.1

Description Validate data in local data frames, local 'tibble' objects, in 'CSV' and 'TSV' files, and in database tables ('PostgreSQL' and 'MySQL'). Validation pipelines can be made using easily-readable, consecutive validation steps and such pipelines allow for switching of the data table context. Upon execution of the validation plan, several reporting options are available. User-defined thresholds for failure rates allow for the determination of appropriate reporting actions (e.g., sending email notifications).

Depends R (>= 3.4.0)

License MIT + file LICENSE

Imports digest (>= 0.6.12), dplyr (>= 0.7.2), Hmisc (>= 4.0-3), htmltools (>= 0.3.6), knitr (>= 1.15.1), lazyWeave (>= 3.0.1), lubridate (>= 1.6.0), magrittr (>= 1.5), mailR (>= 0.4.1), pixiedust (>= 0.7.5), purrr (>= 0.2.3), readr (>= 1.1.1), rJava (>= 0.9-8), rlang (>= 0.1.1), rmarkdown (>= 1.6), stringr (>= 1.2.0), tibble (>= 1.3.3), tidyr (>= 0.6.3)

Suggests testthat

URL https://github.com/rich-iannone/pointblank

BugReports https://github.com/rich-iannone/pointblank/issues

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation no

Author Rich Iannone [aut, cre]

Maintainer Rich Iannone <riannone@me.com>

Repository CRAN

Date/Publication 2017-08-22 15:26:31 UTC

R topics documented:

all_cols *With any 'all_cols()' call, return a wildcard operator*

Description

With any 'all_cols()' call, return a wildcard operator

Usage

```
all_cols()
```

all_passed *Did all of the validations pass?*

Description

Given a completed validation pass, this function will return `TRUE` or `FALSE` on whether all of the validations passed without any fails.

Usage

```
all_passed(agent)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.

Value

an agent object.

col_exists *Verify that one or more columns exist*

Description

Set a verification step that checks whether one or several specified columns exist in the target table.

Usage

```
col_exists(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column or multiple columns in the same table.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.
description	an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Validate that column `a` exists in
# the `small_table` CSV file; do this
```

```
# by creating an agent, focussing on
# that table, creating a `col_exists()`
# step, and then interrogating the table
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_exists(column = a) %>%
  interrogate()

# Determine if this column validation
# passed by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_character` *Verify that a column contains character values*

Description

Set a verification step where a table column is expected to consist of floating point values.

Usage

```
col_is_character(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.

notify_fraction
: the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name
: the name of the local or remote table.

db_type
: if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file
: if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql
: when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path
: an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types
: if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions
: an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description
: an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Validate that the `b` column in the
# `small_table` CSV file is classed as
# `character`
agent <-
  create_agent() %>%
  focus_on(
```

```
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_is_character(
    column = b) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_date` *Verify that a column contains R Date objects*

Description

Set a verification step where a table column is expected to consist entirely of R `Date` objects.

Usage

```
col_is_date(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Validate that the `date` column in the
# `small_table` CSV file is classed as
# `Date`
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_is_date(column = date) %>%
```

```
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_factor` *Verify that a column contains R factor objects*

Description

Set a verification step where a table column is expected to consist entirely of R `factor` objects.

Usage

```
col_is_factor(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `factor`
df <-
  data.frame(
    a = c("one", "two"))

# Validate that column `a`
# in the data frame is classed
# as a factor
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_factor(column = a) %>%
  interrogate()
```

```
# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_is_integer` *Verify that a column contains integer values*

Description

Set a verification step where a table column is expected to consist of integer values.

Usage

```
col_is_integer(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.
`creds_file`	if a connection to a database is required for reaching the table specified in `tbl_name`, then a path to a credentials file can be used to establish that connection. The credentials file is an `RDS` containing a character vector with the following items in the specified order: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

`initial_sql` when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

`preconditions` an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.

`description` an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the `html_summary` function.

Value

an agent object.

Examples

```
# Validate that the `a` column in
# the `small_table` CSV file is
# classed as `integer`
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_is_integer(
    column = a) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_is_logical *Verify that a column contains R logical objects*

Description

Set a verification step where a table column is expected to consist entirely of R `logical` objects.

Usage

```
col_is_logical(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the name of a single table column, multiple columns in the same table, or, a helper function such as `all_cols()`.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.
`creds_file`	if a connection to a database is required for reaching the table specified in `tbl_name`, then a path to a credentials file can be used to establish that connection. The credentials file is an `RDS` containing a character vector with the following items in the specified order: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.
`initial_sql`	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

`preconditions` an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.

`description` an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the `html_summary` function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `logical`
df <-
  data.frame(
    a = c(TRUE, FALSE))

# Validate that column `a` in
# the data frame is classed as
# `logical`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_logical(column = a) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_is_numeric *Verify that a column contains numeric values*

Description

Set a verification step where a table column is expected to consist of floating point values.

Usage

```
col_is_numeric(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column, multiple columns in the same table, or, a helper function such as all_cols().
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

`preconditions` an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.

`description` an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the `html_summary` function.

Value

an agent object.

Examples

```
# Validate that column `d` in
# the `small_table` CSV file
# is classed as `numeric`
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_is_numeric(
    column = d) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_is_posix *Verify that a column contains POSIXct dates*

Description

Set a verification step where a table column is expected to consist entirely of POSIXct dates.

Usage

```
col_is_posix(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the name of a single table column, multiple columns in the same table, or, a helper function such as all_cols().
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

`preconditions` an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.

`description` an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the `html_summary` function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column containing data
# classed as `POSIXct`
df <-
  data.frame(
    a = as.POSIXct(
      strptime(
        "2011-03-27 01:30:00",
        "%Y-%m-%d %H:%M:%S")))

# Validate that column `a` in
# the data frame is classed as
# `POSIXct`
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_is_posix(column = a) %>%
  interrogate()

# Determine if this column
# validation has passed by
# using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_vals_between *Verify whether column data are between two values*

Description

Set a verification step where column data should be between two values.

Usage

```
col_vals_between(agent, column, left, right, preconditions = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL, description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
left	the lower bound for the range. The validation includes this bound value in addition to values greater than left.
right	the upper bound for the range. The validation includes this bound value in addition to values lower than right.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.
description	an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5.6, 8.2, 6.3, 7.8, 3.4))

# Validate that values in
# column `a` are all between
# 1 and 9
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_between(
    column = a,
    left = 1,
    right = 9) %>%
  interrogate()

# Determine if this column
```

```
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_equal` *Verify whether numerical column data are equal to a specified value*

Description

Set a verification step where numeric values in a table column should be equal to a specified value.

Usage

```
col_vals_equal(agent, column, value, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`value`	a numeric value used to test for equality.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with 2 columns of numerical values
df <-
  data.frame(
    a = c(1, 1, 1, 2, 2, 2),
    b = c(5, 5, 5, 3, 6, 3))

# Validate that values in column
# `b` are equal to 5 when values
# in column `a` are equal to 1
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_equal(
    column = b,
```

```
    value = 5,
    preconditions = a == 1) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_vals_gt *Verify whether numerical column data are greater than a specified value*

Description

Set a verification step where numeric values in a table column should be greater than a specified value.

Usage

```
col_vals_gt(agent, column, value, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used for this test. Any column values > value are considered passing.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
description	an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))

# Validate that values in column
# `a` are always greater than 4
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
```

```
  col_vals_gt(
    column = a,
    value = 4) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_gte` *Verify whether numerical column data are greater than or equal to a specified value*

Description

Set a verification step where numeric values in a table column should be greater than or equal to a specified value.

Usage

```
col_vals_gte(agent, column, value, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`value`	a numeric value used for this test. Any column values `>= value` are considered passing.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.

notify_fraction
: the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name
: the name of the local or remote table.

db_type
: if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file
: if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql
: when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path
: an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types
: if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions
: an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description
: an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5, 7, 6, 5, 8, 7))
```

```
# Validate that values in column
# `a` are always greater than or
# equal to 5
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_gte(
    column = a,
    value = 5) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_in_set` *Verify whether column data are part of a set of values*

Description

Set a verification step where numeric values in a table column should be part of a set of values.

Usage

```
col_vals_in_set(agent, column, set, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`set`	a vector of numeric or string-based elements, where column values found within this `set` will be considered as passing.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame with
# 2 columns: one with numerical
# values and the other with strings
```

```
df <-
  data.frame(
    a = c(1, 2, 3, 4),
    b = c("one", "two", "three", "four"),
    stringsAsFactors = FALSE)

# Validate that all numerical values
# in column `a` belong to a numerical
# set, and, create an analogous
# validation check for column `b` with
# a set of string values
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_in_set(
    column = a,
    set = 1:4) %>%
  col_vals_in_set(
    column = b,
    set = c("one", "two",
            "three", "four")) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_lt` *Verify whether numerical column data are less than a specified value*

Description

Set a verification step where numeric values in a table column should be less than a specified value.

Usage

```
col_vals_lt(agent, column, value, preconditions = NULL, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.

`column` the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).

`value` a numeric value used for this test. Any column values `< value` are considered passing.

`preconditions` an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return `TRUE` or `FALSE` for every row evaluated, where rows evaluated as `TRUE` are the rows that are retained for the validation step). For example, if a table has columns `a`, `b`, and `c`, and, column `a` has numerical data, we can write a statement `a < 5` that filters all rows in the table where values in column a are less than five.

`warn_count` the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.

`notify_count` the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.

`warn_fraction` the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.

`notify_fraction`
the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.

`tbl_name` the name of the local or remote table.

`db_type` if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

`creds_file` if a connection to a database is required for reaching the table specified in `tbl_name`, then a path to a credentials file can be used to establish that connection. The credentials file is an `RDS` containing a character vector with the following items in the specified order: (1) database name (`dbname`), (2) the `host` name, (3) the `port`, (4) the username (`user`), and (5) the `password`. This file can be easily created using the `create_creds_file()` function.

`initial_sql` when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial `SELECT...` statement can be omitted for simple queries (e.g., `WHERE a > 1 AND b = 'one'`).

`file_path` an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

`col_types` if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: `c` -> character, `i` -> integer, `n` -> number, `d` -> double, `l` -> logical, `D` -> date, `T` -> date time, `t` -> time, `?` -> guess, or `_/-`, which skips the column.

description an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the `html_summary` function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5, 4, 3, 5, 1, 2))

# Validate that values in
# column `a` are always less
# than 6
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_lt(
    column = a,
    value = 6) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_vals_lte *Verify whether numerical column data are less than or equal to a specified value*

Description

Set a verification step where numeric values in a table column should be less than or equal to a specified value.

Usage

```
col_vals_lte(agent, column, value, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used for this test. Any column values <= value are considered passing.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a

table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column of numerical
# values
df <-
  data.frame(
    a = c(5, 4, 3, 5, 1, 2),
    b = c(3, 2, 4, 3, 5, 6))

# Validate that the sum of
# values across columns `a`
# and `b` are always less
# than or equal to 10
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_lte(
    column = a + b,
    value = 10) %>%
  interrogate()

# Determine if this column
# validation has passed by using
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

---

col_vals_not_between *Verify that column data are not between two values*

Description

Set a verification step where column data should not be between two values.

Usage

```
col_vals_not_between(agent, column, left, right, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL,
  preconditions = NULL, description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
left	the lower bound for the range. The validation includes this bound value in addition to values greater than left. Any values >= left and <= right will be considered as failing.
right	the upper bound for the range. The validation includes this bound value in addition to values lower than right. Any values <= right and >= left will be considered as failing.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
description	an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with a column a numerical values
df <-
  data.frame(
    a = c(5.6, 8.2, 6.3, 7.8, 3.4))

# Validate that none of the values
# in column `a` are between 9 and 10,
# or, between 0 and 2
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_between(
    column = a,
    left = 9,
    right = 10) %>%
  col_vals_not_between(
    column = a,
    left = 0,
    right = 2) %>%
  interrogate()

# Determine if these column
# validations have all passed by
# using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

col_vals_not_equal *Verify whether numerical column data are not equal to a specified value*

Description

Set a verification step where numeric values in a table column should not be equal to a specified value.

Usage

```
col_vals_not_equal(agent, column, value, warn_count = 1,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = NULL, db_type = NULL, creds_file = NULL,
  initial_sql = NULL, file_path = NULL, col_types = NULL,
  preconditions = NULL, description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
column	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., a + b or a + sum(a)).
value	a numeric value used to test for non-equality.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame
# with 2 columns of numerical values
df <-
  data.frame(
    a = c(1, 1, 1, 2, 2, 2),
    b = c(5, 5, 5, 3, 6, 3))

# Validate that values in
# column `b` are not equal to 5
# when values in column `a`
# are equal to 2
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_equal(
    column = b,
    value = 5,
    preconditions = a == 2) %>%
  interrogate()

# Determine if this column
# validation has passed by using
```

```
# `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_not_in_set` *Verify that column data are not part of a set of values*

Description

Set a verification step where numeric values in a table column should be part of a set of values.

Usage

```
col_vals_not_in_set(agent, column, set, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`set`	a vector of numeric or string-based elements, where column values found within this `set` will be considered as failing.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame with 2 columns: one
# with numerical values and the other with strings
df <-
  data.frame(
    a = c(1, 2, 3, 4),
    b = c("one", "two", "three", "four"),
    stringsAsFactors = FALSE)

# Validate that all numerical
# values in column `a` do not
# belong to a specified numerical
# set, and, create an analogous
# validation check for column `b`
# within a set of string values
agent <-
```

```
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_in_set(
    column = a,
    set = 7:10) %>%
  col_vals_not_in_set(
    column = b,
    set = c("seven", "eight",
            "nine", "ten")) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_not_null` *Verify whether all column values are not NULL*

Description

Set a verification step where no values in a table column are expected to be NULL.

Usage

```
col_vals_not_null(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.

notify_fraction
: the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name
: the name of the local or remote table.

db_type
: if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file
: if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql
: when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path
: an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types
: if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions
: an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description
: an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame with
# 2 columns: one with numerical
# values and the other with strings
df <-
  data.frame(
    a = c(1, 2, NA, NA),
```

```
    b = c(2, 2, 5, 5),
    stringsAsFactors = FALSE)

# Validate that all values in
# column `a` are not NULL when
# values in column `b` are equal
# to 2
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_not_null(
    column = a,
    preconditions = b == 2) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_null` *Verify whether all column values are NULL*

Description

Set a verification step where all values in a table column are expected to be NULL.

Usage

```
col_vals_null(agent, column, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame with
# 2 columns: one with numerical
# values and the other with strings
```

```
df <-
  data.frame(
    a = c(1, 2, NA, NA),
    b = c(2, 2, 5, 5),
    stringsAsFactors = FALSE)

# Validate that all values in
# column `a` are NULL when
# values in column `b` are
# equal to 5
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_null(
    column = a,
    preconditions = b == 5) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`col_vals_regex` *Verify whether string column data corresponds to a regex matching expression*

Description

Set a verification step where string column data should correspond to a regex matching expression.

Usage

```
col_vals_regex(agent, column, regex, warn_count = 1, notify_count = NULL,
  warn_fraction = NULL, notify_fraction = NULL, tbl_name = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  file_path = NULL, col_types = NULL, preconditions = NULL,
  description = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `a + b` or `a + sum(a)`).
`regex`	a regex pattern to test for matching strings.

warn_count — the threshold number for individual validations returning a FALSE result before applying the warn flag.

notify_count — the threshold number for individual validations returning a FALSE result before applying the notify flag.

warn_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.

notify_fraction — the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

tbl_name — the name of the local or remote table.

db_type — if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

preconditions — an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame with a column
# containing strings
df <-
  data.frame(
    a = c("s_0131", "s_0231",
          "s_1389", "s_2300"),
    stringsAsFactors = FALSE)

# Validate that all string values in
# column `a` match a regex statement
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_regex(
    column = a,
    regex = "^s_[0-9]{4}$") %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`create_agent` *Create an agent object*

Description

Creates an agent object.

Usage

```
create_agent(name = NULL, email_creds_file_path = NULL,
  notification_recipient_emails = NULL, notification_emails_active = FALSE)
```

Arguments

`name`	optional name for the agent that will eventually carry out the interrogation process.
`email_creds_file_path`	an optional path to an email credentials file.
`notification_recipient_emails`	an optional vector of email addresses to which notification emails should be sent.
`notification_emails_active`	an option to enable notification emails when tests trigger a `notify` status.

Value

an agent object.

Examples

```
# Create an `agent` object in order to begin
# defining validation steps
agent <- create_agent()

## Not run:
# Should notifications be required through
# email, we first create an email credentials
# file, with `create_email_creds_file()`, and
# then reference that file with `create_agent()`
create_email_creds_file(
  file = "~/.pb_notify",
  sender = "point@blank.org",
  host = "smtp.blank.org",
  port = 465,
  user = "point@blank.org",
  password = "************")

agent_notify <-
  create_agent(
  email_creds_file_path = "~/.pb_notify",
  notification_recipient_emails =
    c("a@b.net", "c@d.com", "e@f.org"),
  notification_emails_active = TRUE)

## End(Not run)

# Then, as with any `ptblank_agent` object,
# we can focus on different table, add
# validation steps, and then eventually use
# `interrogate()` to perform the validations
agent <-
  agent %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  col_exists(column = c("a", "b")) %>%
  interrogate()

# A basic summary can be produced using
# the `get_summary()` function
get_summary(agent)[, 1:7]
#> # A tibble: 2 x 7
#>      tbl_name    db_type assertion_type column value regex all_passed
#>         <chr>      <chr>          <chr>  <chr> <dbl> <chr>      <lgl>
```

```
#> 1 small_table local_file     col_exists      a    NA  <NA>       TRUE
#> 2 small_table local_file     col_exists      b    NA  <NA>       TRUE
```

`create_creds_file` *Create a file with DB access credentials*

Description

Creates a file containing access credentials for a database.

Usage

```
create_creds_file(file, dbname, host, port, user, password)
```

Arguments

`file`	a file path for the credentials file to be stored on disk.
`dbname`	the database name.
`host`	the `host` name.
`port`	the port number.
`user`	the username for the database
`password`	the password associated with the `user`.

Examples

```
## Not run:
# Create a credentials file for access to
# remote databases (where the tables to be
# validated reside); place in the user's
# home directory
create_creds_file(
  file = "~/.pb_credentials",
  dbname = "********",
  host = "***************",
  port = ***,
  user = "********",
  password = "*****************")

## End(Not run)
```

create_email_creds_file *Create a file with email access credentials*

Description

Creates a file with access credentials for the purpose of automatically emailing notification messages.

Usage

```
create_email_creds_file(file, sender, host, port, user, password,
  use_ssl = TRUE, authenticate = TRUE)
```

Arguments

file	a file path for the credentials file to be stored on disk.
sender	the sender name.
host	the host name.
port	the port number.
user	the username for the email account.
password	the password associated with the user's email address.
use_ssl	an option as to whether to use SSL; supply a TRUE or FALSE value (TRUE is the default value).
authenticate	an option as to whether to authenticate; supply a TRUE or FALSE value (TRUE is the default value).

Examples

```
## Not run:
# Create a credentials file for automatic
# email notifications; place in the user's
# home directory
create_email_creds_file(
  file = "~/.pb_notify",
  sender = "point@blank.org",
  host = "smtp.blank.org",
  port = 465,
  user = "point@blank.org",
  password = "************")

## End(Not run)
```

`create_validation_step` *Add properly formatted validation steps.*

Description

Add properly formatted validation steps.

Usage

```
create_validation_step(agent, assertion_type, column, value = NULL,
  set = NULL, regex = NULL, preconditions = NULL, warn_count = NULL,
  notify_count = NULL, warn_fraction = NULL, notify_fraction = NULL,
  tbl_name = as.character(NA), db_type = as.character(NA),
  creds_file = as.character(NA), init_sql = as.character(NA),
  file_path = as.character(NA), col_types = as.character(NA))
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`assertion_type`	a string providing the name of the validation function.
`column`	the column (or a set of columns, provided as a character vector) to which this validation should be applied. Aside from a single column name, column operations can be used to create one or more computed columns (e.g., `"a + b"` or `"a + sum(a)"`).
`value`	a numeric value used for this test.
`set`	a vector of numeric or string-based elements.
`regex`	a regex pattern to test for matching strings.
`preconditions`	an optional vector of filtering statements for filtering the table before this validation step.
`warn_count`	the threshold number for individual validations returning a `FALSE` result before applying the `warn` flag.
`notify_count`	the threshold number for individual validations returning a `FALSE` result before applying the `notify` flag.
`warn_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `warn` flag will be applied.
`notify_fraction`	the threshold fraction for individual validations returning a `FALSE` over all the entire set of individual validations. Beyond this threshold, the `notify` flag will be applied.
`tbl_name`	the name of the local or remote table.
`db_type`	if the table is located in a database, the type of database is required here. Currently, this can be either `PostgreSQL` or `MySQL`.

creds_file	a path to a credentials file used for establishing a database connection.
init_sql	an initially-applied SQL statement for transforming tabular data in a database before validation occurs.
file_path	an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

`determine_action` *Determine the course of action for a given verification step. Based on a recent judgment, what actions are taken now?*

Description

Determine the course of action for a given verification step. Based on a recent judgment, what actions are taken now?

Usage

```
determine_action(n, false_count, warn_count, notify_count, warn_fraction,
  notify_fraction)
```

Arguments

n	the total number of validation checks in the validation step.
false_count	the number of validation checks that returned a FALSE result.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.

focus_on *Place certain access details more to the fore*

Description

Allows for a change of the focus on the table name, database type, and the location of the credentials file.

Usage

```
focus_on(agent, tbl_name = NULL, file_name = NULL, col_types = NULL,
  db_type = NULL, creds_file = NULL, initial_sql = NULL,
  description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
tbl_name	the name of the local or remote table.
file_name	the name of a file to be loaded as a table. Valid types are CSV and TSV files.
col_types	if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').
description	an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Create a simple data frame with a column
# of numerical values
df <-
  data.frame(
    a = c(5, 4, 3, 5, 1, 2))

# Validate that values in column `a` are
# always less than 6; this requires the
# use of `create_agent()` (to begin the
# validation process) and `focus_on()`
# to point to the table that will undergo
# validation in subsequent steps
agent <-
  create_agent() %>%
  focus_on(tbl_name = "df") %>%
  col_vals_lt(
    column = "a",
    value = 6) %>%
  interrogate()

# Determine if this column validation has
# passed by using `all_passed()`
all_passed(agent)
#> [1] TRUE
```

`generate_img_files_plan` *Generate SVG files for the plan of a validation pipeline*

Description

Generate SVG files for the plan of a validation pipeline

Usage

```
generate_img_files_plan(agent)
```

Arguments

`agent`	agent an agent object of class `ptblank_agent`.

generate_img_files_results *Generate summary SVG files for the results of a validation pipeline*

Description

Generate summary SVG files for the results of a validation pipeline

Usage

```
generate_img_files_results(agent)
```

Arguments

agent	agent an agent object of class ptblank_agent.

get_all_cols *Get all column names from the table currently in focus*

Description

Get all column names from the table currently in focus

Usage

```
get_all_cols(agent)
```

Arguments

agent	an agent object of class ptblank_agent.

get_summary *Get a simple summary of the interrogation*

Description

Gets the essential information from an agent object after an interrogation is complete.

Usage

```
get_summary(agent)
```

Arguments

agent	an agent object of class ptblank_agent.

Value

an agent object.

`html_summary` *Create an HTML summary file for the interrogation*

Description

Get the essential information from an agent object after an interrogation is complete and then generates an HTML file for a visual summary.

Usage

```
html_summary(agent, intro_text = NULL, footer_text = NULL,
  filename = NULL, output_dir = NULL)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.
`intro_text`	HTML text to be placed at the top of the summary.
`footer_text`	HTML text to be placed in the footer of the summary.
`filename`	an optional filename to use for the output HTML file. If not provided, the filename `validation_report.html` will be used.
`output_dir`	an optional path to place the output HTML file.

`interrogate` *Given an agent that is fully loaded with tasks, perform an interrogation*

Description

The agent has all the information on what to do, so now all interrogations can proceed efficiently, and, according to plan.

Usage

```
interrogate(agent)
```

Arguments

`agent`	an agent object of class `ptblank_agent`.

Value

an agent object.

is_ptblank_agent *Is the object a pointblank agent?*

Description

Determines whether the object is actually a pointbank agent object (i.e., of type `ptblank_agent`).

Usage

```
is_ptblank_agent(object)
```

Arguments

`object`	the object to test for whether it is of class `ptblank_agent`.

Value

an logical value.

pb_notify *Send an email notification*

Description

Send an email notification

Usage

```
pb_notify(agent, recipients, creds_file)
```

Arguments

`agent`	agent an agent object of class `ptblank_agent`.
`recipients`	a vector of email addresses to which the notification email will be sent.
`creds_file`	a path to a credentials file used for sending an email message.

rows_not_duplicated *Verify that row data are not duplicated*

Description

Set a verification step where row data should contain no duplicates.

Usage

```
rows_not_duplicated(agent, cols = NULL, preconditions = NULL,
  warn_count = 1, notify_count = NULL, warn_fraction = NULL,
  notify_fraction = NULL, tbl_name = NULL, db_type = NULL,
  creds_file = NULL, initial_sql = NULL, file_path = NULL,
  col_types = NULL, description = NULL)
```

Arguments

agent	an agent object of class ptblank_agent.
cols	an optional grouping of columns to check for duplication. If not provided, the validation checks for duplicate records using data across all columns.
preconditions	an optional statement of filtering conditions that may reduce the number of rows for validation for the current validation step. The statements are executed for every row of the table in focus and are often referred as predicate statements (they either return TRUE or FALSE for every row evaluated, where rows evaluated as TRUE are the rows that are retained for the validation step). For example, if a table has columns a, b, and c, and, column a has numerical data, we can write a statement a < 5 that filters all rows in the table where values in column a are less than five.
warn_count	the threshold number for individual validations returning a FALSE result before applying the warn flag.
notify_count	the threshold number for individual validations returning a FALSE result before applying the notify flag.
warn_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the warn flag will be applied.
notify_fraction	the threshold fraction for individual validations returning a FALSE over all the entire set of individual validations. Beyond this threshold, the notify flag will be applied.
tbl_name	the name of the local or remote table.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.

creds_file — if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.

initial_sql — when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

file_path — an optional path for a tabular data file to be loaded for this verification step. Valid types are CSV and TSV files.

col_types — if validating a CSV or TSV file, an optional column specification can be provided here as a string. This string representation is where each character represents one column and the mappings are: c -> character, i -> integer, n -> number, d -> double, l -> logical, D -> date, T -> date time, t -> time, ? -> guess, or _/-, which skips the column.

description — an optional, text-based description for the validation step. Used primarily in the Logical Plan section of the report generated by the html_summary function.

Value

an agent object.

Examples

```
# Validate that column `a` exists in
# the `small_table` CSV file; do this
# by creating an agent, focussing on
# that table, creating a `col_exists()`
# step, and then interrogating the table
agent <-
  create_agent() %>%
  focus_on(
    file_name =
      system.file(
        "extdata", "small_table.csv",
        package = "pointblank"),
    col_types = "TDicidlc") %>%
  rows_not_duplicated(
    cols = a & b) %>%
  interrogate()

# Determine if these column
# validations have all passed
# by using `all_passed()`
all_passed(agent)
#> [1] FALSE
```

set_entry_point *Acquire information on the coordinates of a remote table.*

Description

If a table is remote (i.e., in a database), this function will be invoked to set an entry point for the interrogation query.

Usage

```
set_entry_point(table, db_type = NULL, creds_file = NULL,
  initial_sql = NULL)
```

Arguments

table	the table with which an entry point is required.
db_type	if the table is located in a database, the type of database is required here. Currently, this can be either PostgreSQL or MySQL.
creds_file	if a connection to a database is required for reaching the table specified in tbl_name, then a path to a credentials file can be used to establish that connection. The credentials file is an RDS containing a character vector with the following items in the specified order: (1) database name (dbname), (2) the host name, (3) the port, (4) the username (user), and (5) the password. This file can be easily created using the create_creds_file() function.
initial_sql	when accessing a remote table, this provides an option to provide an initial query component before conducting validations. An entire SQL statement can be provided here, or, as a shortcut, the initial SELECT... statement can be omitted for simple queries (e.g., WHERE a > 1 AND b = 'one').

%>% *The magrittr pipe*

Description

pointblank uses the pipe function, %>% to turn function composition into a series of imperative statements.

Index