

# Package ‘RPresto’

May 7, 2026

**Title** DBI Connector to Presto

**Version** 1.4.8

**Copyright** Meta Platforms, Inc. 2015-present.

**Description** Implements a 'DBI' compliant interface to Presto. Presto is an open source distributed SQL query engine for running interactive analytic queries against data sources of all sizes ranging from gigabytes to petabytes: [<https://prestodb.io/>](https://prestodb.io/).

**Depends** R (>= 3.1.0), methods

**Imports** DBI (>= 0.3.0), httr (>= 0.6), openssl, jsonlite, stringi, stats, utils, purrr, dplyr (>= 0.7.0), dbplyr (>= 2.3.3), tibble, tidyselect, bit64, rlang, lifecycle, lubridate, progress, vctrs, stringdist

**Suggests** testthat, hms, knitr, rmarkdown, withr

**License** BSD\_3\_clause + file LICENSE

**URL** <https://github.com/prestodb/RPresto>

**BugReports** <https://github.com/prestodb/RPresto/issues>

**Encoding** UTF-8

**Collate** 'PrestoDriver.R' 'Presto.R' 'PrestoSession.R'  
'PrestoConnection.R' 'PrestoQuery.R' 'PrestoResult.R'  
'RPresto-package.R' 'chunk.R' 'create.dummy.tables.R' 'cte.R'  
'dbAppendTable.R' 'dbAppendTableAs.R' 'dbClearResult.R'  
'dbColumnType.R' 'dbColumnInfo.R' 'dbConnect.R'  
'dbCreateTable.R' 'dbCreateTableAs.R' 'dbDataType.R'  
'dbDisconnect.R' 'dbExistsTable.R' 'dbFetch.R' 'dbGetInfo.R'  
'dbGetQuery.R' 'dbGetRowCount.R' 'dbGetRowsAffected.R'  
'dbGetStatement.R' 'dbHasCompleted.R' 'dbIsValid.R'  
'dbListFields.R' 'dbListTables.R' 'dbQuoteIdentifier.R'  
'dbQuoteLiteral.R' 'dbReadTable.R' 'dbRemoveTable.R'  
'dbRenameTable.R' 'dbSendQuery.R' 'dbUnloadDriver.R'  
'dbWriteTable.R' 'dbplyr-db.R' 'dbplyr-sql.R' 'dbplyr-src.R'  
'default.R' 'fetch.R' 'presto.field.R'  
'presto.field\_utilities.R' 'presto\_type.R' 'presto\_unnest.R'

'request\_headers.R' 'sqlAppendTableAs.R' 'sqlCreateTable.R'  
 'sqlCreateTableAs.R' 'zzz.R'

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Onur Ismail Filiz [aut],  
 Sergey Goder [aut],  
 Jarod G.R. Meng [aut, cre],  
 Thomas J. Leeper [ctb],  
 John Myles White [ctb]

**Maintainer** Jarod G.R. Meng <jarodm@fb.com>

**Repository** CRAN

**Date/Publication** 2025-12-02 06:10:53 UTC

## Contents

add_chunk . . . . .	2
dbAppendTableAs . . . . .	3
dbCreateTableAs . . . . .	4
dbDataType,PrestoDriver-method . . . . .	5
dbGetInfo,PrestoDriver-method . . . . .	6
dbplyr_edition.PrestoConnection . . . . .	7
dbRenameTable . . . . .	7
db_list_tables.PrestoConnection . . . . .	8
kerberos_configs . . . . .	10
Presto . . . . .	10
presto_default . . . . .	12
presto_type . . . . .	13
presto_unnest . . . . .	14
sqlAppendTableAs . . . . .	15
sqlCreateTableAs . . . . .	16
sql_query_save.PrestoConnection . . . . .	16
src_presto . . . . .	17

**Index** **19**

---

add_chunk	<i>Add a chunk field to a data frame</i>
-----------	--

---

## Description

This auxiliary function adds a field, if necessary, to a data frame so that each compartment of the data frame that corresponds to a unique combination of the chunk fields has a size below a certain threshold. This resulting data frame can then be safely used in dbAppendTable() because Presto has a size limit on any discrete INSERT INTO statement.

**Usage**

```
add_chunk(
  value,
  base_chunk_fields = NULL,
  chunk_size = 1e+06,
  new_chunk_field_name = "aux_chunk_idx"
)
```

**Arguments**

value	The original data frame.
base_chunk_fields	A character vector of existing field names that are used to split the data frame before checking the chunk size.
chunk_size	Maximum size (in bytes) of the VALUES statement encoding each unique chunk. Default to 1,000,000 bytes (i.e. 1Mb).
new_chunk_field_name	A string indicating the new chunk field name. Default to "aux_chunk_idx".

**Examples**

```
## Not run:
# returns the original data frame because it's within size
add_chunk(iris)
# add a new aux_chunk_idx field
add_chunk(iris, chunk_size = 2000)
# the new aux_chunk_idx field is added on top of Species
add_chunk(iris, chunk_size = 2000, base_chunk_fields = c("Species"))

## End(Not run)
```

---

dbAppendTableAs	<i>Append to a table in database using a statement</i>
-----------------	--

---

**Description**

Append to a table in database using a statement

**Usage**

```
dbAppendTableAs(conn, name, sql, auto_reorder = TRUE, ...)
```

**Arguments**

conn	a PrestoConnection object, as returned by <code>DBI::dbConnect()</code> .
name	The table name, passed on to <code>DBI::dbQuoteIdentifier()</code> . Options are: <ul style="list-style-type: none"> <li>• a character string with the unquoted DBMS table name, e.g. "table_name",</li> <li>• a call to <code>DBI::Id()</code> with components to the fully qualified table name, e.g. <code>Id(schema = "my_schema", table = "table_name")</code></li> <li>• a call to <code>DBI::SQL()</code> with the quoted and fully qualified table name given verbatim, e.g. <code>SQL('"my_schema"."table_name"')</code></li> </ul>
sql	a character string containing SQL statement, or a <code>tbl_presto</code> object.
auto_reorder	Logical (default TRUE). If TRUE, automatically reorders query columns to match table column order when column names match but order differs. If FALSE, columns are inserted in query order.
...	Other arguments used by individual methods.

---

dbCreateTableAs	<i>Create a table in database using a statement</i>
-----------------	---

---

**Description**

Create a table in database using a statement

**Usage**

```
dbCreateTableAs(conn, name, sql, overwrite = FALSE, with = NULL, ...)
```

**Arguments**

conn	a PrestoConnection object, as returned by <code>DBI::dbConnect()</code> .
name	The table name, passed on to <code>DBI::dbQuoteIdentifier()</code> . Options are: <ul style="list-style-type: none"> <li>• a character string with the unquoted DBMS table name, e.g. "table_name",</li> <li>• a call to <code>DBI::Id()</code> with components to the fully qualified table name, e.g. <code>Id(schema = "my_schema", table = "table_name")</code></li> <li>• a call to <code>DBI::SQL()</code> with the quoted and fully qualified table name given verbatim, e.g. <code>SQL('"my_schema"."table_name"')</code></li> </ul>
sql	a character string containing SQL statement.
overwrite	A boolean indicating if an existing table should be overwritten. Default to FALSE.
with	An optional WITH clause for the CREATE TABLE statement.
...	Other arguments used by individual methods.

---

`dbDataType,PrestoDriver-method`*Return the corresponding presto data type for the given R object*

---

**Description**

Return the corresponding presto data type for the given R object

**Usage**

```
## S4 method for signature 'PrestoDriver'  
dbDataType(dbObj, obj, ...)
```

**Arguments**

<code>dbObj</code>	A <a href="#">PrestoDriver</a> object
<code>obj</code>	Any R object
<code>...</code>	Extra optional parameters, not currently used

**Details**

The default value for unknown classes is 'VARCHAR'.

**Value**

A character value corresponding to the Presto type for obj

**Examples**

```
drv <- RPresto::Presto()  
dbDataType(drv, 1)  
dbDataType(drv, NULL)  
dbDataType(drv, as.POSIXct("2015-03-01 00:00:00", tz = "UTC"))  
dbDataType(drv, Sys.time())  
dbDataType(  
  drv,  
  list(  
    c("a" = 1L, "b" = 2L),  
    c("a" = 3L, "b" = 4L)  
  )  
)  
dbDataType(  
  drv,  
  list(  
    c(as.Date("2015-03-01"), as.Date("2015-03-02")),  
    c(as.Date("2016-03-01"), as.Date("2016-03-02"))  
  )  
)  
dbDataType(drv, iris)
```

---

dbGetInfo,PrestoDriver-method

*Metadata about database objects*

---

## Description

Metadata about database objects

For the [PrestoResult](#) object, the implementation returns the additional `stats` field which can be used to implement things like progress bars. See the examples section.

## Usage

```
## S4 method for signature 'PrestoDriver'
dbGetInfo(dbObj)

## S4 method for signature 'PrestoConnection'
dbGetInfo(dbObj)

## S4 method for signature 'PrestoResult'
dbGetInfo(dbObj)
```

## Arguments

`dbObj` A [PrestoDriver](#), [PrestoConnection](#) or [PrestoResult](#) object

## Value

[PrestoResult](#) A `list()` with elements

**statement** The SQL sent to the database

**row.count** Number of rows fetched so far

**has.completed** Whether all data has been fetched

**stats** Current stats on the query

## Examples

```
## Not run:
conn <- dbConnect(Presto(), "localhost", 7777, "onur", "datascience")
result <- dbSendQuery(conn, "SELECT * FROM jonchang_iris")
iris <- data.frame()
progress.bar <- NULL
while (!dbHasCompleted(result)) {
  chunk <- dbFetch(result)
  if (!NROW(iris)) {
    iris <- chunk
  } else if (NROW(chunk)) {
    iris <- rbind(iris, chunk)
  }
}
```



---

db\_list\_tables.PrestoConnection  
*dbplyr database methods*

---

## Description

dbplyr database methods

## Usage

```
## S3 method for class 'PrestoConnection'  
db_list_tables(con)
```

```
## S3 method for class 'PrestoConnection'  
db_has_table(con, table)
```

```
## S3 method for class 'PrestoConnection'  
db_write_table(  
  con,  
  table,  
  types,  
  values,  
  temporary = FALSE,  
  overwrite = FALSE,  
  ...,  
  with = NULL  
)
```

```
## S3 method for class 'PrestoConnection'  
db_copy_to(  
  con,  
  table,  
  values,  
  overwrite = FALSE,  
  types = NULL,  
  temporary = TRUE,  
  unique_indexes = NULL,  
  indexes = NULL,  
  analyze = TRUE,  
  ...,  
  in_transaction = TRUE,  
  with = NULL  
)
```

```
## S3 method for class 'PrestoConnection'  
db_compute(  
  con,
```

```

    table,
    sql,
    temporary = TRUE,
    unique_indexes = list(),
    indexes = list(),
    analyze = TRUE,
    with = NULL,
    ...
)

## S3 method for class 'PrestoConnection'
db_save_query(
  con,
  sql,
  name,
  temporary = TRUE,
  overwrite = FALSE,
  ...,
  with = NULL
)

## S3 method for class 'PrestoConnection'
db_sql_render(con, sql, ..., use_presto_cte = TRUE)

```

## Arguments

con	A PrestoConnection as returned by dbConnect().
table	Table name
types	Column types. If not provided, column types are inferred using dbDataType.
values	A data.frame.
temporary	If a temporary table should be created. Default to TRUE in the <code>dplyr::db_save_query()</code> generic. The default value generates an error in Presto. Using <code>temporary = FALSE</code> to save the query in a permanent table.
overwrite	If an existing table should be overwritten. Default to FALSE. When TRUE, uses smart overwriting: renames the existing table, creates the new table, and drops the old table on success. If creation fails, the original table is restored.
...	Extra arguments to be passed to individual methods.
with	An optional WITH clause for the CREATE TABLE statement.
unique_indexes, indexes, analyze, in_transaction	Ignored. Included for compatibility with generics.
sql	A SQL statement.
name	The table name, passed on to <code>DBI::dbQuoteIdentifier()</code> . Options are: a character string with the unquoted DBMS table name, a call to <code>DBI::Id()</code> with components to the fully qualified table name, or a call to <code>DBI::SQL()</code> with the quoted and fully qualified table name given verbatim.

`use_presto_cte` A logical value indicating if to use common table expressions stored in Presto-Connection when possible. Default to TRUE. See `vignette("common-table-expressions")`.

---

`kerberos_configs` *A convenient wrapper around Kerberos config*

---

### Description

The configs specify authentication protocol and additional settings.

### Usage

```
kerberos_configs(user = "", password = "", service_name = "presto")
```

### Arguments

`user` User name to pass to `httr::authenticate()`. Default to "".

`password` Password to pass to `httr::authenticate()`. Default to "".

`service_name` The service name. Default to "presto".

### Value

A `httr::config()` output that can be passed to the `request.config` argument of `dbConnect()`.

---

`Presto` *Connect to a Presto database*

---

### Description

Connect to a Presto database

### Usage

```
Presto(...)

## S4 method for signature 'PrestoDriver'
dbConnect(
  drv,
  catalog,
  schema,
  user,
  host = "localhost",
  port = 8080,
  source = methods::getPackageName(),
  session.timezone = "",
```

```

    output.timezone = "",
    parameters = list(),
    ctes = list(),
    request.config = httr::config(),
    use.trino.headers = FALSE,
    extra.credentials = "",
    bigint = c("integer", "integer64", "numeric", "character"),
    ...
)

## S4 method for signature 'PrestoConnection'
dbDisconnect(conn)

```

## Arguments

...	currently ignored
drv	A driver object generated by <a href="#">Presto()</a>
catalog	The catalog to be used
schema	The schema to be used
user	The current user
host	The presto host to connect to
port	Port to use for the connection
source	Source to specify for the connection
session.timezone	Time zone of the Presto server. Presto returns timestamps without time zones with respect to this value. The time arithmetic (e.g. adding hours) will also be done in the given time zone. This value is passed to Presto server via the request headers.
output.timezone	The time zone using which TIME WITH TZ and TIMESTAMP values in the output should be represented. Default to the Presto server timezone (use <code>show(&lt;PrestoConnection&gt;)</code> to see).
parameters	A <a href="#">list()</a> of extra parameters to be passed in the 'X-Presto-Session' header
ctes	A list of common table expressions (CTEs) that can be used in the WITH clause. See <code>vignette("common-table-expressions")</code> .
request.config	An optional config list, as returned by <code>httr::config()</code> , to be sent with every HTTP request.
use.trino.headers	A boolean to indicate whether Trino request headers should be used. Default to FALSE.
extra.credentials	Extra credentials to be passed in the X-Presto-Extra-Credential or X-Trino-Extra-Credential header ( depending on the value of the <code>use.trino.headers</code> argument). Default to an empty string.

bigint	The R type that Presto's 64-bit integer (BIGINT) class should be translated to. The default is "integer", which returns R's integer type, but results in NA for values above/below +/-2147483647. "integer64" returns a <code>bit64::integer64</code> , which allows the full range of 64 bit integers. "numeric" coerces into R's double type but might result in precision loss. Lastly, "character" casts into R's character type.
conn	A <code>PrestoConnection</code> object

### Value

`Presto` A `PrestoDriver` object

`DBI::dbConnect()` A `PrestoConnection` object

`DBI::dbDisconnect()` A `logical()` value indicating success

### Examples

```
## Not run:
conn <- dbConnect(Presto(),
  catalog = "hive", schema = "default",
  user = "onur", host = "localhost", port = 8080,
  session.timezone = "US/Eastern", bigint = "character"
)
dbListTables(conn, "%_iris")
dbDisconnect(conn)

## End(Not run)
```

---

```
presto_default
```

```
Check if default database is available.
```

---

### Description

`presto_default()` works similarly but returns a connection on success and throws a testthat skip condition on failure, making it suitable for use in tests.

RPresto examples and tests connect to a default database via `dbConnect(Presto(), ...)`. This function checks if that database is available, and if not, displays an informative message.

### Usage

```
presto_default(...)
```

```
presto_has_default(...)
```

### Arguments

... Additional arguments passed on to `DBI::dbConnect()`

**Examples**

```

if (presto_has_default()) {
  db <- presto_default()
  print(dbListTables(db))
  dbDisconnect(db)
} else {
  message("No database connection.")
}

```

---

```

presto_type          Get column type information for a Presto table or query

```

---

**Description**

Returns column type information for a `tbl_presto` object, including Presto types for complex and nested types.

**Usage**

```

presto_type(data, ...)

## S3 method for class 'tbl_presto'
presto_type(data, ...)

```

**Arguments**

```

data          A tbl_presto object
...          Additional arguments (currently unused)

```

**Value**

A data frame with columns:

- `name`: Column name (character)
- `type`: R type (character)
- `.presto_type`: Presto type as string (character)

**Examples**

```

## Not run:
# Connect to Presto
con <- DBI::dbConnect(RPresto::Presto(), ...)

# Get column types for a table
tbl(con, "my_table") %>%
  presto_type()

# Get column types for a query

```

```
tbl(con, "my_table") %>%
  dplyr::filter(id > 100) %>%
  presto_type()

## End(Not run)
```

---

```
presto_unnest      Unnest array columns in Presto tables
```

---

## Description

### [Experimental]

Expands array columns into rows using Presto's `CROSS JOIN UNNEST` syntax. This is similar to `tidyr::unnest()` but works with Presto database tables.

## Usage

```
presto_unnest(
  data,
  cols,
  ...,
  values_to = NULL,
  with_ordinality = FALSE,
  ordinality_to = NULL
)

## S3 method for class 'tbl_presto'
presto_unnest(
  data,
  cols,
  ...,
  values_to = NULL,
  with_ordinality = FALSE,
  ordinality_to = NULL
)
```

## Arguments

<code>data</code>	A <code>tbl_presto</code> object
<code>cols</code>	Column(s) to unnest. Currently only supports a single column.
<code>...</code>	Additional arguments (currently unused)
<code>values_to</code>	Name of column to store unnested values. If <code>NULL</code> , uses the original column name with <code>"_elem"</code> appended (e.g., <code>arr</code> becomes <code>arr_elem</code> ).
<code>with_ordinality</code>	If <code>TRUE</code> , includes an ordinality column with the position of each element in the array.
<code>ordinality_to</code>	Name of ordinality column when <code>with_ordinality = TRUE</code> . Must be provided if <code>with_ordinality = TRUE</code> .

**Value**

A tbl\_presto object with the array column unnested into rows.

**Examples**

```
## Not run:
# Connect to Presto
con <- DBI::dbConnect(RPresto::Presto(), ...)

# Create a table with an array column
DBI::dbExecute(con, "CREATE TABLE test (id BIGINT, arr ARRAY(BIGINT))")
DBI::dbExecute(con, "INSERT INTO test VALUES (1, ARRAY[10, 20, 30])")

# Unnest the array column
tbl(con, "test") %>%
  presto_unnest(arr) %>%
  collect()
# Without values_to, the unnested column is named "arr_elem"

# Or specify a custom name
tbl(con, "test") %>%
  presto_unnest(arr, values_to = "elem") %>%
  collect()

## End(Not run)
```

---

sqlAppendTableAs	<i>Compose query to append to a table using a statement</i>
------------------	---

---

**Description**

Compose query to append to a table using a statement

**Usage**

```
sqlAppendTableAs(con, name, sql, auto_reorder = TRUE, ...)
```

**Arguments**

con	A database connection.
name	The table name, passed on to <code>DBI::dbQuoteIdentifier()</code> . Options are: <ul style="list-style-type: none"> <li>• a character string with the unquoted DBMS table name, e.g. "table_name",</li> <li>• a call to <code>DBI::Id()</code> with components to the fully qualified table name, e.g. <code>Id(schema = "my_schema", table = "table_name")</code></li> <li>• a call to <code>DBI::SQL()</code> with the quoted and fully qualified table name given verbatim, e.g. <code>SQL('"my_schema"."table_name"')</code></li> </ul>
sql	a character string containing SQL statement, or a tbl_presto object.

auto_reorder	Logical (default TRUE). If TRUE, automatically reorders query columns to match table column order when column names match but order differs. If FALSE, columns are inserted in query order.
...	Other arguments used by individual methods.

---

sqlCreateTableAs	<i>Compose query to create a simple table using a statement</i>
------------------	---

---

### Description

Compose query to create a simple table using a statement

### Usage

```
sqlCreateTableAs(con, name, sql, with = NULL, ...)
```

### Arguments

con	A database connection.
name	The table name, passed on to <code>DBI::dbQuoteIdentifier()</code> . Options are: <ul style="list-style-type: none"> <li>• a character string with the unquoted DBMS table name, e.g. "table_name",</li> <li>• a call to <code>DBI::Id()</code> with components to the fully qualified table name, e.g. <code>Id(schema = "my_schema", table = "table_name")</code></li> <li>• a call to <code>DBI::SQL()</code> with the quoted and fully qualified table name given verbatim, e.g. <code>SQL('"my_schema"."table_name"')</code></li> </ul>
sql	a character string containing SQL statement.
with	An optional WITH clause for the CREATE TABLE statement.
...	Other arguments used by individual methods.

---

sql_query_save.PrestoConnection	<i>dbplyr SQL methods</i>
---------------------------------	---------------------------

---

### Description

dbplyr SQL methods

### Usage

```
## S3 method for class 'PrestoConnection'
sql_query_save(con, sql, name, temporary = TRUE, ..., with = NULL)
```

**Arguments**

con	A database connection.
sql	a character string containing SQL statement.
name	The table name, passed on to <code>DBI::dbQuoteIdentifier()</code> . Options are: <ul style="list-style-type: none"> <li>• a character string with the unquoted DBMS table name, e.g. "table_name",</li> <li>• a call to <code>DBI::Id()</code> with components to the fully qualified table name, e.g. <code>Id(schema = "my_schema", table = "table_name")</code></li> <li>• a call to <code>DBI::SQL()</code> with the quoted and fully qualified table name given verbatim, e.g. <code>SQL('"my_schema"."table_name"')</code></li> </ul>
temporary	If a temporary table should be created. Default to TRUE in the <code>dbplyr::sql_query_save()</code> generic. The default value generates an error in Presto. Using <code>temporary = FALSE</code> to save the query in a permanent table.
...	Other arguments used by individual methods.
with	An optional WITH clause for the CREATE TABLE statement.

---

src\_presto

*dplyr integration to connect to a Presto database.*


---

**Description**

Allows you to connect to an existing database through a presto connection.

**Usage**

```
src_presto(
  catalog = NULL,
  schema = NULL,
  user = NULL,
  host = NULL,
  port = NULL,
  source = NULL,
  session.timezone = NULL,
  parameters = NULL,
  bigint = c("integer", "integer64", "numeric", "character"),
  con = NULL,
  ...
)
```

**Arguments**

catalog	Catalog to use in the connection
schema	Schema to use in the connection
user	User name to use in the connection

host	Host name to connect to the database
port	Port number to use with the host name
source	Source to specify for the connection
session.timezone	Time zone for the connection
parameters	Additional parameters to pass to the connection
bigint	The R type that Presto's 64-bit integer (BIGINT) types should be translated to. The default is "integer", which returns R's integer type, but results in NA for values above/below +/-2147483647. "integer64" returns a <code>bit64::integer64</code> , which allows the full range of 64 bit integers. "numeric" coerces into R's double type but might result in precision loss. Lastly, "character" casts into R's character type.
con	An object that inherits from <code>PrestoConnection</code> , typically generated by <code>DBI::dbConnect</code> . When a valid connection object is supplied, Other arguments are ignored.
...	For <code>src_presto</code> other arguments passed on to the underlying database connector <code>dbConnect</code> . For <code>tbl.src_presto</code> , it is included for compatibility with the generic, but otherwise ignored.

## Examples

```
## Not run:
# To connect to a database
my_db <- src_presto(
  catalog = "memory",
  schema = "default",
  user = Sys.getenv("USER"),
  host = "http://localhost",
  port = 8080,
  session.timezone = "Asia/Kathmandu"
)
# Use a PrestoConnection
my_con <- DBI::dbConnect(
  catalog = "memory",
  schema = "default",
  user = Sys.getenv("USER"),
  host = "http://localhost",
  port = 8080,
  session.timezone = "Asia/Kathmandu"
)
my_db2 <- src_presto(con = my_con)

## End(Not run)
```

# Index

`add_chunk`, 2

`bit64::integer64`, 12, 18

`db_compute.PrestoConnection`  
(`db_list_tables.PrestoConnection`),  
8

`db_copy_to.PrestoConnection`  
(`db_list_tables.PrestoConnection`),  
8

`db_has_table.PrestoConnection`  
(`db_list_tables.PrestoConnection`),  
8

`db_list_tables.PrestoConnection`, 8

`db_save_query.PrestoConnection`  
(`db_list_tables.PrestoConnection`),  
8

`db_sql_render.PrestoConnection`  
(`db_list_tables.PrestoConnection`),  
8

`db_write_table.PrestoConnection`  
(`db_list_tables.PrestoConnection`),  
8

`dbAppendTableAs`, 3

`dbConnect`, `PrestoDriver`-method (`Presto`),  
10

`dbCreateTableAs`, 4

`dbDataType`, `PrestoDriver`-method, 5

`dbDisconnect`, `PrestoConnection`-method  
(`Presto`), 10

`dbGetInfo`, `PrestoConnection`-method  
(`dbGetInfo`, `PrestoDriver`-method),  
6

`dbGetInfo`, `PrestoDriver`-method, 6

`dbGetInfo`, `PrestoResult`-method  
(`dbGetInfo`, `PrestoDriver`-method),  
6

`DBI::dbConnect`, 18

`DBI::dbConnect()`, 4, 12

`DBI::dbDisconnect()`, 12

`DBI::dbQuoteIdentifier()`, 4, 9, 15–17

`DBI::Id()`, 4, 9, 15–17

`DBI::SQL()`, 4, 9, 15–17

`dbplyr::sql_query_save()`, 17

`dbplyr_edition.PrestoConnection`, 7

`dbRenameTable`, 7

`dplyr::db_save_query()`, 9

`kerberos_configs`, 10

`list()`, 6, 11

`logical()`, 12

`Presto`, 10, 12

`Presto()`, 11

`presto_default`, 12

`presto_has_default` (`presto_default`), 12

`presto_type`, 13

`presto_unnest`, 14

`PrestoConnection`, 6, 12, 18

`PrestoDriver`, 5, 6, 12

`PrestoResult`, 6

`sql_query_save.PrestoConnection`, 16

`sqlAppendTableAs`, 15

`sqlCreateTableAs`, 16

`src_presto`, 17