Package: ckdata (via r-universe)

November 10, 2026

Title Query Amazon Athena

Version 0.1.0

Description A read-only DBI and dbplyr interface to Amazon Athena. A shared Rust core executes Athena queries through Parquet and returns Arrow results, including schemas for empty results. Researchers can use lazy dplyr queries, SQL, temporary saved results and reproducible snapshots.

License MIT + file LICENSE

URL https://github.com/coursekata/ckdata

BugReports https://github.com/coursekata/ckdata/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.94.1, xz, AWS CLI v2 (optional, for interactive AWS SSO sign-in)

Depends R (>= 4.2)

Imports arrow, bit64, cachem, cli, DBI (>= 1.3.0), dbplyr (>= 2.6.0), digest, dplyr (>= 1.1.0), fs, jsonlite, lubridate, methods, nanoarrow, processx, purrr, rlang (>= 1.2.0), stringr, tibble, withr

Suggests DBItest (>= 1.8.3), RSQLite, testthat (>= 3.1.7)

Config/testthat/edition 3

Config/Needs/development devtools, lintr, rcmdcheck, rextendr, roxygen2

Config/pak/sysreqs
cmake make libicu-dev libuv1-dev libxml2-dev libzstd-dev libssl-dev xz-utils libclang-dev

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

Date/Publication 2026-11-10 22:09:46 UTC

RemoteUrl https://github.com/coursekata/ckdata

RemoteRef v0.1.0

RemoteSha c5e7a0a9fb87f627dce57a27e48c06bd37a9e8f0

RemoteSubdir r

Contents

approx_quantile . . . 2
ckdata . . . 3
ckdata_options . . . 7
ckdata_schema . . . 9
ckdata_settings . . . 10
ckdata_types . . . 11
collect_saved . . . 12
daily_at . . . 13
lake_disconnect . . . 14
lake_query . . . 15
lake_tbl . . . 15

Index **17**

`approx_quantile` *Explicit approximate summaries for remote queries*

Description

These functions are used inside dbplyr expressions. Athena evaluates `APPROX_PERCENTILE`; they do not compute a local R summary. `probs` is a single probability between zero and one. NULL values follow SQL semantics.

Usage

```
approx_quantile(x, probs = 0.5, na.rm = FALSE)

approx_median(x, na.rm = FALSE)
```

Arguments

`x`	A remote numeric column.
`probs`	One probability, such as `0.95`.
`na.rm`	Whether to suppress dbplyr's reminder that SQL drops NULLs.

ckdata *A read-only DBI driver for Amazon Athena*

Description

Connect with `DBI::dbConnect(ckdata::ckdata(), workgroup = ..., schema = ...)`. Credentials and region follow the standard AWS SDK configuration. Supplying `profile` or `region` overrides discovery for this connection. An explicit `role_arn` requires an additional role using AWS-managed credential renewal.

Usage

```
ckdata()

## S4 method for signature 'CkDriver'
dbConnect(
  drv,
  workgroup = getOption("ckdata.workgroup", config_env("CKDATA_WORKGROUP")),
  schema = ckdata_schema(),
  profile = config_env("CKDATA_PROFILE"),
  region = config_env("CKDATA_REGION"),
  role_arn = config_env("CKDATA_ROLE_ARN"),
  freshness = NULL,
  login = getOption("ckdata.login", rlang::is_interactive()),
  output_location = config_env("CKDATA_OUTPUT_LOCATION"),
  sso_start_url = NULL,
  sso_region = NULL,
  sso_account_id = NULL,
  sso_role_name = NULL,
  sso_session = NULL,
  sso_registration_scopes = NULL,
  ...
)

## S4 method for signature 'CkConnection'
dbDisconnect(conn, ...)

## S4 method for signature 'CkDriver'
dbIsValid(dbObj, ...)

## S4 method for signature 'CkConnection'
dbIsValid(dbObj, ...)

## S4 method for signature 'CkResult'
dbIsValid(dbObj, ...)

## S4 method for signature 'CkConnection'
```

```
dbGetInfo(dbObj, ...)

## S4 method for signature 'CkConnection'
show(object)

## S4 method for signature 'CkConnection,character'
dbSendQuery(conn, statement, ..., params = NULL, immediate = NULL)

## S4 method for signature 'CkConnection,character'
dbSendQueryArrow(conn, statement, ..., params = NULL, immediate = NULL)

## S4 method for signature 'CkConnection,character'
dbGetQuery(conn, statement, ..., n = -1, params = NULL, immediate = NULL)

## S4 method for signature 'CkConnection,character'
dbGetQueryArrow(conn, statement, ..., params = NULL, immediate = NULL)

## S4 method for signature 'CkResultArrow'
dbGetInfo(dbObj, ...)

## S4 method for signature 'CkResultArrow'
dbFetchArrowChunk(res, ...)

## S4 method for signature 'CkResult'
dbFetch(res, n = -1, ..., row.names = FALSE)

## S4 method for signature 'CkResultArrow'
dbFetchArrow(res, ...)

## S4 method for signature 'CkResult'
dbClearResult(res, ...)

## S4 method for signature 'CkResult'
dbHasCompleted(res, ...)

## S4 method for signature 'CkResult'
dbGetRowCount(res, ...)

## S4 method for signature 'CkResult'
dbGetRowsAffected(res, ...)

## S4 method for signature 'CkResult'
dbGetStatement(res, ...)

## S4 method for signature 'CkResult'
dbColumnInfo(res, ...)

## S4 method for signature 'CkConnection,character'
```

```
dbQuoteIdentifier(conn, x, ...)

## S4 method for signature 'CkConnection,SQL'
dbQuoteIdentifier(conn, x, ...)

## S4 method for signature 'CkConnection,character'
dbQuoteString(conn, x, ...)

## S4 method for signature 'CkConnection,SQL'
dbQuoteString(conn, x, ...)

## S4 method for signature 'CkConnection'
dbQuoteLiteral(conn, x, ...)

## S4 method for signature 'CkConnection,character'
dbListFields(conn, name, ...)

## S4 method for signature 'CkConnection,Id'
dbListFields(conn, name, ...)

## S4 method for signature 'CkConnection'
dbListTables(conn, ...)

## S4 method for signature 'CkConnection,character'
dbExistsTable(conn, name, ...)

## S4 method for signature 'CkConnection,Id'
dbExistsTable(conn, name, ...)

## S4 method for signature 'CkConnection'
dbListObjects(conn, prefix = NULL, ...)

## S4 method for signature 'CkConnection,character'
dbSendStatement(conn, statement, ...)

## S4 method for signature 'CkConnection'
dbBegin(conn, ...)

## S4 method for signature 'CkConnection'
dbCommit(conn, ...)

## S4 method for signature 'CkConnection'
dbRollback(conn, ...)

## S4 method for signature 'CkResult'
dbBind(res, params, ...)

## S4 method for signature 'CkResult'
```

```
dbBindArrow(res, params, ...)

## S4 method for signature 'CkDriver'
dbGetInfo(dbObj, ...)

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

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

## S4 method for signature 'CkDriver'
dbUnloadDriver(drv, ...)

## S4 method for signature 'CkDriver'
dbIsReadOnly(dbObj, ...)

## S4 method for signature 'CkConnection'
dbIsReadOnly(dbObj, ...)

## S4 method for signature 'CkResult'
dbIsReadOnly(dbObj, ...)
```

Arguments

drv	A driver returned by ckdata().
workgroup	Athena workgroup, or CKDATA_WORKGROUP.
schema	Default database, or ckdata_schema().
profile, region	Optional explicit AWS profile and region.
role_arn	Optional required role; defaults to CKDATA_ROLE_ARN. AWS renews its credentials while the source permits. Failure never falls back to the source identity for data access.
freshness	Optional default policy for collect_saved() on this connection. NULL inherits application defaults at collection time.
login	Allow AWS SSO browser sign-in when credentials require it. Defaults to option ckdata.login, or interactive() when unset. FALSE fails immediately without starting login. TRUE permits one login per top-level connection, query or fetch. Requires AWS CLI v2; existing credentials and non-SSO authentication do not require it.
output_location	Optional S3 result prefix, or CKDATA_OUTPUT_LOCATION. Enforced workgroup settings take precedence. Otherwise this overrides the workgroup's location, or supplies one when it is missing.
sso_start_url, sso_region, sso_account_id, sso_role_name	Independent SSO configuration. Each argument overrides its CKDATA_SSO_* environment value. An explicit profile ignores SSO environment settings; individual SSO arguments may override a plain SSO profile. Overrides of chained

profiles are rejected. Complete direct SSO does not inherit an ambient role chain. The SSO region is independent from the Athena service region.

`sso_session` Optional shared AWS SSO login-session name, or `CKDATA_SSO_SESSION`. Direct SSO generates a stable name by default.

`sso_registration_scopes`
Optional AWS SSO registration scopes, or `CKDATA_SSO_REGISTRATION_SCOPES`; defaults to `sso:account:access`.

`...` Additional arguments; unsupported arguments raise an error.

`conn` A ckdata DBI connection.

`dbObj` A ckdata driver, connection, or result.

`object` A ckdata connection to display.

`statement` SQL query.

`params` An unnamed list of equal-length parameter vectors for Athena's positional ? placeholders. Data frame columns bind by position. Values are sent separately using Athena execution parameters. Each row executes once; results are concatenated. For statements with placeholders and `params = NULL`, call `dbBind()` before fetching. For `dbBindArrow()`, supply an Arrow stream with one column per placeholder. Arrow integer and decimal columns retain their precision before R conversion, within Athena's supported types.

`immediate` Accepted for DBI compatibility; Athena has one query API.

`n` Maximum rows to fetch; `-1` or `Inf` fetches all remaining rows.

`res` A ckdata DBI result.

`row.names` `FALSE` or `NULL`; columns are not converted into row names.

`x` Names or values to quote.

`name` A table name or `DBI::Id(schema = ..., table = ...)`.

`prefix` Optional schema or table path for catalog discovery.

`obj` An R object for SQL type description.

Value

`ckdata()` returns a DBI driver. The methods return the values described by their corresponding DBI generics.

`ckdata_options` *Options that control ckdata*

Description

Set these with `options()`, for one session or in your `.Rprofile`.

Details

- `ckdata.mode`: what ckdata does when a query reads a column that is deprecated or may change without notice. One of `"explore"` (the default), `"audit"` or `"publish"`; see Modes below.
- `ckdata.allow`: flagged columns to accept anyway, as a named list of `table = column names`, for example `list(mart_responses = c("lrn_type", "item_id"))`. An allowed column is never reported or removed, in any mode.
- `ckdata.verbose`: `TRUE` reports when `collect_saved()` reuses a saved result. Defaults to `FALSE`. Core failures remain visible in either setting.
- `ckdata.timeout`: query timeout in seconds, including result download and empty-schema recovery. Defaults to 1800; a cancellation stop request can require additional time as documented by the shared core.
- `ckdata.login`: allow browser-based AWS SSO sign-in on a connection and its later operations. Captured when the connection is opened. Defaults to `rlang::is_interactive()`. FALSE fails immediately; an explicit `DBI::dbConnect(..., login = ...)` overrides this option. Browser sign-in uses AWS CLI v2 and retries credential acquisition once per top-level operation, preserving the existing connection and query.
- `ckdata.freshness`: default `collect_saved()` policy; see `daily_at()`. Explicit collection, table and connection policies take precedence, in that order. The option overrides `CKDATA_FRESHNESS_AT` and `CKDATA_FRESHNESS_TZ`, which together specify a daily schedule. Without a policy, collection fetches anew. Application defaults are read at collection time. A schedule assumes the source has refreshed by that time.
- `ckdata.saved_dir`: normal saved-result directory. Defaults to a ckdata subdirectory of R's session temporary directory, normally removed at exit. An explicit persistent directory keeps files across sessions; freshness still applies, and ckdata does not automatically delete those files.
- `ckdata.snapshot_dir`: indefinite snapshot directory. Defaults to the current directory. Snapshots never automatically refresh or expire.
- `ckdata.schema`: the database `lake_tbl()` opens a table from when the call names none; see `ckdata_schema()`.

Modes

Two kinds of column are flagged:

- **Deprecated** columns, which the lake keeps under an old name until a set removal date. The report names each one's replacement.
- Every column of a **staging** (`stg_`) or **intermediate** (`int_`) table, which can change without notice, unless a published table in the same query has it too, as a join key does.

ckdata checks a query when `dplyr::collect()` or `collect_saved()` runs it. The check reads only the query's SQL and columns, never the data.

- `"explore"`: nothing is checked. Use it while you work things out.
- `"audit"`: a warning names the flagged columns the query uses or returns, and nothing else changes, so one run of a script lists what to fix and the script still finishes.

- `"publish"`: the same warning, and flagged columns the query returns are removed from it before it runs, so code that reads them fails now rather than on the removal date. A query that would return nothing else fails instead. Use it for the final run of a document.

The modes are a helper for the common cases, not a guarantee. Because the check reads only the SQL, a flagged column that is renamed, or that a join returns as `.x` or `.y`, is reported but not removed; a query whose first table did not come from `lake_tbl()` is checked only by `collect_saved()` on a ckdata connection; and now and then a warning names a column the query did not need. Other DBI drivers do not use lake column rules.

See Also

ckdata_settings

`ckdata_schema` *The database lake_tbl() reads from by default*

Description

`lake_tbl()` opens a table from this database when the call names none. It is the `ckdata.schema` option when that is set, then `CKDATA_SCHEMA` (or its legacy alias `CK_DATALAKE_SCHEMA`), and otherwise `"default"`. An empty variable counts as unset. No organization-specific database or AWS account is selected by the package. The option and the variable are read on every call, so a change applies to the next `lake_tbl()` call.

Usage

```
ckdata_schema()
```

Details

Set the option in a script or `.Rprofile`. Set the variable where the environment rather than the script decides: an environment that may read only the `research` views, such as a research container, sets `CKDATA_SCHEMA=research`, and its scripts name no database.

Value

The database's name.

See Also

ckdata_options, ckdata_settings

Examples

```
ckdata_schema()

old <- options(ckdata.schema = "research")
ckdata_schema()
options(old)
```

ckdata_settings *Connection settings*

Description

Connection arguments take precedence over configuration. `ckdata.workgroup` and `ckdata.schema` options take precedence over their environment settings. The canonical environment names and compatibility aliases are:

Details

Canonical	Legacy R package alias
CKDATA_PROFILE	CK_DATALAKE_INTERNAL_PROFILE
CKDATA_REGION	CK_DATALAKE_REGION
CKDATA_WORKGROUP	CK_DATALAKE_WORKGROUP
CKDATA_SCHEMA	CK_DATALAKE_SCHEMA
CKDATA_ROLE_ARN	CK_DATALAKE_ROLE_ARN
CKDATA_OUTPUT_LOCATION	CK_DATALAKE_S3_STAGING_DIR

A nonempty canonical variable takes precedence over its legacy alias. Empty and absent variables both count as unset on every platform. To bypass an inherited optional setting, pass an explicit NULL connection argument. Aliases are resolved without changing the process's AWS environment.

Both managed helpers and explicit DBI connections use this configuration. Explicit NULL for `profile`, `region`, `role_arn` or `output_location` bypasses that setting. Without package-specific profile or region settings, the normal AWS SDK chain applies, including `AWS_PROFILE`, `AWS_REGION`, shared profiles and workload credentials. No account, role, region or workgroup is built in. The schema defaults to `"default"`; configure it explicitly for CourseKata.

The workgroup supplies the result location and encryption settings by default. `CKDATA_OUTPUT_LOCATION` can supply or override the S3 prefix when the workgroup permits client settings. Enforced workgroup settings always take precedence. The legacy staging-directory variable therefore remains functional without changing the workgroup or creating AWS resources.

A named `profile` argument selects that profile's authentication provider, ignoring SSO environment settings and unrelated ambient access keys. Without an explicit profile, `CKDATA_PROFILE` (or its legacy alias) selects a profile unless direct SSO settings take precedence. Without ckdata authentication settings, the standard AWS credential chain applies, including `AWS_PROFILE`.

Direct SSO uses `sso_start_url`, `sso_region`, `sso_account_id` and `sso_role_name`, or the corresponding `CKDATA_SSO_*` environment variables. Individual arguments override their environment settings. Missing fields can come from one selected plain SSO profile; role chains are never inherited by direct SSO. Individual SSO overrides of an explicitly selected chained profile are rejected. `CKDATA_SSO_SESSION` optionally names a shared login session; `CKDATA_SSO_REGISTRATION_SCOPES` defaults to `sso:account:access`.

`CKDATA_ROLE_ARN` optionally requires an additional role. AWS providers renew temporary credentials while their source permits; there is no fixed connection lifetime. A profile already targeting the

exact required role is reused. Other sources must assume it successfully; failure never permits data access with the source identity. No extra IAM lookup permission is needed.

Interactive R connections offer AWS SSO browser sign-in through AWS CLI v2 when silent renewal cannot restore the resolved SSO source. A running request stays in its original native job during sign-in; SQL is not resubmitted. Set `login = FALSE` in `DBI::dbConnect()` or `options(ckdata.login = FALSE)` to fail instead. The argument overrides the option; otherwise the default is `rlang::is_interactive()`. Scripts and jobs default to fail-fast. This controls ckdata's login process, not an external profile `credential_process` helper. Each top-level connection, query or fetch permits one browser login and one credential retry. Failed login, cancellation or a failed retry stops the operation. Permission denials never trigger browser login or profile hunting. AWS owns token caching; direct SSO uses a private subprocess-only config file that is removed after the login process exits. The R process's AWS environment and the user's AWS configuration are not modified.

Saved-result freshness uses `CKDATA_FRESHNESS_AT` and `CKDATA_FRESHNESS_TZ`, or option `ckdata.freshness`. No schedule or timezone is built in. Collection, table and connection policies take precedence, in that order. See `daily_at()` for schedules and completed-build revision checks. Normal `dplyr::collect()` is unaffected by saved-result configuration.

`ckdata_types` *Result types and ordering*

Description

Data-frame and tibble output use Arrow's R conversion: booleans become logical vectors, 8/16/32-bit integers become R integers, floating point and decimals become doubles, strings become character vectors, dates become Date, and timestamps become POSIXct. Arrow's integer64 downcasting is disabled: 64-bit integers consistently use bit64::integer64, including inside nested values and when results are empty. Lists/maps/structs use Arrow's established nested R representations.

Details

R doubles cannot represent every decimal exactly or retain every timestamp's subsecond precision. R reserves the minimum signed 32-bit integer for NA; bit64 reserves the minimum signed 64-bit value for NA. Request `lake_query(..., as = "arrow")` for the original Arrow types and their full precision/range. No additional coercion is applied to that path. R Date/POSIXt values used in SQL are encoded as typed DATE/TIMESTAMP literals. Timestamp literals use UTC instants; strings that resemble dates remain strings. difftime inputs use DBI's TIME formatter, including their units and fractions. These values can be used in predicates; Athena UNLOAD rejects TIME output columns. Athena UNLOAD timestamps have no timezone annotation. Arrow's R conversion therefore returns POSIXct without a named timezone; local display depends on the R session's timezone. The Arrow output retains the original naive type. INT96 timestamps use the upstream reader's microsecond schema hint to retain Athena's full date range; native nanosecond timestamp columns keep their precision.

NULL and empty strings remain distinct. Empty results retain their columns and corresponding types. Cached tibble results use the same conversions.

Raw SQL follows Athena UNLOAD's output restrictions: expressions need explicit names and standalone NULL outputs need explicit types. For example, use `SELECT 1 AS example` and

`SELECT CAST(NULL AS VARCHAR) AS example`. Duplicate output names and types unsupported by UNLOAD are also rejected by Athena. This interface supports a read-side DBI subset, not full conformance.

Parquet UNLOAD does not guarantee order across files. SQL ORDER BY and remote arrange() do not create a global downloaded-order contract; arrange the collected result locally when needed.

`grepl()` translates to Athena's Unicode Java regular expressions. Literal matching supports `fixed = TRUE`; `perl` and `useBytes` are unsupported. NULL text matches FALSE, while a missing pattern returns a missing logical. `floor_date()` supports whole seconds through years, including quarters. Weeks honor `week_start` (Monday = 1, Sunday = 7), defaulting to the `lubridate.week.start` option or Sunday.

`collect_saved` *Collect a query with temporary reuse or an indefinite Parquet snapshot*

Description

Normal saved results are reused according to `freshness`. Snapshots are created once and reused indefinitely, without automatic refresh or cleanup. Files contain the Arrow result before conversion to R, preserving its types.

Usage

```
collect_saved(
  query,
  freshness = NULL,
  refresh = FALSE,
  snapshot = FALSE,
  dir = NULL,
  source = NULL
)
```

Arguments

`query`	A lazy dbplyr query.
`freshness`	Override the table/connection policy, then the application option/environment defaults. See `daily_at()`. `"always"` fetches each time; NULL inherits. Without a configured policy, results are fetched anew.
`refresh`	Force a fresh query in normal mode. Invalid for snapshots.
`snapshot`	FALSE for normal reuse, TRUE for an automatically named snapshot, or a filename-safe label such as `"baseline"`.
`dir`	A directory. Normal mode defaults to `ckdata.saved_dir` or a session temporary directory. Snapshots default to `ckdata.snapshot_dir` or the current directory. A persistent directory does not disable freshness.

`source` Optional named list describing the source and access context. Required for other DBI drivers, which do not expose a standard complete source identity. Include database/namespace, principal/role, tenant and any session settings affecting visibility or results. Without it, other DBI drivers cannot safely identify a saved result. Never include credentials.

Details

Identity includes exact rendered SQL, source/access context and column-policy settings. dbplyr embeds query values in SQL; queries are not canonicalized. Labels supplement this identity, so unrelated queries never share a snapshot. Lake column-policy rules apply only to ckdata connections.

A failed save returns the fetched data with a warning. A failed query or revision check returns an available saved result with a stale-data warning; without a saved result, a query failure propagates. A failed revision check without saved data warns and fetches without a reusable revision marker. An existing snapshot that cannot be read raises an error and is preserved; repair it or choose another label/directory. An unreadable normal saved result warns and is fetched again.

Normal defaults are temporary and normally removed at R exit. Explicit persistent directories and snapshots are not automatically deleted. Saved files can be read offline; they do not enforce source authorization.

Writes publish complete Parquet files atomically. Concurrent misses may run duplicate queries. Normal results keep the last successful publication; snapshots retain and return the first publication. No global ordering is added. Ordinary `dplyr::collect()` always uses its normal query path.

Value

A tibble with the current query's grouping.

---

daily_at *Reuse results until a daily refresh or a source revision changes*

Description

Freshness is checked when `collect_saved()` is called. A schedule assumes that the source has refreshed by the specified local time. A revision check can instead report an actual completed build, using a source-specific query or metadata API. Neither policy runs queries in the background.

Usage

```
daily_at(at, tz)

freshness_check(check, every = 300)
```

Arguments

at	Daily refresh time in HH:MM format.
tz	An IANA timezone, such as "America/Los_Angeles".
check	A function accepting a DBI connection and returning one nonempty string identifying the completed source revision. It must not change data.
every	Seconds to reuse a revision check, per source/access context. Defaults to 300. Use zero to check on every collection.

Details

Daily schedules follow daylight saving time. A skipped clock time runs at the transition boundary; a repeated time uses its first occurrence.

An ordinary function can also be used directly as a freshness policy. It has the same contract as check, but is called on every collection.

Value

A freshness policy for a connection, lake_tbl(), collect_saved(), or the ckdata.freshness option.

---

lake_disconnect *Close the managed lake session*

Description

The next call to lake_tbl() or lake_query() opens a new connection using the current settings. Saved lazy queries keep their original connection; rebuild them from a fresh lake_tbl() before collecting results. Use this explicitly before changing connection settings. Calling it before a session exists does not connect.

Usage

```
lake_disconnect()
```

Value

Invisibly, TRUE.

lake_query *Execute SQL through the shared query core*

Description

SQL queries do not receive the dbplyr column-policy heuristics. Empty results retain their schema. The Arrow output retains the original Arrow types; tibble output uses the documented R conversion in ckdata_types. The Parquet UNLOAD path requires named output columns and known types. Alias expressions (`SELECT 1 AS example`) and cast standalone NULL outputs (`SELECT CAST(NULL AS VARCHAR) AS example`) SQL is not rewritten to add them.

Usage

```
lake_query(sql, as = c("tibble", "arrow"), con = lake_con())
```

Arguments

`sql`	SQL query text.
`as`	Either `"tibble"` or `"arrow"`.
`con`	A ckdata DBI connection; defaults to the managed connection.

Value

A tibble, or an Arrow Table when `as = "arrow"`.

lake_tbl *A lazy table in Amazon Athena*

Description

Named-table discovery uses catalog metadata without reading rows. Building a pipeline or calling `show_query()` does not execute it. As with other dbplyr tables, printing previews rows and `collect()` retrieves results.

Usage

```
lake_tbl(table, schema = ckdata_schema(), freshness = NULL)
```

Arguments

`table`	A table name.
`schema`	Database name; defaults to `ckdata_schema()`.
`freshness`	Optional `collect_saved()` policy for queries derived from this table. Overrides connection and application defaults without changing other tables. Conflicting policies in a join require a collection override.

Details

Results have no global order across Parquet files. To order a downloaded result, use `collect()` followed by `arrange()`.

A saved lazy query keeps its original connection. After `lake_disconnect()`, rebuild it from a fresh `lake_tbl()` before collecting results.

Value

A lazy dbplyr table using the managed session connection.

Index

approx_median *(approx_quantile)*, 2
approx_quantile, 2

CkConnection-class *(ckdata)*, 3
ckdata, 3
ckdata_options, 7, *9*
ckdata_schema, 9
ckdata_schema(), *6*, *8*, *15*
ckdata_settings, *9*, 10
ckdata_types, 11, *15*
CkDriver-class *(ckdata)*, 3
CkResult-class *(ckdata)*, 3
CkResultArrow-class *(ckdata)*, 3
collect_saved, 12
collect_saved(), *6*, *8*, *9*, *13–15*

daily_at, 13
daily_at(), *8*, *11*, *12*
dbBegin,CkConnection-method *(ckdata)*, 3
dbBind,CkResult-method *(ckdata)*, 3
dbBindArrow,CkResult-method *(ckdata)*, 3
dbClearResult,CkResult-method *(ckdata)*, 3
dbColumnInfo,CkResult-method *(ckdata)*, 3
dbCommit,CkConnection-method *(ckdata)*, 3
dbConnect,CkDriver-method *(ckdata)*, 3
dbDataType,CkConnection-method *(ckdata)*, 3
dbDataType,CkDriver-method *(ckdata)*, 3
dbDisconnect,CkConnection-method *(ckdata)*, 3
dbExistsTable,CkConnection,character-method *(ckdata)*, 3
dbExistsTable,CkConnection,Id-method *(ckdata)*, 3
dbFetch,CkResult-method *(ckdata)*, 3
dbFetchArrow,CkResultArrow-method *(ckdata)*, 3
dbFetchArrowChunk,CkResultArrow-method *(ckdata)*, 3
dbGetInfo,CkConnection-method *(ckdata)*, 3
dbGetInfo,CkDriver-method *(ckdata)*, 3
dbGetInfo,CkResultArrow-method *(ckdata)*, 3
dbGetQuery,CkConnection,character-method *(ckdata)*, 3
dbGetQueryArrow,CkConnection,character-method *(ckdata)*, 3
dbGetRowCount,CkResult-method *(ckdata)*, 3
dbGetRowsAffected,CkResult-method *(ckdata)*, 3
dbGetStatement,CkResult-method *(ckdata)*, 3
dbHasCompleted,CkResult-method *(ckdata)*, 3
dbIsReadOnly,CkConnection-method *(ckdata)*, 3
dbIsReadOnly,CkDriver-method *(ckdata)*, 3
dbIsReadOnly,CkResult-method *(ckdata)*, 3
dbIsValid,CkConnection-method *(ckdata)*, 3
dbIsValid,CkDriver-method *(ckdata)*, 3
dbIsValid,CkResult-method *(ckdata)*, 3
dbListFields,CkConnection,character-method *(ckdata)*, 3
dbListFields,CkConnection,Id-method *(ckdata)*, 3
dbListObjects,CkConnection-method *(ckdata)*, 3
dbListTables,CkConnection-method *(ckdata)*, 3
dbQuoteIdentifier,CkConnection,character-method *(ckdata)*, 3
dbQuoteIdentifier,CkConnection,SQL-method *(ckdata)*, 3
dbQuoteLiteral,CkConnection-method *(ckdata)*, 3

dbQuoteString,CkConnection,character-method
(ckdata), 3
dbQuoteString,CkConnection,SQL-method
(ckdata), 3
dbRollback,CkConnection-method
(ckdata), 3
dbSendQuery,CkConnection,character-method
(ckdata), 3
dbSendQueryArrow,CkConnection,character-method
(ckdata), 3
dbSendStatement,CkConnection,character-method
(ckdata), 3
dbUnloadDriver,CkDriver-method
(ckdata), 3
dplyr::collect(), *8*, *11*, *13*

freshness_check *(*daily_at*)*, 13

lake_disconnect, 14
lake_disconnect(), *16*
lake_query, 15
lake_query(), *14*
lake_tbl, 15
lake_tbl(), *8, 9*, *14*

options(), *7*

rlang::is_interactive(), *8*, *11*

show,CkConnection-method *(*ckdata*)*, 3