

# Package: sedonadb (via r-universe)

October 7, 2026

**Title** Bindings for Apache SedonaDB  
**Version** 0.4.1.9000  
**Description** Provides bindings for Apache SedonaDB, a lightweight query engine optimized for spatial workflows.  
**License** Apache License (>= 2)  
**Encoding** UTF-8  
**Roxygen** list(markdown = TRUE)  
**SystemRequirements** Cargo (Rust's package manager), rustc  
**Depends** R (>= 4.1.0)  
**Suggests** adbcdrivermanager, dplyr, jsonlite, lintr, rlang, tibble, sf, testthat (>= 3.0.0), tidyselect, withr, wk  
**Imports** geoarrow, nanoarrow  
**Config/testthat/edition** 3  
**Config/build/bootstrap** TRUE  
**Config/Needs/check** lintr  
**Config/roxygen2/version** 8.0.0  
**Config/pak/sysreqs** libzstd-dev libclang-dev  
**Repository** <https://r-multiverse-staging.r-universe.dev>  
**Date/Publication** 2026-10-02 03:27:20 UTC  
**RemoteUrl** <https://github.com/apache/sedona-db>  
**RemoteRef** apache-sedona-db-0.5.0  
**RemoteSha** 751453f04cd6e09842adf3073b2b3dac7d1777bc  
**RemoteSubdir** r/sedonadb

## Contents

|                                 |   |
|---------------------------------|---|
| .fns . . . . .                  | 2 |
| .tables . . . . .               | 3 |
| as_sedonadb_dataframe . . . . . | 3 |

|                                  |           |
|----------------------------------|-----------|
| as_sedonadb_literal . . . . .    | 4         |
| sd_arrange . . . . .             | 4         |
| sd_compute . . . . .             | 5         |
| sd_configure_proj . . . . .      | 6         |
| sd_connect . . . . .             | 7         |
| sd_count . . . . .               | 8         |
| sd_drop_view . . . . .           | 8         |
| sd_expr_column . . . . .         | 9         |
| sd_filter . . . . .              | 11        |
| sd_group_by . . . . .            | 11        |
| sd_join . . . . .                | 12        |
| sd_join_by . . . . .             | 14        |
| sd_join_select . . . . .         | 15        |
| sd_join_select_default . . . . . | 16        |
| sd_preview . . . . .             | 17        |
| sd_read . . . . .                | 17        |
| sd_read_parquet . . . . .        | 19        |
| sd_read_sf . . . . .             | 20        |
| sd_register_udf . . . . .        | 21        |
| sd_select . . . . .              | 22        |
| sd_sql . . . . .                 | 22        |
| sd_summarise . . . . .           | 23        |
| sd_to_view . . . . .             | 24        |
| sd_transmute . . . . .           | 24        |
| sd_with_params . . . . .         | 25        |
| sd_write_parquet . . . . .       | 26        |
| sedonadb_adbc . . . . .          | 27        |
| <b>Index</b>                     | <b>29</b> |

---

.fns

*SedonaDB Functions*

---

## Description

This object is an escape hatch for calling SedonaDB/DataFusion functions directly for translations that are not yet registered or are otherwise misbehaving.

## Usage

.fns

---

|         |                                  |
|---------|----------------------------------|
| .tables | <i>SedonaDB Table Qualifiers</i> |
|---------|----------------------------------|

---

**Description**

This object is an escape hatch for referring to the right or left side of a join when constructing `sd_join_by()` or `sd_join_select()` expressions.

**Usage**

.tables

---

|                       |                                         |
|-----------------------|-----------------------------------------|
| as_sedonadb_dataframe | <i>Convert an object to a DataFrame</i> |
|-----------------------|-----------------------------------------|

---

**Description**

Convert an object to a DataFrame

**Usage**

as\_sedonadb\_dataframe(x, ..., schema = NULL, ctx = NULL)

**Arguments**

- x                    An object to convert
- ...                  Extra arguments passed to/from methods
- schema              The requested schema
- ctx                  A SedonaDB context. This should always be passed to inner calls to SedonaDB functions; NULL implies the global context.

**Value**

A sedonadb\_dataframe

**Examples**

as\_sedonadb\_dataframe(data.frame(x = 1:3))

---

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| as_sedonadb_literal | <i>S3 Generic to create a SedonaDB literal expression</i> |
|---------------------|-----------------------------------------------------------|

---

### Description

This generic provides the opportunity for objects to register a mechanism to be understood as literals in the context of a SedonaDB expression. Users constructing expressions directly should use `sd_expr_literal()`.

### Usage

```
as_sedonadb_literal(x, ..., type = NULL, factory = NULL)
```

### Arguments

|         |                                                                                                                            |
|---------|----------------------------------------------------------------------------------------------------------------------------|
| x       | An object to convert to a SedonaDB literal                                                                                 |
| ...     | Passed to/from methods                                                                                                     |
| type    | An optional data type to request for the output                                                                            |
| factory | An <code>sd_expr_factory()</code> that should be passed to any other calls to <code>as_sedonadb_literal()</code> if needed |

### Value

An object of class `SedonaDBExpr`

### Examples

```
as_sedonadb_literal("abcd")
```

---

|            |                                                                |
|------------|----------------------------------------------------------------|
| sd_arrange | <i>Order rows of a SedonaDB data frame using column values</i> |
|------------|----------------------------------------------------------------|

---

### Description

Order rows of a SedonaDB data frame using column values

### Usage

```
sd_arrange(.data, ...)
```

### Arguments

|       |                                                                                                                                                                                                                                                   |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .data | A <code>sedonadb_dataframe</code> or an object that can be coerced to one.                                                                                                                                                                        |
| ...   | Unnamed expressions for arrange expressions. These are evaluated in the same way as <code>dplyr::arrange()</code> except does not support extra dplyr features such as <code>across()</code> , <code>.by_group</code> , or <code>.locale</code> . |

**Value**

An object of class `sedonadb_dataframe`

**Examples**

```
data.frame(x = c(10:1, NA)) |> sd_arrange(x)
data.frame(x = c(1:10, NA)) |> sd_arrange(desc(x))
```

---

|            |                                        |
|------------|----------------------------------------|
| sd_compute | <i>Collect a DataFrame into memory</i> |
|------------|----------------------------------------|

---

**Description**

Use `sd_compute()` to collect and return the result as a `DataFrame`; use `sd_collect()` to collect and return the result as an R `data.frame`.

**Usage**

```
sd_compute(.data)

sd_collect(.data, ptype = NULL)
```

**Arguments**

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| <code>.data</code> | A <code>sedonadb_dataframe</code> or an object that can be coerced to one. |
| <code>ptype</code> | The target R object. See <a href="#">nanoarrow::convert_array_stream</a> . |

**Value**

`sd_compute()` returns a `sedonadb_dataframe`; `sd_collect()` returns a `data.frame` (or subclass according to `ptype`).

**Examples**

```
sd_sql("SELECT 1 as one") |> sd_compute()
sd_sql("SELECT 1 as one") |> sd_collect()
```

---

|                   |                       |
|-------------------|-----------------------|
| sd_configure_proj | <i>Configure PROJ</i> |
|-------------------|-----------------------|

---

## Description

Performs a runtime configuration of PROJ, which can be used in place of a build-time linked version of PROJ or to add in support if PROJ was not linked at build time.

## Usage

```
sd_configure_proj(
    preset = NULL,
    shared_library = NULL,
    database_path = NULL,
    search_path = NULL
)
```

## Arguments

|                |                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| preset         | One of: <ul style="list-style-type: none"> <li>• "homebrew": Look for PROJ installed by Homebrew. This is the easiest option on MacOS.</li> <li>• "system": Look for PROJ in the platform library load path (e.g., after installing system proj on Linux).</li> <li>• "auto": Try all presets in the order listed above, issuing a warning if none can be configured.</li> </ul> |
| shared_library | An absolute or relative path to a shared library valid for the platform.                                                                                                                                                                                                                                                                                                         |
| database_path  | A path to proj.db                                                                                                                                                                                                                                                                                                                                                                |
| search_path    | A path to the data files required by PROJ for some transforms.                                                                                                                                                                                                                                                                                                                   |

## Value

NULL, invisibly

## Examples

```
sd_configure_proj("auto")
```

---

|            |                           |
|------------|---------------------------|
| sd_connect | Create a SedonaDB context |
|------------|---------------------------|

---

**Description**

Runtime options configure the execution environment. Use `global = TRUE` to configure the global context or use the returned object as a scoped context. A scoped context is recommended for programmatic usage as it prevents named views from interfering with each other.

**Usage**

```
sd_connect(  
  ...,  
  global = FALSE,  
  memory_limit = NULL,  
  temp_dir = NULL,  
  memory_pool_type = NULL,  
  unspillable_reserve_ratio = NULL  
)
```

**Arguments**

|                           |                                                                                                                                                     |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| ...                       | Reserved for future options                                                                                                                         |
| global                    | Use TRUE to set options on the global context.                                                                                                      |
| memory_limit              | Maximum memory for query execution, as a human-readable string (e.g., "4gb", "512m") or NULL for unbounded (the default).                           |
| temp_dir                  | Directory for temporary/spill files, or NULL to use the DataFusion default.                                                                         |
| memory_pool_type          | Memory pool type: "greedy" (default) or "fair". Only takes effect when memory_limit is set.                                                         |
| unspillable_reserve_ratio | Fraction of memory (0–1) reserved for unspillable consumers. Only applies when memory_pool_type is "fair". Defaults to 0.2 when not explicitly set. |

**Value**

The constructed context, invisibly.

**Examples**

```
sd_connect(memory_limit = "100mb", memory_pool_type = "fair")
```

---

|          |                           |
|----------|---------------------------|
| sd_count | Count rows in a DataFrame |
|----------|---------------------------|

---

**Description**

Count rows in a DataFrame

**Usage**

sd\_count(.data)

**Arguments**

.data                    A sedonadb\_dataframe or an object that can be coerced to one.

**Value**

The number of rows after executing the query

**Examples**

sd\_sql("SELECT 1 as one") |> sd\_count()

---

|              |                             |
|--------------|-----------------------------|
| sd_drop_view | Create or Drop a named view |
|--------------|-----------------------------|

---

**Description**

Remove a view created with `sd_to_view()` from the context.

**Usage**

sd\_drop\_view(table\_ref)  
  
sd\_ctx\_drop\_view(ctx, table\_ref)  
  
sd\_view(table\_ref)  
  
sd\_ctx\_view(ctx, table\_ref)

**Arguments**

table\_ref                The name of the view reference  
ctx                        A SedonaDB context.

**Value**

The context, invisibly

**Examples**

```
sd_sql("SELECT 1 as one") |> sd_to_view("foofy")
sd_view("foofy")
sd_drop_view("foofy")
try(sd_view("foofy"))
```

---

|                |                                            |
|----------------|--------------------------------------------|
| sd_expr_column | <i>Create SedonaDB logical expressions</i> |
|----------------|--------------------------------------------|

---

**Description**

Create SedonaDB logical expressions

**Usage**

```
sd_expr_column(column_name, qualifier = NULL, factory = sd_expr_factory())

sd_expr_literal(x, type = NULL, factory = sd_expr_factory())

sd_expr_binary(op, lhs, rhs, factory = sd_expr_factory())

sd_expr_negative(expr, factory = sd_expr_factory())

sd_expr_is_null(expr, factory = sd_expr_factory())

sd_expr_is_not_null(expr, factory = sd_expr_factory())

sd_expr_not(expr, factory = sd_expr_factory())

sd_expr_any_function(
  function_name,
  args,
  ...,
  na.rm = NULL,
  factory = sd_expr_factory()
)

sd_expr_scalar_function(function_name, args, factory = sd_expr_factory())

sd_expr_aggregate_function(
  function_name,
  args,
```

```

    ...,
    na.rm = FALSE,
    distinct = FALSE,
    factory = sd_expr_factory()
)

sd_expr_cast(expr, type, factory = sd_expr_factory())

sd_expr_alias(expr, alias, factory = sd_expr_factory())

sd_expr_parse_binary(expr)

as_sd_expr(x, factory = sd_expr_factory())

is_sd_expr(x)

sd_expr_factory(ctx = NULL)

```

### Arguments

|               |                                                                                                                                                                                    |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| column_name   | A column name                                                                                                                                                                      |
| qualifier     | An optional qualifier (e.g., table reference) that may be used to disambiguate a specific reference                                                                                |
| factory       | A <a href="#">sd_expr_factory()</a> . This factory wraps a SedonaDB context and is used to resolve scalar functions and/or retrieve options.                                       |
| x             | An object to convert to a SedonaDB literal (constant).                                                                                                                             |
| type          | A destination type into which expr should be cast.                                                                                                                                 |
| op            | Operator name for a binary expression. In general these follow R function names (e.g., >, <, +, -).                                                                                |
| lhs, rhs      | Arguments to a binary expression                                                                                                                                                   |
| expr          | A SedonaDBExpr or object coercible to one with <a href="#">as_sd_expr()</a> .                                                                                                      |
| function_name | The name of the function to call. This name is resolved from the context associated with factory.                                                                                  |
| args          | A list of SedonaDBExpr or object coercible to one with <a href="#">as_sd_expr()</a> .                                                                                              |
| ...           | Reserved for future use                                                                                                                                                            |
| na.rm         | For aggregate expressions, should nulls be ignored? The R idiom is to respect null; however, the SQL idiom is to drop them. The default value follows the R idiom (na.rm = FALSE). |
| distinct      | For aggregate expressions, use only distinct values.                                                                                                                               |
| alias         | An alias to apply to expr.                                                                                                                                                         |
| ctx           | A SedonaDB context or NULL to use the default context.                                                                                                                             |

### Value

An object of class SedonaDBExpr

**Examples**

```
sd_expr_column("foofy")
sd_expr_literal(1L)
sd_expr_scalar_function("abs", list(1L))
sd_expr_cast(1L, nanoarrow::na_int64())
sd_expr_alias(1L, "foofy")
```

sd\_filter

*Keep rows of a SedonaDB DataFrame that match a condition***Description**

Keep rows of a SedonaDB DataFrame that match a condition

**Usage**

```
sd_filter(.data, ...)
```

**Arguments**

|       |                                                                                                                                                                                                                 |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .data | A sedonadb_dataframe or an object that can be coerced to one.                                                                                                                                                   |
| ...   | Unnamed expressions for filter conditions. These are evaluated in the same way as <code>dplyr::filter()</code> except does not support extra dplyr features such as <code>across()</code> or <code>.by</code> . |

**Value**

An object of class `sedonadb_dataframe`

**Examples**

```
data.frame(x = 1:10) |> sd_filter(x > 5)
```

sd\_group\_by

*Group SedonaDB DataFrames by one or more expressions***Description**

Note that unlike `dplyr::group_by()`, these groups are dropped after any transformations.

**Usage**

```
sd_group_by(.data, ...)
```

```
sd_ungroup(.data)
```

**Arguments**

`.data` A `sedonadb_dataframe` or an object that can be coerced to one.

`...` Named expressions whose unique combination will be used as groups to potentially compute a future aggregate expression. These are evaluated in the same way as `dplyr::group_by()` except `.add` nor `.drop` are supported.

**Value**

An object of class `sedonadb_dataframe`

**Examples**

```
data.frame(letter = c(rep("a", 3), rep("b", 4), rep("c", 3)), x = 1:10) |>
  sd_group_by(letter) |>
  sd_summarise(x = sum(x))
```

---

|         |                                     |
|---------|-------------------------------------|
| sd_join | <i>Join two SedonaDB DataFrames</i> |
|---------|-------------------------------------|

---

**Description**

Perform a join operation between two dataframes. Use `sd_join_by()` to specify join conditions using `x$column` and `y$column` syntax to reference columns from the left and right tables respectively.

**Usage**

```
sd_join(
  x,
  y,
  by = NULL,
  join_type = "inner",
  select = sd_join_select_default(),
  keep = NULL
)

sd_left_join(x, y, by = NULL, select = sd_join_select_default(), keep = NULL)

sd_right_join(x, y, by = NULL, select = sd_join_select_default(), keep = NULL)

sd_inner_join(x, y, by = NULL, select = sd_join_select_default(), keep = NULL)

sd_full_join(x, y, by = NULL, select = sd_join_select_default(), keep = NULL)

sd_semi_join(x, y, by = NULL)
```

```
sd_anti_join(x, y, by = NULL)

sd_cross_join(x, y, select = sd_join_select_default())
```

## Arguments

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x         | The left dataframe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| y         | The right dataframe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| by        | Join specification. One of: <ul style="list-style-type: none"> <li>• A <code>sedonadb_join_by</code> object from <code>sd_join_by()</code></li> <li>• A <code>sedonadb_join_by</code> object from <code>sd_join_intersects()</code>, <code>sd_join_contains()</code>, <code>sd_join_within()</code>, <code>sd_join_covers()</code>, <code>sd_join_coveredby()</code>, <code>sd_join_touches()</code>, <code>sd_join_crosses()</code>, <code>sd_join_overlaps()</code>, or <code>sd_join_equals()</code>.</li> <li>• A character vector of column names to join on in both tables</li> <li>• A named character vector mapping left-table column names to right-table column names, e.g. <code>c(x_val = "y_val")</code></li> <li>• <code>NULL</code> for a natural join on columns with matching names</li> </ul> |
| join_type | The type of join to perform. One of "inner", "left", "right", "full", "leftsemi", "rightsemi", "leftanti", "rightanti", "leftmark", or "rightmark".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| select    | Post-join column selection. One of <ul style="list-style-type: none"> <li>• <code>NULL</code> for no modification, which may result in duplicate (unqualified) column names. The column may still be referred to with a qualifier in advanced usage using <code>sd_expr_column()</code>.</li> <li>• <code>sd_join_select_default()</code> for dplyr-like behaviour (equi-join keys or spatial join keys removed for inner/left joins, intersecting names suffixed)</li> <li>• <code>sd_join_select()</code> for a custom selection</li> </ul>                                                                                                                                                                                                                                                                    |
| keep      | Use <code>TRUE</code> to keep all key columns in an equijoin or spatial join. This is only applied when using <code>sd_join_select_default()</code> for left/inner joins and right equijoins (full joins and right spatial joins always keep spatial join keys).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Value

An object of class `sedonadb_dataframe`

## Examples

```
df1 <- data.frame(x = letters[1:10], y = 1:10)
df2 <- data.frame(y = 10:1, z = LETTERS[1:10])
df1 |> sd_join(df2)
```

---

|            |                                |
|------------|--------------------------------|
| sd_join_by | <i>Specify join conditions</i> |
|------------|--------------------------------|

---

## Description

Use `sd_join_by()` to specify join conditions for `sd_join()` using expressions that reference columns from both tables. Table references are specified using `x$column` and `y$column` syntax to disambiguate columns from the left and right tables, and the special helper `x$geom()` and `y$geom()` may be used for tables with exactly one geometry column. Spatial joins can use spatial predicates in the join by expression (e.g., `sd_join_by(st_intersects(x$geom(), y$geom()))`) or the shorthand `sd_join_intersects()`.

## Usage

```
sd_join_by(...)
sd_join_intersects()
sd_join_contains()
sd_join_within()
sd_join_covers()
sd_join_coveredby()
sd_join_touches()
sd_join_crosses()
sd_join_overlaps()
sd_join_equals()
sd_join_dwithin(distance)
```

## Arguments

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ...      | Expressions specifying join conditions. These should be comparison expressions (e.g., <code>x\$id == y\$id</code> , <code>x\$value &gt; y\$threshold</code> ) or spatial predicate expressions (e.g., <code>st_intersects(x\$geometry, y\$geometry)</code> ). Multiple conditions are combined with AND. Like <code>dplyr</code> 's <code>join_by()</code> , single columns are parsed as an equijoin condition (e.g., <code>id</code> becomes <code>x\$id == y\$id</code> ). |
| distance | For a within-distance join, the distance threshold.                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Details**

For programmatic usage, the `.tables` pronoun may be used to unambiguously refer to a table qualifier (similar to the `.data` pronoun which may be used in single-table SedonaDB verbs).

**Value**

An object of class `sedonadb_join_by` containing the unevaluated join condition expressions.

**Examples**

```
# Equality join on id column
sd_join_by(x$id == y$id)

# Can use just the column name as a shorthand
sd_join_by(id)

# Multiple conditions (combined with AND)
sd_join_by(x$id == y$id, x$date >= y$start_date)

# Inequality join
sd_join_by(x$value > y$threshold)

# Spatial joins
sd_join_intersects()
sd_join_dwithin(100)
```

---

sd\_join\_select

Specify custom post-join column selection

---

**Description**

Use `sd_join_select()` to specify which columns to include in the join result and optionally rename them. Columns may be referenced using `x$column` and `y$column` syntax to disambiguate columns from the left and right tables, or by bare column name when the name exists on only one side of the join.

**Usage**

```
sd_join_select(...)
```

**Arguments**

```
...
```

Named expressions specifying output columns. Each expression may reference a column using `x$column` or `y$column` syntax, or use a bare column name when it is unambiguous. If the same column name exists on both sides of the join, it must be qualified with `x$` or `y$`. The name of the argument becomes the output column name. Unnamed arguments use the original column name (without table prefix).

**Value**

An object of class `sedonadb_join_select` containing the unevaluated column selection expressions.

**Examples**

```
# Select and rename columns
sd_join_select(id = x$id, left_value = x$value, right_value = y$value)

# Unnamed arguments keep original column name
sd_join_select(x$id, x$name, y$value)
```

---

```
sd_join_select_default
```

*Specify default post-join column selection*

---

**Description**

Use `sd_join_select_default()` to specify that the join result should remove duplicate equijoin key columns (keeping the x-side version) and apply suffixes to any remaining overlapping column names.

**Usage**

```
sd_join_select_default(suffix = c(".x", ".y"))
```

**Arguments**

|                     |                                                                                                                               |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <code>suffix</code> | A character vector of length 2 specifying suffixes to add to overlapping column names from the left (x) and right (y) tables. |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------|

**Value**

An object of class `sedonadb_join_select_default` specifying the default column selection behavior.

**Examples**

```
# Default suffixes
sd_join_select_default()

# Custom suffixes
sd_join_select_default(suffix = c("_left", "_right"))
```

---

|            |                                                         |
|------------|---------------------------------------------------------|
| sd_preview | <i>Preview and print the results of running a query</i> |
|------------|---------------------------------------------------------|

---

### Description

This is used to implement `print()` for the `sedonadb_dataframe` or can be used to explicitly preview if `options(sedonadb.interactive = FALSE)`.

### Usage

```
sd_preview(.data, n = NULL, ascii = NULL, width = NULL)
```

### Arguments

|                    |                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.data</code> | A <code>sedonadb_dataframe</code> or an object that can be coerced to one.                                                                                                                                                                  |
| <code>n</code>     | The number of rows to preview. Use <code>Inf</code> to preview all rows. Defaults to <code>getOption("pillar.print_max")</code> .                                                                                                           |
| <code>ascii</code> | Use <code>TRUE</code> to force ASCII table formatting or <code>FALSE</code> to force unicode formatting. By default, use a heuristic to determine if the output is unicode-friendly or the value of <code>getOption("cli.unicode")</code> . |
| <code>width</code> | The character width of the output. Defaults to <code>getOption("width")</code> .                                                                                                                                                            |

### Value

`.data`, invisibly

### Examples

```
sd_sql("SELECT 1 as one") |> sd_preview()
```

---

|         |                                                |
|---------|------------------------------------------------|
| sd_read | <i>Read one or more files into a DataFrame</i> |
|---------|------------------------------------------------|

---

### Description

Resolves the reader from `format`, or from the file extension when `format` is `NULL`. The query is executed lazily when results are requested.

**Usage**

```
sd_read(
  file_or_files,
  options = list(),
  format = NULL,
  partitioning = NULL,
  check_extension = FALSE
)

sd_ctx_read(
  ctx,
  file_or_files,
  options = list(),
  format = NULL,
  partitioning = NULL,
  check_extension = FALSE
)
```

**Arguments**

|                              |                                                                                                                                                                            |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>file_or_files</code>   | One or more paths or URIs.                                                                                                                                                 |
| <code>options</code>         | A named list of scalar reader or object-store options.                                                                                                                     |
| <code>format</code>          | An optional file format name such as "parquet", "csv", or "json". By default the format is inferred from the path extension.                                               |
| <code>partitioning</code>    | Optional character vector of hive-style partition column names. NULL auto-discovers partitions; <code>character()</code> disables discovery.                               |
| <code>check_extension</code> | Whether to check extensions for explicitly selected formats, including compression inference. Directory listings are always filtered to matching files. Defaults to FALSE. |
| <code>ctx</code>             | A SedonaDB context.                                                                                                                                                        |

**Value**

A `sedonadb_dataframe`

**Examples**

```
path <- system.file("files/natural-earth_cities_geo.parquet", package = "sedonadb")
sd_read(path) |> head(5) |> sd_preview()
```

sd\_read\_parquet

*Create a DataFrame from one or more Parquet files***Description**

The query will only be executed when requested.

**Usage**

```
sd_read_parquet(
  path,
  options = NULL,
  geometry_columns = NULL,
  validate = FALSE,
  partitioning = NULL
)

sd_ctx_read_parquet(
  ctx,
  path,
  options = NULL,
  geometry_columns = NULL,
  validate = FALSE,
  partitioning = NULL
)
```

**Arguments**

|                  |                                                                                                                                                                                                                                                           |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| path             | One or more paths or URIs to Parquet files                                                                                                                                                                                                                |
| options          | A named list of options to pass to the Parquet reader.                                                                                                                                                                                                    |
| geometry_columns | A JSON string or named list mapping column names to GeoParquet column metadata. This can be used to mark binary WKB columns as geometry columns or override existing GeoParquet metadata. The suggested jsonlite package is required when passing a list. |
| validate         | Whether to validate geometry column contents against their metadata. Currently, this validates WKB input.                                                                                                                                                 |
| partitioning     | A character vector of column names for hive-style partitioning. By default (NULL), partition columns are auto-discovered. Use character() to disable partitioning.                                                                                        |
| ctx              | A SedonaDB context.                                                                                                                                                                                                                                       |

**Value**

A sedonadb\_dataframe

**Examples**

```
path <- system.file("files/natural-earth_cities_geo.parquet", package = "sedonadb")
sd_read_parquet(path) |> head(5) |> sd_preview()
```

sd\_read\_sf

*Read GDAL/OGR via the sf package***Description**

Uses the ArrowArrayStream interface to GDAL exposed via the sf package to read GDAL/OGR-based data sources.

**Usage**

```
sd_read_sf(
  dsn,
  layer = NULL,
  ...,
  query = NA,
  options = NULL,
  drivers = NULL,
  filter = NULL,
  fid_column_name = NULL,
  lazy = FALSE
)
```

```
sd_ctx_read_sf(
  ctx,
  dsn,
  layer = NULL,
  ...,
  query = NA,
  options = NULL,
  drivers = NULL,
  filter = NULL,
  fid_column_name = NULL,
  lazy = FALSE
)
```

**Arguments**

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| dsn, layer | Description of datasource and layer. See <a href="#">sf::read_sf()</a> for details. |
| ...        | Currently unused and must be empty                                                  |
| query      | A SQL query to pass on to GDAL/OGR.                                                 |
| options    | A character vector with layer open options in the form "KEY=VALUE".                 |

|                 |                                                                                                                                                                                                                                                                                     |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| drivers         | A list of drivers to try if the dsn cannot be guessed.                                                                                                                                                                                                                              |
| filter          | A spatial object that may be used to filter while reading. In the future SedonaDB will automatically calculate this value based on the query. May be any spatial object that can be converted to WKT via <code>wk::as_wkt()</code> . This filter's CRS must match that of the data. |
| fid_column_name | An optional name for the feature id (FID) column.                                                                                                                                                                                                                                   |
| lazy            | Use TRUE to stream the data from the source rather than collect first. This can be faster for large data sources but can also be confusing because the data may only be scanned exactly once.                                                                                       |
| ctx             | A SedonaDB context created using <code>sd_connect()</code> .                                                                                                                                                                                                                        |

**Value**

A SedonaDB DataFrame.

**Examples**

```
nc_gpkg <- system.file("gpkg/nc.gpkg", package = "sf")
sd_read_sf(nc_gpkg)
```

---

|                 |                                         |
|-----------------|-----------------------------------------|
| sd_register_udf | <i>Register a user-defined function</i> |
|-----------------|-----------------------------------------|

---

**Description**

Several types of user-defined functions can be registered into a session context. Currently, the only implemented variety is an external pointer to a Rust FFI\_ScalarUDF, an example of which is available from the [DataFusion Python documentation](#).

**Usage**

```
sd_register_udf(udf)

sd_ctx_register_udf(ctx, udf)
```

**Arguments**

|     |                                            |
|-----|--------------------------------------------|
| udf | An object of class 'datafusion_scalar_udf' |
| ctx | A SedonaDB context.                        |

**Value**

NULL, invisibly

---

|           |                                                     |
|-----------|-----------------------------------------------------|
| sd_select | <i>Keep or drop columns of a SedonaDB DataFrame</i> |
|-----------|-----------------------------------------------------|

---

**Description**

Keep or drop columns of a SedonaDB DataFrame

**Usage**

```
sd_select(.data, ...)
```

**Arguments**

- .data            A sedonadb\_dataframe or an object that can be coerced to one.
- ...            One or more bare names. Evaluated like `dplyr::select()`.

**Value**

An object of class sedonadb\_dataframe

**Examples**

```
data.frame(x = 1:10, y = letters[1:10]) |> sd_select(x)
```

---

|        |                                    |
|--------|------------------------------------|
| sd_sql | <i>Create a DataFrame from SQL</i> |
|--------|------------------------------------|

---

**Description**

The query will only be executed when requested.

**Usage**

```
sd_sql(sql, ..., params = NULL)

sd_ctx_sql(ctx, sql, ..., params = NULL)
```

**Arguments**

- sql            A SQL string to execute
- ...            These dots are for future extensions and currently must be empty.
- params        A list of parameters to fill placeholders in the query.
- ctx            A SedonaDB context.

**Value**

A sedonadb\_dataframe

**Examples**

```
sd_sql("SELECT ST_Point(0, 1) as geom")
sd_sql("SELECT ST_Point($1, $2) as geom", params = list(1, 2))
sd_sql("SELECT ST_Point($x, $y) as geom", params = list(x = 1, y = 2))
```

---

|              |                                                                |
|--------------|----------------------------------------------------------------|
| sd_summarise | <i>Aggregate SedonaDB DataFrames to a single row per group</i> |
|--------------|----------------------------------------------------------------|

---

**Description**

Aggregate SedonaDB DataFrames to a single row per group

**Usage**

```
sd_summarise(.data, ..., .env = parent.frame())

sd_summarize(.data, ..., .env = parent.frame())
```

**Arguments**

- .data            A sedonadb\_dataframe or an object that can be coerced to one.
- ...            Aggregate expressions. These are evaluated in the same way as `dplyr::summarise()` except the outer expression must be an aggregate expression (e.g., `sum(x) + 1` is not currently possible).
- .env            The calling environment for programmatic usage

**Value**

An object of class sedonadb\_dataframe

**Examples**

```
data.frame(x = c(10:1, NA)) |> sd_summarise(x = sum(x, na.rm = TRUE))
```

---

|            |                                             |
|------------|---------------------------------------------|
| sd_to_view | <i>Register a DataFrame as a named view</i> |
|------------|---------------------------------------------|

---

### Description

This is useful for creating a view that can be referenced in a SQL statement. Use `sd_drop_view()` to remove it.

### Usage

```
sd_to_view(.data, table_ref, overwrite = FALSE, ctx = NULL)
```

### Arguments

|           |                                                                |
|-----------|----------------------------------------------------------------|
| .data     | A sedonadb_dataframe or an object that can be coerced to one.  |
| table_ref | The name of the view reference                                 |
| overwrite | Use TRUE to overwrite a view with the same name (if it exists) |
| ctx       | A SedonaDB context.                                            |

### Value

.data, invisibly

### Examples

```
sd_sql("SELECT 1 as one") |> sd_to_view("foofy")
sd_sql("SELECT * FROM foofy")
```

---

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| sd_transmute | <i>Create, modify, and delete columns of a SedonaDB DataFrame</i> |
|--------------|-------------------------------------------------------------------|

---

### Description

Create, modify, and delete columns of a SedonaDB DataFrame

### Usage

```
sd_transmute(.data, ...)
```

### Arguments

|       |                                                                                                                                                                                                                      |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .data | A sedonadb_dataframe or an object that can be coerced to one.                                                                                                                                                        |
| ...   | Named expressions for new columns to create. These are evaluated in the same way as <code>dplyr::transmute()</code> except does not support extra dplyr features such as <code>across()</code> or <code>.by</code> . |

**Value**

An object of class `sedonadb_dataframe`

**Examples**

```
data.frame(x = 1:10) |>  
  sd_transmute(y = x + 1L)
```

---

|                |                             |
|----------------|-----------------------------|
| sd_with_params | <i>Fill in placeholders</i> |
|----------------|-----------------------------|

---

**Description**

This is a slightly more verbose form of `sd_sql()` with params that is useful if a data frame is to be repeatedly queried.

**Usage**

```
sd_with_params(.data, ...)
```

**Arguments**

|                    |                                                                                                        |
|--------------------|--------------------------------------------------------------------------------------------------------|
| <code>.data</code> | A <code>sedonadb_dataframe</code> or an object that can be coerced to one.                             |
| <code>...</code>   | Named or unnamed parameters that will be coerced to literals with <code>as_sedonadb_literal()</code> . |

**Value**

A `sedonadb_dataframe` with the provided parameters filled into the query

**Examples**

```
sd_sql("SELECT ST_Point($1, $2) as pt") |>  
  sd_with_params(11, 12)  
sd_sql("SELECT ST_Point($x, $y) as pt") |>  
  sd_with_params(x = 11, y = 12)
```

sd\_write\_parquet

*Write DataFrame to (Geo)Parquet files***Description**

Write this DataFrame to one or more (Geo)Parquet files. For input that contains geometry columns, GeoParquet metadata is written such that suitable readers can recreate Geometry/Geography types when reading the output and potentially read fewer row groups when only a subset of the file is needed for a given query.

**Usage**

```
sd_write_parquet(
  .data,
  path,
  options = NULL,
  partition_by = character(0),
  sort_by = character(0),
  single_file_output = NULL,
  geoparquet_version = "1.0",
  overwrite_bbox_columns = FALSE,
  max_row_group_size = NULL,
  compression = NULL
)
```

**Arguments**

|                    |                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .data              | A sedonadb_dataframe or an object that can be coerced to one.                                                                                                                                                                                                                                                                                                            |
| path               | A filename or directory to which parquet file(s) should be written                                                                                                                                                                                                                                                                                                       |
| options            | A named list of key/value options to be used when constructing a parquet writer. Common options are exposed as other arguments to sd_write_parquet(); however, this argument allows setting any DataFusion Parquet writer option. If an option is specified here and by another argument to this function, the value specified as an explicit argument takes precedence. |
| partition_by       | A character vector of column names to partition by. If non-empty, applies hive-style partitioning to the output                                                                                                                                                                                                                                                          |
| sort_by            | A character vector of column names to sort by. Currently only ascending sort is supported                                                                                                                                                                                                                                                                                |
| single_file_output | Use TRUE or FALSE to force writing a single Parquet file vs. writing one file per partition to a directory. By default, a single file is written if partition_by is unspecified and path ends with .parquet                                                                                                                                                              |
| geoparquet_version | GeoParquet metadata version to write if output contains one or more geometry columns. The default ("1.0") is the most widely supported and will result in                                                                                                                                                                                                                |

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        | geometry columns being recognized in many readers; however, only includes statistics at the file level. Use "1.1" to compute an additional bounding box column for every geometry column in the output: some readers can use these columns to prune row groups when files contain an effective spatial ordering. The extra columns will appear just before their geometry column and will be named "[geom_col_name]_bbox" for all geometry columns except "geometry", whose bounding box column name is just "bbox" |
| overwrite_bbox_columns | Use TRUE to overwrite any bounding box columns that already exist in the input. This is useful in a read -> modify -> write scenario to ensure these columns are up-to-date. If FALSE (the default), an error will be raised if a bbox column already exists                                                                                                                                                                                                                                                        |
| max_row_group_size     | Target maximum number of rows in each row group. Defaults to the global configuration value (1M rows).                                                                                                                                                                                                                                                                                                                                                                                                              |
| compression            | Sets the Parquet compression codec. Valid values are: uncompressed, snappy, gzip(level), brotli(level), lz4, zstd(level), and lz4_raw. Defaults to the global configuration value (zstd(3)).                                                                                                                                                                                                                                                                                                                        |

**Value**

The input, invisibly

**Examples**

```
tmp_parquet <- tempfile(fileext = ".parquet")

sd_sql("SELECT ST_Point(1, 2, 4326) as geom") |>
  sd_write_parquet(tmp_parquet)

sd_read_parquet(tmp_parquet)
unlink(tmp_parquet)
```

---

|               |                             |
|---------------|-----------------------------|
| sedonadb_adbc | <i>SedonaDB ADBC Driver</i> |
|---------------|-----------------------------|

---

**Description**

SedonaDB ADBC Driver

**Usage**

```
sedonadb_adbc()
```

**Value**

An `adbcdrivermanager::adbc_driver()` of class 'sedonadb\_driver\_sedonadb'

**Examples**

```
library(adbcdrivermanager)

con <- sedonadb_adbc() |>
  adbc_database_init() |>
  adbc_connection_init()
con |>
  read_adbc("SELECT ST_Point(0, 1) as geometry") |>
  as.data.frame()
```

# Index

.fns, [2](#)  
.tables, [3](#)  
  
adbcdrivermanager::adbc\_driver(), [27](#)  
as\_sd\_expr(sd\_expr\_column), [9](#)  
as\_sd\_expr(), [10](#)  
as\_sedonadb\_dataframe, [3](#)  
as\_sedonadb\_literal, [4](#)  
as\_sedonadb\_literal(), [25](#)  
  
dplyr::arrange(), [4](#)  
dplyr::filter(), [11](#)  
dplyr::group\_by(), [11](#), [12](#)  
dplyr::select(), [22](#)  
dplyr::summarise(), [23](#)  
dplyr::transmute(), [24](#)  
  
is\_sd\_expr(sd\_expr\_column), [9](#)  
  
nanoarrow::convert\_array\_stream, [5](#)  
  
sd\_anti\_join(sd\_join), [12](#)  
sd\_arrange, [4](#)  
sd\_collect(sd\_compute), [5](#)  
sd\_compute, [5](#)  
sd\_configure\_proj, [6](#)  
sd\_connect, [7](#)  
sd\_connect(), [21](#)  
sd\_count, [8](#)  
sd\_cross\_join(sd\_join), [12](#)  
sd\_ctx\_drop\_view(sd\_drop\_view), [8](#)  
sd\_ctx\_read(sd\_read), [17](#)  
sd\_ctx\_read\_parquet(sd\_read\_parquet),  
[19](#)  
sd\_ctx\_read\_sf(sd\_read\_sf), [20](#)  
sd\_ctx\_register\_udf(sd\_register\_udf),  
[21](#)  
sd\_ctx\_sql(sd\_sql), [22](#)  
sd\_ctx\_view(sd\_drop\_view), [8](#)  
sd\_drop\_view, [8](#)  
sd\_drop\_view(), [24](#)  
  
sd\_expr\_aggregate\_function  
    (sd\_expr\_column), [9](#)  
sd\_expr\_alias(sd\_expr\_column), [9](#)  
sd\_expr\_any\_function(sd\_expr\_column), [9](#)  
sd\_expr\_binary(sd\_expr\_column), [9](#)  
sd\_expr\_cast(sd\_expr\_column), [9](#)  
sd\_expr\_column, [9](#)  
sd\_expr\_column(), [13](#)  
sd\_expr\_factory(sd\_expr\_column), [9](#)  
sd\_expr\_factory(), [10](#)  
sd\_expr\_is\_not\_null(sd\_expr\_column), [9](#)  
sd\_expr\_is\_null(sd\_expr\_column), [9](#)  
sd\_expr\_literal(sd\_expr\_column), [9](#)  
sd\_expr\_literal(), [4](#)  
sd\_expr\_negative(sd\_expr\_column), [9](#)  
sd\_expr\_not(sd\_expr\_column), [9](#)  
sd\_expr\_parse\_binary(sd\_expr\_column), [9](#)  
sd\_expr\_scalar\_function  
    (sd\_expr\_column), [9](#)  
sd\_filter, [11](#)  
sd\_full\_join(sd\_join), [12](#)  
sd\_group\_by, [11](#)  
sd\_inner\_join(sd\_join), [12](#)  
sd\_join, [12](#)  
sd\_join(), [14](#)  
sd\_join\_by, [14](#)  
sd\_join\_by(), [3](#), [12](#), [13](#)  
sd\_join\_contains(sd\_join\_by), [14](#)  
sd\_join\_contains(), [13](#)  
sd\_join\_coveredby(sd\_join\_by), [14](#)  
sd\_join\_coveredby(), [13](#)  
sd\_join\_covers(sd\_join\_by), [14](#)  
sd\_join\_covers(), [13](#)  
sd\_join\_crosses(sd\_join\_by), [14](#)  
sd\_join\_crosses(), [13](#)  
sd\_join\_dwithin(sd\_join\_by), [14](#)  
sd\_join\_equals(sd\_join\_by), [14](#)  
sd\_join\_equals(), [13](#)  
sd\_join\_intersects(sd\_join\_by), [14](#)

`sd_join_intersects()`, [13](#)  
`sd_join_overlaps(sd_join_by)`, [14](#)  
`sd_join_overlaps()`, [13](#)  
`sd_join_select`, [15](#)  
`sd_join_select()`, [3](#), [13](#)  
`sd_join_select_default`, [16](#)  
`sd_join_select_default()`, [13](#)  
`sd_join_touches(sd_join_by)`, [14](#)  
`sd_join_touches()`, [13](#)  
`sd_join_within(sd_join_by)`, [14](#)  
`sd_join_within()`, [13](#)  
`sd_left_join(sd_join)`, [12](#)  
`sd_preview`, [17](#)  
`sd_read`, [17](#)  
`sd_read_parquet`, [19](#)  
`sd_read_sf`, [20](#)  
`sd_register_udf`, [21](#)  
`sd_right_join(sd_join)`, [12](#)  
`sd_select`, [22](#)  
`sd_semi_join(sd_join)`, [12](#)  
`sd_sql`, [22](#)  
`sd_sql()`, [25](#)  
`sd_summarise`, [23](#)  
`sd_summarize(sd_summarise)`, [23](#)  
`sd_to_view`, [24](#)  
`sd_to_view()`, [8](#)  
`sd_transmute`, [24](#)  
`sd_ungroup(sd_group_by)`, [11](#)  
`sd_view(sd_drop_view)`, [8](#)  
`sd_with_params`, [25](#)  
`sd_write_parquet`, [26](#)  
`sedonadb_adbc`, [27](#)  
`sf::read_sf()`, [20](#)  
  
`wk::as_wkt()`, [21](#)