Package: sedonafns (via r-universe)

July 1, 2026

Title Apache SedonaDB Function Definitions and Documentation

Version 0.4.0

Description Provides function definitions and documentation for use in 'SedonaDB'.

License Apache License (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Suggests sedonadb, testthat (>= 3.0.0)

Config/build/bootstrap TRUE

Config/Needs/build glue, here, rlang, roxygen2, yaml

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-21 03:18:21 UTC

RemoteUrl https://github.com/apache/sedona-db

RemoteRef apache-sedona-db-0.4.0

RemoteSha b6f66a887eb77ab638b49eddd1aaaa6aebf353cd

RemoteSubdir r/sedonafns

Contents

`rs_band_no_data_value` *Returns the nodata value of the specified band as a double. Returns null if the band has no nodata value defined.*

Description

Returns the nodata value of the specified band as a double. Returns null if the band has no nodata value defined.

Usage

```
rs_band_no_data_value(rast = sd_missing_arg(), band = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`band`	(int): Band index (1-based). Defaults to 1 if not specified.

Value

(float64)

See Also

SedonaDB SQL documentation for RS_BANDNODATAVALUE

`rs_band_path` *Retrieves the file path of an out-of-database (out-db) raster band, returning the external raster file location referenced by the raster.*

Description

Primarily used with out-db rasters, where only raster path and geo-referencing metadata are stored in the database.

Usage

```
rs_band_path(rast = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster

Value

(utf8)

See Also

SedonaDB SQL documentation for RS_BANDPATH

`rs_band_pixel_type` *Returns the pixel data type of the specified band as a string.*

Description

Returns the pixel data type of the specified band as a string.

Usage

```
rs_band_pixel_type(rast = sd_missing_arg(), band = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`band`	(int): Band index (1-based). Defaults to 1 if not specified.

Value

(utf8)

See Also

SedonaDB SQL documentation for RS_BANDPIXELTYPE

`rs_band_to_dim` *Collapses all bands into a single band with a new dimension.*

Description

Collapses all bands in a raster into a single band by introducing a new dimension. The new dimension is prepended to the existing dimensions, with size equal to the number of bands. Band data is concatenated in band order. All bands must have identical dimension names, shapes, data types, and nodata values. If any of these differ, an error is returned — collapsing bands with different nodata sentinels would silently lose information about which sentinel applies where. The new dimension name must not already exist on the input bands. For example, `RS_BandToDim(raster_with_y_x, 'y')` is rejected because the output would carry duplicate y dimensions and the round-trip through RS_DimToBand cannot recover the original layout. This is the inverse of RS_DimToBand. A round-trip `RS_BandToDim(RS_DimToBand(raster, 'dim'), 'dim')` recovers the original data layout. This is useful for converting GeoTIFF-style multi-band rasters into a single N-dimensional chunk for export to formats like Zarr.

Usage

```
rs_band_to_dim(rast = sd_missing_arg(), dim_name = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`dim_name`	(utf8): Name for the new dimension (e.g., 'time', 'band').

Value

(raster)

See Also

SedonaDB SQL documentation for RS_BANDTODIM

`rs_contains` *Returns true if the first argument's extent contains the second.*

Description

Returns `true` if the convex hull of the first argument completely contains the second argument. Both rasters and geometries are accepted in either argument position. When two rasters are provided, their convex hulls are compared. If the arguments have different CRSes, the geometry is transformed into the raster's CRS before evaluating the predicate. For two rasters, the second raster's extent is transformed into the first raster's CRS. If the preferred transformation fails, the extent of both sides are transformed to WGS 84 as a fallback. If either argument has no CRS set, the comparison is performed directly without CRS transformation.

Usage

```
rs_contains(...)
```

Arguments

...	Supported combinations: • rast (raster), geom (geometry) • geom (geometry), rast (raster) • rast_a (raster), rast_b (raster)

Value

(boolean)

See Also

SedonaDB SQL documentation for RS_CONTAINS

---

rs_convex_hull *Returns the convex hull geometry of a raster.*

Description

Returns the convex hull geometry of a raster.

Usage

```
rs_convex_hull(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_CONVEXHULL

rs_crs *Returns the CRS string for a raster.*

Description

Returns the CRS string for a raster.

Usage

```
rs_crs(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(string)

See Also

SedonaDB SQL documentation for RS_CRS

rs_dim_names *Returns the ordered list of dimension names for a raster band.*

Description

Returns the ordered list of dimension names for a raster band. Standard 2D rasters have dimensions `["y", "x"]`. N-dimensional rasters may include additional dimensions such as `time`, `pressure`, `wavelength`, etc. The dimension names correspond to the entries in RS_Shape — the i-th name matches the i-th size. When the band index is omitted, all bands must agree on their dimension names; if they disagree, an error is returned prompting the user to specify a band index.

Usage

```
rs_dim_names(rast = sd_missing_arg(), band = sd_missing_arg())
```

Arguments

rast	(raster): Input raster
band	(int): Band index (1-based). A null or out-of-range index returns null.

Value

(list)

See Also

SedonaDB SQL documentation for RS_DIMNAMES

`rs_dim_size` *Returns the size of a named dimension in a raster band.*

Description

Returns the size of a named dimension in a raster. For example, `RS_DimSize(raster, 'x')` returns the width and `RS_DimSize(raster, 'time')` returns the number of time steps. This is equivalent to looking up a specific entry in RS_Shape by name rather than by position. ### Heterogeneous rasters Bands within a raster can have different dimensionality. When the band index is omitted, `RS_DimSize` considers only the bands that carry the named dimension ("pass-through"): - If at least one band carries the dimension and they all agree on its size, that size is returned. - If the bands that carry the dimension disagree on its size, an error is returned prompting the user to specify a band index. - If **no** band carries the dimension, an error is returned. This usually signals a typo'd dimension name. When the band index is supplied explicitly, returns the size of the dimension in that band, or `NULL` if the band does not carry it.

Usage

```
rs_dim_size(
  rast = sd_missing_arg(),
  dim_name = sd_missing_arg(),
  band = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`dim_name`	(utf8): Name of the dimension to query.
`band`	(int): Band index (1-based). A null or out-of-range index returns null.

Value

(int64)

See Also

SedonaDB SQL documentation for RS_DIMSIZE

rs_dim_to_band *Promotes a within-band dimension into separate bands.*

Description

Promotes a named non-spatial dimension into separate bands. Each index along the dimension becomes its own band with that dimension removed. This bridges the gap between the dimension model (where all indices live in one band) and the band model (where each index is a separate band accessible by number). For example, a raster with 1 band of shape `[wavelength=200, y=256, x=256]` becomes a raster with 200 bands, each of shape `[y=256, x=256]`. Standard band math functions can then operate on individual wavelength bands by index. Bands that do not contain the named dimension are passed through unchanged. If **no** band carries the named dimension, the function errors — that condition usually signals a typo'd dimension name. The spatial dimensions (`x_dim` and `y_dim`) cannot be promoted. The inverse operation is RS_BandToDim.

Usage

```
rs_dim_to_band(rast = sd_missing_arg(), dim_name = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`dim_name`	(utf8): Name of the dimension to promote (e.g., 'wavelength', 'time').

Value

(raster)

See Also

SedonaDB SQL documentation for RS_DIMTOBAND

rs_envelope *Returns the envelope (bounding box) of a raster as a geometry.*

Description

Returns the envelope (bounding box) of a raster as a geometry.

Usage

```
rs_envelope(rast = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_ENVELOPE

`rs_example` *Creates a simple example raster for testing and demos.*

Description

Creates a simple example raster for testing and demos.

Usage

```
rs_example()
```

Value

(raster)

See Also

SedonaDB SQL documentation for RS_EXAMPLE

`rs_from_path` *Creates an out-of-database raster from a raster file path.*

Description

Loads raster metadata from the file at `path` and returns a raster whose bands reference the source file as out-db bands. This is useful when you want to work with rasters stored on disk without copying their pixel data into the raster value itself.

Usage

```
rs_from_path(path = sd_missing_arg())
```

Arguments

`path`	(string): Input string

Value

(raster)

See Also

SedonaDB SQL documentation for RS_FROMPATH

`rs_geo_reference` *Returns the georeference metadata of raster as a string in GDAL or ESRI format as commonly seen in a world file. Default is GDAL if not specified. Both formats output six lines: scalex, skewy, skewx, scaley, upperleftx, upperlefty. In GDAL format the upper-left coordinates refer to the corner of the upper-left pixel, while in ESRI format they are shifted to the center of the upper-left pixel.*

Description

Returns the georeference metadata of raster as a string in GDAL or ESRI format as commonly seen in a world file. Default is GDAL if not specified. Both formats output six lines: scalex, skewy, skewx, scaley, upperleftx, upperlefty. In GDAL format the upper-left coordinates refer to the corner of the upper-left pixel, while in ESRI format they are shifted to the center of the upper-left pixel.

Usage

```
rs_geo_reference(rast = sd_missing_arg(), format = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`format`	(utf8): Output format, either 'GDAL' (default) or 'ESRI'. GDAL reports the upper-left corner of the upper-left pixel; ESRI shifts the coordinates to the center of the upper-left pixel.

Value

(utf8)

See Also

SedonaDB SQL documentation for RS_GEOREFERENCE

rs_height — *Returns the height of a raster in pixels.*

Description

Returns the height of a raster in pixels.

Usage

```
rs_height(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(bigint)

See Also

SedonaDB SQL documentation for RS_HEIGHT

rs_intersects — *Returns true if the extents of the two arguments intersect.*

Description

Returns `true` if the convex hull of the first argument intersects the second argument. Both rasters and geometries are accepted in either argument position. When two rasters are provided, their convex hulls are compared. If the arguments have different CRSes, the geometry is transformed into the raster's CRS before evaluating the predicate. For two rasters, the second raster's extent is transformed into the first raster's CRS. If the preferred transformation fails, the extent of both sides are transformed to WGS 84 as a fallback. If either argument has no CRS set, the comparison is performed directly without CRS transformation.

Usage

```
rs_intersects(...)
```

Arguments

...	Supported combinations: • rast (raster), geom (geometry) • geom (geometry), rast (raster) • rast_a (raster), rast_b (raster)

Value

(boolean)

See Also

SedonaDB SQL documentation for RS_INTERSECTS

`rs_is_empty` *Returns true when the raster has zero visible volume — some dimension has length 0, so it holds no cells. This is visible-volume emptiness, not byte-emptiness: a lazy raster whose pixels are resolved on demand carries no bytes but is not empty.*

Description

Returns true when the raster has zero visible volume — some dimension has length 0, so it holds no cells. This is visible-volume emptiness, not byte-emptiness: a lazy raster whose pixels are resolved on demand carries no bytes but is not empty.

Usage

```
rs_is_empty(rast = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster

Value

(boolean)

See Also

SedonaDB SQL documentation for RS_ISEMPTY

rs_meta_data *Returns raster metadata as a struct.*

Description

RS_MetaData() returns a struct containing: - upperLeftX, upperLeftY: origin of the raster geotransform - gridWidth, gridHeight: raster dimensions in pixels - scaleX, scaleY: pixel scale - skewX, skewY: geotransform skew terms - srid: raster SRID if available, otherwise 0 - numSampleDimensions: band count - tileWidth, tileHeight: GDAL block size derived tile dimensions For rasters with no bands, tileWidth and tileHeight are 0.

Usage

```
rs_meta_data(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(struct)

See Also

SedonaDB SQL documentation for RS_METADATA

rs_num_bands *Returns the number of bands in the raster.*

Description

Returns the number of bands in the raster.

Usage

```
rs_num_bands(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(uint32)

See Also

SedonaDB SQL documentation for RS_NUMBANDS

`rs_num_dimensions` *Returns the number of dimensions in a raster band.*

Description

Returns the number of dimensions in a raster band. A standard 2D raster has 2 dimensions (`y` and `x`). N-dimensional rasters (e.g., from Zarr or NetCDF sources) may have additional dimensions such as `time`, `pressure`, or `wavelength`. When the band index is omitted, all bands must agree on the number of dimensions; if they disagree, an error is returned prompting the user to specify a band index.

Usage

```
rs_num_dimensions(rast = sd_missing_arg(), band = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`band`	(int): Band index (1-based). A null or out-of-range index returns null.

Value

(int32)

See Also

SedonaDB SQL documentation for RS_NUMDIMENSIONS

`rs_pixel_as_centroid` *Returns the centroid of the specified pixel as a Point geometry.*

Description

Returns the centroid of the specified pixel as a Point geometry.

Usage

```
rs_pixel_as_centroid(
  rast = sd_missing_arg(),
  colX = sd_missing_arg(),
  rowY = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`colX`	(integer): Input integer
`rowY`	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_PIXELASCENTROID

`rs_pixel_as_point` *Returns the upper-left corner of the specified pixel as a Point geometry.*

Description

Returns the upper-left corner of the specified pixel as a Point geometry.

Usage

```
rs_pixel_as_point(
  rast = sd_missing_arg(),
  colX = sd_missing_arg(),
  rowY = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`colX`	(integer): Input integer
`rowY`	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_PIXELASPOINT

rs_pixel_as_polygon *Returns the bounding polygon of the specified pixel.*

Description

Returns the bounding polygon of the specified pixel.

Usage

```
rs_pixel_as_polygon(
  rast = sd_missing_arg(),
  colX = sd_missing_arg(),
  rowY = sd_missing_arg()
)
```

Arguments

rast	(raster): Input raster
colX	(integer): Input integer
rowY	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_PIXELASPOLYGON

rs_raster_to_world_coord *Converts raster pixel coordinates to world coordinates as a point.*

Description

Converts raster pixel coordinates to world coordinates as a point.

Usage

```
rs_raster_to_world_coord(
  rast = sd_missing_arg(),
  x = sd_missing_arg(),
  y = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`x`	(integer): Input integer
`y`	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_RASTERTOWORLDCOORD

`rs_raster_to_world_coord_x`

Converts raster pixel coordinates to world X coordinate.

Description

Converts raster pixel coordinates to world X coordinate.

Usage

```
rs_raster_to_world_coord_x(
  rast = sd_missing_arg(),
  x = sd_missing_arg(),
  y = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`x`	(integer): Input integer
`y`	(integer): Input integer

Value

(double)

See Also

SedonaDB SQL documentation for RS_RASTERTOWORLDCOORDX

rs_raster_to_world_coord_y

Converts raster pixel coordinates to world Y coordinate.

Description

Converts raster pixel coordinates to world Y coordinate.

Usage

```
rs_raster_to_world_coord_y(
  rast = sd_missing_arg(),
  x = sd_missing_arg(),
  y = sd_missing_arg()
)
```

Arguments

rast	(raster): Input raster
x	(integer): Input integer
y	(integer): Input integer

Value

(double)

See Also

SedonaDB SQL documentation for RS_RASTERTOWORLDCOORDY

rs_rotation

Returns the raster rotation in radians based on skew parameters.

Description

Returns the raster rotation in radians based on skew parameters.

Usage

```
rs_rotation(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_ROTATION

---

rs_scale_x *Returns the pixel width (scale X) of a raster.*

Description

Returns the pixel width (scale X) of a raster.

Usage

```
rs_scale_x(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_SCALEX

---

rs_scale_y *Returns the pixel height (scale Y) of a raster.*

Description

Returns the pixel height (scale Y) of a raster.

Usage

```
rs_scale_y(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_SCALEY

`rs_set_crs` *Sets the Coordinate Reference System (CRS) for a raster.*

Description

Sets a CRS on a raster. This is metadata-only: raster cell values and geotransform are not transformed.

Usage

```
rs_set_crs(rast = sd_missing_arg(), target_crs = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`target_crs`	(string): Input string

Value

(raster)

See Also

SedonaDB SQL documentation for RS_SETCRS

`rs_set_srid` *Sets the SRID (spatial reference identifier) for a raster.*

Description

Sets the SRID of a raster. This is metadata-only: raster cell values and geotransform are not transformed.

Usage

```
rs_set_srid(rast = sd_missing_arg(), srid = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`srid`	(integer): Input integer

Value

(raster)

See Also

SedonaDB SQL documentation for RS_SETSRID

`rs_shape` *Returns the shape (size of each dimension) of a raster band.*

Description

Returns the shape of a raster band as a list of dimension sizes. The entries correspond to the dimension names returned by RS_DimNames. For a standard 2D raster this returns `[height, width]`. For an N-dimensional raster with a time dimension it might return `[12, 256, 256]` meaning 12 time steps at 256x256 spatial resolution. When the band index is omitted, all bands must agree on their shape; if they disagree, an error is returned prompting the user to specify a band index.

Usage

```
rs_shape(rast = sd_missing_arg(), band = sd_missing_arg())
```

Arguments

`rast`	(raster): Input raster
`band`	(int): Band index (1-based). A null or out-of-range index returns null.

Value

(list)

See Also

SedonaDB SQL documentation for RS_SHAPE

rs_skew_x *Returns the X skew (rotation) parameter of a raster.*

Description

Returns the X skew (rotation) parameter of a raster.

Usage

```
rs_skew_x(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_SKEWX

rs_skew_y *Returns the Y skew (rotation) parameter of a raster.*

Description

Returns the Y skew (rotation) parameter of a raster.

Usage

```
rs_skew_y(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_SKEWY

`rs_slice` *Selects a single index along a dimension, removing that dimension from the output.*

Description

Extracts a single slice along a named non-spatial dimension, removing that dimension from every band that carries it in the output raster. For example, slicing a raster with shape `[time=12, y=256, x=256]` on `'time'` at index 5 produces a 2D raster with shape `[y=256, x=256]` containing the data at time step 5. The spatial dimensions (`x_dim` and `y_dim`) cannot be sliced. ### Heterogeneous rasters Bands within a raster can have different dimensionality. Bands that do not carry the named dimension are emitted unchanged ("pass-through") rather than producing an error. This matches the convention used by `RS_DimToBand` and `xarray`'s `ds.isel({dim: i})`. For example, with a raster containing an elevation band (`[y, x]`) and a temperature band (`[time, y, x]`), `RS_Slice(raster, 'time', 5)` slices the temperature band and passes the elevation band through unchanged. If **no** band carries the named dimension, the function errors — that condition usually signals a typo'd dimension name. Returns an error if the index is out of range for any band that does carry the dimension. See also RS_SliceRange to extract a range of indices while keeping the dimension.

Usage

```
rs_slice(
  rast = sd_missing_arg(),
  dim_name = sd_missing_arg(),
  index = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`dim_name`	(utf8): Name of the dimension to slice (e.g., 'time').
`index`	(int): Zero-based index along the dimension.

Value

(raster)

See Also

SedonaDB SQL documentation for RS_SLICE

`rs_slice_range` *Narrows a dimension to a half-open range, keeping the dimension with reduced size.*

Description

Narrows a named non-spatial dimension to the half-open range `[start, end)` on every band that carries it, keeping the dimension in the output with reduced size. For example, narrowing a raster with shape `[time=12, y=256, x=256]` on `'time'` with range `[2, 7)` produces a raster with shape `[time=5, y=256, x=256]`. The spatial dimensions (`x_dim` and `y_dim`) cannot be narrowed. ### Heterogeneous rasters Bands within a raster can have different dimensionality. Bands that do not carry the named dimension are emitted unchanged ("pass-through") rather than producing an error. This matches the convention used by `RS_Slice`, `RS_DimToBand`, and `xarray`'s `ds.isel`. If **no** band carries the named dimension, the function errors — that condition usually signals a typo'd dimension name. Returns an error if `start >= end` or the range is out of bounds for any band that does carry the dimension. See also RS_Slice to extract a single index and remove the dimension entirely.

Usage

```
rs_slice_range(
  rast = sd_missing_arg(),
  dim_name = sd_missing_arg(),
  start = sd_missing_arg(),
  end = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`dim_name`	(utf8): Name of the dimension to narrow (e.g., 'time').
`start`	(int): Start index (inclusive, zero-based).
`end`	(int): End index (exclusive).

Value

(raster)

See Also

SedonaDB SQL documentation for RS_SLICERANGE

rs_srid — *Returns the SRID of a raster.*

Description

Returns the SRID of a raster.

Usage

```
rs_srid(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(integer)

See Also

SedonaDB SQL documentation for RS_SRID

rs_upper_left_x — *Returns the upper-left X coordinate of a raster.*

Description

Returns the upper-left X coordinate of a raster.

Usage

```
rs_upper_left_x(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_UPPERLEFTX

rs_upper_left_y *Returns the upper-left Y coordinate of a raster.*

Description

Returns the upper-left Y coordinate of a raster.

Usage

```
rs_upper_left_y(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(double)

See Also

SedonaDB SQL documentation for RS_UPPERLEFTY

rs_width *Returns the width of a raster in pixels.*

Description

Returns the width of a raster in pixels.

Usage

```
rs_width(rast = sd_missing_arg())
```

Arguments

rast	(raster): Input raster

Value

(bigint)

See Also

SedonaDB SQL documentation for RS_WIDTH

rs_within — *Returns true if the first argument's extent is within the second.*

Description

Returns `true` if the convex hull of the first argument is completely within the second argument. This is the inverse of `RS_Contains`: `RS_Within(A, B)` is equivalent to `RS_Contains(B, A)`. Both rasters and geometries are accepted in either argument position. When two rasters are provided, their convex hulls are compared. If the arguments have different CRSes, the geometry is transformed into the raster's CRS before evaluating the predicate. For two rasters, the second raster's extent is transformed into the first raster's CRS. If the preferred transformation fails, the extent of both sides are transformed to WGS 84 as a fallback. If either argument has no CRS set, the comparison is performed directly without CRS transformation.

Usage

```
rs_within(...)
```

Arguments

`...`	Supported combinations: • rast (raster), geom (geometry) • geom (geometry), rast (raster) • rast_a (raster), rast_b (raster)

Value

(boolean)

See Also

SedonaDB SQL documentation for RS_WITHIN

rs_world_to_raster_coord — *Converts world coordinates to raster coordinates as a point.*

Description

Converts world coordinates to raster coordinates as a point.

Usage

```
rs_world_to_raster_coord(
  rast = sd_missing_arg(),
  x = sd_missing_arg(),
  y = sd_missing_arg()
)
```

Arguments

rast	(raster): Input raster
x	(double): Input double
y	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for RS_WORLDTORASTERCOORD

---

rs_world_to_raster_coord_x

Converts world coordinates to raster X coordinate.

Description

Converts world coordinates to raster X coordinate.

Usage

```
rs_world_to_raster_coord_x(
  rast = sd_missing_arg(),
  x = sd_missing_arg(),
  y = sd_missing_arg()
)
```

Arguments

rast	(raster): Input raster
x	(double): Input double
y	(double): Input double

Value

(integer)

See Also

SedonaDB SQL documentation for RS_WORLDTORASTERCOORDX

`rs_world_to_raster_coord_y` *Converts world coordinates to raster Y coordinate.*

Description

Converts world coordinates to raster Y coordinate.

Usage

```
rs_world_to_raster_coord_y(
  rast = sd_missing_arg(),
  x = sd_missing_arg(),
  y = sd_missing_arg()
)
```

Arguments

`rast`	(raster): Input raster
`x`	(double): Input double
`y`	(double): Input double

Value

(integer)

See Also

SedonaDB SQL documentation for RS_WORLDTORASTERCOORDY

`sd_affine` *Applies an affine transformation to a geometry.*

Description

Applies an affine transformation to a geometry.

Usage

```
sd_affine(...)
```

Arguments

...	Supported combinations: • geom (geometry), a (double), b (double), d (double), e (double), xOff (double), yOff (double) • geom (geometry), a (double), b (double), c (double), d (double), e (double), f (double), g (double), h (double), i (double), xOff (double), yOff (double), zOff (double)

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_AFFINE

---

sd_analyze_agg *Computes the statistics of geometries for the input geometry or geography.*

Description

`ST_Analyze_Agg()` provides a high-level summary of its input geometries. The fields of its struct return type are: - `count`: Number of input geometries - `minx`, `miny`, `maxx`, `maxy`: Minimum bounding rectangle (envelope). For geography, this represents geographical bounds such that minx may be greater than maxx where this represents more compact bounds (e.g., a small feature crossing the antimeridian). - `mean_size_in_bytes` - `mean_points_per_geometry` - `puntal_count`: Number of point or multipoint geometries - `lineal_count`: Number of line or multilinestring geometries - `polygonal_count`: Number of polygon or multipolygon geometries - `geometrycollection_count`: Number of geometrycollection geometries - `mean_envelope_width` - `mean_envelope_height` - `mean_envelope_area`

Usage

```
sd_analyze_agg(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(struct)

See Also

SedonaDB SQL documentation for ST_ANALYZE_AGG

sd_area — *Returns the area of a geometry or geography.*

Description

`ST_Area()` Returns the area of a polygon or multi-polygon. For geometry, returns the area in the units of the coordinate reference system (squared); for geography, returns the area in square meters, calculated on a spherical Earth model.

Usage

```
sd_area(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_AREA

sd_as_binary — *Converts a geometry or geography to Well-Known Binary (WKB) format.*

Description

`ST_AsBinary()` Returns the Well-Known Binary representation of a geometry or geography. This function also has the alias ST_AsWKB. `ST_AsBinary()` may preserve the underlying Arrow storage type of the input geometry. For example, when the input geometry is backed by Arrow `BinaryView` storage, the WKB output may also use `BinaryView` storage. If a downstream consumer requires simpler Arrow storage such as `Binary`, wrap the result with `sd_simplifystorage()`.

Usage

```
sd_as_binary(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(binary)

See Also

SedonaDB SQL documentation for ST_ASBINARY

---

sd_as_ewkb *Returns the EWKB representation of a geometry or geography.*

Description

EWKB extends WKB to include SRID information in the binary header and is useful for PostGIS compatibility. This function preserves item-level CRSes or will repeat a type-level CRS for all elements if present.

Usage

```
sd_as_ewkb(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(binary)

See Also

SedonaDB SQL documentation for ST_ASEWKB

---

sd_as_geo_json *Returns the GeoJSON representation of a geometry.*

Description

Returns the GeoJSON representation of a geometry.

Usage

```
sd_as_geo_json(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(string)

See Also

SedonaDB SQL documentation for ST_ASGEOJSON

---

sd_as_text

Returns the Well-Known Text string representation of a geometry or geography.

Description

Alias: `ST_AsWKT`.

Usage

```
sd_as_text(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(string)

See Also

SedonaDB SQL documentation for ST_ASTEXT

---

sd_azimuth

Returns the azimuth between two points in radians, or NULL if not available.

Description

Returns the azimuth (angle) from the first point to the second point, measured in **radians** clockwise from north. Returns NULL if either input is not a point.

Usage

```
sd_azimuth(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_AZIMUTH

`sd_boundary` *Returns the closure of the combinatorial boundary of this geometry or geography.*

Description

For a polygon, the boundary is the exterior and interior rings; for a linestring it is the endpoints.

Usage

```
sd_boundary(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_BOUNDARY

`sd_buffer` *Computes a geometry or geography that represents all points whose distance from the input is less than or equal to a specified distance.*

Description

For geography, the distance is specified in meters. The buffer is computed on the sphere, producing a result that considers polar regions and the antimeridian. Negative distances are supported for polygon inputs to shrink the polygon.

Usage

```
sd_buffer(...)
```

Arguments

...	Supported combinations: • geom (geometry), distance (float64) • geom (geometry), distance (float64), params (string) • geom (geography), distance (float64) • geom (geography), distance (float64), num_quad_segs (integer) • geom (geography), distance (float64), params (string)

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_BUFFER

`sd_cell_id_from_point` *Returns the S2 cell ID containing the given point geography at the maximum S2 level (30).*

Description

Returns the 64-bit S2 cell ID at level 30 (the finest resolution) that contains the given point geography. Returns NULL for empty point geometries. This is useful for spatial indexing and efficient spatial queries.

Usage

```
sd_cell_id_from_point(geom = sd_missing_arg())
```

Arguments

geom	(geography): Input geography

Value

(int64)

See Also

SedonaDB SQL documentation for S2_CELLIDFROMPOINT

---

sd_centroid *Returns the centroid of a geometry or geography.*

Description

For geography, the centroid is calculated using spherical geometry.

Usage

```
sd_centroid(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_CENTROID

sd_closest_point — *Returns the 2-dimensional point on geom1 that is closest to geom2.*

Description

Returns the 2-dimensional point on geom1 that is closest to geom2.

Usage

```
sd_closest_point(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_CLOSESTPOINT

sd_collect_agg — *Combines multiple geometries from a set of rows into a single collection.*

Description

An aggregate function that collects multiple geometries into a single GeometryCollection or the appropriate Multi-type if all inputs share the same geometry type.

Usage

```
sd_collect_agg(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_COLLECT_AGG

sd_concave_hull *Returns a concave hull enclosing the input geometry.*

Description

Returns a concave hull enclosing the input geometry. The `pct_convex` parameter controls how "tight" the hull is: 1.0 produces the convex hull, while smaller values produce tighter, more concave shapes.

Usage

```
sd_concave_hull(geom = sd_missing_arg(), pct_convex = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`pct_convex`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_CONCAVEHULL

sd_contains *Returns true if geomA contains geomB.*

Description

Returns `true` if no points of B lie outside A and at least one point of B lies inside A.

Usage

```
sd_contains(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_CONTAINS

`sd_convex_hull` *Returns the Convex Hull of a geometry or geography.*

Description

Returns the smallest convex polygon that encloses all points in the input geometry.

Usage

```
sd_convex_hull(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_CONVEXHULL

`sd_covered_by` *Returns true if geomA is covered by geomB.*

Description

Returns `true` if no point in geometry A is outside geometry B. Inverse of `ST_Covers`: `ST_CoveredBy(A, B)` is equivalent to `ST_Covers(B, A)`.

Usage

```
sd_covered_by(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_COVEREDBY

`sd_covering_cell_ids` *Returns an array of S2 cell IDs that cover the given geography.*

Description

Returns an array of 64-bit S2 cell IDs that form a covering of the given geography. A covering is a set of cells that completely contains the geography with minimal overlap. For a point, returns an array with a single cell ID. For empty geometries, returns an empty array. This is useful for spatial indexing and efficient spatial queries. The returned cells may be at different S2 levels depending on the size and shape of the geography.

Usage

```
sd_covering_cell_ids(geom = sd_missing_arg())
```

Arguments

`geom`	(geography): Input geography

Value

(array)

See Also

SedonaDB SQL documentation for S2_COVERINGCELLIDS

`sd_covers` *Returns true if geomA covers geomB.*

Description

Returns `true` if no point in geometry B is outside geometry A. Similar to `ST_Contains` but does not require interior intersection.

Usage

```
sd_covers(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_COVERS

`sd_crosses` *Returns true if A crosses B.*

Description

Returns `true` if the two geometries have some (but not all) interior points in common and the dimension of the intersection is less than that of either input.

Usage

```
sd_crosses(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_CROSSES

`sd_crs` *Returns the Coordinate Reference System (CRS) metadata associated with a geometry or geography object.*

Description

Returns the CRS (Coordinate Reference System) string associated with a geometry or geography. Whether it was set as an authority code (e.g. `'EPSG:3857'`), PROJJSON, or WKT, the CRS is returned in the same form it was set.

Usage

```
sd_crs(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(string)

See Also

SedonaDB SQL documentation for ST_CRS

`sd_d_within` *Returns true if two geometries or geographies are within a specified distance of each other.*

Description

For geography, the distance is specified in meters and represents the spherical approximation of the geodesic distance between the two geographies.

Usage

```
sd_d_within(
  geom_a = sd_missing_arg(),
  geom_b = sd_missing_arg(),
  distance = sd_missing_arg()
)
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry
`distance`	(double): Input double

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_DWITHIN

sd_difference *Computes the difference between geomA and geomB.*

Description

Returns the part of geometry A that does not intersect with geometry B.

Usage

```
sd_difference(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_DIFFERENCE

sd_dimension *Returns the dimension of the geometry or geography.*

Description

Returns the inherent dimension of the geometry: - 0 for points or multipoints

- 1 for linestring or multilinestrings - 2 for polygons or multipolygons For geometry collections, returns the largest dimension among the components.

Usage

```
sd_dimension(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_DIMENSION

sd_disjoint *Returns true if geomA is disjoint from geomB.*

Description

Returns `true` if the two geometries do not share any points. This is the inverse of `ST_Intersects`.

Usage

```
sd_disjoint(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_DISJOINT

sd_distance *Returns the distance between two geometries or geographies.*

Description

For geometry, returns the 2D Cartesian (planar) distance between two geometries, in the units of the coordinate reference system. For geography, returns the spherical approximation of the geodesic distance between two geographies in meters.

Usage

```
sd_distance(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_DISTANCE

`sd_dump` *Expands multi-part geometries into child parts*

Description

If the geometry is simple it returns the geometry itself, if the geometry is collection or multi it returns record for each of collection components.

Usage

```
sd_dump(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(struct)

See Also

SedonaDB SQL documentation for ST_DUMP

`sd_end_point` *Returns last point of a linestring.*

Description

Returns last point of a linestring.

Usage

```
sd_end_point(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_ENDPOINT

sd_envelope — *Returns the bounding box (envelope) of a geometry or geography as a new geometry.*

Description

Returns the minimum axis-aligned bounding box of a geometry. This is usually a polygon but can be a linestring for perfectly vertical or horizontal input and is a point if the input is also a point. For geography, the return value is still a rectangular polygon geometry that represents the latitude and longitude range of the input. For geographies that span the antimeridian, a multipolygon is returned (one on each side of the antimeridian). Because of slight numerical inaccuracy of geographical calculations, the bounds are expanded slightly compared to the corresponding geometry envelope.

Usage

```
sd_envelope(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_ENVELOPE

sd_envelope_agg — *An aggregate function that returns the collective bounding box (envelope) of a set of geometries.*

Description

An aggregate function that computes the collective bounding box (envelope) of all geometries in a group.

Usage

```
sd_envelope_agg(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_ENVELOPE_AGG

---

sd_equals *Returns true if geomA equals geomB.*

Description

Returns `true` if the two geometries are topologically equal (i.e., they represent the same point set, regardless of vertex order).

Usage

```
sd_equals(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_EQUALS

---

sd_flip_coordinates *Returns a new geometry with the X and Y coordinates of each vertex swapped.*

Description

Returns a new geometry with the X and Y coordinates of each vertex swapped.

Usage

```
sd_flip_coordinates(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_FLIPCOORDINATES

---

sd_force2d *Forces a geometry or geography into a XY coordinate model.*

Description

Discards any Z and M values present (if any).

Usage

```
sd_force2d(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_FORCE2D

sd_force3d

Forces a geometry or geography into a XYZ coordinate model with an optional Z value.

Description

If the geometry already has Z values they are preserved; otherwise the optional `z` argument (default 0) is used.

Usage

```
sd_force3d(geom = sd_missing_arg(), z = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`z`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_FORCE3D

sd_force3dm

Forces a geometry or geography into a XYM coordinate model with an optional M value.

Description

If the geometry already has M values they are preserved; otherwise the optional `m` argument (default 0) is used.

Usage

```
sd_force3dm(geom = sd_missing_arg(), m = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`m`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_FORCE3DM

`sd_force4d` *Forces a geometry or geography into a XYZM coordinate model with optional Z and M values.*

Description

If the geometry already has Z and M values they are preserved; otherwise optional `z` and `m` arguments (both default 0) are used.

Usage

```
sd_force4d(geom = sd_missing_arg(), z = sd_missing_arg(), m = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`z`	(double): Input double
`m`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_FORCE4D

sd_geog_from_wkb *Constructs a Geography from WKB Binary.*

Description

Constructs a Geography from WKB Binary.

Usage

```
sd_geog_from_wkb(wkb = sd_missing_arg())
```

Arguments

wkb	(binary): Input binary

Value

(geography)

See Also

SedonaDB SQL documentation for ST_GEOGFROMWKB

sd_geog_from_wkt *Constructs a Geography from WKT.*

Description

Alias: ST_GeogFromText.

Usage

```
sd_geog_from_wkt(wkt = sd_missing_arg(), srid = sd_missing_arg())
```

Arguments

wkt	(string): Input string
srid	(integer): Input integer

Value

(geography)

See Also

SedonaDB SQL documentation for ST_GEOGFROMWKT

`sd_geog_point` *Creates a geography POINT from given longitude and latitude coordinates.*

Description

Creates a geography Point from longitude and latitude coordinates.

Usage

```
sd_geog_point(longitude = sd_missing_arg(), latitude = sd_missing_arg())
```

Arguments

`longitude`	(double): Input double
`latitude`	(double): Input double

Value

(geography)

See Also

SedonaDB SQL documentation for ST_GEOGPOINT

`sd_geom_from_ewkb` *Constructs a geometry from Extended Well-Known Binary (EWKB).*

Description

Parses an EWKB binary value and returns a geometry. EWKB extends the WKB format to include SRID information in the binary header and is useful for PostGIS compatibility.

Usage

```
sd_geom_from_ewkb(ewkb = sd_missing_arg())
```

Arguments

`ewkb`	(binary): Input binary

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_GEOMFROMEWKB

sd_geom_from_ewkt *Constructs a geometry from Extended Well-Known Text (EWKT).*

Description

Parses an EWKT string in the format SRID=<srid>;<wkt> and returns a geometry with the embedded SRID set. EWKT is used by PostGIS and other systems for human-readable output when an item-level SRID required.

Usage

```
sd_geom_from_ewkt(ewkt = sd_missing_arg())
```

Arguments

ewkt	(string): Input string

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_GEOMFROMEWKT

sd_geom_from_wkb *Constructs a Geometry from Well-Known Binary (WKB).*

Description

Parses a WKB binary value and returns a geometry. The input is validated by default. An optional SRID or CRS can be provided as a second argument.

Usage

```
sd_geom_from_wkb(wkb = sd_missing_arg(), srid = sd_missing_arg())
```

Arguments

wkb	(binary): Input binary
srid	(crs): Input crs

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_GEOMFROMWKB

`sd_geom_from_wkt` *Constructs a Geometry from Well-Known Text (WKT).*

Description

An optional SRID or CRS can be provided as a second argument to set the spatial reference. Aliases: `ST_GeomFromText`, `ST_GeometryFromText`.

Usage

```
sd_geom_from_wkt(wkt = sd_missing_arg(), srid = sd_missing_arg())
```

Arguments

`wkt`	(string): Input string
`srid`	(crs): Input crs

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_GEOMFROMWKT

`sd_geometry_n` *Returns the 1-based Nth geometry if the geometry is a GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINESTRING, MULTICURVE or (MULTI)POLYGON.*

Description

Returns NULL if the index is out of range.

Usage

```
sd_geometry_n(geom = sd_missing_arg(), n = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`n`	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_GEOMETRYN

---

sd_geometry_type *Returns the type of a geometry or geography.*

Description

Returns the OGC geometry type name prefixed with `ST_`. Possible return values include: `'ST_Point'`, `'ST_LineString'`, `'ST_Polygon'`, `'ST_MultiPoint'`, `'ST_MultiLineString'`, `'ST_MultiPolygon'`, and `'ST_GeometryCollection'`.

Usage

```
sd_geometry_type(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(string)

See Also

SedonaDB SQL documentation for ST_GEOMETRYTYPE

---

sd_has_m *Returns true if the geometry has a M dimension.*

Description

Returns true if the geometry has a M dimension.

Usage

```
sd_has_m(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_HASM

---

sd_has_z *Returns true if the geometry has a Z dimension.*

Description

Returns true if the geometry has a Z dimension.

Usage

```
sd_has_z(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_HASZ

---

sd_interior_ring_n *Returns the Nth interior ring of a polygon.*

Description

Returns the Nth interior `LINESTRING` ring of a `POLYGON`. Returns `NULL` if the geometry is not a polygon or `n` is out of range. `n` is 1-based in SedonaDB.

Usage

```
sd_interior_ring_n(geom = sd_missing_arg(), n = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
n	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_INTERIORRINGN

---

sd_intersection *Computes the intersection of two geometries or geographies.*

Description

Computes the intersection of two geometries or geographies.

Usage

```
sd_intersection(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_INTERSECTION

---

sd_intersection_agg *Returns the cumulative intersection of all geometries in the input.*

Description

Returns the cumulative intersection of all geometries in the input.

Usage

```
sd_intersection_agg(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_INTERSECTION_AGG

---

sd_intersects *Returns true if geomA intersects geomB.*

Description

Returns true if geomA intersects geomB.

Usage

```
sd_intersects(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_INTERSECTS

---

sd_is_closed *Returns true if the LINESTRING start and end point are the same.*

Description

Returns `true` if the LineString's start and end points are the same. For MultiLineStrings, all elements must be closed.

Usage

```
sd_is_closed(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_ISCLOSED

`sd_is_collection` *Returns true if the geometry or geography type of the input is a collection type.*

Description

Returns TRUE if the geometry type of the input is a geometry collection type. Collection types are the following: - GEOMETRYCOLLECTION - MULTIPOINT - MULTIPOLYGON - MULTILINESTRING

Usage

```
sd_is_collection(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_ISCOLLECTION

`sd_is_empty` *Returns true if the geometry or geography is empty.*

Description

Returns true if the geometry or geography is empty.

Usage

```
sd_is_empty(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_ISEMPTY

`sd_is_ring` *Returns true if a linestring is ST_IsClosed and ST_IsSimple.*

Description

Returns `true` if the linestring is both closed (`ST_IsClosed`) and simple (`ST_IsSimple`). In other words, the input forms a ring with no self-intersections.

Usage

```
sd_is_ring(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_ISRING

`sd_is_simple` *Tests if geometry's only self-intersections are at boundary points.*

Description

Returns `true` if the geometry has no anomalous geometric points such as self-intersections or self-tangency.

Usage

```
sd_is_simple(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_ISSIMPLE

`sd_is_valid` *Checks whether a geometry meets the rules of a valid spatial object according to the OGC standard.*

Description

Returns `true` if the geometry is well-formed according to OGC rules. Use `ST_IsValidReason` to get a diagnostic message for invalid geometries and `ST_MakeValid` to attempt to repair them.

Usage

```
sd_is_valid(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_ISVALID

`sd_is_valid_reason` *Returns a text explanation describing why a geometry is invalid.*

Description

Returns a text string explaining why a geometry is invalid, or `"Valid Geometry"` if the geometry is valid. Useful for debugging geometry construction.

Usage

```
sd_is_valid_reason(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(string)

See Also

SedonaDB SQL documentation for ST_ISVALIDREASON

---

sd_knn *Returns true if geomA finds k nearest neighbors from geomB.*

Description

ST_KNN is used in a join condition to find the k nearest neighbors of one geometry from another set of geometries. The actual k-nearest neighbors logic is handled by the spatial join execution engine: the function itself is a stub that simply defines the expected signature.

Usage

```
sd_knn(
  geomA = sd_missing_arg(),
  geomB = sd_missing_arg(),
  k = sd_missing_arg(),
  use_spheroid = sd_missing_arg()
)
```

Arguments

geomA	(geometry): The geometry around which to search.
geomB	(geometry): Column containing candidate geometries.
k	(integer): The number of nearest neighbours to return. Defaults to 1.
use_spheroid	(boolean): true to use spherical distance, false for Euclidean. Defaults to false.

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_KNN

sd_length *Returns the length of a geometry or geography.*

Description

Returns the length of geom. This function only supports LineString, MultiLineString, and GeometryCollections containing linear geometries. Use ST_Perimeter for polygons. For geometry, returns the length in the units of the coordinate reference system; for geography, returns the spherical approximation of the geodesic length in meters.

Usage

```
sd_length(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_LENGTH

sd_line_interpolate_point *Returns a point interpolated along a line.*

Description

Returns a point interpolated along a line. First argument must be a LINESTRING. Second argument is a double between 0 and 1 representing fraction of total linestring length the point has to be located. For geography, the interpolation is computed using spherical interpolation between vertices.

Usage

```
sd_line_interpolate_point(geom = sd_missing_arg(), fraction = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
fraction	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_LINEINTERPOLATEPOINT

`sd_line_locate_point` *Returns the location of the closest point on a LineString as a fraction of its total length.*

Description

Returns a double between 0 and 1, representing the location of the closest point on the LineString as a fraction of its total length. The first argument must be a LINESTRING, and the second argument is a POINT geometry or geography. For geography, the fraction is computed based on spherical interpolation between vertices.

Usage

```
sd_line_locate_point(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geography): Input geography
`geom_b`	(geography): Input geography

Value

(double)

See Also

SedonaDB SQL documentation for ST_LINELOCATEPOINT

`sd_line_merge` *Merges a collection of potentially connected line segments into the fewest possible LineStrings.*

Description

Merges a collection of potentially connected line segments into the fewest possible LineStrings.

Usage

```
sd_line_merge(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_LINEMERGE

`sd_line_substring` *Returns a linestring being a substring of the input one starting and ending at the given fractions of total 2d length.*

Description

Returns a linestring being a substring of the input one starting and ending at the given fractions of total 2d length.

Usage

```
sd_line_substring(
  geom = sd_missing_arg(),
  start_fraction = sd_missing_arg(),
  end_fraction = sd_missing_arg()
)
```

Arguments

`geom`	(geometry): Input geometry
`start_fraction`	(double): The fraction from which the returned line will start (between 0 and 1)
`end_fraction`	(double): The fraction from which the returned line will end (between 0 and 1)

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_LINESUBSTRING

`sd_m` *Returns the M (measure) coordinate of a Point geometry.*

Description

Extracts the M (measure) coordinate from a Point geometry or geography. Returns NULL if the geometry has no M dimension or for non-point geometries.

Usage

```
sd_m(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_M

`sd_m_max` *Returns the maximum M (measure) value from a geometry or geography's bounding box.*

Description

Returns the maximum M (measure) value from a geometry or geography's bounding box.

Usage

```
sd_m_max(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_MMAX

`sd_m_min` *Returns the minimum M-coordinate (measure) of a geometry or geography's bounding box.*

Description

Returns the minimum M-coordinate (measure) of a geometry or geography's bounding box.

Usage

```
sd_m_min(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_MMIN

`sd_make_line` *Creates a LineString from two or more input geometries.*

Description

Creates a `LineString` from two or more input `Point`, `MultiPoint`, or `LineString` geometries. The function connects the input geometries in the order they are provided to form a single continuous line.

Usage

```
sd_make_line(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_MAKELINE

---

sd_make_valid *Creates a valid representation of an invalid geometry.*

Description

Collapsed geometries are either converted to empty (keepCollapsed=false) or a valid geometry of lower dimension (keepCollapsed=true). Default is keepCollapsed=false.

Usage

```
sd_make_valid(geom = sd_missing_arg(), keepCollapsed = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
keepCollapsed	(boolean): Input boolean

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_MAKEVALID

sd_max_distance *Returns the maximum distance between any pair of points in two geometries or geographies.*

Description

For geography, returns the maximum distance in meters calculated using a spherical approximation.

Usage

```
sd_max_distance(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geography): Input geography
geom_b	(geography): Input geography

Value

(double)

See Also

SedonaDB SQL documentation for ST_MAXDISTANCE

sd_minimum_clearance *Returns the minimum clearance of a geometry.*

Description

The minimum clearance is a metric that quantifies a geometry's tolerance to changes in coordinate precision or vertex positions. It represents the maximum distance by which vertices can be adjusted without introducing invalidity to the geometry's structure. A larger minimum clearance value indicates greater robustness against such perturbations. For a geometry with a minimum clearance of x, the following conditions hold: - No two distinct vertices are separated by a distance less than x. - No vertex lies within a distance x from any line segment it is not an endpoint of. For geometries with no definable minimum clearance, such as single Point geometries or MultiPoint geometries where all points occupy the same location, the function returns Double.MAX_VALUE.

Usage

```
sd_minimum_clearance(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_MINIMUMCLEARANCE

`sd_minimum_clearance_line` *Returns a LineString representing the minimum clearance distance of the input geometry.*

Description

This function returns a two-point LineString geometry representing the minimum clearance distance of the input geometry. If the input geometry does not have a defined minimum clearance, such as for single Points or coincident MultiPoints, an empty LineString geometry is returned instead.

Usage

```
sd_minimum_clearance_line(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_MINIMUMCLEARANCELINE

`sd_missing_arg` *Missing argument sentinel*

Description

Missing argument sentinel

Usage

```
sd_missing_arg()
```

Value

An object of class 'sd_missing_arg'

`sd_n_points` *Returns the number of points of the geometry or geography.*

Description

Returns the total number of coordinate points in the geometry, counting all vertices across all components.

Usage

```
sd_n_points(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_NPOINTS

`sd_n_rings` *Returns the total number of rings in a geometry or geography (both exterior and interior rings).*

Description

Returns the total number of rings (exterior and interior) in a polygon or multipolygon. For a simple polygon, this is 1 plus the number of holes. For a multipolygon, it is the sum of all rings in all polygons. Returns 0 for non-polygon geometry types.

Usage

```
sd_n_rings(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_NRINGS

`sd_normalize` *Returns the geometry or geography in its canonical form.*

Description

Returns the geometry or geography in its canonical form.

Usage

```
sd_normalize(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_NORMALIZE

`sd_num_geometries` *Returns the number of geometries in a geometry or geography collection.*

Description

Returns the number of Geometries. If geometry is a GEOMETRYCOLLECTION (or MULTI*) return the number of geometries, for single geometries will return 1.

Usage

```
sd_num_geometries(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_NUMGEOMETRIES

`sd_num_interior_rings` *Returns the number of interior rings (holes) in a polygon geometry or geography.*

Description

Returns the number of interior rings (holes) in a polygon. For a polygon without holes, returns 0. For multipolygons, returns the number of interior rings in the first polygon. Returns NULL for non-polygon geometry types. Alias: `ST_NumInteriorRing`.

Usage

```
sd_num_interior_rings(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_NUMINTERIORRINGS

`sd_num_points` *Returns the number of points of a linestring geometry or geography.*

Description

Returns the total number of coordinate points in a linestring geometry. Returns NULL for other geometry types. Use `ST_NPoints()` to calculate the number of vertices for all geometry types.

Usage

```
sd_num_points(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_NUMPOINTS

---

sd_overlaps *Returns true if A overlaps B.*

Description

Returns `true` if the two geometries share space and have the same dimension, but are not completely contained by each other.

Usage

```
sd_overlaps(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_OVERLAPS

---

sd_perimeter *Calculates the perimeter of a given geometry or geography.*

Description

This function calculates the perimeter of a given geometry or geography. It supports Polygon, MultiPolygon, and GeometryCollection geometries (as long as the GeometryCollection contains polygonal geometries). For other types, it returns 0. To measure lines, use ST_Length. For geography, returns the spherical approximation of the geodesic perimeter in meters.

Usage

```
sd_perimeter(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_PERIMETER

---

sd_point *Constructs a Point Geometry from X and Y.*

Description

Constructs a Point geometry from X and Y coordinates. An optional SRID or CRS can be provided as a third argument.

Usage

```
sd_point(x = sd_missing_arg(), y = sd_missing_arg(), srid = sd_missing_arg())
```

Arguments

x	(double): Input double
y	(double): Input double
srid	(crs): Input crs

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POINT

`sd_point_m` *Constructs a Point with an M (measure) coordinate from X, Y, and M values.*

Description

Constructs a Point with an M (measure) coordinate from X, Y, and M values.

Usage

```
sd_point_m(x = sd_missing_arg(), y = sd_missing_arg(), m = sd_missing_arg())
```

Arguments

`x`	(double): Input double
`y`	(double): Input double
`m`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POINTM

`sd_point_n` *Returns the Nth point in a linestring.*

Description

Return the Nth point in a single linestring. Negative values are counted backwards from the end of the LineString such that -1 is the last point. Returns NULL if there is no linestring in the geometry.

Usage

```
sd_point_n(geom = sd_missing_arg(), n = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`n`	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POINTN

`sd_point_on_surface` *Returns a point guaranteed to lie on the surface of a geometry or geography.*

Description

Unlike ST_Centroid which does not necessarily intersect a geometry or geography, ST_PointOnSurface makes an attempt to find an interior point close to the middle (using deterministic heuristics).

Usage

```
sd_point_on_surface(geom = sd_missing_arg())
```

Arguments

`geom`	(geography): Input geography

Value

(geography)

See Also

SedonaDB SQL documentation for ST_POINTONSURFACE

`sd_point_z` *Constructs a Point with a Z coordinate from X, Y, and Z values.*

Description

Constructs a Point with a Z coordinate from X, Y, and Z values.

Usage

```
sd_point_z(x = sd_missing_arg(), y = sd_missing_arg(), z = sd_missing_arg())
```

Arguments

`x`	(double): Input double
`y`	(double): Input double
`z`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POINTZ

`sd_point_zm` *Constructs a Point with X, Y, Z and M coordinates.*

Description

Constructs a Point with X, Y, Z and M coordinates.

Usage

```
sd_point_zm(
  x = sd_missing_arg(),
  y = sd_missing_arg(),
  z = sd_missing_arg(),
  m = sd_missing_arg()
)
```

Arguments

`x`	(double): Input double
`y`	(double): Input double
`z`	(double): Input double
`m`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POINTZM

`sd_points` *Returns a MultiPoint geometry consisting of all the coordinates of the input geometry or geography.*

Description

Returns a MultiPoint geometry consisting of all the coordinates of the input geometry. It preserves duplicate points as well as M and Z coordinates.

Usage

```
sd_points(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POINTS

`sd_polygonize` *Builds a polygonal geometry from linear components in the input geometry.*

Description

Builds a polygonal geometry from linear components in the input geometry. Handles GeometryCollection and multi-geometries by extracting linear components. Typically used with collections of complete, simple linework that forms polygon boundaries. Unlike many geometry constructors, this doesn't require explicit closure of rings—any closed rings in the input will become polygon boundaries.

Usage

```
sd_polygonize(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POLYGONIZE

`sd_polygonize_agg` *Creates polygons from a set of geometries that contain linework representing the edges of a polygon.*

Description

Creates polygons from a set of geometries that contain linework representing the edges of a polygon.

Usage

```
sd_polygonize_agg(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_POLYGONIZE_AGG

`sd_reduce_precision` *Reduces the coordinate precision of a geometry or geography to the specified grid size.*

Description

Rebuilds a new geometry after specifying all vertices to the specified size. Z and M values are preserved but not affected by the grid snapping. The implementation is similar to a unary union and is designed to output valid geometry (e.g., by avoiding creating new crossing edges when snapping vertices).

Usage

```
sd_reduce_precision(geom = sd_missing_arg(), grid_size = sd_missing_arg())
```

Arguments

`geom`	(geography): Input geography
`grid_size`	(double): Input double

Value

(geography)

See Also

SedonaDB SQL documentation for ST_REDUCEPRECISION

`sd_relate` *Returns the DE-9IM intersection matrix string for two geometries, or tests whether two geometries satisfy a given intersection matrix pattern.*

Description

When called with two geometry arguments, returns the DE-9IM (Dimensionally Extended 9-Intersection Model) intersection matrix as a 9-character string describing the spatial relationship between two geometries. When called with a third string argument, returns `true` if the two geometries satisfy the given DE-9IM intersection matrix pattern, `false` otherwise. The pattern can use wildcards (`*`) and `T` (any non-empty intersection) in addition to exact dimension values (0, 1, 2, F).

Usage

```
sd_relate(
  geom_a = sd_missing_arg(),
  geom_b = sd_missing_arg(),
  intersectionMatrixPattern = sd_missing_arg()
)
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry
`intersectionMatrixPattern`	(string): A 9-character DE-9IM pattern string. Each character can be 0-2, T, F, or * (wildcard).

Value

(string)

See Also

SedonaDB SQL documentation for ST_RELATE

sd_reverse — *Returns the geometry or geography with vertex order reversed.*

Description

Returns the geometry or geography with vertex order reversed.

Usage

```
sd_reverse(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_REVERSE

sd_rotate — *Rotates a geometry counter-clockwise around the Z axis by an angle in radians.*

Description

Rotates a geometry counter-clockwise around the Z axis by an angle in radians.

Usage

```
sd_rotate(geom = sd_missing_arg(), rot = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
rot	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_ROTATE

sd_rotate_x *Rotates a geometry around the X axis by an angle in radians.*

Description

Rotates a geometry around the X axis by an angle in radians.

Usage

```
sd_rotate_x(geom = sd_missing_arg(), rot = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
rot	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_ROTATEX

sd_rotate_y *Rotates a geometry around the Y axis by an angle in radians.*

Description

Rotates a geometry around the Y axis by an angle in radians.

Usage

```
sd_rotate_y(geom = sd_missing_arg(), rot = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
rot	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_ROTATEY

sd_scale — *Scales a geometry by multiplying ordinates with scale factors.*

Description

Scales a geometry by multiplying each coordinate by the corresponding scale factor. A 3D variant accepts `scaleZ`.

Usage

```
sd_scale(
  geom = sd_missing_arg(),
  scaleX = sd_missing_arg(),
  scaleY = sd_missing_arg(),
  scaleZ = sd_missing_arg()
)
```

Arguments

`geom`	(geometry): Input geometry
`scaleX`	(double): Input double
`scaleY`	(double): Input double
`scaleZ`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SCALE

sd_segmentize — *Densifies a geometry by adding intermediate points along segments that exceed a maximum length.*

Description

Densifies a geometry or geography by adding intermediate points along line segments that exceed the specified maximum segment length. For geometry, uses planar distance; for geography, uses the spherical approximation of the geodesic distance in meters and points are added along the great circle arc. In both cases, points are added so that each resulting segment is no longer than `max_segment_length`.

Usage

```
sd_segmentize(geom = sd_missing_arg(), max_segment_length = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
max_segment_length	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SEGMENTIZE

---

sd_set_crs *Sets the Coordinate Reference System (CRS) for a geometry.*

Description

Sets a CRS to a geometry/geography. This is metadata-only - no coordinates are transformed. This is different from ST_Transform which actually transforms coordinates. `target_crs` accepts an authority code (e.g. `'EPSG:4326'`), PROJJSON, or WKT. It is stored as given and read back unchanged by `ST_CRS`.

Usage

```
sd_set_crs(geom = sd_missing_arg(), target_crs = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
target_crs	(string): Input string

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SETCRS

sd_set_srid — *Sets the SRID (spatial reference identifier) for a geometry.*

Description

Sets the SRID (spatial reference system identifier) of the geometry. This is metadata only - no coordinate transformation occurs. Use ST_Transform to actually reproject coordinates.

Usage

```
sd_set_srid(geom = sd_missing_arg(), srid = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
srid	(integer): Input integer

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SETSRID

sd_simplify — *Simplifies an input geometry or geography using the Douglas-Peucker algorithm.*

Description

Simplifies a geometry or geography using the Douglas-Peucker algorithm. The `tolerance` parameter controls the degree of simplification: higher values produce simpler geometries. For geography, the tolerance is specified in meters and a different algorithm is used. This function may produce invalid geometries; use `ST_SimplifyPreserveTopology` when validity must be maintained.

Usage

```
sd_simplify(geom = sd_missing_arg(), tolerance = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry
tolerance	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SIMPLIFY

`sd_simplify_preserve_topology` *Simplifies a geometry, ensuring the result is a valid geometry with the same topology.*

Description

Simplifies a geometry, ensuring that the result is a valid geometry having the same dimension and number of components as the input, and with the components having the same topological relationship.

Usage

```
sd_simplify_preserve_topology(
  geom = sd_missing_arg(),
  tolerance = sd_missing_arg()
)
```

Arguments

`geom`	(geometry): Input geometry
`tolerance`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SIMPLIFYPRESERVETOPOLOGY

sd_snap *Snaps input geometry to reference geometry within tolerance.*

Description

Snaps input geometry to reference geometry within tolerance. Result geometry aligns to the reference geometry where within tolerance distance.

Usage

```
sd_snap(
  geom_a = sd_missing_arg(),
  geom_b = sd_missing_arg(),
  tolerance = sd_missing_arg()
)
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry
tolerance	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SNAP

sd_srid *Returns the SRID (spatial reference identifier) of a geometry.*

Description

Returns the SRID (spatial reference identifier) of a geometry.

Usage

```
sd_srid(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_SRID

---

sd_start_point *Returns the start point of a linestring geometry.*

Description

Returns the start point of a linestring geometry.

Usage

```
sd_start_point(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_STARTPOINT

---

sd_sym_difference *Returns the parts of geometries or geographies A and B that do not overlap.*

Description

Returns the parts of geometries or geographies A and B that do not overlap.

Usage

```
sd_sym_difference(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_SYMDIFFERENCE

`sd_tessellate_geog` *Converts a geometry to geography, densifying edges to approximate planar lines as geodesic segments.*

Description

Converts a geometry to a geography while tessellating edges using spherical (geodesic) interpolation. The tolerance parameter specifies the maximum deviation in meters between the original edge and the tessellated approximation. This is useful when converting planar geometries to geographies where edges need to follow geodesic paths on the sphere.

Usage

```
sd_tessellate_geog(geom = sd_missing_arg(), tolerance = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry
`tolerance`	(double): Input double

Value

(geography)

See Also

SedonaDB SQL documentation for ST_TESSELLATEGEOG

sd_tessellate_geom

Converts a geography to geometry, densifying edges to approximate geodesic segments as planar lines.

Description

Converts a geography to a geometry while tessellating edges using spherical (geodesic) interpolation. The tolerance parameter specifies the maximum deviation in meters between the original spherical edge and the tessellated planar approximation. This is useful when converting geographies to geometries while preserving the shape of geodesic edges as approximated line segments. When an edge crosses the antimeridian, the returned coordinates are "wrapped" such that the returned longitude is either greater than 180 or less than -180. This helps produce valid geometry in more cases (e.g., linestring or polygon crossing the antimeridian); however, it does not produce valid geometry in the case of a polygon ring that spans the full longitude range (e.g., Antarctica).

Usage

```
sd_tessellate_geom(geom = sd_missing_arg(), tolerance = sd_missing_arg())
```

Arguments

geom	(geography): Input geography
tolerance	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_TESSELLATEGEOM

sd_to_geography

Converts a geometry to a geography by changing the edge interpretation to spherical.

Description

Converts a geometry to a geography by changing the edge interpretation from planar to spherical. This is a metadata-only operation that does not modify the underlying coordinate data. If the input is already a geography, it is returned unchanged.

Usage

```
sd_to_geography(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geography)

See Also

SedonaDB SQL documentation for ST_TOGEOGRAPHY

---

sd_to_geometry *Converts a geography to a geometry by changing the edge interpretation to planar.*

Description

Converts a geography to a geometry by changing the edge interpretation from spherical to planar. This is a metadata-only operation that does not modify the underlying coordinate data. If the input is already a geometry, it is returned unchanged.

Usage

```
sd_to_geometry(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_TOGEOMETRY

sd_touches — *Returns true if A touches B.*

Description

Returns `true` if the two geometries have at least one boundary point in common but no interior points in common.

Usage

```
sd_touches(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_TOUCHES

sd_transform — *Transforms a geometry from one coordinate reference system to another.*

Description

Transforms the coordinate reference system of a geometry between EPSG reference systems. If provided with 2 arguments, target_crs can be specified as an EPSG code (e.g., 'EPSG:4326') or as a PROJ string. When 3 arguments are provided, a source_crs can be used to specify the source CRS in case the input geometry doesn't have an SRID defined. The source_crs takes precedence over the geometry's SRID.

Usage

```
sd_transform(...)
```

Arguments

`...`	Supported combinations: • geom (geometry), target_crs (string) • geom (geometry), source_crs (string), target_crs (string)

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_TRANSFORM

`sd_translate` *Returns a geometry with coordinates translated by deltaX and deltaY.*

Description

Translates (shifts) a geometry by the given deltaX and deltaY offsets. A 3D variant accepts `deltaZ`.

Usage

```
sd_translate(
  geom = sd_missing_arg(),
  deltaX = sd_missing_arg(),
  deltaY = sd_missing_arg(),
  deltaZ = sd_missing_arg()
)
```

Arguments

`geom`	(geometry): Input geometry
`deltaX`	(double): Input double
`deltaY`	(double): Input double
`deltaZ`	(double): Input double

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_TRANSLATE

sd_unary_union *Returns a single geometry which is the union of all components.*

Description

Returns a single geometry which is the union of all components of the input geometry. Useful for computing the union of a set of geometries stored in a single geometry (e.g., a GEOMETRYCOLLECTION or MULTI* geometry).

Usage

```
sd_unary_union(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_UNARYUNION

sd_union *Returns a geometry or geography that represents the point set union of two geometries or geographies.*

Description

Returns a geometry or geography that represents the point set union of two geometries or geographies.

Usage

```
sd_union(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

geom_a	(geometry): Input geometry
geom_b	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_UNION

`sd_union_agg` *Returns a geometry that represents the point set union of all geometries.*

Description

Returns a geometry that represents the point set union of all geometries.

Usage

```
sd_union_agg(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(geometry)

See Also

SedonaDB SQL documentation for ST_UNION_AGG

`sd_within` *Returns true if A is completely inside B.*

Description

Returns `true` if geometry A is completely inside geometry B. This is the inverse of `ST_Contains`: `ST_Within(A, B)` is equivalent to `ST_Contains(B, A)`.

Usage

```
sd_within(geom_a = sd_missing_arg(), geom_b = sd_missing_arg())
```

Arguments

`geom_a`	(geometry): Input geometry
`geom_b`	(geometry): Input geometry

Value

(boolean)

See Also

SedonaDB SQL documentation for ST_WITHIN

---

sd_x

Returns the X coordinate (longitude for geography) of the point, or NULL if not available.

Description

Returns NULL for non-point geometries or NULL input.

Usage

```
sd_x(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_X

---

sd_x_max

Returns the maximum X coordinate (longitude for geography) of a geometry or geography's bounding box.

Description

Returns the maximum X coordinate (longitude for geography) of a geometry or geography's bounding box.

Usage

```
sd_x_max(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_XMAX

`sd_x_min` *Returns the minimum X coordinate (longitude for geography) of a geometry or geography's bounding box.*

Description

Returns the minimum X coordinate (longitude for geography) of a geometry or geography's bounding box.

Usage

```
sd_x_min(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_XMIN

`sd_y` *Returns the Y coordinate (latitude for geography) of the point, or NULL if not available.*

Description

Returns NULL for non-point geometries or NULL input.

Usage

```
sd_y(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_Y

`sd_y_max` *Returns the maximum Y coordinate (latitude for geography) of a geometry or geography's bounding box.*

Description

For geography, the maximum latitude is computed considering spherical geometry (e.g., the maximum latitude of a linestring that crosses the North Pole is 90).

Usage

```
sd_y_max(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_YMAX

`sd_y_min` *Returns the minimum Y coordinate (latitude for geography) of a geometry or geography's bounding box.*

Description

For geography, the minimum latitude is computed considering spherical geometry (e.g., the minimum latitude of a linestring that crosses the South Pole is -90).

Usage

```
sd_y_min(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_YMIN

---

sd_z *Returns the Z coordinate of the point, or NULL if not available.*

Description

Returns NULL if the geometry has no Z dimension or for non-point geometries.

Usage

```
sd_z(geom = sd_missing_arg())
```

Arguments

geom	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_Z

`sd_z_max` *Returns the maximum Z coordinate of a geometry or geography's bounding box.*

Description

Returns the maximum Z coordinate of a geometry or geography's bounding box.

Usage

```
sd_z_max(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_ZMAX

`sd_z_min` *Returns the minimum Z coordinate of a geometry or geography's bounding box.*

Description

Returns the minimum Z coordinate of a geometry or geography's bounding box.

Usage

```
sd_z_min(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(double)

See Also

SedonaDB SQL documentation for ST_ZMIN

`sd_zmflag` *Returns a code indicating the dimension of the coordinates in a geometry.*

Description

Returns a code indicating the coordinate dimension: - 0 = 2D (XY) - 1 = 3D with M coordinate (XYM) - 2 = 3D with Z coordinate (XYZ) - 3 = 4D (XYZM)

Usage

```
sd_zmflag(geom = sd_missing_arg())
```

Arguments

`geom`	(geometry): Input geometry

Value

(integer)

See Also

SedonaDB SQL documentation for ST_ZMFLAG

Index