PostGIS 3.3 Cheatsheet

New in this release [1] Enhanced in this release [2] aggregate [agg] window function [W] Requires GEOS 3.9 or higher[g3.9] 2.5/3D support[3d] SQL-MM[mm] Supports geography [G]

PostGIS Geometry/Geography/Box Data Types

box2d The type representing a 2-dimensional bounding box.

box3d The type representing a 3-dimensional bounding box.

geometry The type representing spatial features with planar coordinate systems.

geometry_dump A composite type used to describe the parts of complex geometry.

geography The type representing spatial features with geodetic (ellipsoidal) coordinate systems.

Table Management Functions

AddGeometryColumn[3d] Adds a geometry column to an existing table.

```
1. table_name, column_name, srid, type, dimension, use_typmod=true
2. schema_name, table_name, column_name, srid, type, dimension, use_typmod=true
3. catalog_name, schema_name, table_name, column_name, srid, type, dimension, use_typmod=true
```

DropGeometryColumn[3d] Removes a geometry column from a spatial table.

```
1. table_name, column_name
2. schema_name, table_name, column_name
3. catalog_name, schema_name, table_name, column_name
```

DropGeometryTable Drops a table and all its references in geometry_columns.

```
1. table_name
2. schema_name, table_name
3. catalog_name, schema_name, table_name
```

Find_SRID (a_schema_name, a_table_name, a_geomfield_name) Returns the SRID defined for a geometry column.

Populate_Geometry_Columns Ensures geometry columns are defined with type modifiers or have appropriate spatial constraints.

```
1. use_typmod=true
2. relation_oid, use_typmod=true
```

UpdateGeometrySRID[3d] Updates the SRID of all features in a geometry column, and the table metadata.

```
1. table_name, column_name, srid
2. schema_name, table_name, column_name, srid
3. catalog_name, schema_name, table_name, column_name, srid
```

Geometry Constructors

ST_Collect[3d] Creates a GeometryCollection or Multi* geometry from a set of geometries.

1. `g1, g2`
2. `g1_array`
3. `g1field` [agg]

ST_LineFromMultiPoint[3d] (aMultiPoint) Creates a LineString from a MultiPoint geometry.

ST_MakeEnvelope (xmin, ymin, xmax, ymax, srid=unknown) Creates a rectangular Polygon from minimum and maximum coordinates.

ST_MakeLine[3d] Creates a LineString from Point, MultiPoint, or LineString geometries.

1. `geom1, geom2`
2. `geoms_array`
3. `geoms` [agg]

ST_MakePoint[3d] Creates a 2D, 3DZ or 4D Point.

1. `x, y`
2. `x, y, z`
3. `x, y, z, m`

ST_MakePointM (x, y, m) Creates a Point from X, Y and M values.

ST_MakePolygon[3d] Creates a Polygon from a shell and optional list of holes.

1. `linestring`
2. `outerlinestring, interiorlinestrings`

ST_Point[mm] Creates a Point with X, Y and SRID values.

1. `x, y`
2. `x, y, srid=unknown`

ST_PointZ (x, y, z, srid=unknown) Creates a Point with X, Y, Z and SRID values.

ST_PointM (x, y, m, srid=unknown) Creates a Point with X, Y, M and SRID values.

ST_PointZM (x, y, z, m, srid=unknown) Creates a Point with X, Y, Z, M and SRID values.

ST_Polygon[mm 3d] (lineString, srid) Creates a Polygon from a LineString with a specified SRID.

ST_TileEnvelope (tileZoom, tileX, tileY, bounds=SRID=3857;LINESTRING(-20037508.342789 -20037508.342789,20037508.342789 20037508.342789), margin=0.0) Creates a rectangular Polygon in Web Mercator (SRID:3857) using the XYZ tile system.

ST_HexagonGrid (size, bounds) Returns a set of hexagons and cell indices that completely cover the bounds of the geometry argument.

ST_Hexagon (size, cell_i, cell_j, origin) Returns a single hexagon, using the provided edge size and cell coordinate within the hexagon grid space.

ST_SquareGrid (size, bounds) Returns a set of grid squares and cell indices that completely cover the bounds of the geometry argument.

ST_Square (size, cell_i, cell_j, origin) Returns a single square, using the provided edge size and cell coordinate within the square grid space.

ST_Letters[1] (letters, font) Returns the input letters rendered as geometry with a default start position at the origin and default text height of 100.

Geometry Accessors

GeometryType[3d] (geomA) Returns the type of a geometry as text.

ST_Boundary[mm 3d] (geomA) Returns the boundary of a geometry.

ST_BoundingDiagonal[3d] (geom, fits=false) Returns the diagonal of a geometry's bounding box.

ST_CoordDim[mm 3d] (geomA) Return the coordinate dimension of a geometry.

ST_Dimension[mm] (g) Returns the topological dimension of a geometry.

ST_Dump[3d] (g1) Returns a set of geometry_dump rows for the components of a geometry.

ST_DumpPoints[3d] (geom) Returns a set of geometry_dump rows for the coordinates in a geometry.

ST_DumpSegments[3d] (geom) Returns a set of geometry_dump rows for the segments in a geometry.

ST_DumpRings[3d] (a_polygon) Returns a set of geometry_dump rows for the exterior and interior rings of a Polygon.

ST_EndPoint[mm 3d] (g) Returns the last point of a LineString or CircularLineString.

ST_Envelope[mm] (g1) Returns a geometry representing the bounding box of a geometry.

ST_ExteriorRing[mm 3d] (a_polygon) Returns a LineString representing the exterior ring of a Polygon.

ST_GeometryN[mm 3d] (geomA, n) Return an element of a geometry collection.

ST_GeometryType[mm 3d] (g1) Returns the SQL-MM type of a geometry as text.

ST_HasArc[3d] (geomA) Tests if a geometry contains a circular arc

ST_InteriorRingN[mm 3d] (a_polygon, n) Returns the Nth interior ring (hole) of a Polygon.

ST_IsClosed[mm 3d] (g) Tests if a LineStrings's start and end points are coincident. For a PolyhedralSurface tests if it is closed (volumetric).

ST_IsCollection[3d] (g) Tests if a geometry is a geometry collection type.

ST_IsEmpty[mm] (geomA) Tests if a geometry is empty.

ST_IsPolygonCCW [3d] (geom) Tests if Polygons have exterior rings oriented counter-clockwise and interior rings oriented clockwise.

ST_IsPolygonCW [3d] (geom) Tests if Polygons have exterior rings oriented clockwise and interior rings oriented counter-clockwise.

ST_IsRing[mm] (g) Tests if a LineString is closed and simple.

ST_IsSimple[mm 3d] (geomA) Tests if a geometry has no points of self-intersection or self-tangency.

ST_M[mm 3d] (a_point) Returns the M coordinate of a Point.

ST_MemSize[3d] (geomA) Returns the amount of memory space a geometry takes.

ST_NDims[3d] (g1) Returns the coordinate dimension of a geometry.

ST_NPoints[3d] (g1) Returns the number of points (vertices) in a geometry.

ST_NRings[3d] (geomA) Returns the number of rings in a polygonal geometry.

ST_NumGeometries[mm 3d] (geom) Returns the number of elements in a geometry collection.

ST_NumInteriorRings[mm] (a_polygon) Returns the number of interior rings (holes) of a Polygon.

ST_NumInteriorRing (a_polygon) Returns the number of interior rings (holes) of a Polygon. Aias for ST_NumInteriorRings

ST_NumPatches[mm 3d] (g1) Return the number of faces on a Polyhedral Surface. Will return null for non-polyhedral geometries.

ST_NumPoints[mm] (g1) Returns the number of points in a LineString or CircularString.

ST_PatchN[mm 3d] (geomA, n) Returns the Nth geometry (face) of a PolyhedralSurface.

ST_PointN[mm 3d] (a_linestring, n) Returns the Nth point in the first LineString or circular LineString in a geometry.

ST_Points[3d] (geom) Returns a MultiPoint containing the coordinates of a geometry.

ST_StartPoint[mm 3d] (geomA) Returns the first point of a LineString.

ST_Summary[G] Returns a text summary of the contents of a geometry.

```
1. g
2. g
```

ST_X[mm 3d] (a_point) Returns the X coordinate of a Point.

ST_Y[mm 3d] (a_point) Returns the Y coordinate of a Point.

ST_Z[mm 3d] (a_point) Returns the Z coordinate of a Point.

ST_Zmflag[3d] (geomA) Returns a code indicating the ZM coordinate dimension of a geometry.

Geometry Editors

ST_AddPoint[3d] Add a point to a LineString.

```
1. linestring, point
2. linestring, point, position = -1
```

ST_CollectionExtract Given a geometry collection, returns a multi-geometry containing only elements of a specified type.

```
1. collection
2. collection, type
```

ST_CollectionHomogenize (collection) Returns the simplest representation of a geometry collection.

ST_CurveToLine[mm 3d] (curveGeom, tolerance, tolerance_type, flags) Converts a geometry containing curves to a linear geometry.

ST_Scroll[3d] (linestring, point) Change start point of a closed LineString.

ST_FlipCoordinates[3d] (geom) Returns a version of a geometry with X and Y axis flipped.

ST_Force2D[3d] (geomA) Force the geometries into a "2-dimensional mode".

ST_Force3D[3d] (geomA, Zvalue = 0.0) Force the geometries into XYZ mode. This is an alias for ST_Force3DZ.

ST_Force3DZ[3d] (geomA, Zvalue = 0.0) Force the geometries into XYZ mode.

ST_Force3DM (geomA, Mvalue = 0.0) Force the geometries into XYM mode.

ST_Force4D[3d] (geomA, Zvalue = 0.0, Mvalue = 0.0) Force the geometries into XYZM mode.

ST_ForcePolygonCCW [3d] (geom) Orients all exterior rings counter-clockwise and all interior rings clockwise.

ST_ForceCollection[3d] (geomA) Convert the geometry into a GEOMETRYCOLLECTION.

ST_ForcePolygonCW [3d] (geom) Orients all exterior rings clockwise and all interior rings counter-clockwise.

ST_ForceSFS[3d] Force the geometries to use SFS 1.1 geometry types only.

```
1. geomA
2. geomA, version
```

ST_ForceRHR[3d] (g) Force the orientation of the vertices in a polygon to follow the Right-Hand-Rule.

ST_ForceCurve[3d] (g) Upcast a geometry into its curved type, if applicable.

ST_LineToCurve[3d] (geomANoncircular) Converts a linear geometry to a curved geometry.

ST_Multi (geom) Return the geometry as a MULTI* geometry.

ST_Normalize (geom) Return the geometry in its canonical form.

ST_QuantizeCoordinates (g, prec_x, prec_y, prec_z, prec_m) Sets least significant bits of coordinates to zero

ST_RemovePoint[3d] (linestring, offset) Remove a point from a linestring.

ST_RemoveRepeatedPoints[3d] (geom, tolerance) Returns a version of a geometry with duplicate points removed.

ST_Reverse[3d] (g1) Return the geometry with vertex order reversed.

ST_Segmentize[G] Return a modified geometry/geography having no segment longer than the given distance.

```
1. geom, max_segment_length
2. geog, max_segment_length
```

ST_SetPoint[3d] (linestring, zerobasedposition, point) Replace point of a linestring with a given point.

ST_ShiftLongitude[3d] (geom) Shifts the longitude coordinates of a geometry between -180..180 and 0..360.

ST_WrapX[3d] (geom, wrap, move) Wrap a geometry around an X value.

ST_SnapToGrid[3d] Snap all points of the input geometry to a regular grid.

```
1. geomA, originX, originY, sizeX, sizeY
2. geomA, sizeX, sizeY
3. geomA, size
4. geomA, pointOrigin, sizeX, sizeY, sizeZ, sizeM
```

ST_Snap (input, reference, tolerance) Snap segments and vertices of input geometry to vertices of a reference geometry.

ST_SwapOrdinates[3d] (geom, ords) Returns a version of the given geometry with given ordinate values swapped.

Geometry Validation

ST_IsValid[mm] Tests if a geometry is well-formed in 2D.

```
1. g
2. g, flags
```

ST_IsValidDetail (geom, flags) Returns a valid_detail row stating if a geometry is valid or if not a reason and a location.

ST_IsValidReason Returns text stating if a geometry is valid, or a reason for invalidity.

```
1. geomA
2. geomA, flags
```

ST_MakeValid[3d] Attempts to make an invalid geometry valid without losing vertices.

```
1. input
2. input, params
```

Spatial Reference System Functions

ST_SetSRID (geom, srid) Set the SRID on a geometry.

ST_SRID[mm] (g1) Returns the spatial reference identifier for a geometry.

ST_Transform[mm] Return a new geometry with coordinates transformed to a different spatial reference system.

```
1. g1, srid
2. geom, to_proj
3. geom, from_proj, to_proj
4. geom, from_proj, to_srid
```

Geometry Input

ST_BdPolyFromText (WKT, srid) Construct a Polygon given an arbitrary collection of closed linestrings as a MultiLineString Well-Known text representation.

ST_BdMPolyFromText (WKT, srid) Construct a MultiPolygon given an arbitrary collection of closed linestrings as a MultiLineString text representation Well-Known text representation.

ST_GeogFromText[G] (EWKT) Return a specified geography value from Well-Known Text representation or extended (WKT).

ST_GeographyFromText[G] (EWKT) Return a specified geography value from Well-Known Text representation or extended (WKT).

ST_GeomCollFromText[mm] Makes a collection Geometry from collection WKT with the given SRID. If SRID is not given, it defaults to 0.

```
1. WKT, srid
2. WKT
```

ST_GeomFromEWKT[3d] (EWKT) Return a specified ST_Geometry value from Extended Well-Known Text representation (EWKT).

ST_GeomFromMARC21[1] (marcxml) Takes MARC21/XML geographic data as input and returns a PostGIS geometry object.

ST_GeometryFromText[mm] Return a specified ST_Geometry value from Well-Known Text representation (WKT). This is an alias name for ST_GeomFromText

```
1. WKT
2. WKT, srid
```

ST_GeomFromText[mm] Return a specified ST_Geometry value from Well-Known Text representation (WKT).

```
1. WKT
2. WKT, srid
```

ST_LineFromText[mm] Makes a Geometry from WKT representation with the given SRID. If SRID is not given, it defaults to 0.

```
1. WKT
2. WKT, srid
```

ST_MLineFromText[mm] Return a specified ST_MultiLineString value from WKT representation.

```
1. WKT, srid
2. WKT
```

ST_MPointFromText[mm] Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

```
1. WKT, srid
2. WKT
```

ST_MPolyFromText[mm] Makes a MultiPolygon Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

```
1. WKT, srid
2. WKT
```

ST_PointFromText[mm] Makes a point Geometry from WKT with the given SRID. If SRID is not given, it defaults to unknown.

```
1. WKT
2. WKT, srid
```

ST_PolygonFromText[mm] Makes a Geometry from WKT with the given SRID. If SRID is not given, it defaults to 0.

```
1. WKT
2. WKT, srid
```

ST_WKTToSQL[mm] (WKT) Return a specified ST_Geometry value from Well-Known Text representation (WKT). This is an alias name for ST_GeomFromText

ST_GeogFromWKB[G] (wkb) Creates a geography instance from a Well-Known Binary geometry representation (WKB) or extended Well Known Binary (EWKB).

ST_GeomFromEWKB[3d] (EWKB) Return a specified ST_Geometry value from Extended Well-Known Binary representation (EWKB).

ST_GeomFromWKB[mm] Creates a geometry instance from a Well-Known Binary geometry representation (WKB) and optional SRID.

```
1. geom
2. geom, srid
```

ST_LineFromWKB[mm] Makes a LINESTRING from WKB with the given SRID

```
1. WKB
2. WKB, srid
```

ST_LinestringFromWKB[mm] Makes a geometry from WKB with the given SRID.

```
1. WKB
2. WKB, srid
```

ST_PointFromWKB[mm 3d] Makes a geometry from WKB with the given SRID

```
1. geom
2. geom, srid
```

ST_WKBToSQL[mm] (WKB) Return a specified ST_Geometry value from Well-Known Binary representation (WKB). This is an alias name for ST_GeomFromWKB that takes no srid

ST_Box2dFromGeoHash (geohash, precision=full_precision_of_geohash) Return a BOX2D from a GeoHash string.

ST_GeomFromGeoHash (geohash, precision=full_precision_of_geohash) Return a geometry from a GeoHash string.

ST_GeomFromGML[3d] Takes as input GML representation of geometry and outputs a PostGIS geometry object

```
1. geomgml
2. geomgml, srid
```

ST_GeomFromGeoJSON[3d] Takes as input a geojson representation of a geometry and outputs a PostGIS geometry object

```
1. geomjson
2. geomjson
3. geomjson
```

ST_GeomFromKML[3d] (geomkml) Takes as input KML representation of geometry and outputs a PostGIS geometry object

ST_GeomFromTWKB (twkb) Creates a geometry instance from a TWKB ("Tiny Well-Known Binary") geometry representation.

ST_GMLToSQL[mm] Return a specified ST_Geometry value from GML representation. This is an alias name for ST_GeomFromGML

```
1. geomgml
2. geomgml, srid
```

ST_LineFromEncodedPolyline (polyline, precision=5) Creates a LineString from an Encoded Polyline.

ST_PointFromGeoHash (geohash, precision=full_precision_of_geohash) Return a point from a GeoHash string.

ST_FromFlatGeobufToTable (schemaname, tablename, FlatGeobuf input data) Creates a table based on the structure of FlatGeobuf data.

ST_FromFlatGeobuf (Table reference, FlatGeobuf input data) Reads FlatGeobuf data.

Geometry Output

ST_AsEWKT[G 3d] Return the Well-Known Text (WKT) representation of the geometry with SRID meta data.

1. g1
2. g1, maxdecimaldigits=15
3. g1
4. g1, maxdecimaldigits=15

ST_AsText[mm G] Return the Well-Known Text (WKT) representation of the geometry/geography without SRID metadata.

1. g1
2. g1, maxdecimaldigits = 15
3. g1
4. g1, maxdecimaldigits = 15

ST_AsBinary[mm G 3d] Return the OGC/ISO Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.

1. g1
2. g1, NDR_or_XDR
3. g1
4. g1, NDR_or_XDR

ST_AsEWKB[3d] Return the Extended Well-Known Binary (EWKB) representation of the geometry with SRID meta data.

1. g1
2. g1, NDR_or_XDR

ST_AsHEXEWKB[3d] Returns a Geometry in HEXEWKB format (as text) using either little-endian (NDR) or big-endian (XDR) encoding.

1. g1, NDRorXDR
2. g1

ST_AsEncodedPolyline (geom, precision=5) Returns an Encoded Polyline from a LineString geometry.

ST_AsFlatGeobuf Return a FlatGeobuf representation of a set of rows.

1. row [agg]
2. row, index
3. row, index, geom_name

ST_AsGeobuf Return a Geobuf representation of a set of rows.

1. row [agg]
2. row, geom_name

ST_AsGeoJSON[G 3d] Return a geometry as a GeoJSON element.

1. feature, geomcolumnname, maxdecimaldigits=9, pretty_bool=false
2. geom, maxdecimaldigits=9, options=8
3. geog, maxdecimaldigits=9, options=0

ST_AsGML[mm G 3d] Return the geometry as a GML version 2 or 3 element.

1. geom, maxdecimaldigits=15, options=0
2. geog, maxdecimaldigits=15, options=0, nprefix=null, id=null
3. version, geom, maxdecimaldigits=15, options=0, nprefix=null, id=null
4. version, geog, maxdecimaldigits=15, options=0, nprefix=null, id=null

ST_AsKML[G 3d] Return the geometry as a KML element.

1. geom, maxdecimaldigits=15, nprefix=NULL
2. geog, maxdecimaldigits=15, nprefix=NULL

ST_AsLatLonText (pt, format='') Return the Degrees, Minutes, Seconds representation of the given point.

ST_AsMARC21[1] (geom, format='hdddmmss') Returns geometry as a MARC21/XML record with a geographic datafield (034).

ST_AsMVTGeom (geom, bounds, extent=4096, buffer=256, clip_geom=true) Transforms a geometry into the coordinate space of a MVT tile.

ST_AsMVT Aggregate function returning a MVT representation of a set of rows.

1. row [agg]
2. row, name
3. row, name, extent
4. row, name, extent, geom_name
5. row, name, extent, geom_name, feature_id_name

ST_AsSVG[G] Returns SVG path data for a geometry.

```
1. geom, rel=0, maxdecimaldigits=15
2. geog, rel=0, maxdecimaldigits=15
```

ST_AsTWKB Returns the geometry as TWKB, aka "Tiny Well-Known Binary"

```
1. g1, decimaldigits_xy=0, decimaldigits_z=0, decimaldigits_m=0,
   include_sizes=false, include_bounding boxes=false
2. geometries, unique_ids, decimaldigits_xy=0, decimaldigits_z=0, decimaldigits_m=0,
   include_sizes=false, include_bounding_boxes=false
```

ST_AsX3D[3d] (g1, maxdecimaldigits=15, options=0) Returns a Geometry in X3D xml node element format: ISO-IEC-19776-1.2-X3DEncodings-XML

ST_GeoHash (geom, maxchars=full_precision_of_point) Return a GeoHash representation of the geometry.

Spatial Relationships

ST_3DIntersects[mm 3d] (geomA, geomB) Tests if two geometries spatially intersect in 3D - only for points, linestrings, polygons, polyhedral surface (area).

ST_Contains[mm] (geomA, geomB) Tests if no points of B lie in the exterior of A, and A and B have at least one interior point in common.

ST_ContainsProperly (geomA, geomB) Tests if B intersects the interior of A but not the boundary or exterior.

ST_CoveredBy[G] Tests if no point in A is outside B

```
1. geomA, geomB
2. geogA, geogB
```

ST_Covers[G] Tests if no point in B is outside A

```
1. geomA, geomB
2. geogpolyA, geogpointB
```

ST_Crosses[mm] (g1, g2) Tests if two geometries have some, but not all, interior points in common.

ST_Disjoint[mm] (A, B) Tests if two geometries are disjoint (they have no point in common).

ST_Equals[mm] (A, B) Tests if two geometries include the same set of points.

ST_Intersects[mm G] Tests if two geometries intersect (they have at least one point in common).

```
1. geomA, geomB
2. geogA, geogB
```

ST_LineCrossingDirection (linestringA, linestringB) Returns a number indicating the crossing behavior of two LineStrings.

ST_OrderingEquals[mm] (A, B) Tests if two geometries represent the same geometry and have points in the same directional order.

ST_Overlaps[mm] (A, B) Tests if two geometries intersect and have the same dimension, but are not completely contained by each other.

ST_Relate[mm] Tests if two geometries have a topological relationship matching an Intersection Matrix pattern, or computes their Intersection Matrix

```
1. geomA, geomB, intersectionMatrixPattern
2. geomA, geomB
3. geomA, geomB, boundaryNodeRule
```

ST_RelateMatch (intersectionMatrix, intersectionMatrixPattern) Tests if a DE-9IM Intersection Matrix matches an Intersection Matrix pattern

ST_Touches[mm] (A, B) Tests if two geometries have at least one point in common, but their interiors do not intersect.

ST_Within[mm] (A, B) Tests if no points of A lie in the exterior of B, and A and B have at least one interior point in common.

ST_3DDWithin[mm 3d] (g1, g2, distance_of_srid) Tests if two 3D geometries are within a given 3D distance

ST_3DDFullyWithin[3d] (g1, g2, distance) Tests if two 3D geometries are entirely within a given 3D distance

ST_DFullyWithin (g1, g2, distance) Tests if two geometries are entirely within a given distance

ST_DWithin[G] Tests if two geometries are within a given distance

```
1. g1, g2, distance_of_srid
2. gg1, gg2, distance_meters, use_spheroid = true
```

ST_PointInsideCircle (a_point, center_x, center_y, radius) Tests if a point geometry is inside a circle defined by a center and radius.

Operators

&&[G] Returns TRUE if A's 2D bounding box intersects B's 2D bounding box.

```
1. A, B
2. A, B
```

&&(geometry,box2df) (A, B) Returns TRUE if a geometry's (cached) 2D bounding box intersects a 2D float precision bounding box (BOX2DF).

&&(box2df,geometry) (A, B) Returns TRUE if a 2D float precision bounding box (BOX2DF) intersects a geometry's (cached) 2D bounding box.

&&(box2df,box2df) (A, B) Returns TRUE if two 2D float precision bounding boxes (BOX2DF) intersect each other.

&&&[3d] (A, B) Returns TRUE if A's n-D bounding box intersects B's n-D bounding box.

&&&(geometry,gidx)[3d] (A, B) Returns TRUE if a geometry's (cached) n-D bounding box intersects a n-D float precision bounding box (GIDX).

&&&(gidx,geometry)[3d] (A, B) Returns TRUE if a n-D float precision bounding box (GIDX) intersects a geometry's (cached) n-D bounding box.

&&&(gidx,gidx)[3d] (A, B) Returns TRUE if two n-D float precision bounding boxes (GIDX) intersect each other.

&< (A, B) Returns TRUE if A's bounding box overlaps or is to the left of B's.

&<| (A, B) Returns TRUE if A's bounding box overlaps or is below B's.

&> (A, B) Returns TRUE if A' bounding box overlaps or is to the right of B's.

<< (A, B) Returns TRUE if A's bounding box is strictly to the left of B's.

<<| (A, B) Returns TRUE if A's bounding box is strictly below B's.

=[G] Returns TRUE if the coordinates and coordinate order geometry/geography A are the same as the coordinates and coordinate order of geometry/geography B.

```
1. A, B
2. A, B
```

>> (A, B) Returns TRUE if A's bounding box is strictly to the right of B's.

@ (A, B) Returns TRUE if A's bounding box is contained by B's.

@(geometry,box2df) (A, B) Returns TRUE if a geometry's 2D bounding box is contained into a 2D float precision bounding box (BOX2DF).

@(box2df,geometry) (A, B) Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into a geometry's 2D bounding box.

@(box2df,box2df) (A, B) Returns TRUE if a 2D float precision bounding box (BOX2DF) is contained into another 2D float precision bounding box.

|&> (A, B) Returns TRUE if A's bounding box overlaps or is above B's.

|>> (A, B) Returns TRUE if A's bounding box is strictly above B's.

~ (A, B) Returns TRUE if A's bounding box contains B's.

~(geometry,box2df) (A, B) Returns TRUE if a geometry's 2D bonding box contains a 2D float precision bounding box (GIDX).

~(box2df,geometry) (A, B) Returns TRUE if a 2D float precision bounding box (BOX2DF) contains a geometry's 2D bonding box.

~(box2df,box2df) (A, B) Returns TRUE if a 2D float precision bounding box (BOX2DF) contains another 2D float precision bounding box (BOX2DF).

~= (A, B) Returns TRUE if A's bounding box is the same as B's.

<->[G] Returns the 2D distance between A and B.

```
1. A, B
2. A, B
```

|=| (A, B) Returns the distance between A and B trajectories at their closest point of approach.

<#> (A, B) Returns the 2D distance between A and B bounding boxes.

<<->> (A, B) Returns the n-D distance between the centroids of A and B bounding boxes.

<<#>> (A, B) Returns the n-D distance between A and B bounding boxes.

Measurement Functions

ST_Area[mm G] Returns the area of a polygonal geometry.

```
1. g1
2. geog, use_spheroid=true
```

ST_Azimuth[G] Returns the north-based azimuth of a line between two points.

```
1. origin, target
2. origin, target
```

ST_Angle Returns the angle between two vectors defined by 3 or 4 points, or 2 lines.

```
1. point1, point2, point3, point4
2. line1, line2
```

ST_ClosestPoint (geom1, geom2) Returns the 2D point on g1 that is closest to g2. This is the first point of the shortest line from one geometry to the other.

ST_3DClosestPoint[3d] (g1, g2) Returns the 3D point on g1 that is closest to g2. This is the first point of the 3D shortest line.

ST_Distance[mm G] Returns the distance between two geometry or geography values.

```
1. g1, g2
2. geog1, geog2, use_spheroid=true
```

ST_3DDistance[mm 3d] (g1, g2) Returns the 3D cartesian minimum distance (based on spatial ref) between two geometries in projected units.

ST_DistanceSphere (geomlonlatA, geomlonlatB, radius=6371008) Returns minimum distance in meters between two lon/lat geometries using a spherical earth model.

ST_DistanceSpheroid (geomlonlatA, geomlonlatB, measurement_spheroid=WGS84) Returns the minimum distance between two lon/lat geometries using a spheroidal earth model.

ST_FrechetDistance (g1, g2, densifyFrac = -1) Returns the Fréchet distance between two geometries.

ST_HausdorffDistance Returns the Hausdorff distance between two geometries.

```
1. g1, g2
2. g1, g2, densifyFrac
```

ST_Length[mm G] Returns the 2D length of a linear geometry.

```
1. a_2dlinestring
2. geog, use_spheroid=true
```

ST_Length2D (a_2dlinestring) Returns the 2D length of a linear geometry. Alias for ST_Length

ST_3DLength[mm 3d] (a_3dlinestring) Returns the 3D length of a linear geometry.

ST_LengthSpheroid[3d] (a_geometry, a_spheroid) Returns the 2D or 3D length/perimeter of a lon/lat geometry on a spheroid.

ST_LongestLine (g1, g2) Returns the 2D longest line between two geometries.

ST_3DLongestLine[3d] (g1, g2) Returns the 3D longest line between two geometries

ST_MaxDistance (g1, g2) Returns the 2D largest distance between two geometries in projected units.

ST_3DMaxDistance[3d] (g1, g2) Returns the 3D cartesian maximum distance (based on spatial ref) between two geometries in projected units.

ST_MinimumClearance (g) Returns the minimum clearance of a geometry, a measure of a geometry's robustness.

ST_MinimumClearanceLine (g) Returns the two-point LineString spanning a geometry's minimum clearance.

ST_Perimeter[mm G] Returns the length of the boundary of a polygonal geometry or geography.

```
1. g1
2. geog, use_spheroid=true
```

ST_Perimeter2D (geomA) Returns the 2D perimeter of a polygonal geometry. Alias for ST_Perimeter.

ST_3DPerimeter[mm 3d] (geomA) Returns the 3D perimeter of a polygonal geometry.

ST_Project[G] (g1, distance, azimuth) Returns a point projected from a start point by a distance and bearing (azimuth).

ST_ShortestLine (geom1, geom2) Returns the 2D shortest line between two geometries

ST_3DShortestLine[3d] (g1, g2) Returns the 3D shortest line between two geometries

Overlay Functions

ST_ClipByBox2D (geom, box) Computes the portion of a geometry falling within a rectangle.

ST_Difference[mm g3.9 3d] (geomA, geomB, gridSize = -1) Computes a geometry representing the part of geometry A that does not intersect geometry B.

ST_Intersection[mm G g3.9 3d] Computes a geometry representing the shared portion of geometries A and B.

1. `geomA, geomB, gridSize = -1`
2. `geogA, geogB`

ST_MemUnion[3d] (geomfield) Aggregate function which unions geometries in a memory-efficent but slower way

ST_Node[3d] (geom) Nodes a collection of lines.

ST_Split (input, blade) Returns a collection of geometries created by splitting a geometry by another geometry.

ST_Subdivide[g3.9] (geom, max_vertices=256, gridSize = -1) Computes a rectilinear subdivision of a geometry.

ST_SymDifference[mm g3.9 3d] (geomA, geomB, gridSize = -1) Computes a geometry representing the portions of geometries A and B that do not intersect.

ST_UnaryUnion[g3.9 3d] (geom, gridSize = -1) Computes the union of the components of a single geometry.

ST_Union[mm g3.9 3d] Computes a geometry representing the point-set union of the input geometries.

1. `g1, g2`
2. `g1, g2, gridSize`
3. `g1_array`
4. `g1field` [agg]
5. `g1field, gridSize` [agg]

Geometry Processing

ST_Buffer[mm G] Computes a geometry covering all points within a given distance from a geometry.

```
1. g1, radius_of_buffer, buffer_style_parameters = ''
2. g1, radius_of_buffer, num_seg_quarter_circle
3. g1, radius_of_buffer, buffer_style_parameters
4. g1, radius_of_buffer, num_seg_quarter_circle
```

ST_BuildArea (geom) Creates a polygonal geometry formed by the linework of a geometry.

ST_Centroid[mm G] Returns the geometric center of a geometry.

```
1. g1
2. g1, use_spheroid=true
```

ST_ChaikinSmoothing (geom, nIterations = 1, preserveEndPoints = false) Returns a smoothed version of a geometry, using the Chaikin algorithm

ST_ConcaveHull[2] (param_geom, param_pctconvex, param_allow_holes = false) Computes a possibly concave geometry that encloses all input geometry vertices

ST_ConvexHull[mm 3d] (geomA) Computes the convex hull of a geometry.

ST_DelaunayTriangles[3d] (g1, tolerance, flags) Returns the Delaunay triangulation of the vertices of a geometry.

ST_FilterByM (geom, min, max = null, returnM = false) Removes vertices based on their M value

ST_GeneratePoints Generates random points contained in a Polygon or MultiPolygon.

```
1. g, npoints
2. g, npoints, seed
```

ST_GeometricMedian [3d] (geom, tolerance = NULL, max_iter = 10000, fail_if_not_converged = false) Returns the geometric median of a MultiPoint.

ST_LineMerge[2] Return the lines formed by sewing together a MultiLineString.

```
1. amultilinestring
2. amultilinestring, directed
```

ST_MaximumInscribedCircle[g3.9] (geom) Computes the largest circle contained within a geometry.

ST_MinimumBoundingCircle (geomA, num_segs_per_qt_circ=48) Returns the smallest circle polygon that contains a geometry.

ST_MinimumBoundingRadius (geom) Returns the center point and radius of the smallest circle that contains a geometry.

ST_OrientedEnvelope (geom) Returns a minimum-area rectangle containing a geometry.

ST_OffsetCurve (line, signed_distance, style_parameters='') Returns an offset line at a given distance and side from an input line.

ST_PointOnSurface[mm 3d] (g1) Computes a point guaranteed to lie in a polygon, or on a geometry.

ST_Polygonize Computes a collection of polygons formed from the linework of a set of geometries.

```
1. geomfield [agg]
2. geom_array
```

ST_ReducePrecision[g3.9] (g, gridsize) Returns a valid geometry with points rounded to a grid tolerance.

ST_SharedPaths (lineal1, lineal2) Returns a collection containing paths shared by the two input linestrings/multilinestrings.

ST_Simplify Returns a simplified version of a geometry, using the Douglas-Peucker algorithm.

```
1. geomA, tolerance
2. geomA, tolerance, preserveCollapsed
```

ST_SimplifyPreserveTopology (geomA, tolerance) Returns a simplified and valid version of a geometry, using the Douglas-Peucker algorithm.

ST_SimplifyPolygonHull[1] (param_geom, vertex_fraction, is_outer = true) Computes a simplifed topology-preserving outer or inner hull of a polygonal geometry.

ST_SimplifyVW (geomA, tolerance) Returns a simplified version of a geometry, using the Visvalingam-Whyatt algorithm

ST_SetEffectiveArea (geomA, threshold = 0, set_area = 1) Sets the effective area for each vertex, using the Visvalingam-Whyatt algorithm.

ST_TriangulatePolygon[1] (geom) Computes the constrained Delaunay triangulation of polygons

ST_VoronoiLines (g1, tolerance, extend_to) Returns the boundaries of the Voronoi diagram of the vertices of a geometry.

ST_VoronoiPolygons (g1, tolerance, extend_to) Returns the cells of the Voronoi diagram of the vertices of a geometry.

Affine Transformations

ST_Affine[3d] Apply a 3D affine transformation to a geometry.

```
1. geomA, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff
2. geomA, a, b, d, e, xoff, yoff
```

ST_Rotate[3d] Rotates a geometry about an origin point.

```
1. geomA, rotRadians
2. geomA, rotRadians, x0, y0
3. geomA, rotRadians, pointOrigin
```

ST_RotateX[3d] (geomA, rotRadians) Rotates a geometry about the X axis.

ST_RotateY[3d] (geomA, rotRadians) Rotates a geometry about the Y axis.

ST_RotateZ[3d] (geomA, rotRadians) Rotates a geometry about the Z axis.

ST_Scale[3d] Scales a geometry by given factors.

```
1. geomA, XFactor, YFactor, ZFactor
2. geomA, XFactor, YFactor
3. geom, factor
4. geom, factor, origin
```

ST_Translate[3d] Translates a geometry by given offsets.

```
1. g1, deltax, deltay
2. g1, deltax, deltay, deltaz
```

ST_TransScale[3d] (geomA, deltaX, deltaY, XFactor, YFactor) Translates and scales a geometry by given offsets and factors.

Clustering Functions

ST_ClusterDBSCAN (geom, eps, minpoints) Window function that returns a cluster id for each input geometry using the DBSCAN algorithm.

ST_ClusterIntersecting (g) Aggregate function that clusters the input geometries into connected sets.

ST_ClusterKMeans (geom, number_of_clusters, max_radius) Window function that returns a cluster id for each input geometry using the K-means algorithm.

ST_ClusterWithin (g, distance) Aggregate function that clusters the input geometries by separation distance.

Bounding Box Functions

Box2D (geom) Returns a BOX2D representing the 2D extent of a geometry.

Box3D[3d] (geom) Returns a BOX3D representing the 3D extent of a geometry.

ST_EstimatedExtent Returns the estimated extent of a spatial table.

```
1. schema_name, table_name, geocolumn_name, parent_only
2. schema_name, table_name, geocolumn_name
3. table_name, geocolumn_name
```

ST_Expand Returns a bounding box expanded from another bounding box or a geometry.

```
1. geom, units_to_expand
2. geom, dx, dy, dz=0, dm=0
3. box, units_to_expand
4. box, dx, dy
5. box, units_to_expand
6. box, dx, dy, dz=0
```

ST_Extent (geomfield) Aggregate function that returns the bounding box of geometries.

ST_3DExtent[3d] (geomfield) Aggregate function that returns the 3D bounding box of geometries.

ST_MakeBox2D (pointLowLeft, pointUpRight) Creates a BOX2D defined by two 2D point geometries.

ST_3DMakeBox (point3DLowLeftBottom, point3DUpRightTop) Creates a BOX3D defined by two 3D point geometries.

ST_XMax[3d] (aGeomorBox2DorBox3D) Returns the X maxima of a 2D or 3D bounding box or a geometry.

ST_XMin[3d] (aGeomorBox2DorBox3D) Returns the X minima of a 2D or 3D bounding box or a geometry.

ST_YMax[3d] (aGeomorBox2DorBox3D) Returns the Y maxima of a 2D or 3D bounding box or a geometry.

ST_YMin[3d] (aGeomorBox2DorBox3D) Returns the Y minima of a 2D or 3D bounding box or a geometry.

ST_ZMax[3d] (aGeomorBox2DorBox3D) Returns the Z maxima of a 2D or 3D bounding box or a geometry.

ST_ZMin[3d] (aGeomorBox2DorBox3D) Returns the Z minima of a 2D or 3D bounding box or a geometry.

Linear Referencing

ST_LineInterpolatePoint[3d] (a_linestring, a_fraction) Returns a point interpolated along a line at a fractional location.

ST_3DLineInterpolatePoint[3d] (a_linestring, a_fraction) Returns a point interpolated along a 3D line at a fractional location.

ST_LineInterpolatePoints[3d] (a_linestring, a_fraction, repeat) Returns points interpolated along a line at a fractional interval.

ST_LineLocatePoint (a_linestring, a_point) Returns the fractional location of the closest point on a line to a point.

ST_LineSubstring[3d] (a_linestring, startfraction, endfraction) Returns the part of a line between two fractional locations.

ST_LocateAlong[mm] (geom_with_measure, measure, offset = 0) Returns the point(s) on a geometry that match a measure value.

ST_LocateBetween[mm] (geom, measure_start, measure_end, offset = 0) Returns the portions of a geometry that match a measure range.

ST_LocateBetweenElevations[3d] (geom, elevation_start, elevation_end) Returns the portions of a geometry that lie in an elevation (Z) range.

ST_InterpolatePoint[3d] (linear_geom_with_measure, point) Returns the interpolated measure of a geometry closest to a point.

ST_AddMeasure[3d] (geom_mline, measure_start, measure_end) Interpolates measures along a linear geometry.

Trajectory Functions

ST_IsValidTrajectory^{3d} (line) Tests if the geometry is a valid trajectory.

ST_ClosestPointOfApproach^{3d} (track1, track2) Returns a measure at the closest point of approach of two trajectories.

ST_DistanceCPA^{3d} (track1, track2) Returns the distance between the closest point of approach of two trajectories.

ST_CPAWithin^{3d} (track1, track2, dist) Tests if the closest point of approach of two trajectories is within the specified distance.

Long Transaction Support

AddAuth (auth_token) Adds an authorization token to be used in the current transaction.

CheckAuth Creates a trigger on a table to prevent/allow updates and deletes of rows based on authorization token.

```
1. a_schema_name, a_table_name, a_key_column_name
2. a_table_name, a_key_column_name
```

DisableLongTransactions () Disables long transaction support.

EnableLongTransactions () Enables long transaction support.

LockRow Sets lock/authorization for a row in a table.

```
1. a_schema_name, a_table_name, a_row_key, an_auth_token,
   expire_dt
2. a_table_name, a_row_key, an_auth_token, expire_dt
3. a_table_name, a_row_key, an_auth_token
```

UnlockRows (auth_token) Removes all locks held by an authorization token.

Version Functions

PostGIS_Extensions_Upgrade () Packages and upgrades PostGIS extensions (e.g. postgis_raster, postgis_topology, postgis_sfcgal) to latest available version.

PostGIS_Full_Version () Reports full PostGIS version and build configuration infos.

PostGIS_GEOS_Version () Returns the version number of the GEOS library.

PostGIS_Liblwgeom_Version () Returns the version number of the liblwgeom library. This should match the version of PostGIS.

PostGIS_LibXML_Version () Returns the version number of the libxml2 library.

PostGIS_Lib_Build_Date () Returns build date of the PostGIS library.

PostGIS_Lib_Version () Returns the version number of the PostGIS library.

PostGIS_PROJ_Version () Returns the version number of the PROJ4 library.

PostGIS_Wagyu_Version () Returns the version number of the internal Wagyu library.

PostGIS_Scripts_Build_Date () Returns build date of the PostGIS scripts.

PostGIS_Scripts_Installed () Returns version of the PostGIS scripts installed in this database.

PostGIS_Scripts_Released () Returns the version number of the postgis.sql script released with the installed PostGIS lib.

PostGIS_Version () Returns PostGIS version number and compile-time options.

Grand Unified Custom Variables (GUCs)

postgis.backend The backend to service a function where GEOS and SFCGAL overlap. Options: geos or sfcgal. Defaults to geos.

postgis.gdal_datapath A configuration option to assign the value of GDAL's GDAL_DATA option. If not set, the environmentally set GDAL_DATA variable is used.

postgis.gdal_enabled_drivers A configuration option to set the enabled GDAL drivers in the PostGIS environment. Affects the GDAL configuration variable GDAL_SKIP.

postgis.enable_outdb_rasters A boolean configuration option to enable access to out-db raster bands.

postgis.gdal_config_options A string configuration to set options used when working with an out-db raster.

Troubleshooting Functions	
PostGIS_AddBBox (geomA)	Add bounding box to the geometry.
PostGIS_DropBBox (geomA)	Drop the bounding box cache from the geometry.
PostGIS_HasBBox (geomA)	Returns TRUE if the bbox of this geometry is cached, FALSE otherwise.