

Page No:

**Batch:A2 Roll No.:16010421073 Experiment No.:5**

**Aim: To execute spatial queries using PostGIS.**



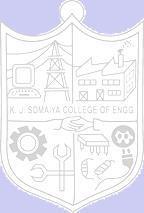
**Resources needed:** PostgreSQL 9.6, PostGIS 2.0



# Theory

**PostGIS** is an open source software program that adds support for geographic objects to the PostgreSQL object-relational database. PostGIS follows the Simple Features for SQL specification from the Open Geospatial Consortium (OGC). PostGIS turns the PostgreSQL Database Management System into a spatial database by adding support for the three features: spatial types, indexes, and functions. Because it is built on PostgreSQL, PostGIS automatically inherits important “enterprise” features as well as open standards for implementation. PostgreSQL is a powerful, object-relational database management system (ORDBMS). It is also open source software.

**Features of** PostGIS

* Geometry types for points, line strings, polygons, multi-points, multi-line-strings, multi- polygons and geometry collections.
* Spatial predicates for determining the interactions of geometries using the 3x3 Egenhofer matrix (provided by the GEOS software library).
* Spatial operators for determining geospatial measurements like area, distance, length and perimeter.
* Spatial operators for determining geospatial set operations, like union, difference, symmetric difference and buffers (provided by GEOS).
* R-tree-over-GiST (Generalised Search Tree) spatial indexes for high speed spatial querying.
* Index selectivity support, to provide high performance query plans for mixed spatial/non- spatial queries.
* For raster data

Geometry is and abstract type and concrete subtypes can be **atomic** or **collection** types

* + Atomic
    - Point : It represents a single location in coordinate space

e.g. POINT(3, 4), POINT (3,5,4,8)

* + - LineString : It is a 1-dimensional line formed by a contiguous sequence of line segments. Each line segment is defined by two points, with the end point of one segment forming the start point of the next segment

e.g. LINESTRING (1 2, 3 4, 5 6)

* + - LineRing : It is a LineString which is both closed and simple. The first and last points must be equal, and the line must not self-intersect

e.g. LINEARRING (0 0 0, 4 0 0, 4 4 0, 0 4 0, 0 0 0)

* + - Polygon : It is a 2-dimensional planar region, delimited by an exterior boundary (the shell) and zero or more interior boundaries (holes). Each boundary is a LinearRing.

e.g. POLYGON ((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1

0))

# Collection

* + - MultiPoint : It is a collection of points

e.g. MULTIPOINT ( (0 0), (1 2) )

* + - MultiLineString : It is a collection of LineStrings. A MultiLineString is closed if each of its elements is closed

e.g. MULTILINESTRING ( (0 0,1 1,1 2), (2 3,3 2,5 4) )

* + - MultiPolygon : It is a collection of non-overlapping, non-adjacent polygons. Polygons in the collection may touch only at a finite number of points.

e.g. MULTIPOLYGON (((1 5, 5 5, 5 1, 1 1, 1 5)), ((6 5, 9 1, 6 1, 6 5)))

* + - GeometryCollection : It is a is a heterogeneous (mixed) collection of geometries

e.g. GEOMETRYCOLLECTION ( POINT(2 3), LINESTRING(2 3, 3 4))

* + - Also there are PolyHedralSurface, Triangle and TIN

PostGIS provides different functions for determining relationships(topological or distance) between geometries, compute measurements, overlays and geometry construction also besides other provisions.

Few of the functions are

# Measurement functions

ST\_Area : **float ST\_Area(geometry *g1*);**

Returns the area of a polygonal geometry

ST\_Length : **float ST\_Length(geometry *a\_2dlinestring*);** R

Returns the 2D Cartesian length of the geometry if it is a LineString, MultiLineString, ST\_Curve, ST\_MultiCurve

ST\_Perimeter **: float ST\_Perimeter(geometry *g1*);**

Returns the 2D perimeter of the geometry/geography if it is a ST\_Surface, ST\_MultiSurface (Polygon, MultiPolygon)

# Named Spatial Relationships

For determining common spatial relationships, OGC SFS defines a set of named spatial relationship predicates. PostGIS provides these as the functions

ST\_Contains : **boolean ST\_Contains(geometry *geomA*, geometry *geomB*);** ST\_Crosses : **boolean ST\_Crosses(geometry *g1*, geometry *g2*);** ST\_Disjoint : **boolean ST\_Disjoint( geometry *A* , geometry *B* );** ST\_Equals : **boolean ST\_Equals(geometry *A*, geometry *B*);**

ST\_Intersects : **boolean ST\_Intersects( geometry *geomA* , geometry *geomB* );**

ST\_Overlaps : **boolean ST\_Overlaps(geometry *A*, geometry *B*);** ST\_Touches : **boolean ST\_Touches(geometry *A*, geometry *B*);** ST\_Within. : **boolean ST\_Within(geometry *A*, geometry *B*);**

It also defines the non-standard relationship predicates

ST\_Covers : **boolean ST\_Covers(geometry *geomA*, geometry *geomB*);** ST\_CoveredBy : **boolean ST\_CoveredBy(geometry *geomA*, geometry *geomB*);** ST\_ContainsProperly : **boolean ST\_ContainsProperly(geometry *geomA*, geometry *geomB*);**

Spatial predicates are usually used as conditions in SQL WHERE or JOIN clauses.

**SELECT city.name, state.name, city.geom**

**FROM city JOIN state ON ST\_Intersects(city.geom, state.geom);**



# Procedure:

1. Installation of relational database PostgreSQL 9.6 (download from <http://www.enterprisedb.com/products-services-training/pgdownload> )
2. Installation of PostGIS using Application stack builder.
3. Download spatial data from [**https://www.diva-gis.org/gdata**](https://www.diva-gis.org/gdata) (OR similar website with FREE usable data) Get it for any country with minimum 3 subjects.
4. Import the data in your PostgreSQL
5. Identify spatial relationship between any two geometric entities (any 3 named relationships)
6. Perform any two measurement functions for geometric data.
7. Execute any one range query

| 0 | | | ALQ | | | 128 |
| --- | --- | --- | --- | --- |
| 5541.57712511724 | | | ALQ | | | 129A |
| 5579.67450712005 | | | ALQ | | | 001 |
| 6083.4207708641 | | | ALQ | | | 131 |
| 7691.2205404848 | | | ALQ | | | 003 |
| 7900.75451037313 | | | ALQ | | | 122 |
| 8694.20710669982 | | | ALQ | | | 129B |
| 9564.24289057111 | | | ALQ | | | 130 |
| 12089.665931705 | | | ALQ | | | 127 |
| **18472.5531479404** | **|** | **ALQ** | **|** | **002** |

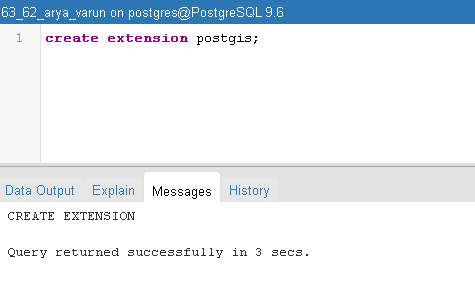
# Range query in Postgis SELECT ST\_Reclass(rast, 1,

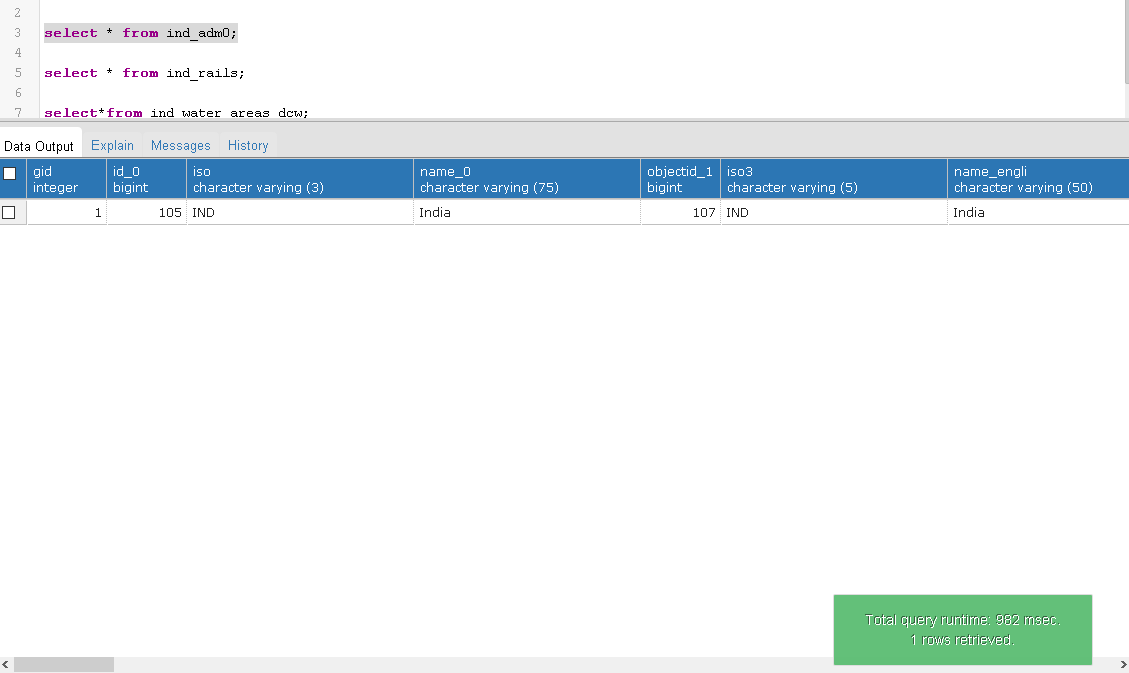
'[0-90]:0,(90-100):1,[100-1000):2',

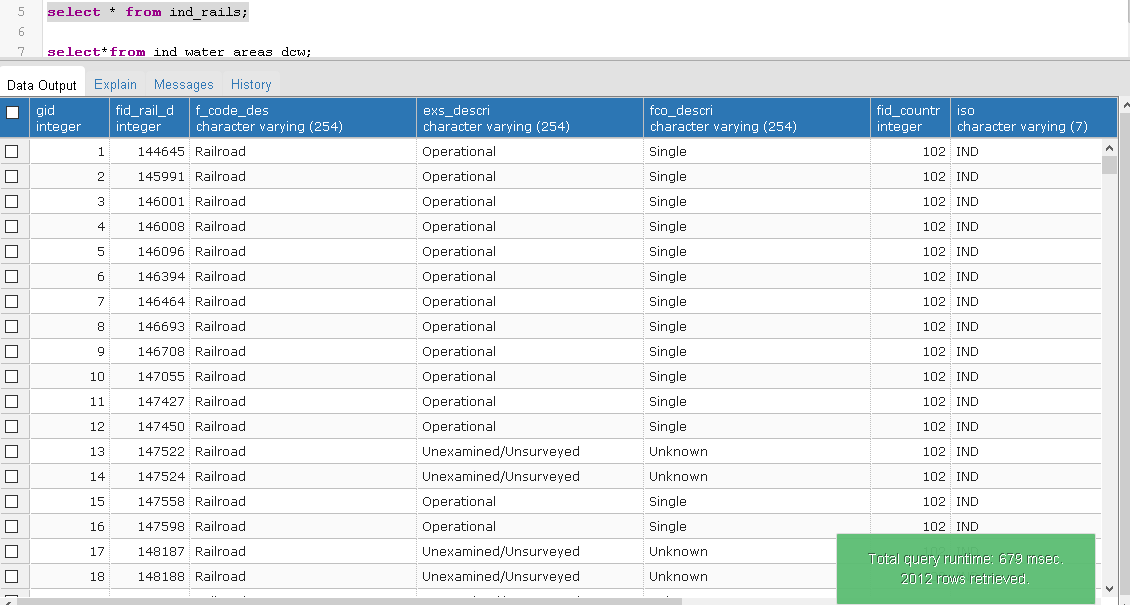
'4BUI', 0) AS rast FROM sometable WHERE filename = '123.tif';

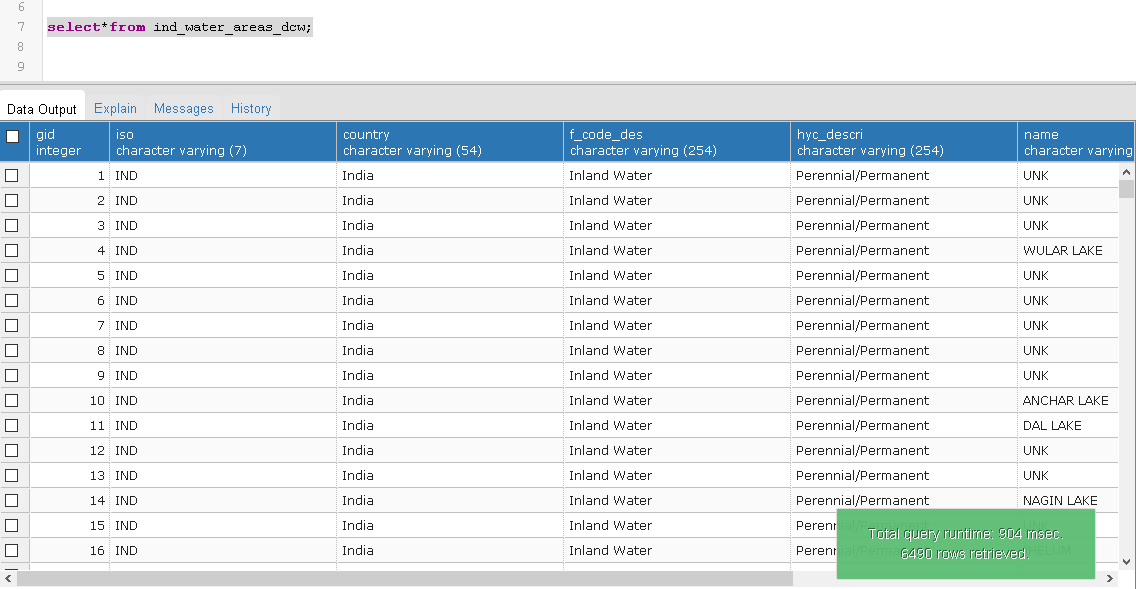


# Results: (Program printout with output)

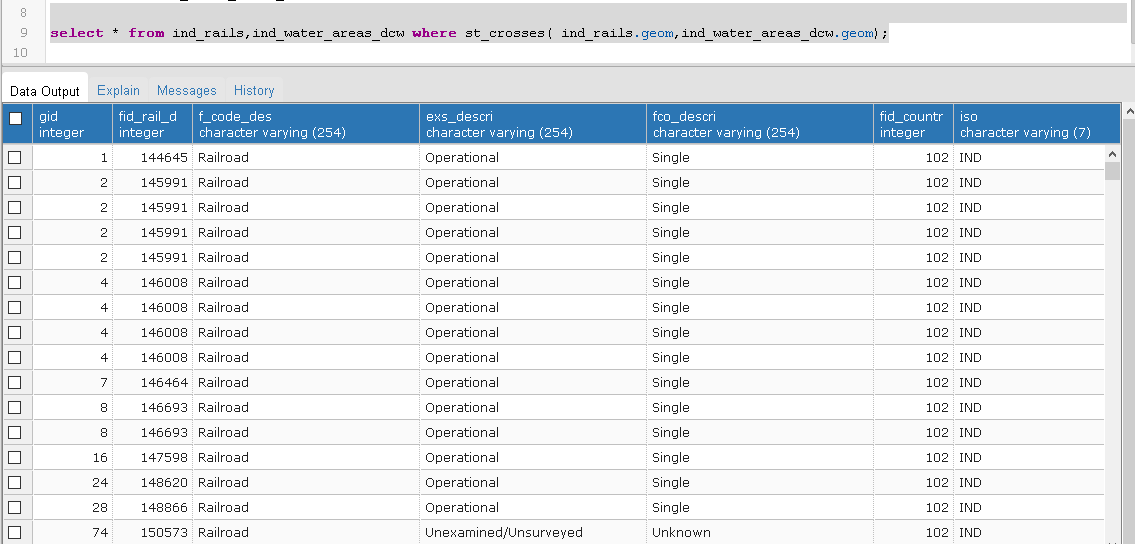




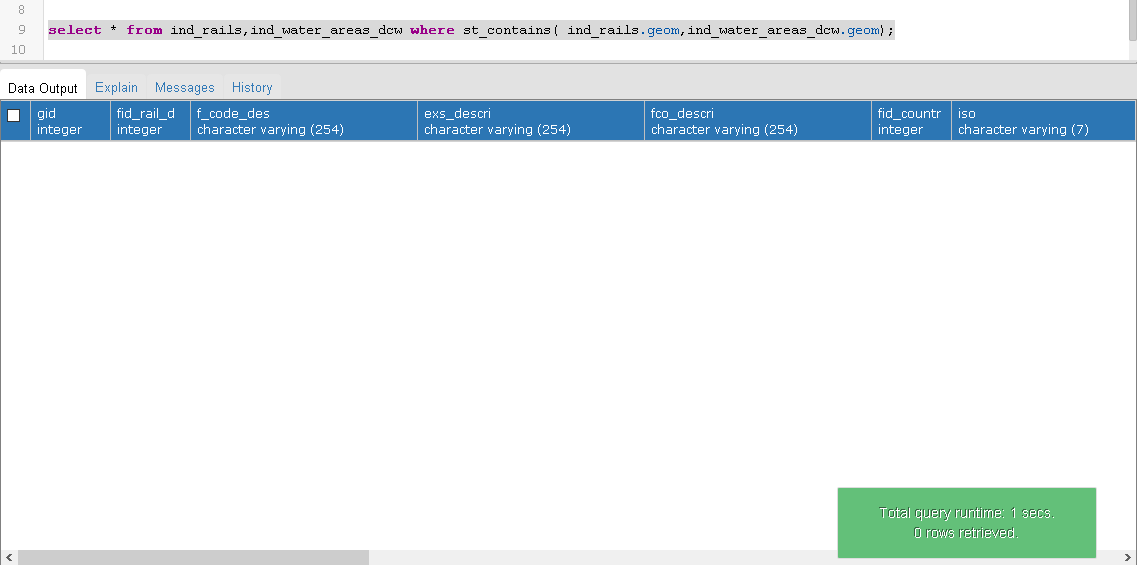


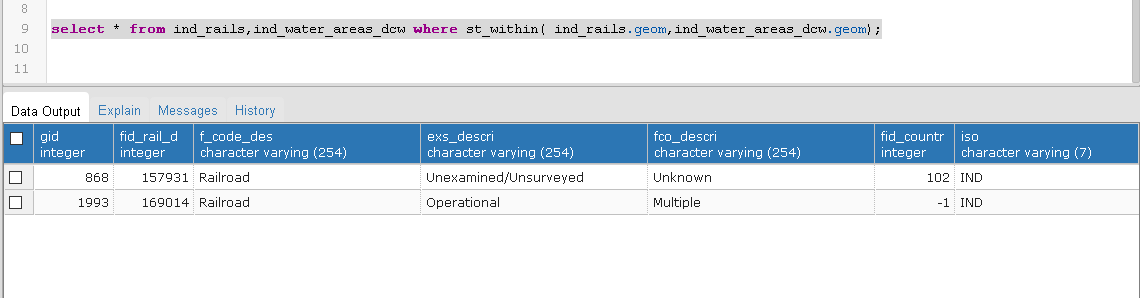


Find rail roads which crosses the water areas

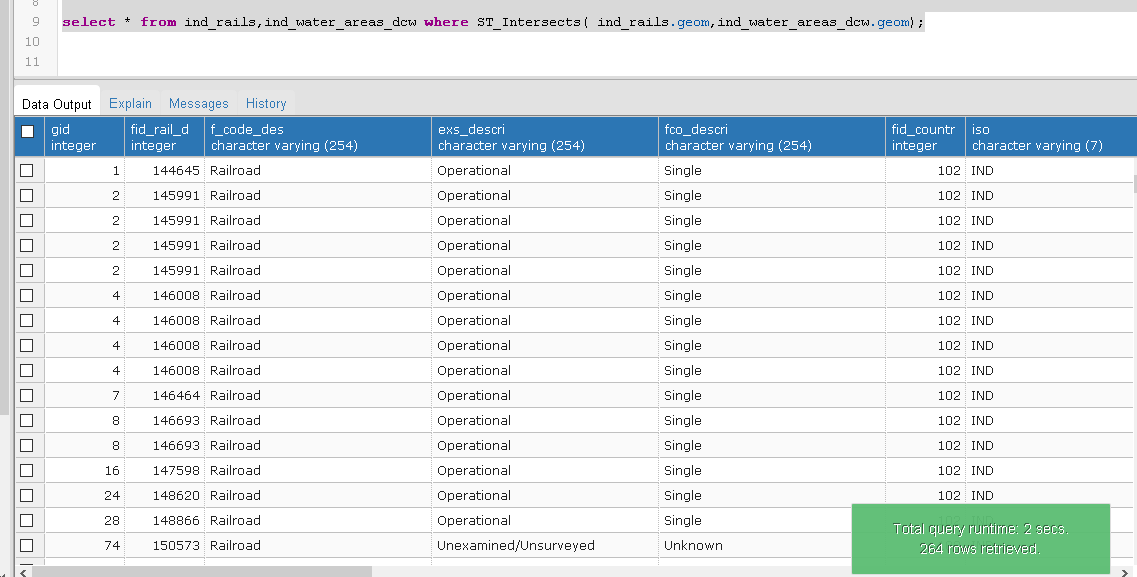


Find rail roads which are contain the water areas

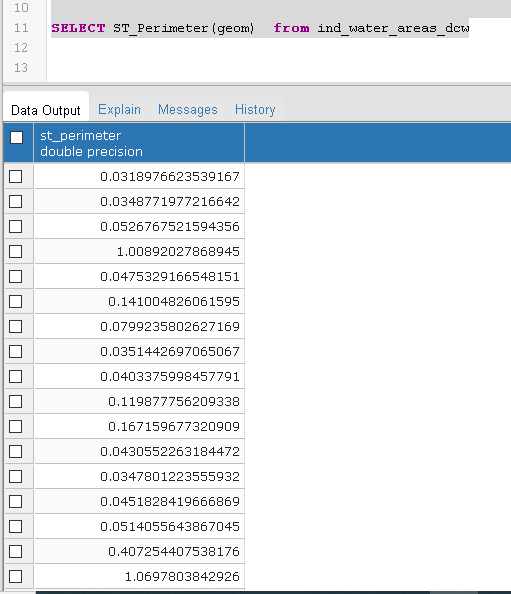


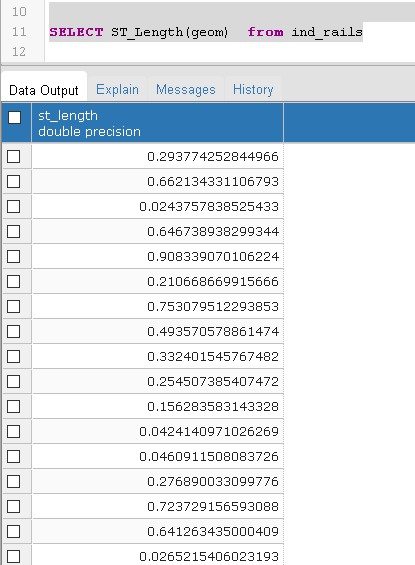
Find rail roads which are within the water areas

Write a query to find all the rails and water areas which intersect each other

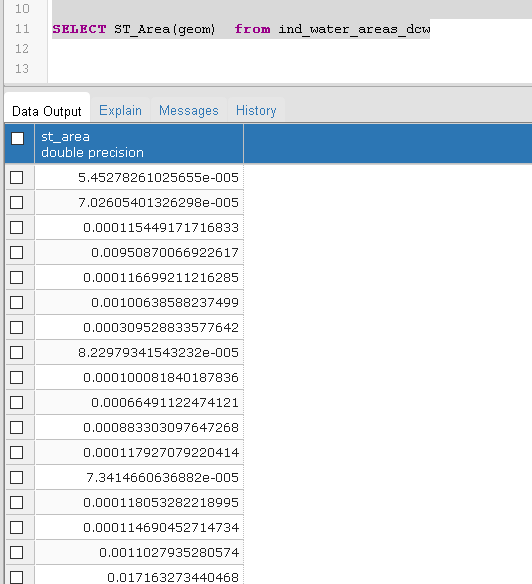


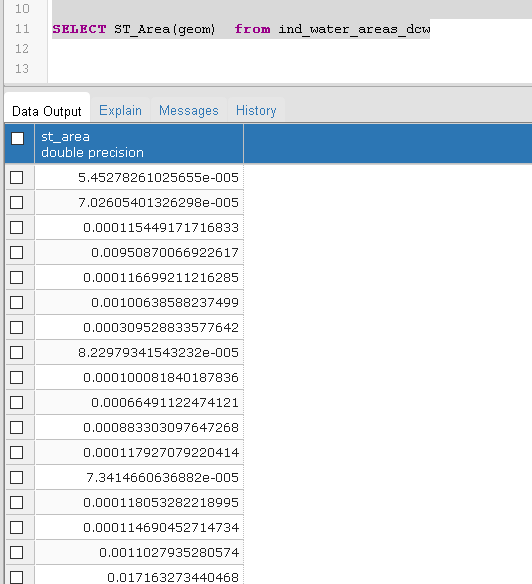
Write a query to find perimeter of water areas using the spatial data

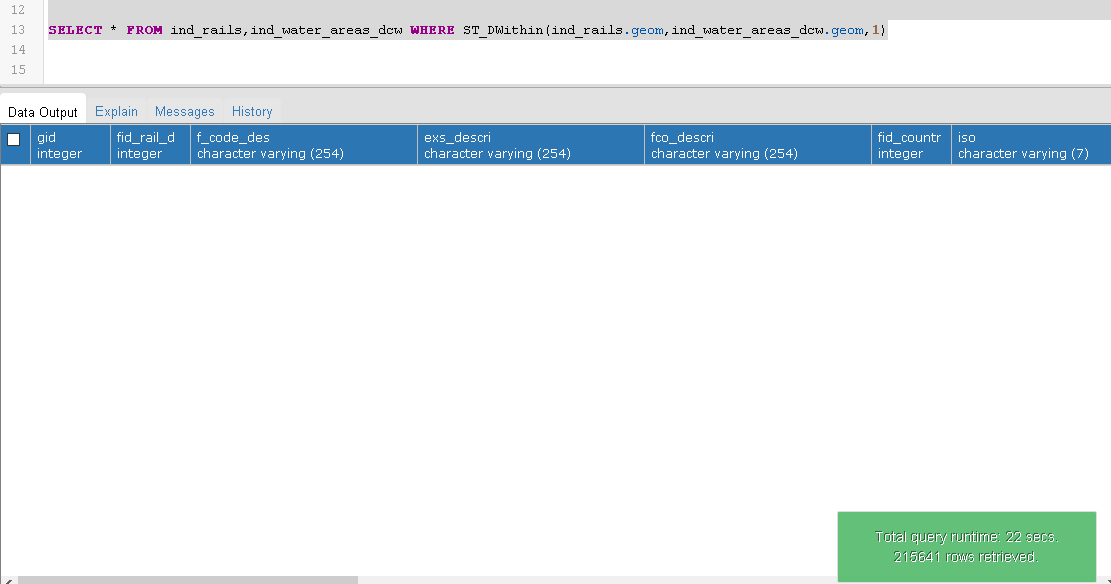


Write a query to find length of rails using the spatial data

Write a query to find Area of water areas using the spatial data





Write a query to Find all the rails and water areas which lie within 1 distance from each other



**Questions:**

# Explain the spatial functions used for these queries in detail.

# Ans:

# Spatial functions are used to query and manipulate spatial data in a database. They allow you to perform operations such as measuring distances between points, finding objects within a certain radius, and computing the area of a polygon. Here are some common spatial functions and their descriptions:

* **ST\_Intersects:** This function tests whether two geometry objects intersect. It returns true if the two objects share any portion of their space. For example, ST\_Intersects(p1, p2) returns true if point p1 and point p2 intersect.
* **ST\_Within:** This function tests whether a geometry object is within another geometry object. It returns true if the first object is completely within the second object. For example, ST\_Within(p1, polygon) returns true if point p1 is within the polygon.
* **ST\_Area:** This function calculates the area of a polygon geometry object. It returns the area in the units specified by the spatial reference system of the object. For example, ST\_Area(polygon) returns the area of the polygon.

1. **Explain any two applications of spatial database.**

**Ans:** Spatial databases are used to store, manage, and manipulate spatial data, which includes information about the location and shape of objects. Here are two applications of spatial databases:

# Geographic Information Systems (GIS):

# A GIS is a system designed to capture,store, manipulate, analyze, manage, and present spatial or geographic data.

# A GIS uses a spatial database to store the spatial data, such as maps, satellite imagery, and demographic data.

# Spatial databases are an integral part of GIS as they provide a structured and organized way of storing the spatial data. GIS is used in various fields, including urban planning, environmental management, forestry, transportation, and public health.

* **Location-Based Services (LBS):**
* LBS is a type of service that uses location data to provide information, entertainment, or security to users.
* Spatial databases are used in LBS to store and retrieve information about points of interest, such as restaurants, hotels, and gas stations.
* The information stored in the spatial database can be used to provide directions, traffic updates, and location-based advertising.
* LBS is used in various applications, such as mobile navigation, social networking, and emergency response systems.



# Outcomes:

# CO2: Design advanced database systems using Object Relational, Spatial and NOSQL Databases and its implementation.

**Conclusion: (Conclusion to be based on outcomes achieved)**

**Thus we successfully implemented the execution of spatial queries.**

Elmasri and Navathe, “Fundamentals of Database Systems”, Pearson Education

* 1. Raghu Ramakrishnan and Johannes Gehrke, “Database Management Systems” 3rd Edition, McGraw Hill,2002
  2. Korth, Silberchatz, Sudarshan, “Database System Concepts” McGraw Hill
  3. <http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_tut01>