

**Experiment No.**

**4**

**Title: Execution of Spatial database queries**

Page No:

# Batch: B2 Roll No.: 1914078 Experiment No.:4

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Resources needed:** PostgreSQL 9.6, PostGIS2.0

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Theory

**PostGIS** is an open-source software program that adds support for geographic objects to thePostgreSQL 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, multipolygons 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/nonspatial queries.
* For raster data

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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. Create spatial database.
4. Execute the spatial queries on given data to perform following task. (Refer http://www.postgis.org/docs/reference.html)
   1. Find the rivers crossing 'United States'.
   2. List the countries with the area covered by water body III. Find cities belong to Canada
5. Execute a KNN query to get K neighbours of the given points.
6. Execute any one range query

“**KNN**” stands for “K nearest neighbours”, where “K” is the number of neighbours you are looking for. **KNN** is a pure index based nearest neighbour search. ... The **KNN**system works by evaluating distances between bounding boxes inside the **PostGIS**R-Tree index.

|  |
| --- |
| SELECT ST\_Distance(geom, 'SRID=3005;POINT(1011102 450541)'::geometry) as d,edabbr, |
| vaabbr  FROM va2005  ORDER BY d limit 10;  d | edabbr | vaabbr  ------------------+--------+--------  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  (10 rows) |

# Range query in Postgis

SELECT ST\_Reclass(rast, 1,

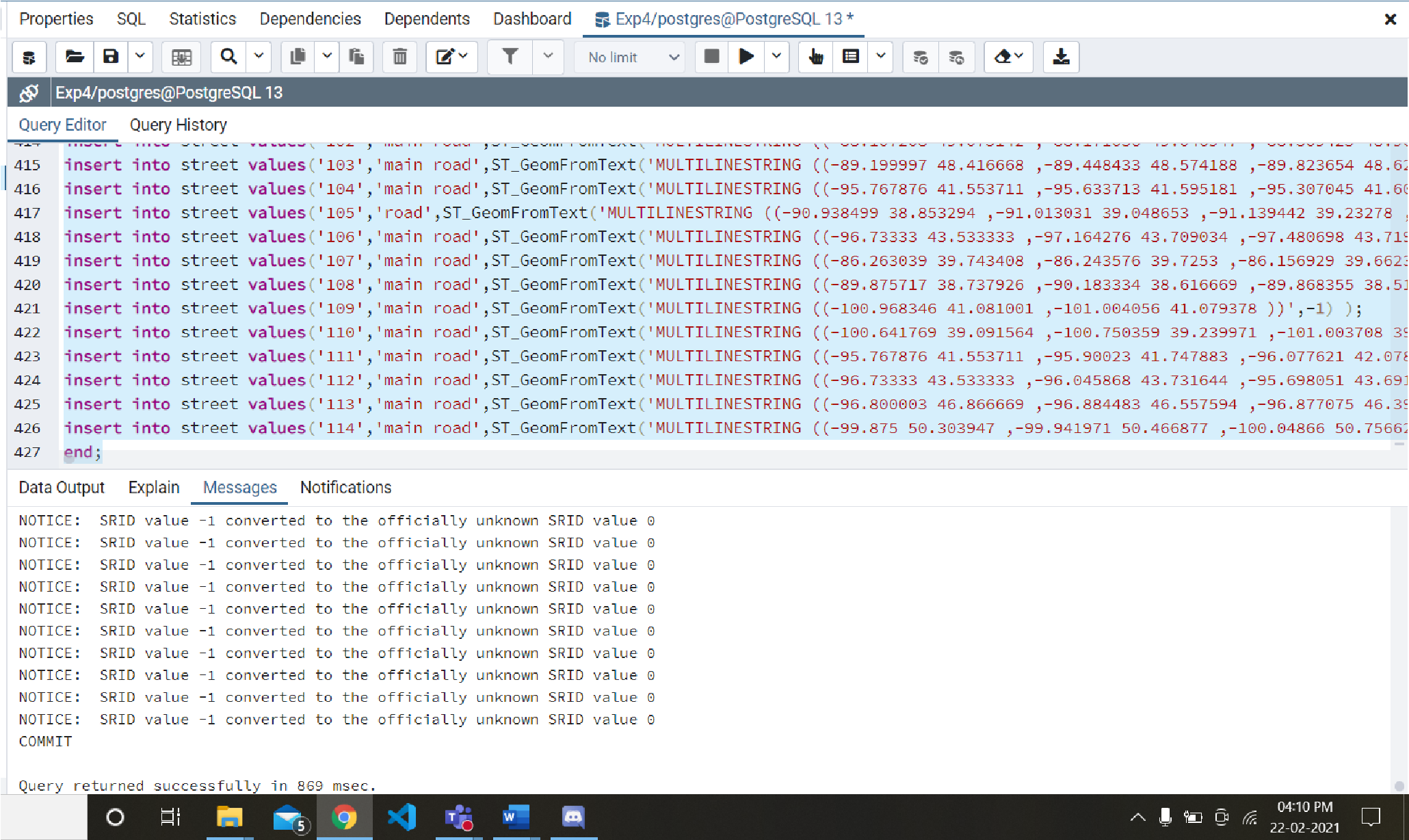
'[0-90]:0,(90-100):1,[100-1000):2',

'4BUI', 0) AS rast FROM sometable

WHERE filename = '123.tif';

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Results: (Program printout with output)

Running given script:-



# Exploring:-

**INT4** is an alias for the INTEGER data type.

**INT8** is an alias for the BIGINT data type.

**FLOAT8** is an alias for the DOUBLE data type.

**NUMERIC** datatype is the superset of all number data types like float, double, int etc. It can be used to reference or make any kind of number data type. An advantage of using numeric type is that we can use varied range of each datatype to conserve storage. The syntax for using numeric data type is NUMERIC(precision, scale). We used NUMERIC(38,9) meaning 38 digits and 9 can be entered in decimal.

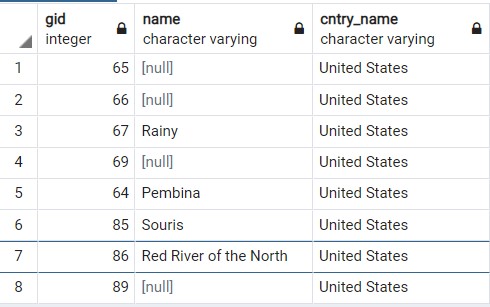
**Q) Find the rivers crossing 'United States'.**

**Ans)** select r.gid,r.name,c.cntry\_name

From river r,country c

where

((ST\_Crosses(r.the\_geom, c.the\_geom )) AND c.cntry\_name= 'United States') ;



## Q) List the countries with the area covered by water body

**Ans)** select distinct c.cntry\_name,w.name from country c, water w

where

(ST\_Within(w.the\_geom,c.the\_geom) );



## Q) Find cities belong to Canada

**Ans)** select c1.city\_name,c2.cntry\_name from city c1,country c2

where((ST\_CoveredBy(c1.the\_geom,c2.the\_geom)) AND

c2.cntry\_name='Canada');

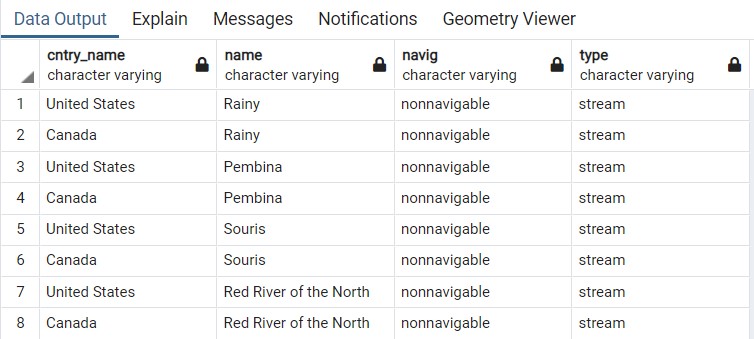


# Extra queries:-

**Q)** Selecting country, river name and river navigation if the type of river is a lake and the river crosses the country

select country.cntry\_name, river.name, river.navig, river.type from country,river

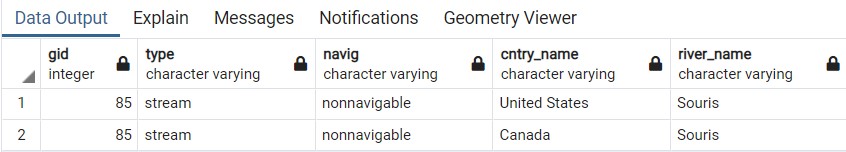
where((ST\_crosses(country.the\_geom,river.the\_geom)) and river.type = 'stream');



**Q)** Selecting the river gid, type , navigation and country name if the river and country cross and the name of the river is Pembina

select r.gid,r.type,r.navig,c.cntry\_name, r.name as river\_name From river r,country c

where((ST\_Crosses(r.the\_geom, c.the\_geom )) and r.name='Souris');



# KNN query:-

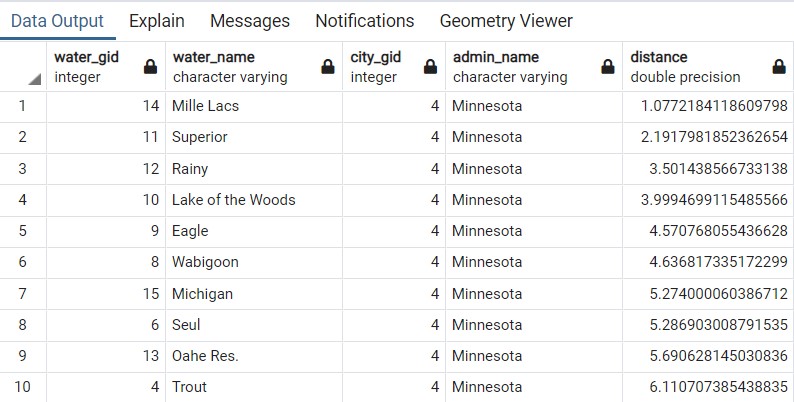
**Q)** Selecting water gid, name, city admin\_name and gid for top 10 waters nearest to city Minneapolis

SELECT water.gid as water\_gid, water.name as water\_name, city.gid as city\_gid, city.admin\_name, ST\_Distance(water.the\_geom, city.the\_geom) as distance

FROM water, city

WHERE city.city\_name = 'Minneapolis'

ORDER BY ST\_Distance(water.the\_geom, city.the\_geom) ASC LIMIT 10;



# Range query:-

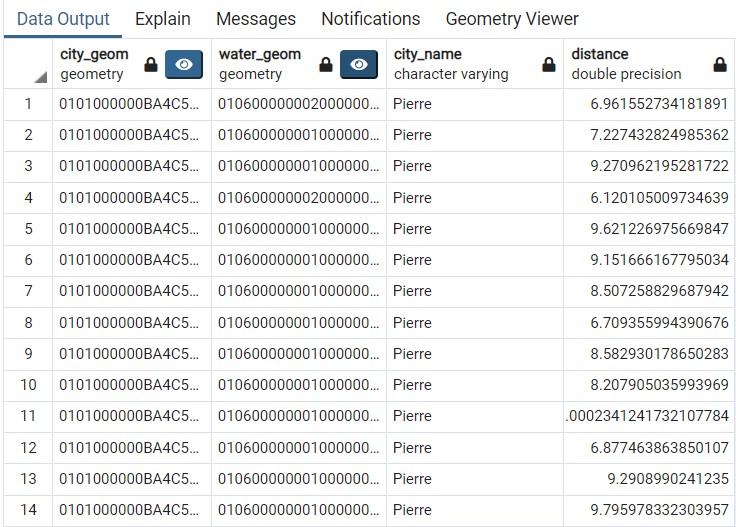
**Q)** Selecting water geom, city name and geom and distance for finding all waters within range of 10 from city Pierre:-

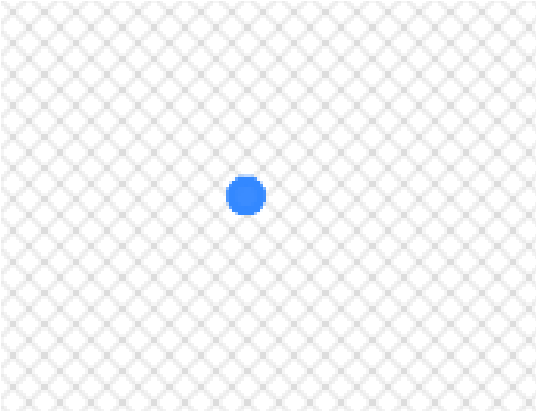
SELECT a.the\_geom as city\_geom, b.the\_geom as water\_geom, a.city\_name,

ST\_Distance(a.the\_geom, b.the\_geom) as distance

FROM city a, water b

WHERE a.city\_name='Pierre' and ST\_DWithin(a.the\_geom, b.the\_geom, 10);



City\_geom:

Water\_geom: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Questions:**

**1. Explain the spatial functions used for these queries in detail.**

**Ans.**

* **ST\_Crosses:** ST\_Crosses takes two geometry objects and returns TRUE if their intersection "spatially cross", that is, the geometries have some, but not all interior points in common. The intersection of the interiors of the geometries must not be the empty set and must have a dimensionality less than the maximum dimension of the two input geometries. Additionally, the intersection of the two geometries must not equal either of the source geometries. Otherwise, it returns FALSE.
* **ST\_Within:** Returns TRUE if geometry A is completely inside geometry B. For this function to make sense, the source geometries must both be of the same coordinate projection, having the same SRID. It is a given that if ST\_Within(A,B) is true and ST\_Within(B,A) is true, then the two geometries are considered spatially equal.
* **ST\_CoveredBy:** Returns 1 (TRUE) if no point in Geometry/Geography A is outside Geometry/Geography B.
* **ST\_Distance:** For geometry types, returns the minimum 2D Cartesian (planar) distance between two geometries, in projected units (spatial ref units).

For geography types, defaults to return the minimum geodesic distance between two geographies in meters, compute on the spheroid determined by the SRID. If use\_spheroid is false, a faster spherical calculation is used.

* **ST\_DWithin:** Returns true if the geometries are within a given distance. For geometry: The distance is specified in units defined by the spatial reference system of the geometries. For this function to make sense, the source geometries must be in the same coordinate system (have the same SRID). For geography: units are in meters and distance measurement defaults to use\_spheroid=true. For faster evaluation use use\_spheroid=false to measure on the sphere.

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Outcomes:**

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

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

Through this experiment we learnt the implementation of spatial queries using Postgis in PostgreSQL. We successfully installed Postgis and executed the queries according to given problem statements. We also learnt to execute KNN and Range Queries.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Grade: AA / AB / BB / BC / CC / CD /DD**

## Signature of faculty in-charge with date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ References:**

1. Elmasri and Navathe, “Fundamentals of Database Systems”, Pearson Education
2. Raghu Ramakrishnan and Johannes Gehrke, “Database Management Systems” 3rd Edition, McGraw Hill,2002
3. Korth, Silberchatz, Sudarshan, “Database System Concepts” McGraw Hill
4. http://www.bostongis.com/PrinterFriendly.aspx?content\_name=postgis\_tut01

Page No: