

**Experiment No.5**

**Title: Execution of Spatial database queries**

**Batch: B-4** **Roll No.: 16010422234 Name: Chandana Ramesh Galgali**

**Experiment No.: 5**

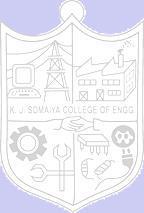
**Aim: To execute spatial queries using PostGIS. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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, 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 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
8. Create account on GEE using Somaiya email id
9. Upload the shapefile
10. Visualize it on GEE

Access the video resources **GE\_shapefile-SMP.mp4** and **shptocsv,mp4** from

<https://drive.google.com/drive/folders/1jB7t4zVtyANA70XfHiwF2qU_JSMlU1_n?usp=drive_link>

var shapefile : Table projects/ee-suchitrapatil23/assets/indian\_waterbodies\_area// this you need to do for your shapefile in assets

// Load the shapefile as a FeatureCollection.

var shapefile = shapefileUrl;

// Display the shapefile on the map.

Map.centerObject(shapefile, 10); // Center the map on the shapefile

Map.addLayer(shapefile, {}, 'Shapefile');

// Print the FeatureCollection to the console to inspect its properties.

print(shapefile);

// Convert the shapefile to a CSV table.

var csvTable = shapefile;

// Print the resulting CSV table.

print(csvTable);

// Export the CSV table to Google Drive.

Export.table.toDrive({

collection: csvTable,

description: 'indian\_waterbodies\_shapefilecsv',

fileFormat: 'CSV'

});

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)**

**create extension postgis;**

**select \* from USA\_water\_areas\_dcw;**

**select \* from USA\_roads;**

**select \* from USA\_adm1;**

**select \* from USA\_roads, USA\_water\_areas\_dcw where st\_crosses(USA\_roads.geom, USA\_water\_areas\_dcw.geom);**

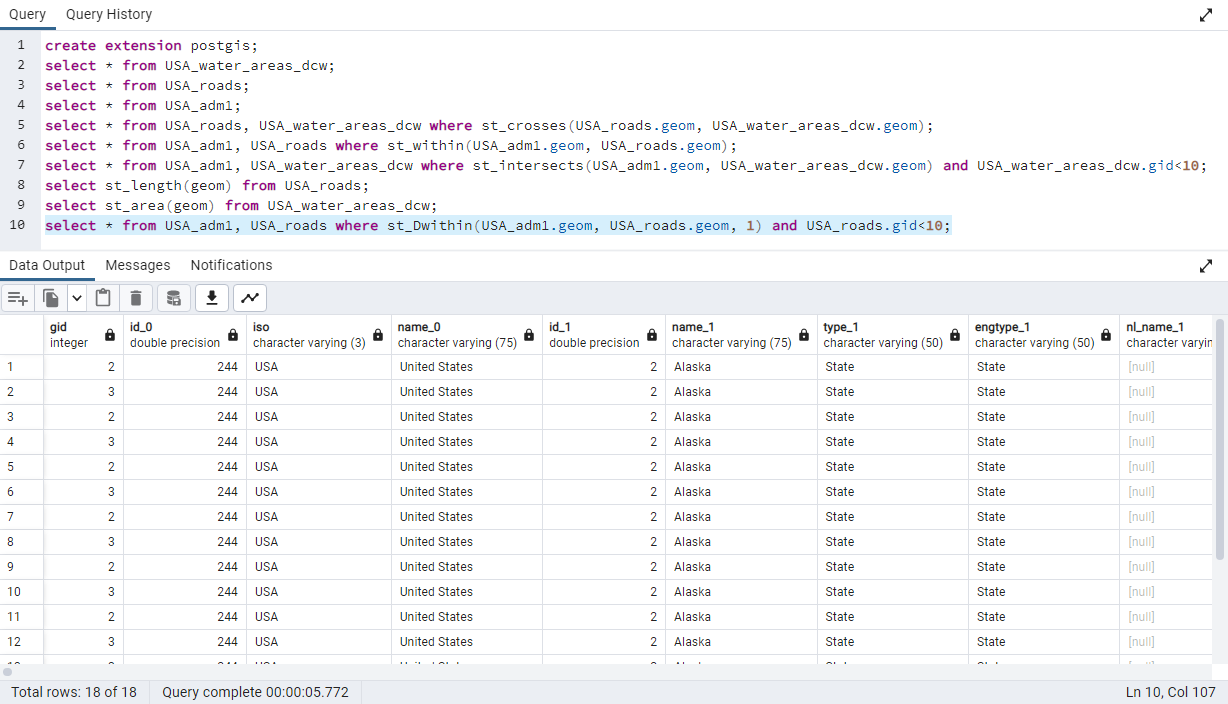
**select \* from USA\_adm1, USA\_roads where st\_within(USA\_adm1.geom, USA\_roads.geom);**

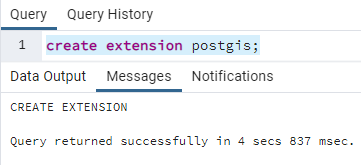
**select \* from USA\_adm1, USA\_water\_areas\_dcw where st\_intersects(USA\_adm1.geom, USA\_water\_areas\_dcw.geom) and USA\_water\_areas\_dcw.gid<10;**

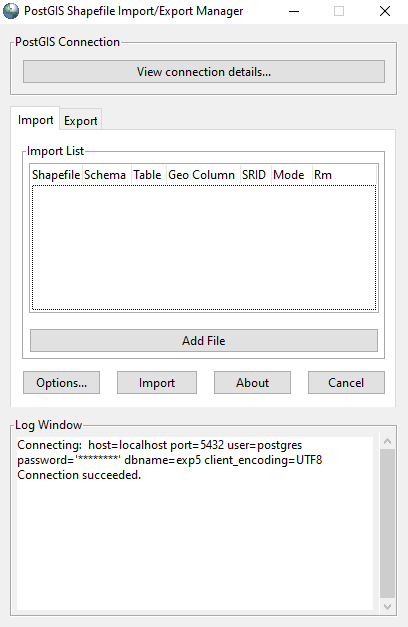
**select st\_length(geom) from USA\_roads;**

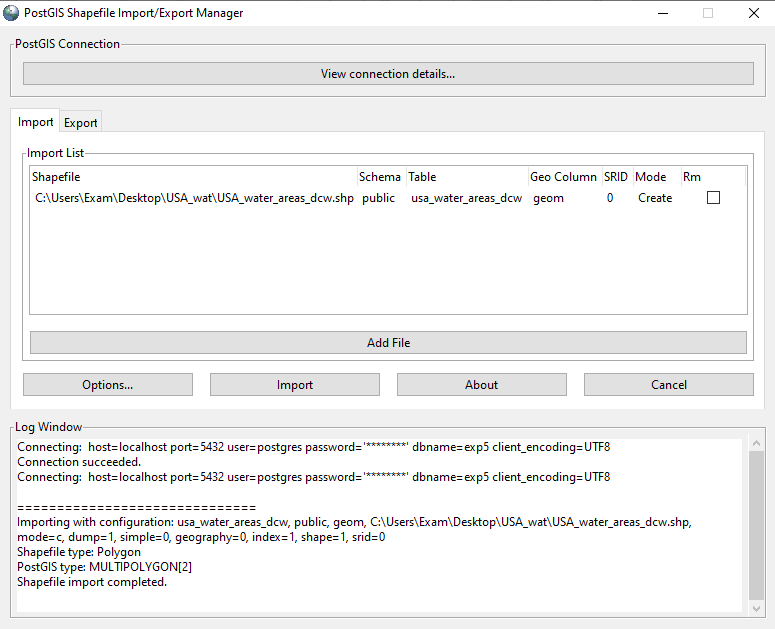
**select st\_area(geom) from USA\_water\_areas\_dcw;**

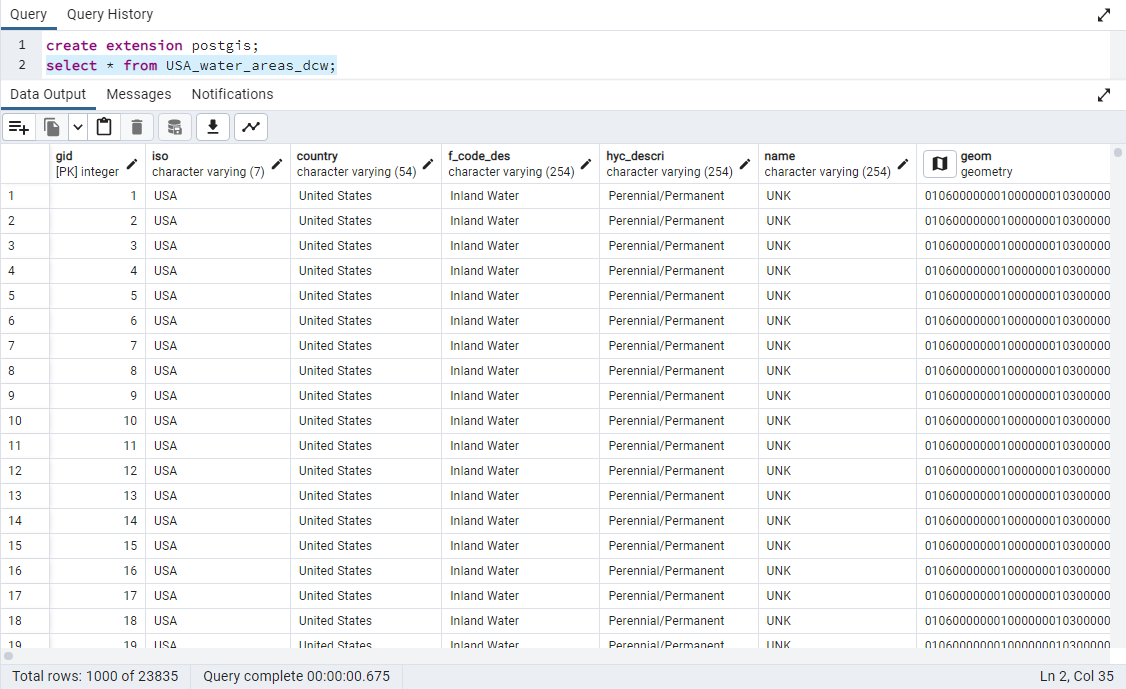
**select \* from USA\_adm1, USA\_roads where st\_Dwithin(USA\_adm1.geom, USA\_roads.geom) and USA\_roads.gid<10;**

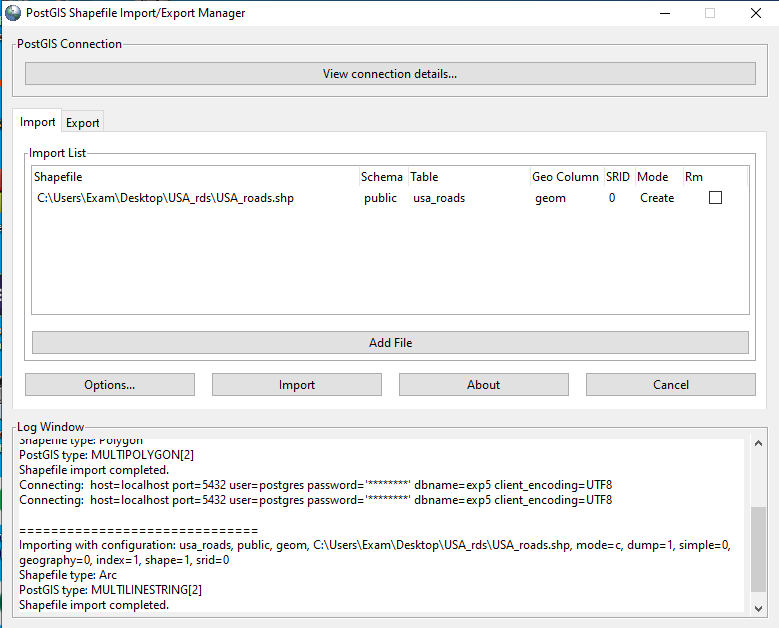
****

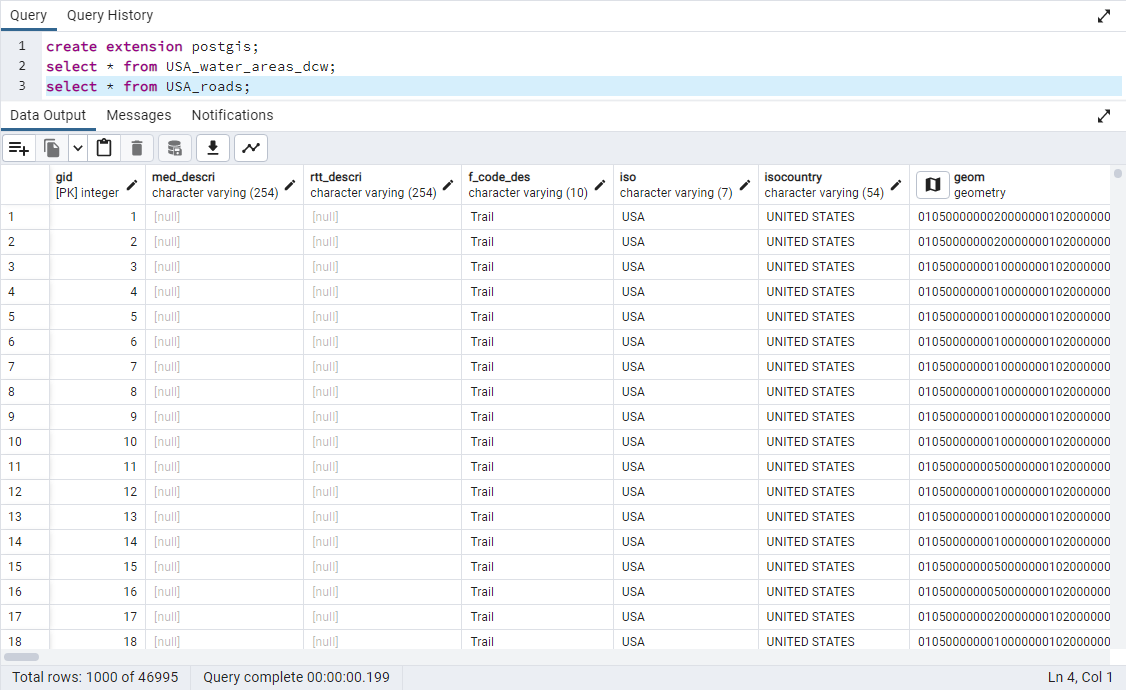


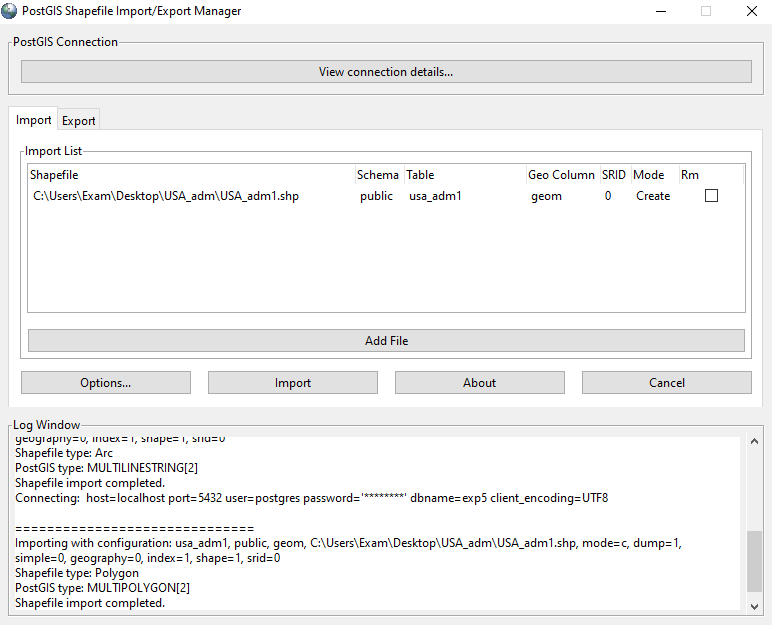


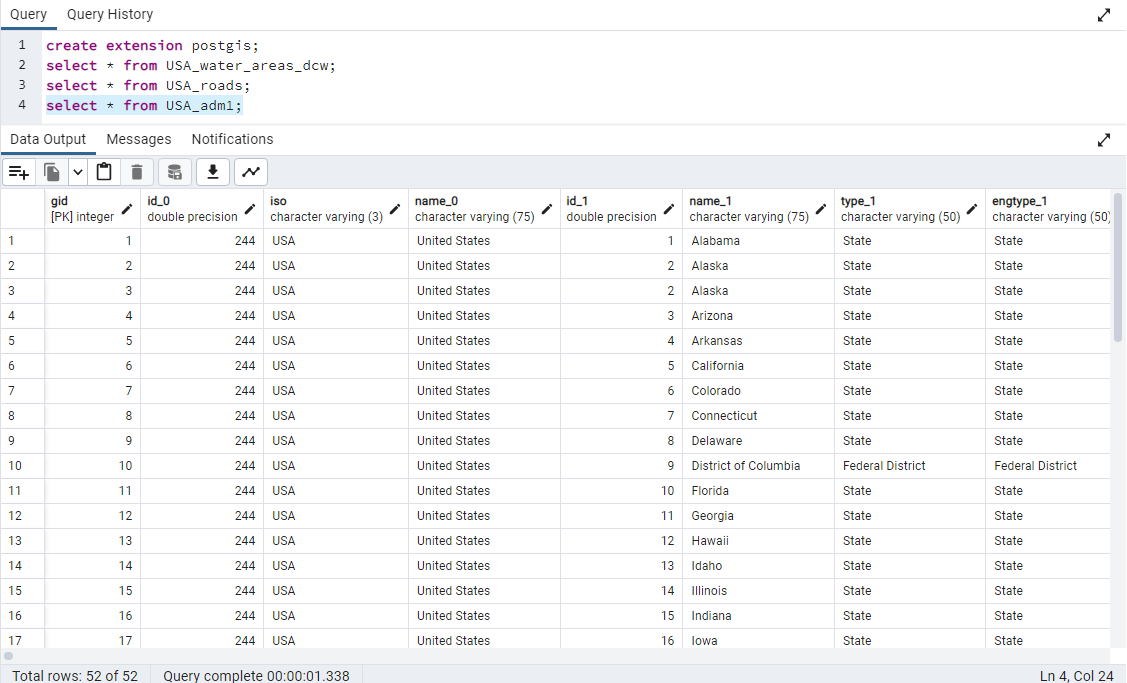


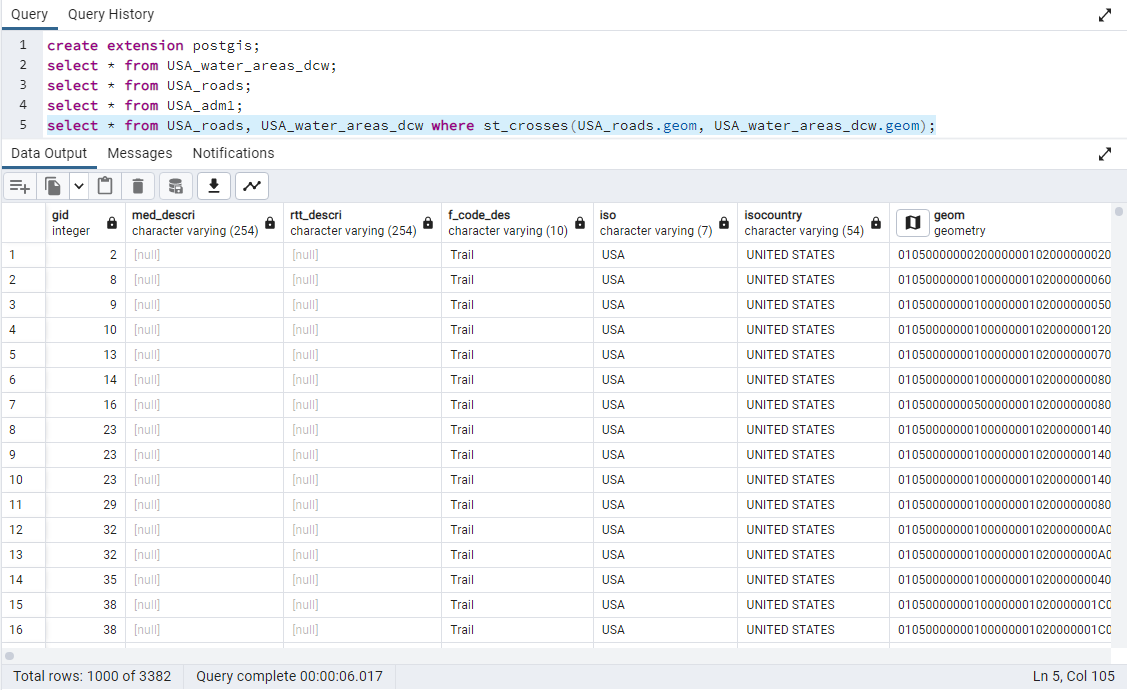


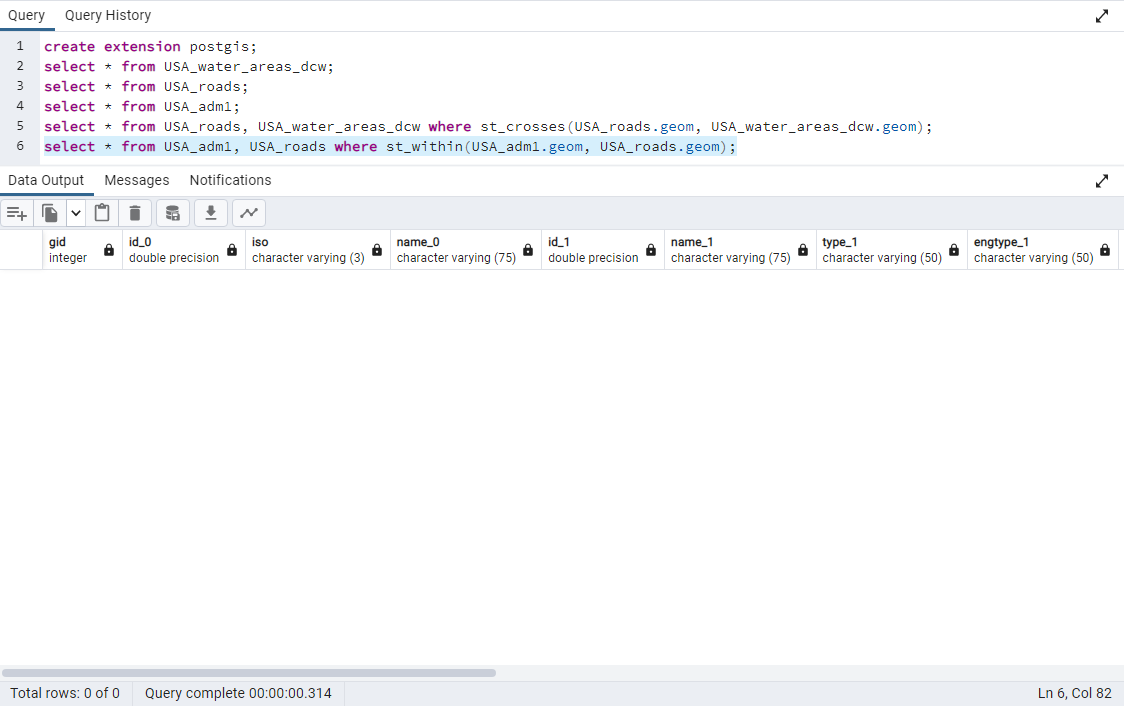


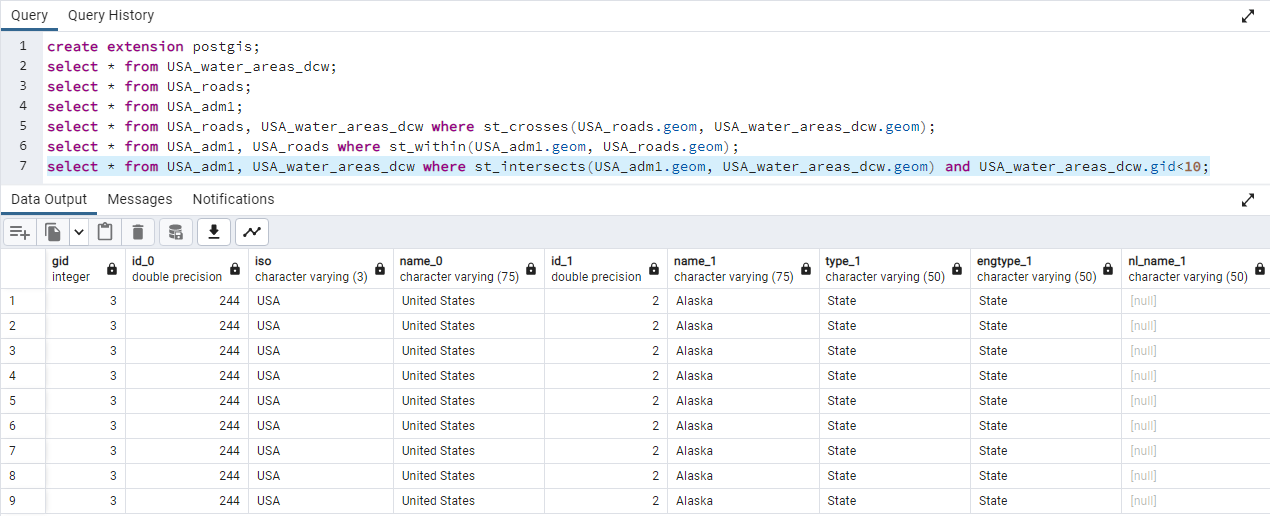


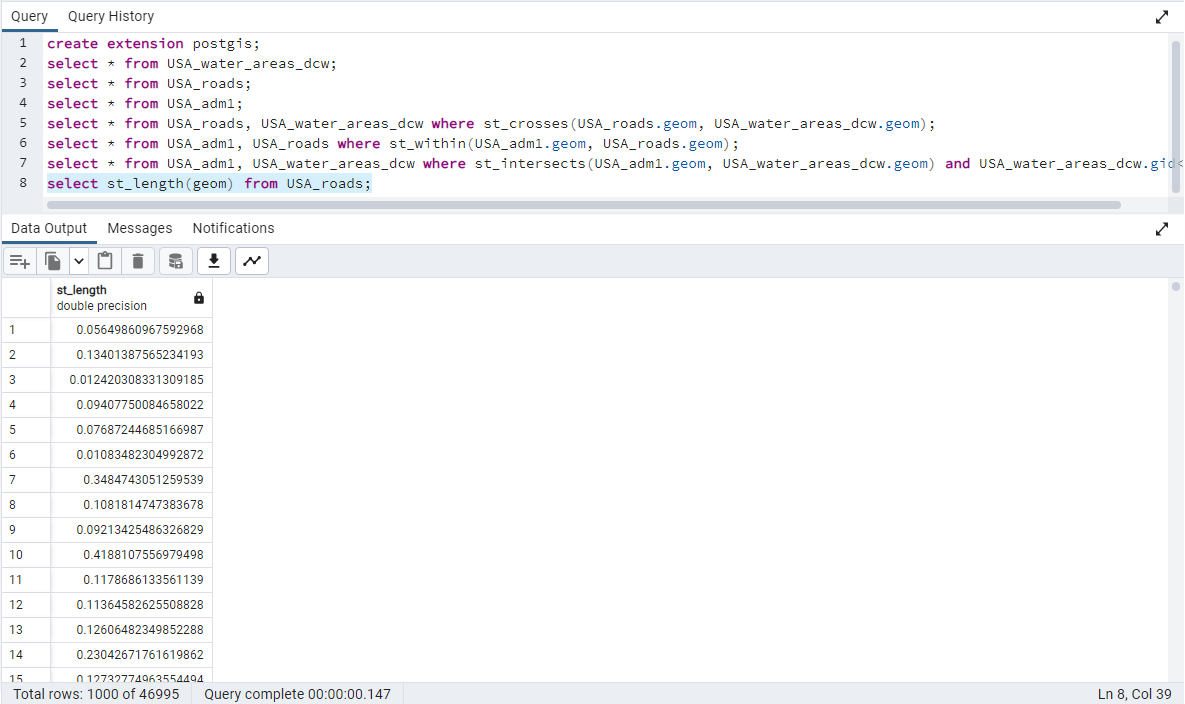


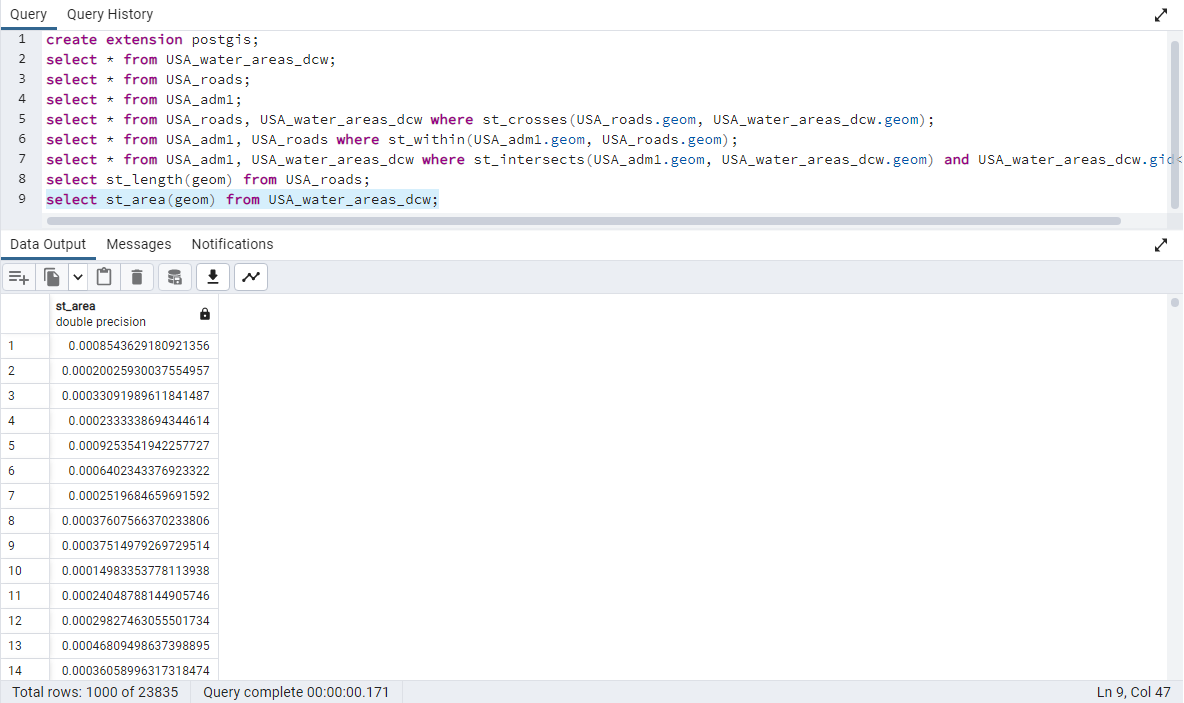


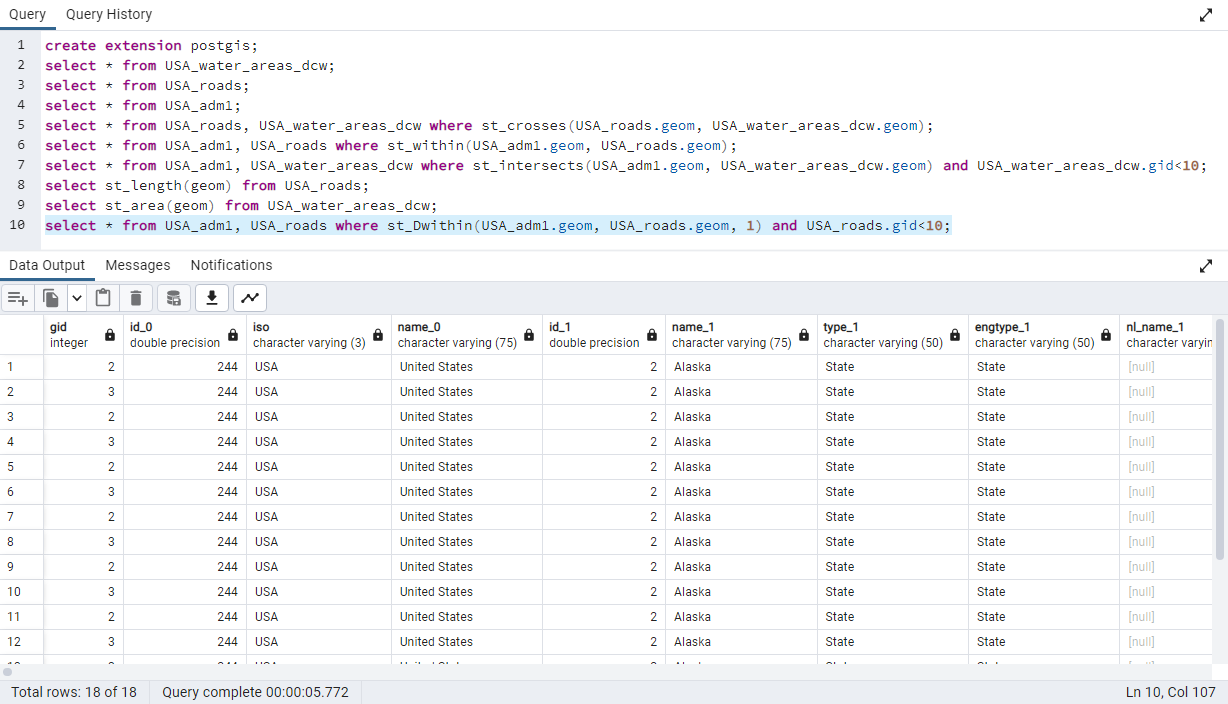


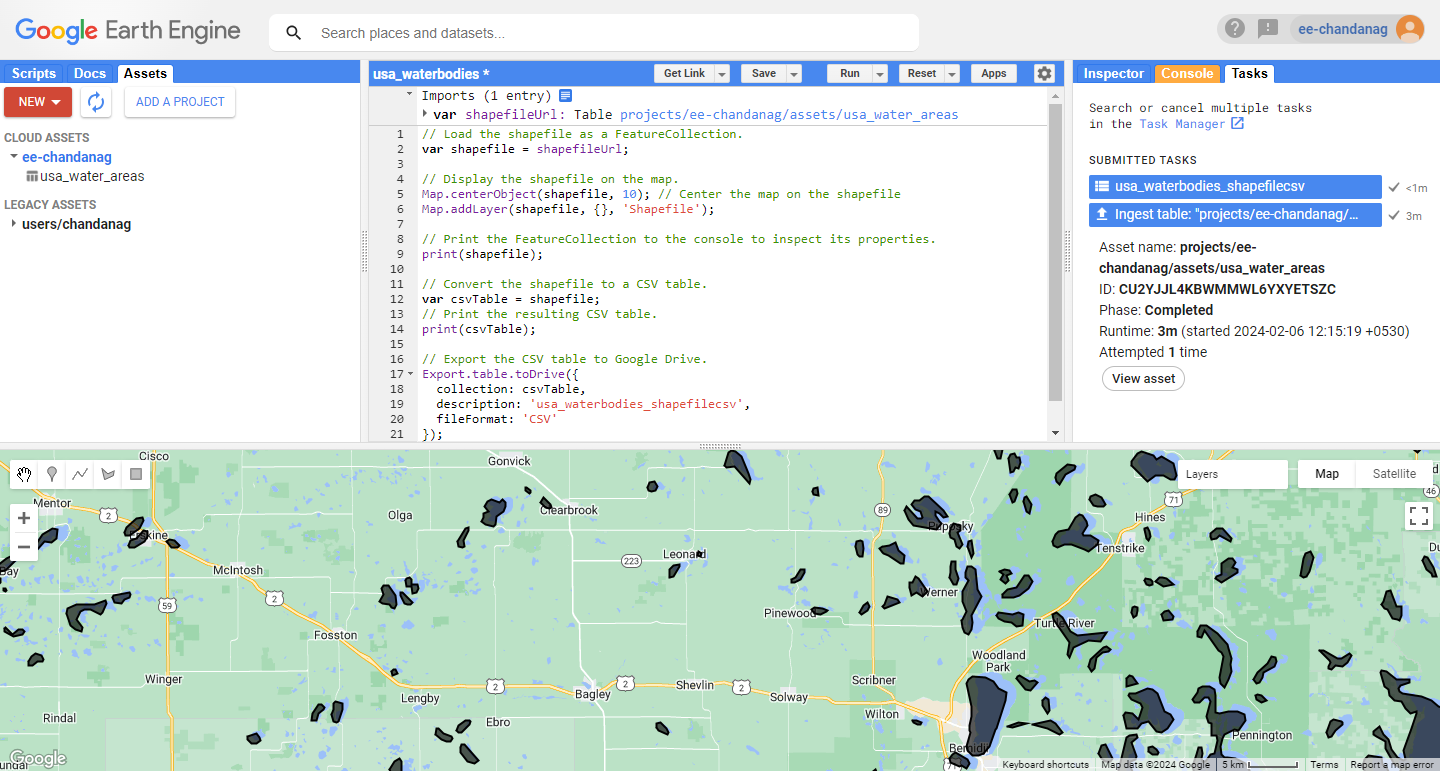


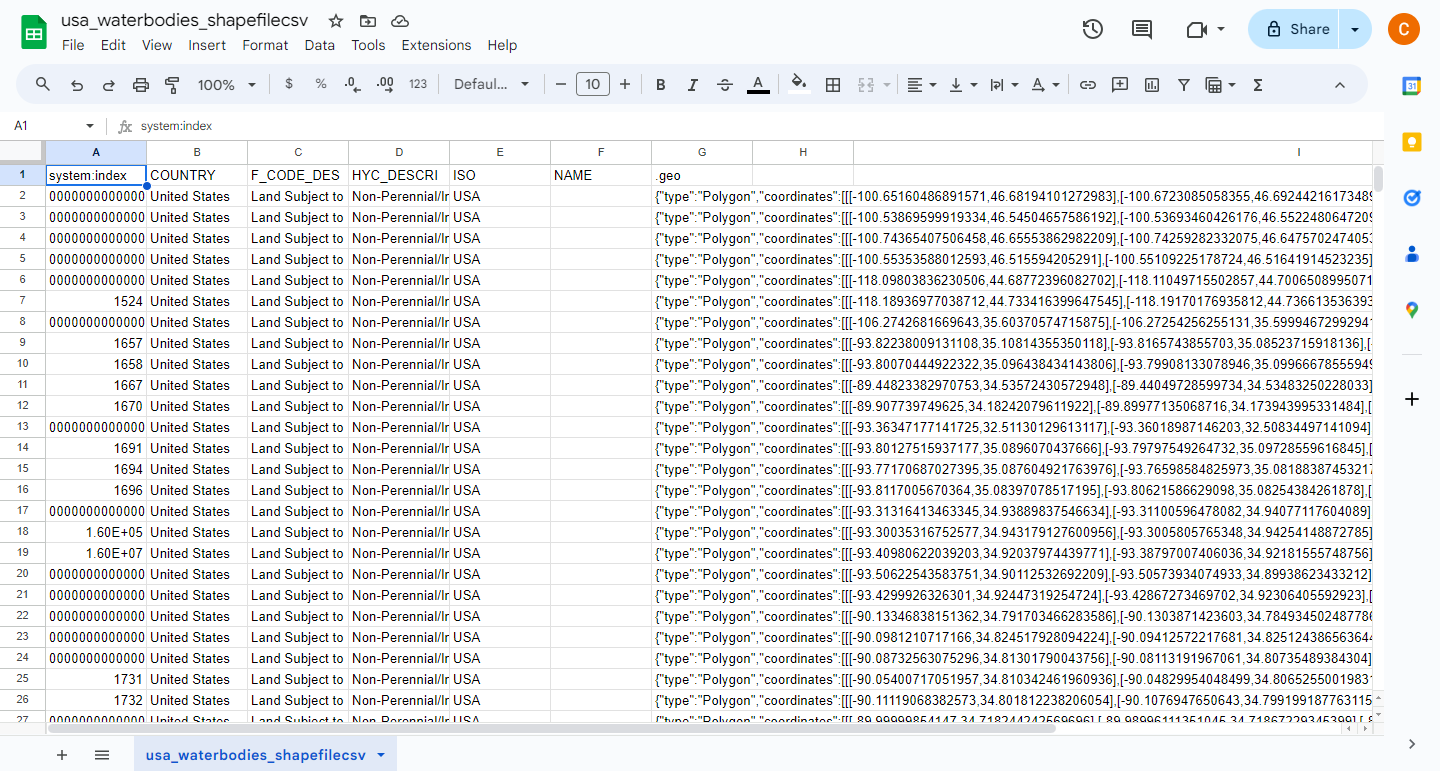












**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Questions:**

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

**Ans:** Spatial databases handle data that has a spatial component, such as geographic information system (GIS) data, maps, or any information related to locations on Earth's surface. Spatial functions are used to perform operations on spatial data. Here are some common spatial functions:

Distance Function:

Example: ST\_Distance(geometry1, geometry2)

Explanation: Calculates the distance between two geometries, which could be points, lines, or polygons. It's often used to find the nearest neighbors or to analyze proximity.

Intersection Function:

Example: ST\_Intersects(geometry1, geometry2)

Explanation: Checks whether two geometries intersect with each other. It's frequently used to find overlapping or intersecting spatial features.

Buffer Function:

Example: ST\_Buffer(geometry, distance)

Explanation: Creates a buffer zone around a geometry. This is useful for proximity analysis and to study the influence of one spatial feature on another within a certain distance.

Area Function:

Example: ST\_Area(polygon)

Explanation: Calculates the area of a polygon. This is useful for tasks such as land-use planning or environmental analysis.

Within Function:

Example: ST\_Within(geometry1, geometry2)

Explanation: Checks whether one geometry is completely within another. This can be used to identify containment relationships between spatial features.

Centroid Function:

Example: ST\_Centroid(geometry)

Explanation: Computes the center point of a geometry. It's often used in mapping applications to label or symbolize features.

**2. Explain any two applications of spatial databases.**

**Ans:** GIS (Geographic Information Systems):

Spatial databases are extensively used in GIS for mapping and analyzing spatial data. They store and manage geographic information such as maps, satellite imagery, and terrain data. GIS applications are widely employed in urban planning, environmental monitoring, natural resource management, and disaster response. For instance, a city planner might use a spatial database to analyze the impact of a new development on the existing infrastructure.

Location-Based Services (LBS):

Spatial databases play a crucial role in Location-Based Services, where the user's location is used to provide relevant information or services. Examples include mapping applications (like Google Maps), location-based advertising, and location-based social networking. Spatial databases help in efficiently storing and querying large datasets of geographical points of interest and providing real-time information based on the user's location.

Emergency Response and Public Safety:

Spatial databases are vital in emergency response systems. They can store information about the locations of hospitals, emergency shelters, fire stations, and other critical infrastructure. During a disaster, such as a natural calamity or a pandemic, spatial databases can be used to quickly analyze affected areas, plan evacuation routes, and allocate resources effectively.

Environmental Monitoring:

Spatial databases are used to store and manage environmental data, including information about ecosystems, habitats, and climate. Scientists and researchers use spatial databases to analyze spatial patterns, track changes over time, and make informed decisions about conservation and environmental management.

Transportation and Logistics:

In transportation and logistics, spatial databases are employed to manage and optimize the movement of goods and people. They are used to store information about road networks, traffic patterns, and distribution centers. Route planning, fleet management, and supply chain optimization are examples of applications where spatial databases play a crucial role.

These applications highlight the versatility and importance of spatial databases in various domains where location-based information is critical for decision-making and analysis.

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

**Outcomes: Design advanced database systems using Object relational, Spatial and NOSQL databases and its implementation.**

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

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

The outcomes of the experiment underscore PostGIS as a powerful tool for executing spatial queries, making it a valuable asset in spatial data management and analysis. The successful execution of diverse spatial functions demonstrates its applicability across different domains, positioning PostGIS as a reliable solution for handling and extracting insights from spatial datasets.

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

**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

1. Korth, Silberchatz, Sudarshan, “Database System Concepts” McGraw Hill
2. http://www.bostongis.com/PrinterFriendly.aspx?content\_name=postgis\_tut01