Experiment No.4

Title: Execution of Spatial database queries

Batch: B1 **Roll No.: 16010420133** **Experiment No.:4**

Aim: To execute spatial queries using PostGIS.

Resources needed: PostgreSQL 9.6, PostGIS 2.0

Theory

A spatial database is a database that is enhanced to store and access spatial data or data that defines a geometric space. These data are often associated with geographic locations and features, or constructed features like cities. Data on spatial databases are stored as coordinates, points, lines, polygons and topology. Some spatial databases handle more complex data like three-dimensional objects, topological coverage and linear networks.

Spatial Database in PostgreSQL

PostGIS turns the PostgreSQL Database Management System into a spatial database by adding support for the three features: spatial types, spatial indexes, and spatial functions. Because it is built on PostgreSQL, PostGIS automatically inherits important "enterprise" features as well as open standards for implementation.

PostGIS uses three data models : OGC Geometry, SQL/MM Curves and WKT and WKB.

Geometry

The Open Geospatial Consortium (OGC) developed the Simple Features Access standard (SFA) to provide a model for geospatial data. It defines the fundamental spatial type of Geometry, along with operations which manipulate and transform geometry values to perform spatial analysis tasks. PostGIS implements the OGC Geometry model as the PostgreSQL data types geometry and geography.

Geometry models shapes in the 2-dimensional Cartesian plane or in 3-dimensional plane for certain types like PolyHedralSurface. Each coordinate has a X and Y ordinate value determining its location in the plane. Shapes are constructed from points or line segments, with points specified by a single coordinate, and line segments by two coordinates.

Coordinates may contain optional Z and M ordinate values. The Z ordinate is often used to represent elevation. The M ordinate contains a measure value, which may represent time or distance.

Geometry values are associated with a spatial reference system indicating the coordinate system in which it is embedded. The spatial reference system is identified by the geometry SRID number. The units of the X and Y axes are determined by the spatial reference system. In planar reference systems the X and Y coordinates typically represent easting and northing, while in geodetic systems they represent longitude and latitude. SRID 0 represents an infinite Cartesian plane with no units assigned to its axes

K. J. SOMAIYA COLLEGE OF ENGG

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:

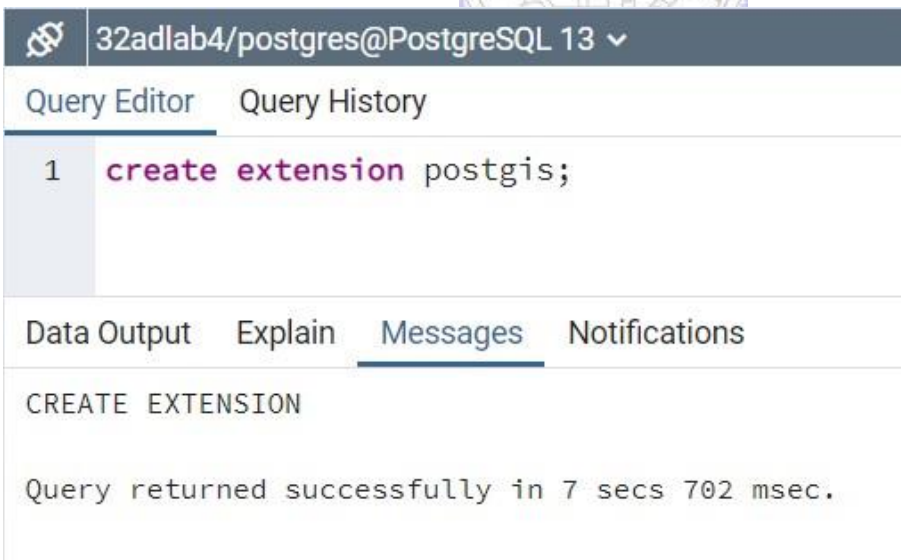
1. Download spatial data from **`https://www.diva-gis.org/gdata`** (OR similar website with FREE usable data) Get it for any country with minimum 3 subjects.
2. Import the data in your PostgreSQL
3. Identify spatial relationship between any two geometric entities (any 3 named relationships)
4. Perform any two measurement functions for geometric data.

Results: (Queries depicting the above said activity performed individually and snapshots of the result(if any))

USE COURIER NEW FONT WITH SIZE =11 FOR QUERY STATEMENTS

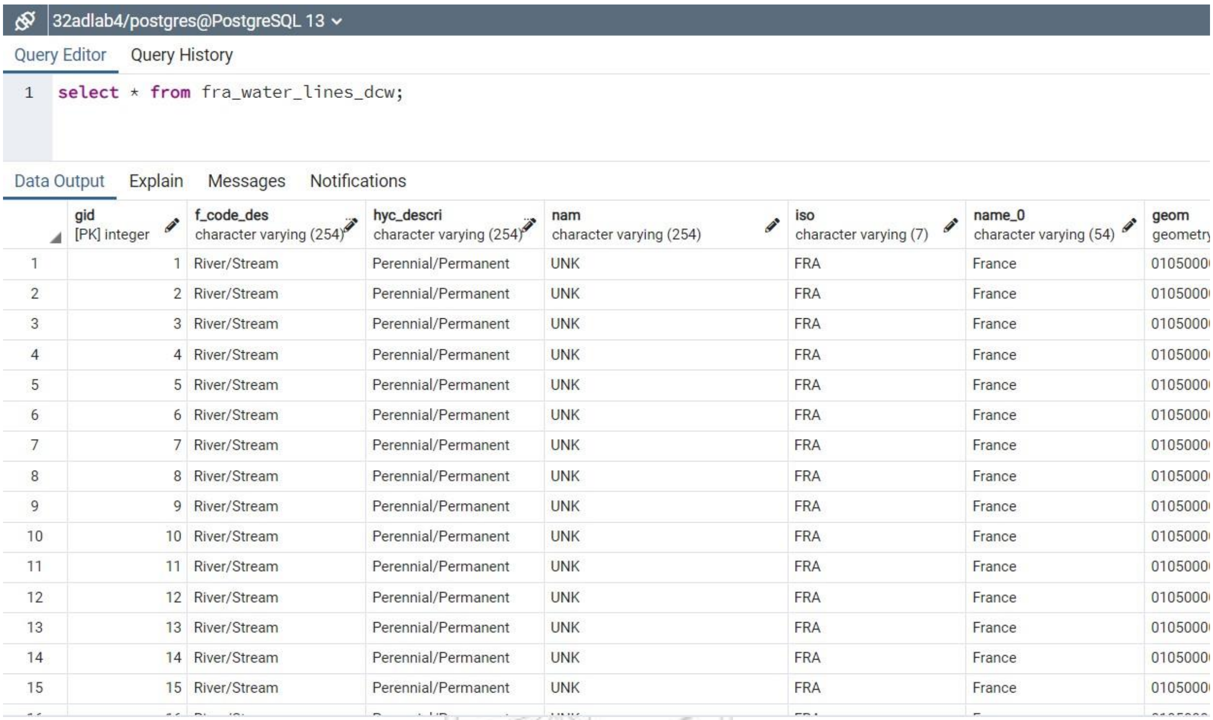
Create postgis extention:

```
create extension postgis;
```



Import water bodies data via postgis:

```
select * from fra_water_lines_dcw;
```



32adlab4/postgres@PostgreSQL 13

Query Editor   Query History

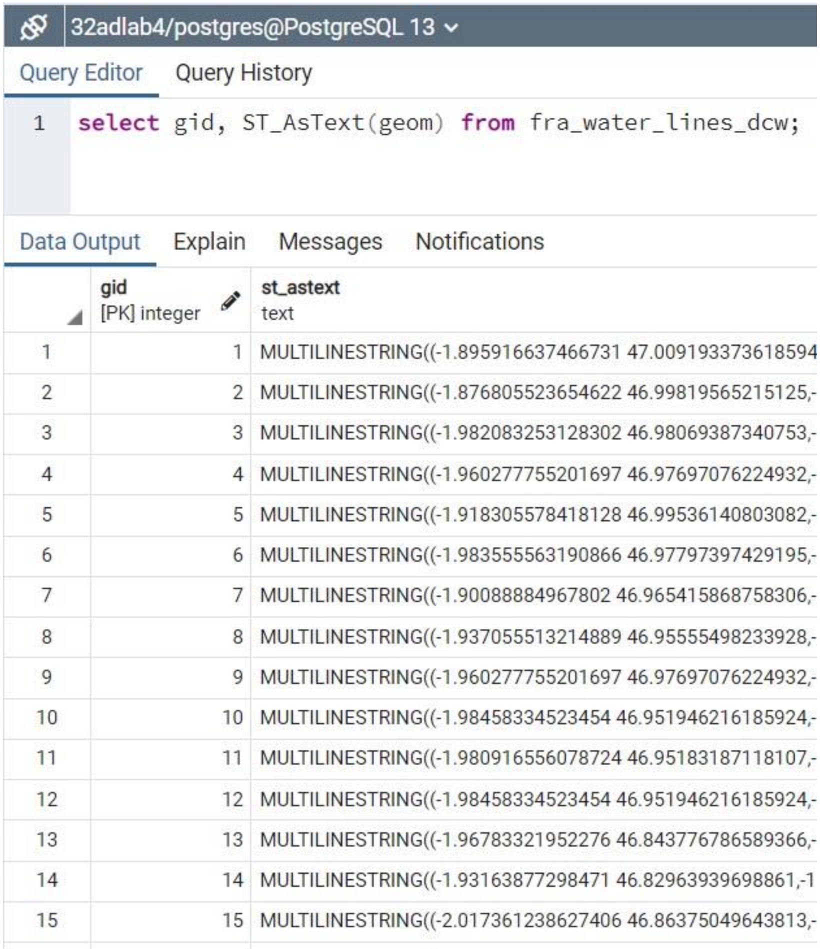
```
1  select * from fra_water_lines_dcw;
```

Data Output   Explain   Messages   Notifications

	gid [PK] integer	f_code_des character varying (254)	hyc_descri character varying (254)	nam character varying (254)	iso character varying (7)	name_0 character varying (54)	geom geometry
1	1	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
2	2	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
3	3	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
4	4	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
5	5	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
6	6	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
7	7	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
8	8	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
9	9	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
10	10	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
11	11	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
12	12	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
13	13	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
14	14	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000
15	15	River/Stream	Perennial/Permanent	UNK	FRA	France	0105000

K. J. SOMAIYA COLLEGE OF ENGG

Add geometry: `select gid, ST_AsText(geom) from fra_water_lines_dcw;`



32adlab4/postgres@PostgreSQL 13

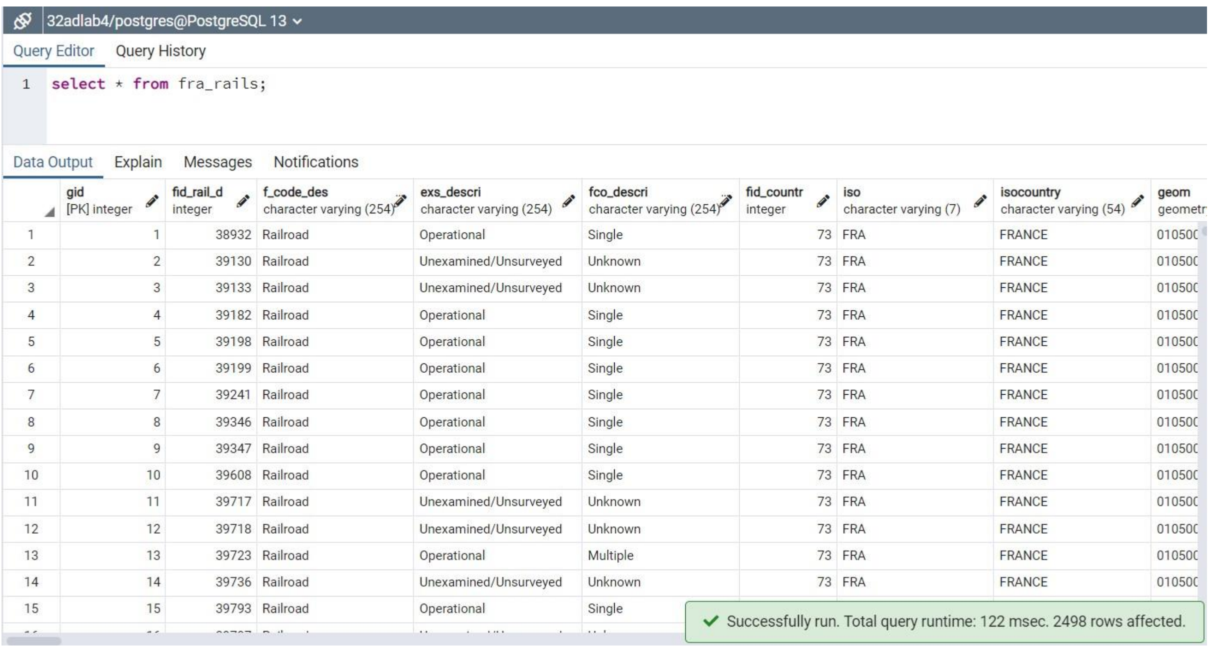
Query Editor | Query History

```sql
select gid, ST_AsText(geom) from fra_water_lines_dcw;
```

Data Output | Explain | Messages | Notifications

	gid [PK] integer	st_astext text
1	1	MULTILINESTRING((-1.895916637466731 47.009193373618594
2	2	MULTILINESTRING((-1.876805523654622 46.99819565215125,-
3	3	MULTILINESTRING((-1.982083253128302 46.98069387340753,-
4	4	MULTILINESTRING((-1.960277755201697 46.97697076224932,-
5	5	MULTILINESTRING((-1.918305578418128 46.99536140803082,-
6	6	MULTILINESTRING((-1.98355563190866 46.97797397429195,-
7	7	MULTILINESTRING((-1.90088884967802 46.965415868758306,-
8	8	MULTILINESTRING((-1.937055513214889 46.95555498233928,-
9	9	MULTILINESTRING((-1.960277755201697 46.97697076224932,-
10	10	MULTILINESTRING((-1.98458334523454 46.951946216185924,-
11	11	MULTILINESTRING((-1.980916556078724 46.95183187118107,-
12	12	MULTILINESTRING((-1.98458334523454 46.951946216185924,-
13	13	MULTILINESTRING((-1.96783321952276 46.843776786589366,-
14	14	MULTILINESTRING((-1.93163877298471 46.82963939698861,-1
15	15	MULTILINESTRING((-2.017361238627406 46.86375049643813,-

import railroads data via postgis: `select * from fra_rails;`



32adlab4/postgres@PostgreSQL 13

Query Editor | Query History

```sql
select * from fra_rails;
```

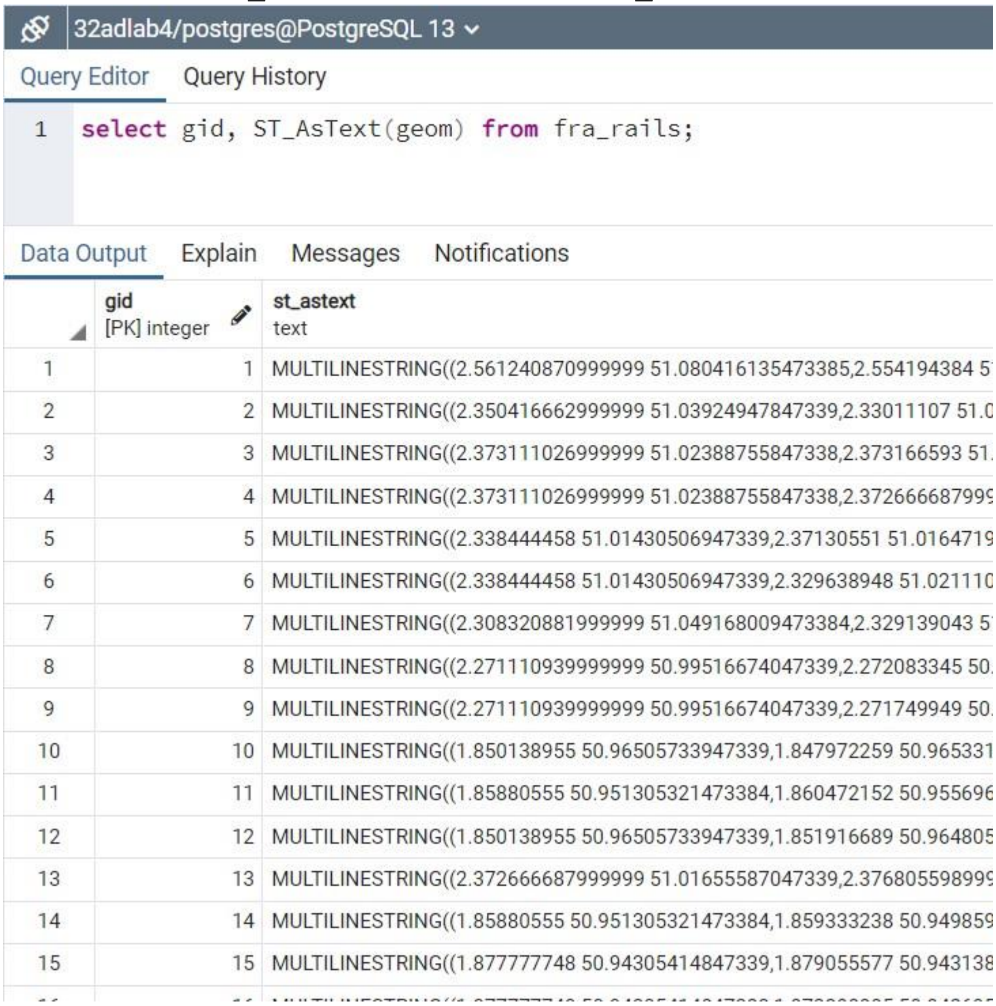
Data Output | Explain | Messages | Notifications

	gid [PK] integer	fid_rail_d integer	f_code_des character varying (254)	exs_descri character varying (254)	fco_descri character varying (254)	fid_countr integer	iso character varying (7)	isocountry character varying (54)	geom geometr
1	1	38932	Railroad	Operational	Single	73	FRA	FRANCE	01050
2	2	39130	Railroad	Unexamined/Unsurveyed	Unknown	73	FRA	FRANCE	01050
3	3	39133	Railroad	Unexamined/Unsurveyed	Unknown	73	FRA	FRANCE	01050
4	4	39182	Railroad	Operational	Single	73	FRA	FRANCE	01050
5	5	39198	Railroad	Operational	Single	73	FRA	FRANCE	01050
6	6	39199	Railroad	Operational	Single	73	FRA	FRANCE	01050
7	7	39241	Railroad	Operational	Single	73	FRA	FRANCE	01050
8	8	39346	Railroad	Operational	Single	73	FRA	FRANCE	01050
9	9	39347	Railroad	Operational	Single	73	FRA	FRANCE	01050
10	10	39608	Railroad	Operational	Single	73	FRA	FRANCE	01050
11	11	39717	Railroad	Unexamined/Unsurveyed	Unknown	73	FRA	FRANCE	01050
12	12	39718	Railroad	Unexamined/Unsurveyed	Unknown	73	FRA	FRANCE	01050
13	13	39723	Railroad	Operational	Multiple	73	FRA	FRANCE	01050
14	14	39736	Railroad	Unexamined/Unsurveyed	Unknown	73	FRA	FRANCE	01050
15	15	39793	Railroad	Operational	Single				

✔ Successfully run. Total query runtime: 122 msec. 2498 rows affected.

Add geometry:

```
select gid, ST_AsText(geom) from fra_rails;
```


32adlab4/postgres@PostgreSQL 13

Query Editor | Query History

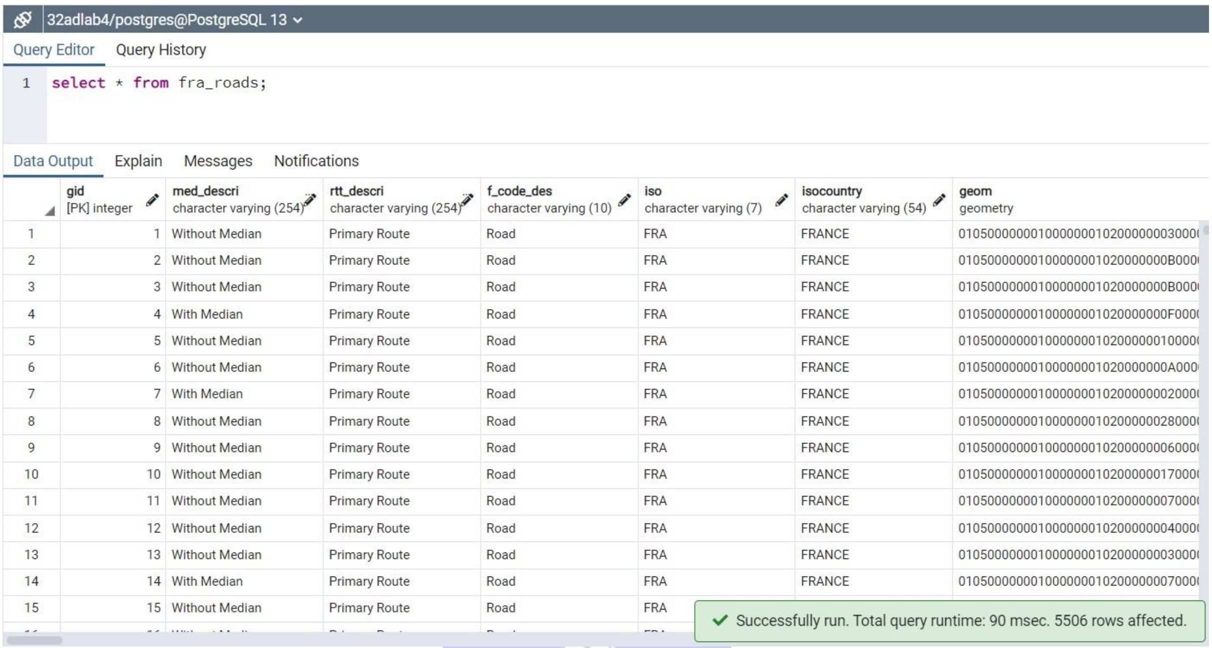
```
1  select gid, ST_AsText(geom) from fra_rails;
```

Data Output | Explain | Messages | Notifications

	gid [PK] integer	st_astext text
1	1	MULTILINESTRING((2.561240870999999 51.080416135473385,2.554194384 5
2	2	MULTILINESTRING((2.350416662999999 51.03924947847339,2.33011107 51.0
3	3	MULTILINESTRING((2.373111026999999 51.02388755847338,2.373166593 51.
4	4	MULTILINESTRING((2.373111026999999 51.02388755847338,2.372666687999
5	5	MULTILINESTRING((2.338444458 51.01430506947339,2.37130551 51.016471
6	6	MULTILINESTRING((2.338444458 51.01430506947339,2.329638948 51.02111
7	7	MULTILINESTRING((2.308320881999999 51.049168009473384,2.329139043 5
8	8	MULTILINESTRING((2.271110939999999 50.99516674047339,2.272083345 50.
9	9	MULTILINESTRING((2.271110939999999 50.99516674047339,2.271749949 50.
10	10	MULTILINESTRING((1.850138955 50.96505733947339,1.847972259 50.965331
11	11	MULTILINESTRING((1.85880555 50.951305321473384,1.860472152 50.955696
12	12	MULTILINESTRING((1.850138955 50.96505733947339,1.851916689 50.964805
13	13	MULTILINESTRING((2.372666687999999 51.01655587047339,2.376805598999
14	14	MULTILINESTRING((1.85880555 50.951305321473384,1.859333238 50.949859
15	15	MULTILINESTRING((1.877777748 50.94305414847339,1.879055577 50.943138

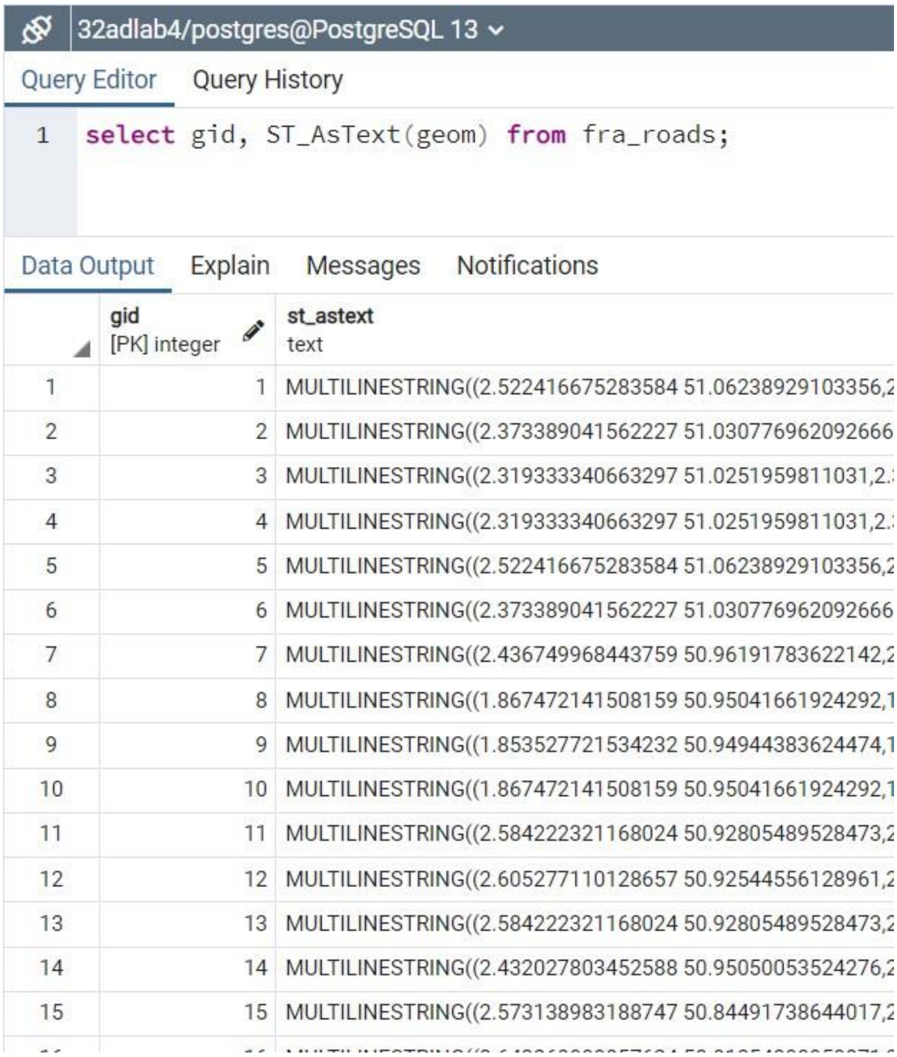
Import roads data via postgis:

```
select * from fra_roads;
```



	gid [PK] integer	med_descri character varying (254)	rtt_descri character varying (254)	f_code_des character varying (10)	iso character varying (7)	isocountry character varying (54)	geom geometry
1	1	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000030000
2	2	Without Median	Primary Route	Road	FRA	FRANCE	01050000000100000001020000000B000
3	3	Without Median	Primary Route	Road	FRA	FRANCE	01050000000100000001020000000B000
4	4	With Median	Primary Route	Road	FRA	FRANCE	01050000000100000001020000000F000
5	5	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000100000
6	6	Without Median	Primary Route	Road	FRA	FRANCE	01050000000100000001020000000A000
7	7	With Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000020000
8	8	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000280000
9	9	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000060000
10	10	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000170000
11	11	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000070000
12	12	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000040000
13	13	Without Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000030000
14	14	With Median	Primary Route	Road	FRA	FRANCE	0105000000010000000102000000070000
15	15	Without Median	Primary Route	Road	FRA		

Add geometry: `select gid, ST_AsText(geom) from fra_roads;`

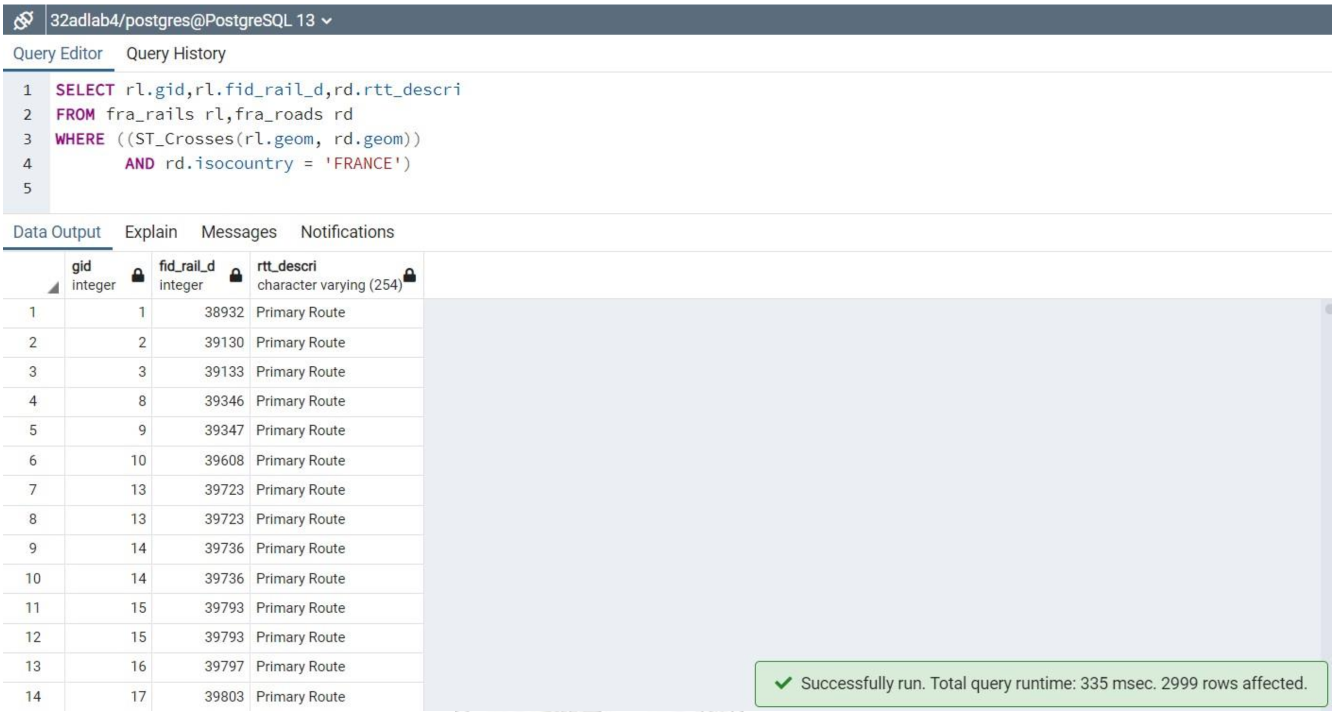


	gid [PK] integer	st_astext text
1	1	MULTILINESTRING((2.522416675283584 51.06238929103356,2
2	2	MULTILINESTRING((2.373389041562227 51.030776962092666
3	3	MULTILINESTRING((2.319333340663297 51.0251959811031,2.:
4	4	MULTILINESTRING((2.319333340663297 51.0251959811031,2.:
5	5	MULTILINESTRING((2.522416675283584 51.06238929103356,2
6	6	MULTILINESTRING((2.373389041562227 51.030776962092666
7	7	MULTILINESTRING((2.436749968443759 50.96191783622142,2
8	8	MULTILINESTRING((1.867472141508159 50.95041661924292,1
9	9	MULTILINESTRING((1.853527721534232 50.94944383624474,1
10	10	MULTILINESTRING((1.867472141508159 50.95041661924292,1
11	11	MULTILINESTRING((2.584222321168024 50.92805489528473,2
12	12	MULTILINESTRING((2.605277110128657 50.92544556128961,2
13	13	MULTILINESTRING((2.584222321168024 50.92805489528473,2
14	14	MULTILINESTRING((2.432027803452588 50.95050053524276,2
15	15	MULTILINESTRING((2.573138983188747 50.84491738644017,2

SPACIAL QUERIES:

1. Find railrods id no. that cross roads in FRANCE:

```
SELECT rl.gid,rl.fid_rail_d,rd.rtt_descri
FROM fra_rails rl,fra_roads rd
WHERE ((ST_Crosses(rl.geom, rd.geom))
AND rd.isocountry = 'FRANCE')
```

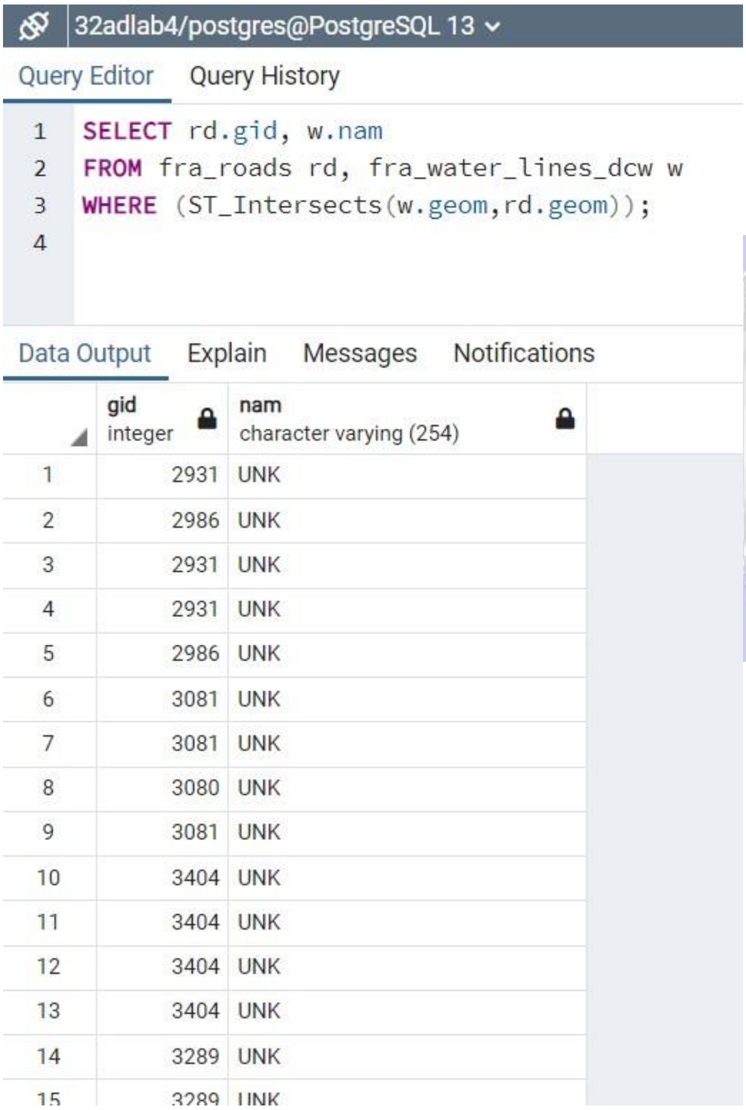



```sql
SELECT rl.gid,rl.fid_rail_d,rd.rtt_descri
FROM fra_rails rl,fra_roads rd
WHERE ((ST_Crosses(rl.geom, rd.geom))
       AND rd.isocountry = 'FRANCE')
```

	gid integer	fid_rail_d integer	rtt_descri character varying (254)
1	1	38932	Primary Route
2	2	39130	Primary Route
3	3	39133	Primary Route
4	8	39346	Primary Route
5	9	39347	Primary Route
6	10	39608	Primary Route
7	13	39723	Primary Route
8	13	39723	Primary Route
9	14	39736	Primary Route
10	14	39736	Primary Route
11	15	39793	Primary Route
12	15	39793	Primary Route
13	16	39797	Primary Route
14	17	39803	Primary Route

2. Find road id no. and river name that intersect at a geographical point:

```
SELECT rd.gid, w.nam
FROM fra_roads rd, fra_water_lines_dcw w
WHERE (ST_Intersects(w.geom,rd.geom));
```

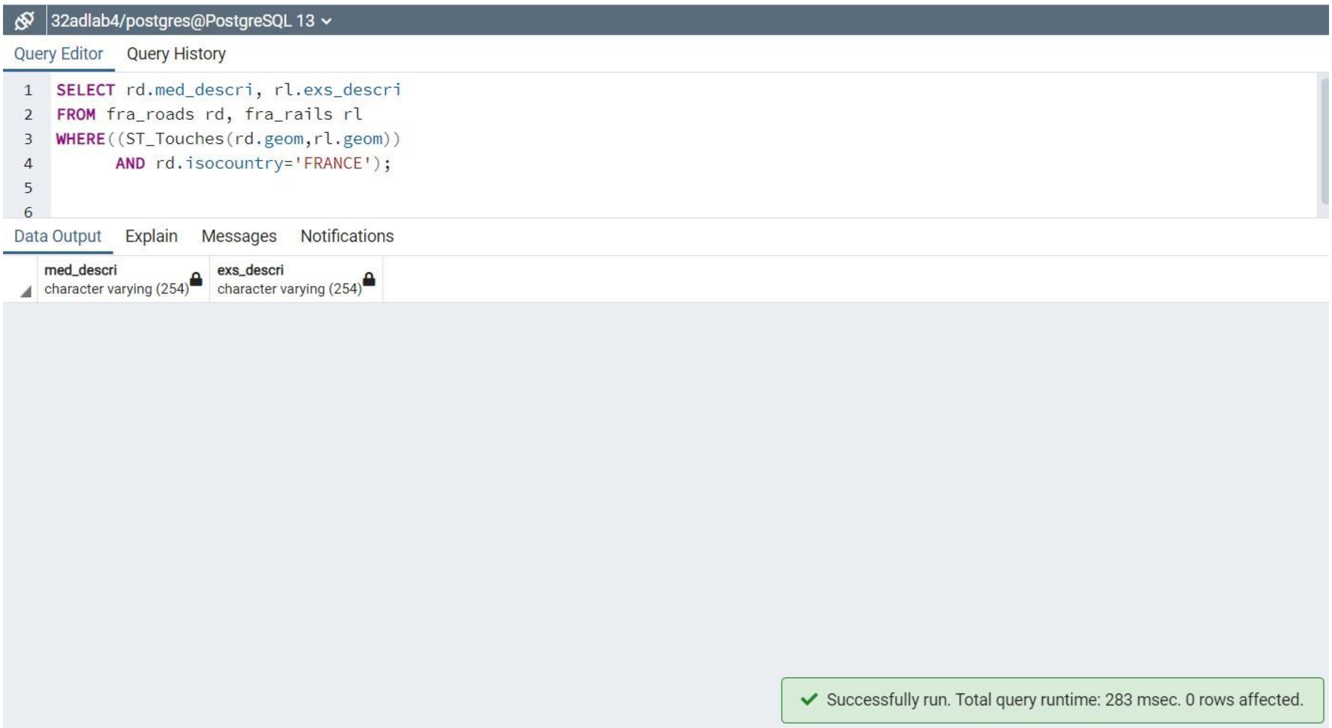


	gid integer	nam character varying (254)
1	2931	UNK
2	2986	UNK
3	2931	UNK
4	2931	UNK
5	2986	UNK
6	3081	UNK
7	3081	UNK
8	3080	UNK
9	3081	UNK
10	3404	UNK
11	3404	UNK
12	3404	UNK
13	3404	UNK
14	3289	UNK
15	3289	UNK

3. Find roads and railroads in FRANCE that touch each other:

```
SELECT rd.med_descri, rl.exs_descri  FROM
fra_roads rd, fra_rails rl
```

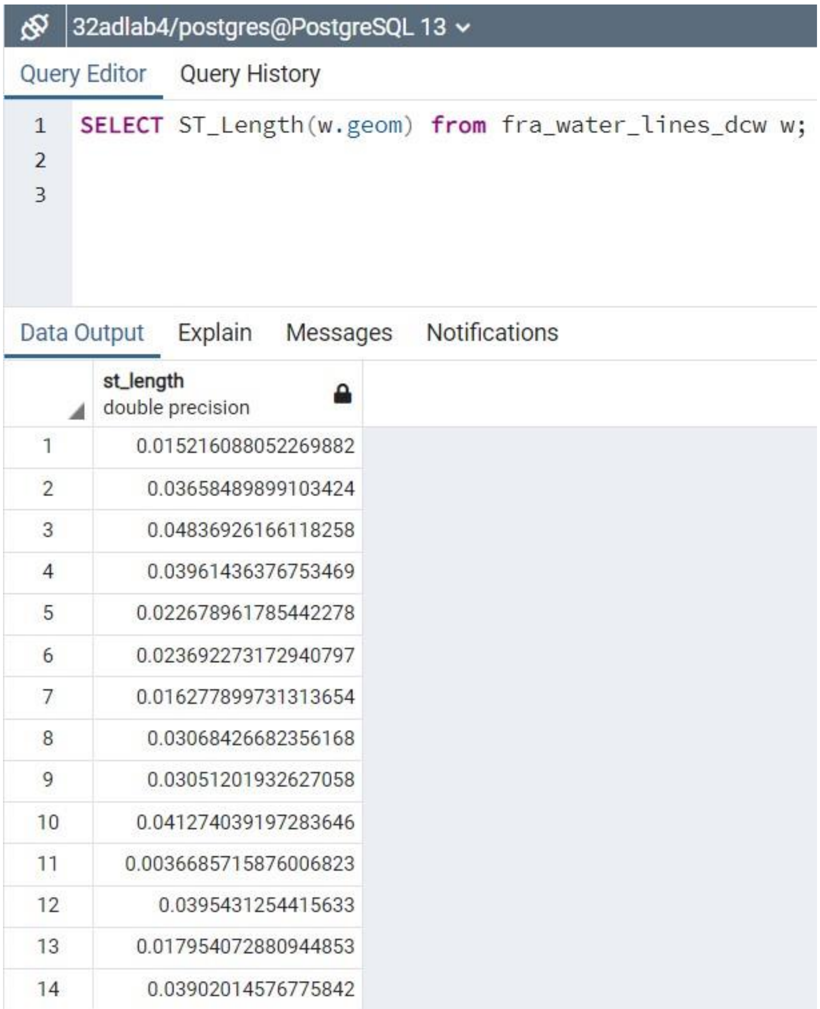
```
WHERE((ST_Touches(rd.geom,rl.geom))
AND rd.isocountry='FRANCE');
```



Measurement Functions:

1. ST_Length

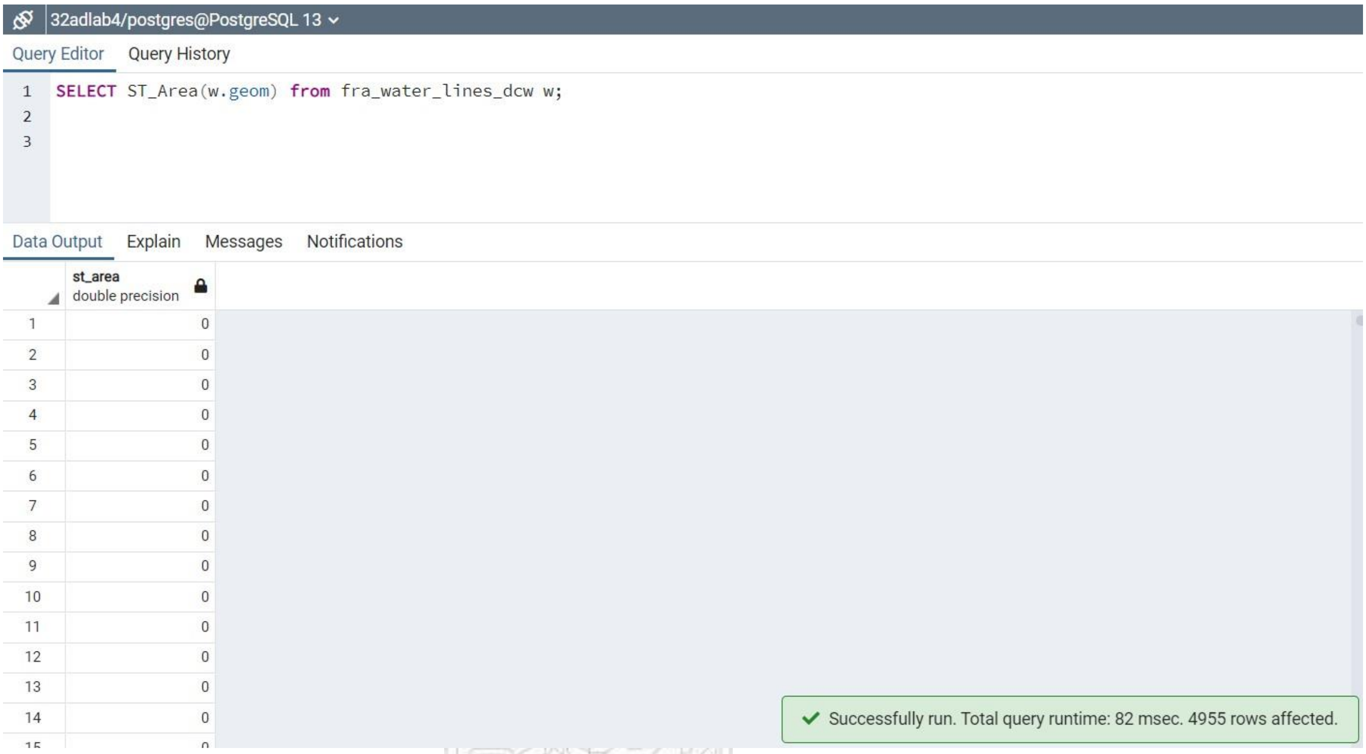
```
SELECT ST_Length(w.geom) from fra_water_lines_dcw w;
```



	st_length double precision
1	0.015216088052269882
2	0.03658489899103424
3	0.04836926166118258
4	0.03961436376753469
5	0.022678961785442278
6	0.023692273172940797
7	0.016277899731313654
8	0.03068426682356168
9	0.03051201932627058
10	0.041274039197283646
11	0.0036685715876006823
12	0.0395431254415633
13	0.017954072880944853
14	0.03902014576775842

2. ST_Perimeter

```
SELECT ST_Area(w.geom) from fra_water_lines_dcw w;
```



Questions:

1. Explain the chosen query along with the spatial components in detail.

____ **Ans:**

ST_Distance — for geometry type Returns the 2D Cartesian distance between two geometries in projected units (based on spatial ref). For geography type defaults to return minimum geodesic distance between two geographies in meters. float ST_Distance(geometry g1, geometry g2);

ST_MakePoint — creates a 2D, 3DZ or 4D point geometry.

ST_CoveredBy — Returns 1 (TRUE) if no point in Geometry/Geography A is outside Geometry/Geography B

ST_Within — Returns true if the geometry A is completely inside geometry B

ST_Crosses — Returns TRUE if the supplied geometries have some, but not all, interior points in common.

ST_GeomFromText method constructs geometry from a character string representation. Collapse/expand section Syntax

ST_Geometry::ST_GeomFromText(character-string[, srid])

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 have studied and executed queries on Spatial Database.

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. *https://www.techopedia.com/definition/17287/spatial-database*
3. *https://postgis.net/docs/using_postgis_dbmanagement.html*
4. *https://postgis.net/docs/using_postgis_query.html*