SGD LAB EXP – 8

Name : Aditi Chhajed; **Reg. No.** : 221081009
Branch : IT ; **Course Instructor** : Prof. Vedashree Awati

Aim:

Spatial Measurement Functions.

Theory:

1. ST_Area(geometry)

- *Calculates the area of a polygonal geometry in the spatial reference system of the geometry.*
- *Example: `SELECT ST_Area(geom) FROM parks WHERE name = 'Yellowstone National Park';`*

2. ST_Centroid(geometry)

- *Returns the centroid (geometric center) of a geometry, which is the average location of all points in the geometry.*
- *Example: `SELECT ST_Centroid(geom) FROM cities WHERE name = 'Los Angeles';`*

3. ST_Distance(geometry, geometry)

- *Calculates the minimum distance between two geometries.*
- *Example: `SELECT ST_Distance(geom1, geom2) FROM roads, landmarks WHERE roads.name = 'Route 66' AND landmarks.name = 'Grand Canyon';`*

4. ST_Distance_Spheroid(geometry, geometry, spheroid)

- *Computes the distance between two geometries on the surface of a spheroid (ellipsoid), which accounts for the Earth's curvature.*
- *Example: `SELECT ST_Distance_Spheroid(geom1, geom2, 'SPHEROID["WGS 84", 6378137, 298.257223563]') FROM cities WHERE name = 'New York' AND name = 'Los Angeles';`*

5. ST_Distance_Sphere(geometry, geometry)

- *Calculates the minimum distance between two geometries on the Earth's surface using a spherical model, considering the curvature of the Earth.*
- *Example: `SELECT ST_Distance_Sphere(geom1, geom2) FROM cities WHERE name = 'New York' AND name = 'Los Angeles';`*

6. ST_Length(geometry)

- *Computes the total length of a linestring geometry.*
- *Example: `SELECT ST_Length(geom) FROM roads WHERE name = 'Route 66';`*

7. ST_Length_Spheroid(geometry, spheroid)

- *Measures the length of a linestring geometry on a spheroid, accounting for the Earth's curvature.*
- *Example: `SELECT ST_Length_Spheroid(geom, 'SPHEROID["GRS 1980",6378137,298.257222101]') FROM roads WHERE name = 'Route 66';`*

8. ST_Length3D(geometry)

- *Calculates the 3D length of a linestring geometry, considering the Z-dimension (height).*
- *Example: `SELECT ST_Length3D(geom) FROM mountain_trails WHERE name = 'Everest Base Camp Trail';`*

9. ST_Length3D_Spheroid(geometry, spheroid)

- *Computes the 3D length of a linestring geometry on the surface of a spheroid, incorporating both the geometry's Z-dimension and the Earth's curvature.*
- *Example: `SELECT ST_Length3D_Spheroid(geom, 'SPHEROID["GRS 1980",6378137,298.257223563]') FROM mountain_trails WHERE name = 'Everest Base Camp Trail';`*

10. ST_Perimeter(geometry)

- *Calculates the perimeter of a polygonal geometry, which is the total length of the boundary.*
- *Example: `SELECT ST_Perimeter(geom) FROM parks WHERE name = 'Grand Canyon';`*

11. ST_Perimeter3D(geometry)

- *Computes the perimeter of a 3D polygon geometry, accounting for the Z-dimension.*
- *Example: `SELECT ST_Perimeter3D(geom) FROM 3d_parks WHERE name = 'Grand Canyon';`*

Implementation:

1. ***ST_Area():***

Query Query History

```sql
SELECT name, ST_Area(geom) AS area
FROM polys
WHERE name IN ('Central Park', 'Lake Tahoe');

--aditi chhajed 221081009
```

Data Output Messages Geometry Viewer × Notifications

	name character varying	area double precision
1	Central Park	0.0003487449999997511
2	Lake Tahoe	2.1344999999887033e-05

2. ***ST_Centroid():***

Query Query History

```sql
SELECT name, ST_AsText(ST_Centroid(geom)) AS centroid
FROM polys
WHERE name = 'Yellowstone National Park';

--aditi chhajed 221081009
```

Data Output Messages Geometry Viewer × Notifications

	name character varying	centroid text
1	Yellowstone National Park	POINT(-110.60569193529622 44.252043749186285)

3. ST_Distance():

Query | Query History

```sql
SELECT a.name AS point1, b.name AS point2, ST_Distance(a.geom, b.geom) AS distance
FROM pts a, pts b
WHERE a.name = 'New York' AND b.name = 'Los Angeles';

--aditi chhajed 221081009
```

Data Output | Messages | Geometry Viewer | Notifications

	point1 character varying	point2 character varying	distance double precision
1	New York	Los Angeles	44.736312919707636

4. ST_Distance_Spheroid() and ST_Distance_Sphere():

Query | Query History

```sql
SELECT cityA.name AS city1, cityB.name AS city2,
    ST_DistanceSphere(cityA.geom, cityB.geom) / 1000 AS distance_km_sphere,
    ST_DistanceSpheroid(cityA.geom, cityB.geom,
    'SPHEROID["GRS 1980",6378137,298.257222101]') / 1000 AS distance_km_spheroid
FROM pts AS cityA CROSS JOIN pts AS cityB
WHERE cityA.name = 'Los Angeles' AND cityB.name = 'New York';

--aditi chhajed 221081009
```

Data Output | Messages | Geometry Viewer | Notifications

	city1 character varying	city2 character varying	distance_km_sphere double precision	distance_km_spheroid double precision
1	Los Angeles	New York	3935.75167321378	3944.422231510842

5. ST_Length():

Query | Query History

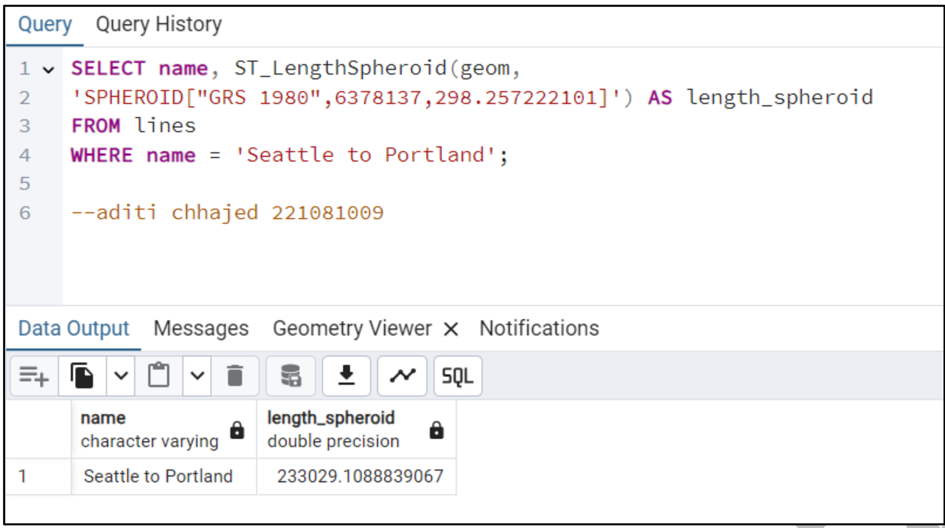
```sql
SELECT name, ST_Length(geom) AS length
FROM lines
WHERE name = 'Route 66 Chicago to Santa Monica';

--aditi chhajed 221081009
```

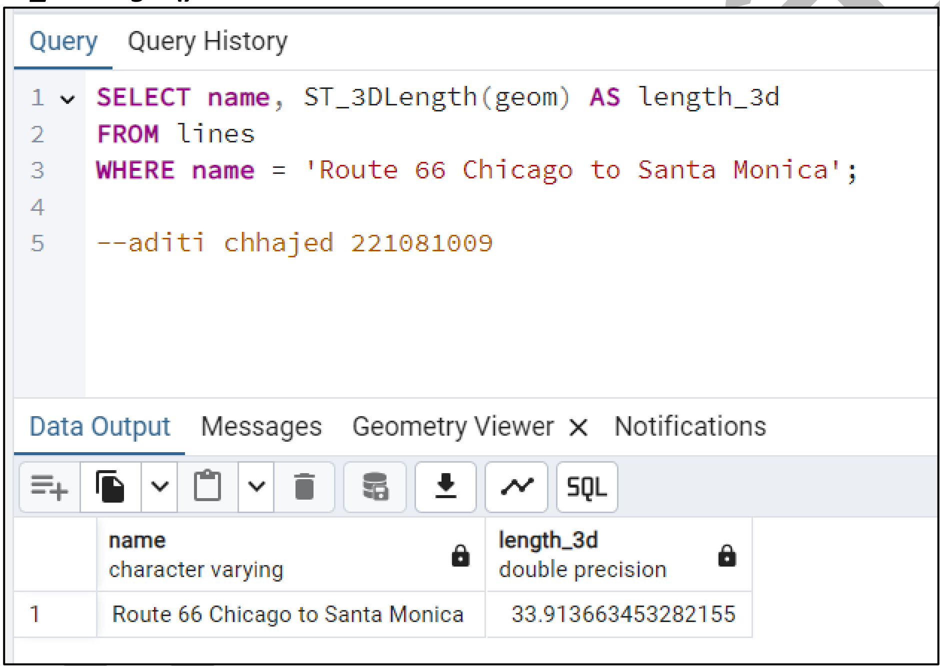
Data Output | Messages | Geometry Viewer | Notifications

	name character varying	length double precision
1	Route 66 Chicago to Santa Monica	33.913663453282155

6. *ST_LengthSpheroid():*



7. *ST_3DLength():*



8. *ST_Length3D_Spheroid():*

- *PostGIS offers several spatial functions for distance and length calculations, but not every theoretical function (like ST_Length3D_Spheroid()) is supported. Instead, ST_DistanceSpheroid() and transformations to geography types are typically used for high-accuracy calculations involving Earth's curvature.*

9. ST_Perimeter():

Query Query History

```sql
SELECT name, ST_Perimeter(geom) AS perimeter
FROM polys
WHERE name = 'Grand Canyon National Park';

--aditi chhajed 221081009
```

Data Output Messages Geometry Viewer × Notifications

	name character varying	perimeter double precision
1	Grand Canyon National Park	0.2245121715478275

10. ST_3DPerimeter():

```sql
SELECT name, ST_3DPerimeter(geom) AS perimeter_3d
FROM polys
WHERE name = 'Grand Canyon National Park';

--aditi chhajed 221081009
```

Data Output Messages Geometry Viewer × Notifications

	name character varying	perimeter_3d double precision
1	Grand Canyon National Park	0.2245121715478275

Conclusion:

In summary, using PostGIS spatial relationship functions allowed me to analyze and understand how different geographic features relate to one another.

By using functions like `ST_Contains()` and `ST_Within()`, I can determine whether a specific geometry lies entirely within another.

With functions like `ST_Intersects()` or `ST_Overlaps()`, I can identify shared spaces or overlapping areas.

These tools helped me explore boundaries, distances, adjacency, and coverage between geographic entities.

Whether I need to check if a point is inside a polygon, find features within a specific distance, or identify touching boundaries, these functions provide me with powerful ways to analyze and interact with spatial data for insightful results and meaningful spatial relationships.