DATABASE MANAGEMENT SYSTEMS

SPRING 2023

CS – 632 PROJECT

Professor: Mr. Buchi Okoli Okoli

Project Members

Kaleeswaran Sivasankaran, James Helloween, Ravi Shankar Dhwarakesh

GOAL

Our goal is to provide the Geographic Information System (GIS) analysis, using database that supports spatial data types which is PostGIS.

DATASOURCE: https://github.com/opengeos/data

In the above git hub link, the Geographic Information of US, China, World Cities and other spatial data will be available from that we will be using US data source in this analysis.

Data Source of US: In this data source 3 different tables will be created for states, cities and counties.

Importing the Data: Data will be imported using PostGIS share file import/export manager, which is a tool that allows us to import shapefiles from a specified path.

After adding the files in the UI alter the SRID to 4326 since it represents spatial data using longitude and latitude coordinates on the Earth's surface and import the data.

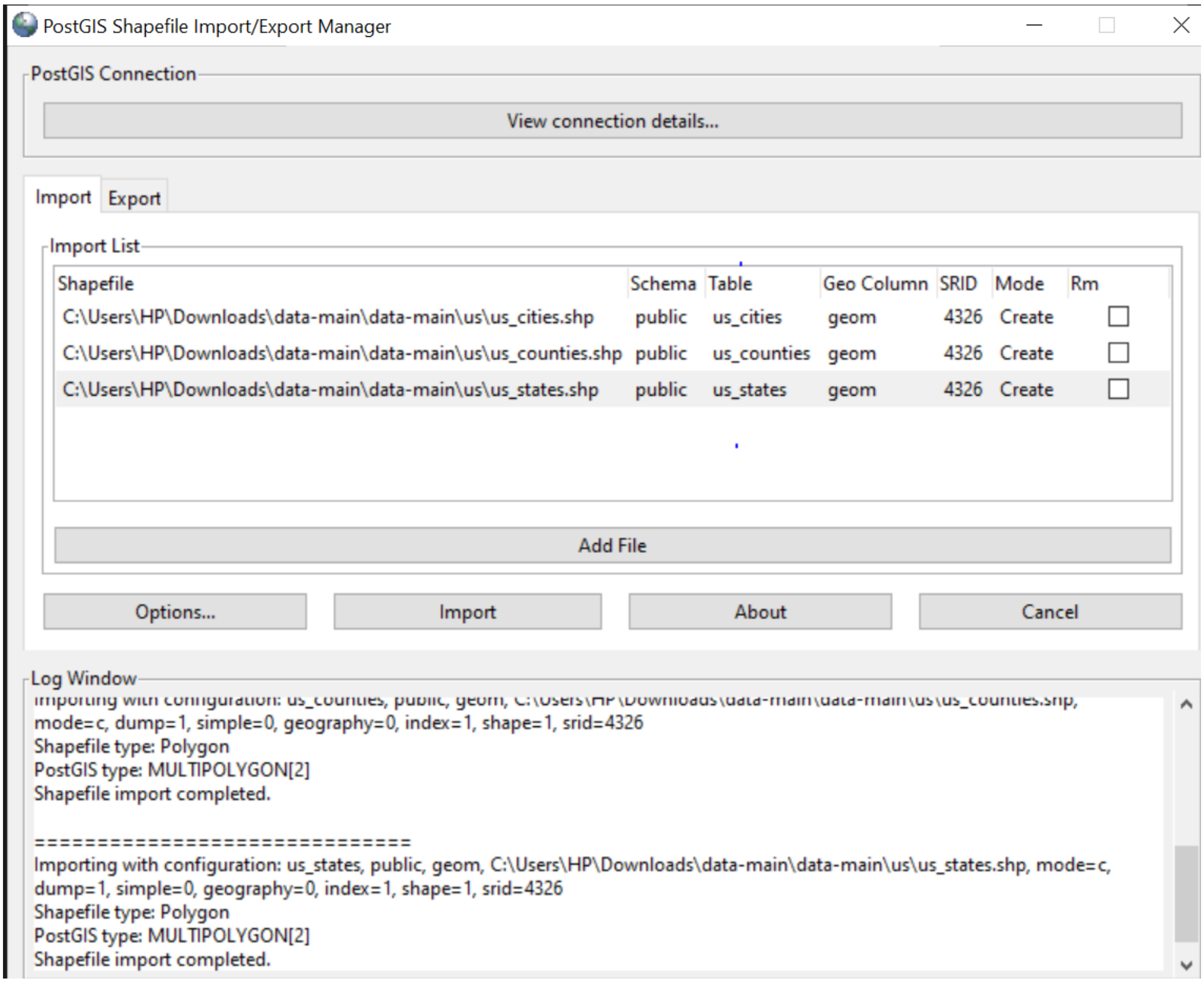


Table Schema of us_cities, us_states and us_counties.

Query Query History

```sql
SELECT column_name, data_type, is_nullable
FROM information_schema.columns
WHERE table_schema = 'public' AND table_name = 'us_cities';
```

Data Output Messages Notifications

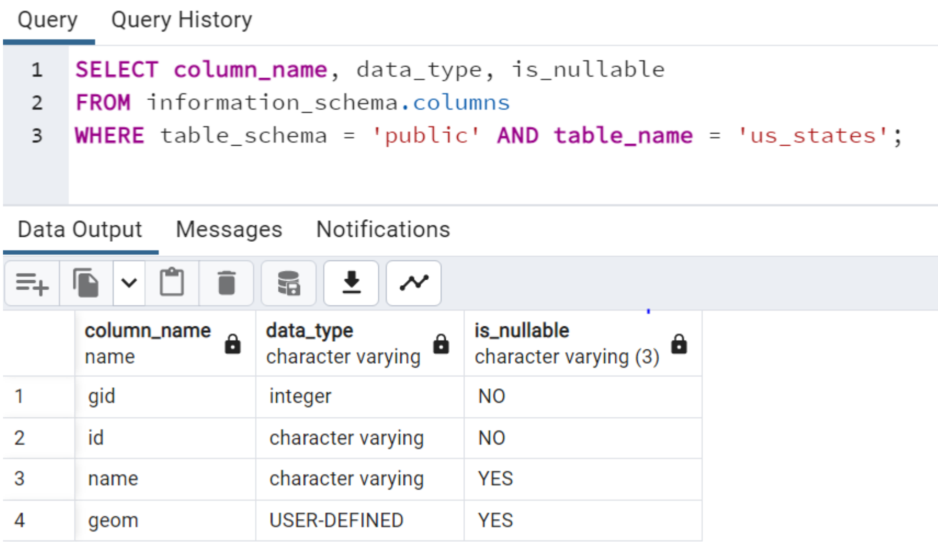
	column_name name	data_type character varying	is_nullable character varying (3)
1	gid	integer	NO
2	id	character varying	YES
3	pop_2010	numeric	YES
4	elev_in_ft	numeric	YES
5	state	character varying	YES
6	geom	USER-DEFINED	YES

Query Query History

```sql
SELECT column_name, data_type, is_nullable
FROM information_schema.columns
WHERE table_schema = 'public' AND table_name = 'us_counties';
```

Data Output Messages Notifications

	column_name name	data_type character varying	is_nullable character varying (3)
1	gid	integer	NO
2	geo_id	character varying	YES
3	state	character varying	YES
4	county	character varying	YES
5	name	character varying	YES
6	lsad	character varying	YES
7	censusarea	numeric	YES
8	geom	USER-DEFINED	YES



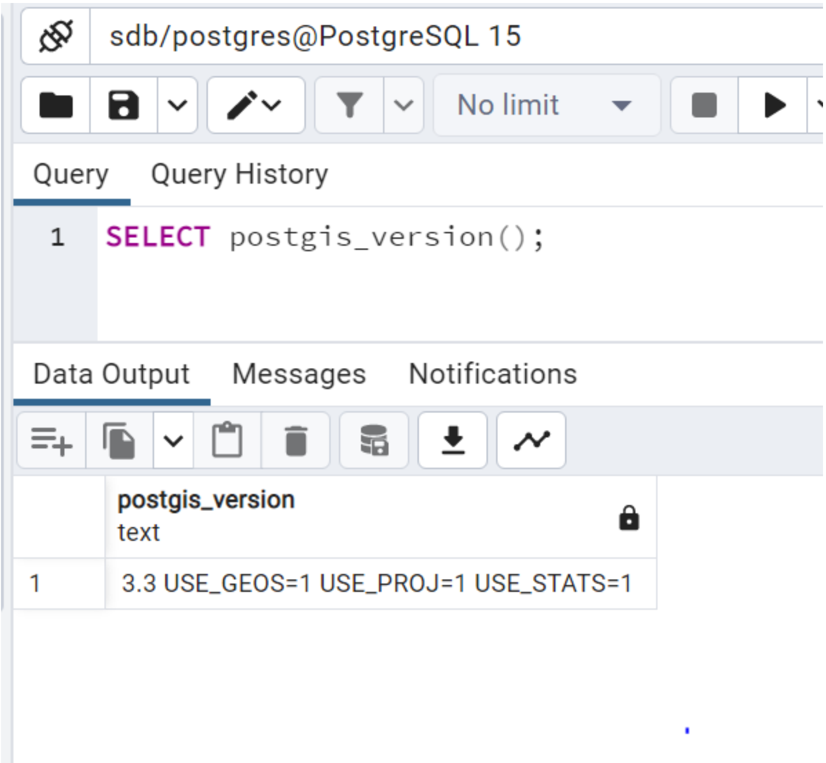
Add and Verify PostGIS extension

Query to add the postgis extenstion

CREATE EXTENSION postgis;

Query to verify the existance

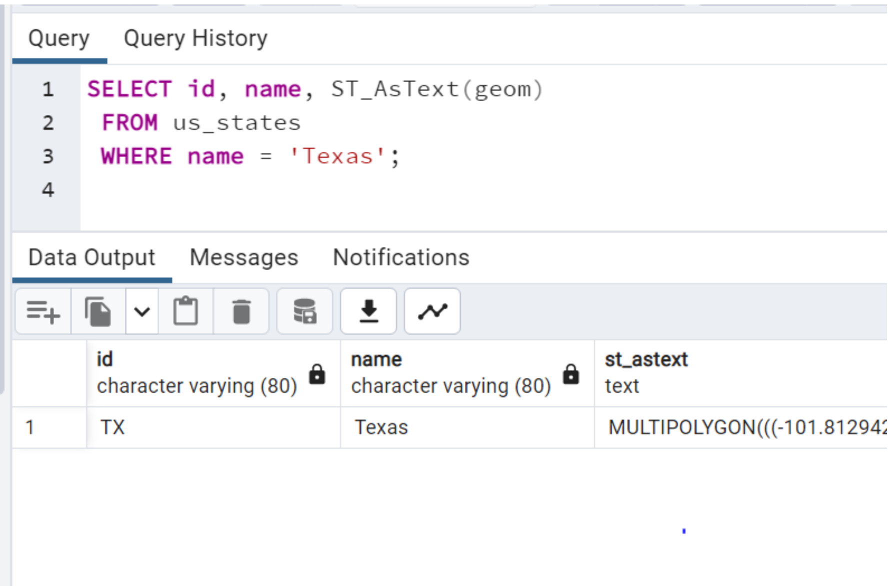
SELECT postgis_version();



Retrieve Locations of specific features

```
SELECT id, name, ST_AsText(geom)
 FROM us_states
 WHERE name = 'Texas';
```

ST_AsText is a method in PostGIS, an extension to the PostgreSQL database system, that returns the Well-Known Text (WKT) representation of a geometry or geography instance

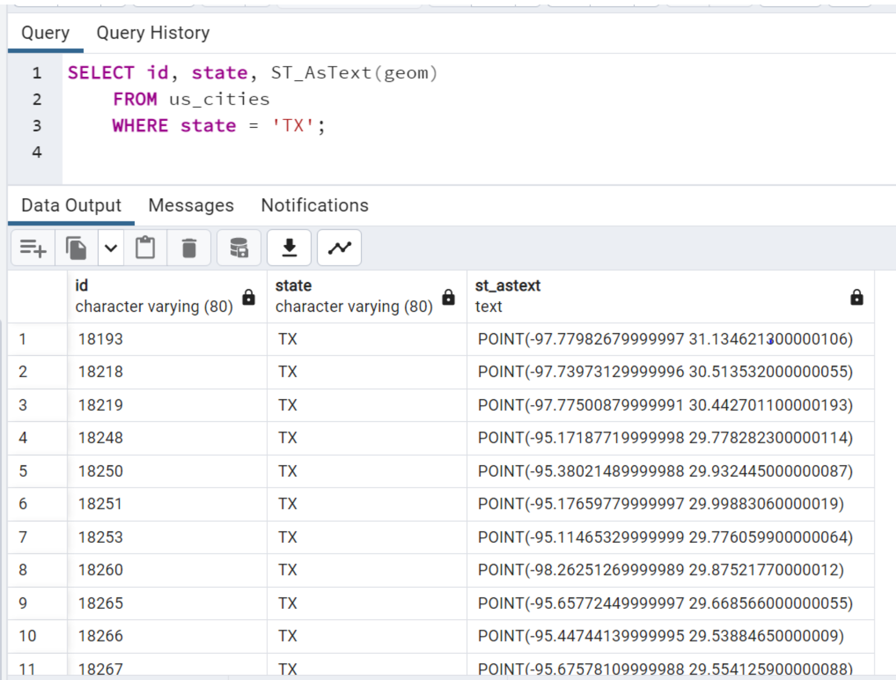

Query Query History

```
SELECT id, name, ST_AsText(geom)
 FROM us_states
 WHERE name = 'Texas';
```

Data Output Messages Notifications

	id character varying (80)	name character varying (80)	st_astext text
1	TX	Texas	MULTIPOLYGON(((-101.812942

```
SELECT id, state, ST_AsText(geom)
        FROM us_cities
        WHERE state = 'TX';
```


Query Query History

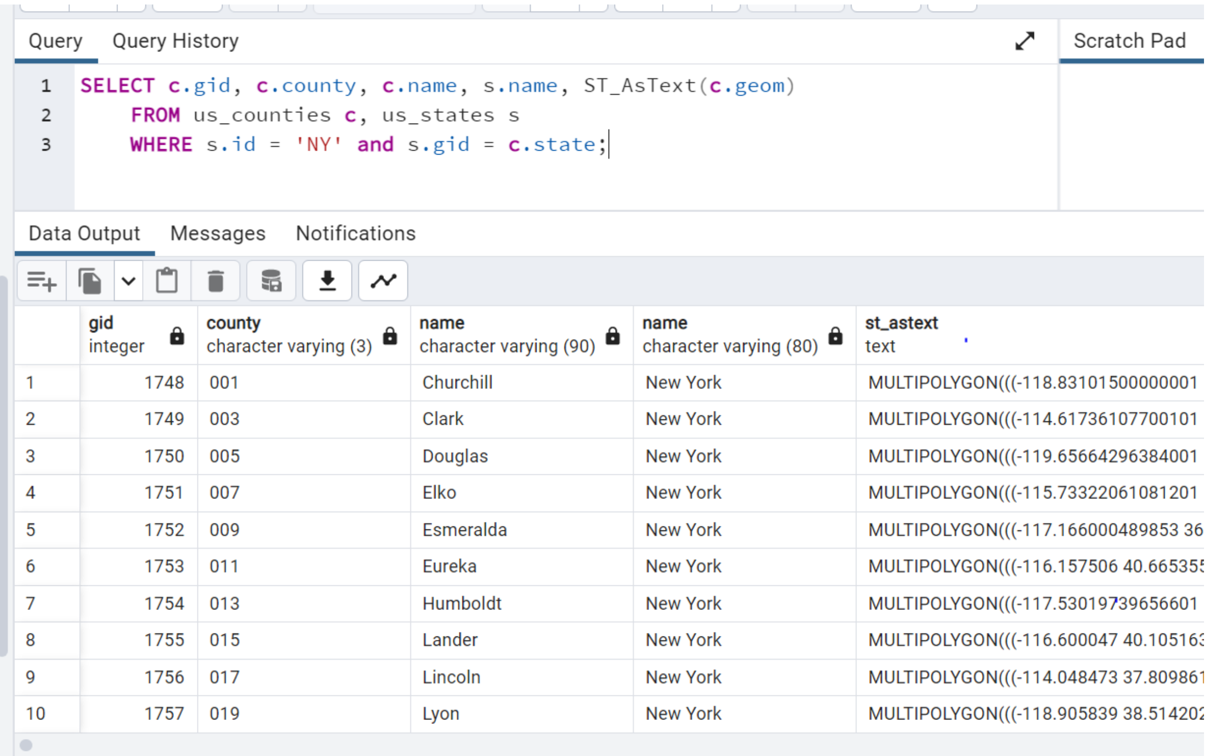
```
SELECT id, state, ST_AsText(geom)
    FROM us_cities
    WHERE state = 'TX';
```

Data Output Messages Notifications

	id character varying (80)	state character varying (80)	st_astext text
1	18193	TX	POINT(-97.77982679999997 31.134621300000106)
2	18218	TX	POINT(-97.73973129999996 30.513532000000055)
3	18219	TX	POINT(-97.77500879999991 30.442701100000193)
4	18248	TX	POINT(-95.17187719999998 29.778282300000114)
5	18250	TX	POINT(-95.38021489999988 29.932445000000087)
6	18251	TX	POINT(-95.17659779999997 29.99883060000019)
7	18253	TX	POINT(-95.11465329999999 29.776059900000064)
8	18260	TX	POINT(-98.26251269999989 29.87521770000012)
9	18265	TX	POINT(-95.65772449999997 29.668566000000055)
10	18266	TX	POINT(-95.44744139999995 29.53884650000009)
11	18267	TX	POINT(-95.67578109999988 29.554125900000088)

SELECT c.gid, c.county, c.name, s.name, ST_AsText(c.geom)

FROM us_counties c, us_states s
WHERE s.id = 'NY' and s.gid = c.state;



Query Query History Scratch Pad

```
SELECT c.gid, c.county, c.name, s.name, ST_AsText(c.geom)
    FROM us_counties c, us_states s
    WHERE s.id = 'NY' and s.gid = c.state;
```

Data Output Messages Notifications

	gid integer	county character varying (3)	name character varying (90)	name character varying (80)	st_astext text
1	1748	001	Churchill	New York	MULTIPOLYGON(((-118.83101500000001
2	1749	003	Clark	New York	MULTIPOLYGON(((-114.61736107700101
3	1750	005	Douglas	New York	MULTIPOLYGON(((-119.65664296384001
4	1751	007	Elko	New York	MULTIPOLYGON(((-115.73322061081201
5	1752	009	Esmeralda	New York	MULTIPOLYGON(((-117.166000489853 36
6	1753	011	Eureka	New York	MULTIPOLYGON(((-116.157506 40.66535
7	1754	013	Humboldt	New York	MULTIPOLYGON(((-117.53019739656601
8	1755	015	Lander	New York	MULTIPOLYGON(((-116.600047 40.10516
9	1756	017	Lincoln	New York	MULTIPOLYGON(((-114.048473 37.80986
10	1757	019	Lyon	New York	MULTIPOLYGON(((-118.905839 38.51420

Query selects the names of US states from a table us_states where the geometry of the state intersects with a polygon defined in the coordinates.

SELECT name FROM us_states WHERE ST_Intersects(geom, ST_GeomFromText(
'POLYGON((-106.65 25.84, -106.65 36.50, -93.51 36.50, -93.51 25.84, -106.65 25.84))', 4326));

Query Query History

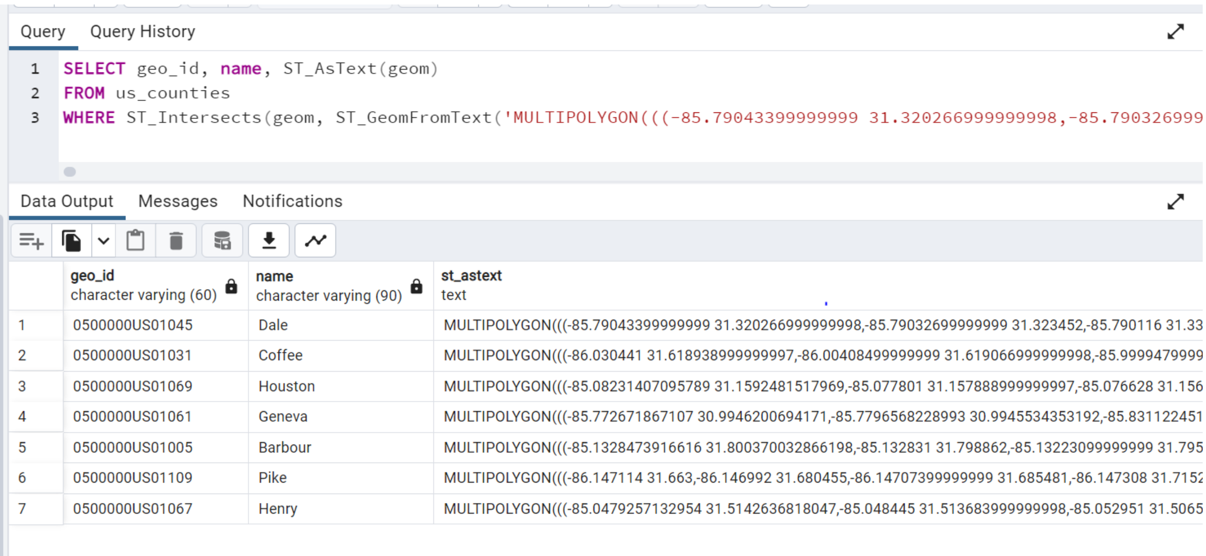
```
SELECT name
FROM us_states
WHERE ST_Intersects(geom, ST_GeomFromText(
    'POLYGON((-106.65 25.84, -106.65 36.50, -93.51 36.50, -93.51 25.84, -106.65 25.84))', 4326));
```

Data Output Messages Notifications

	name character varying (80)
1	Texas
2	Louisiana
3	Arkansas
4	Oklahoma
5	New Mexico

Query gives each counties in us_counties table where the geometry of the county intersects with the MultiPolygon defined by the given coordinates.

SELECT geo_id, name, ST_AsText(geom)
FROM us_counties
WHERE ST_Intersects(geom, ST_GeomFromText('MULTIPOLYGON(((-85.79043399999999 31.320266999999998,-85.79032699999999 31.323452,-85.790116 31.330081999999997,-85.79010000000001 31.336275999999998……)))', 4326));



Query | Query History

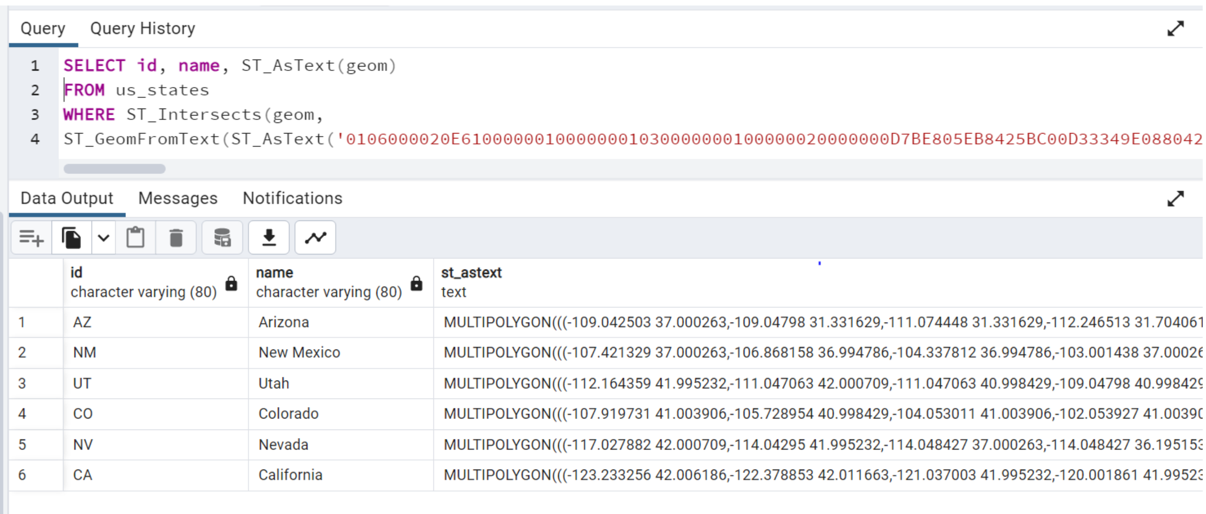
```
SELECT geo_id, name, ST_AsText(geom)
FROM us_counties
WHERE ST_Intersects(geom, ST_GeomFromText('MULTIPOLYGON(((-85.79043399999999 31.320266999999998,-85.790326999
```

Data Output | Messages | Notifications

	geo_id character varying (60)	name character varying (90)	st_astext text
1	0500000US01045	Dale	MULTIPOLYGON(((-85.79043399999999 31.320266999999998,-85.79032699999999 31.323452,-85.790116 31.33
2	0500000US01031	Coffee	MULTIPOLYGON(((-86.030441 31.618938999999997,-86.00408499999999 31.619066999999998,-85.9999479999
3	0500000US01069	Houston	MULTIPOLYGON(((-85.08231407095789 31.1592481517969,-85.077801 31.157888999999997,-85.076628 31.156
4	0500000US01061	Geneva	MULTIPOLYGON(((-85.772671867107 30.9946200694171,-85.7796568228993 30.9945534353192,-85.831122451
5	0500000US01005	Barbour	MULTIPOLYGON(((-85.1328473916616 31.800370032866198,-85.132831 31.798862,-85.13223099999999 31.795
6	0500000US01109	Pike	MULTIPOLYGON(((-86.147114 31.663,-86.146992 31.680455,-86.14707399999999 31.685481,-86.147308 31.715
7	0500000US01067	Henry	MULTIPOLYGON(((-85.0479257132954 31.5142636818047,-85.048445 31.513683999999998,-85.052951 31.5065

Query gives rows in the us_states table where the geometry of the state intersects with a geometry defined by the given Well-Known Binary (WKB) representation.

SELECT id, name, ST_AsText(geom)
FROM us_states
WHERE ST_Intersects(geom,
ST_GeomFromText(ST_AsText('0106000020E61000000100000001030000000 1...'), 4326));



Query | Query History

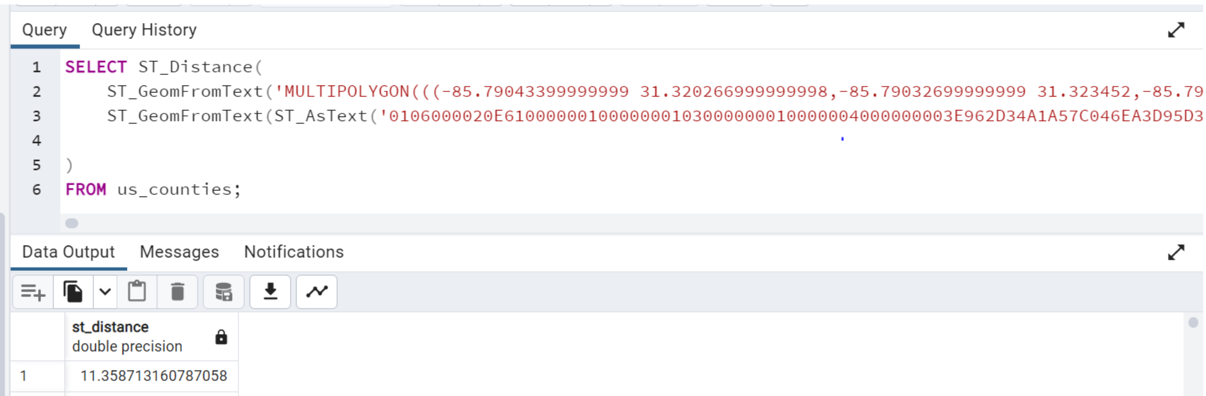
```
SELECT id, name, ST_AsText(geom)
FROM us_states
WHERE ST_Intersects(geom,
ST_GeomFromText(ST_AsText('0106000020E6100000010000000103000000010000002000000D7BE805EB8425BC00D33349E088042
```

Data Output | Messages | Notifications

	id character varying (80)	name character varying (80)	st_astext text
1	AZ	Arizona	MULTIPOLYGON(((-109.042503 37.000263,-109.04798 31.331629,-111.074448 31.331629,-112.246513 31.704061
2	NM	New Mexico	MULTIPOLYGON(((-107.421329 37.000263,-106.868158 36.994786,-104.337812 36.994786,-103.001438 37.0002
3	UT	Utah	MULTIPOLYGON(((-112.164359 41.995232,-111.047063 42.000709,-111.047063 40.998429,-109.04798 40.99842
4	CO	Colorado	MULTIPOLYGON(((-107.919731 41.003906,-105.728954 40.998429,-104.053011 41.003906,-102.053927 41.0039
5	NV	Nevada	MULTIPOLYGON(((-117.027882 42.000709,-114.04295 41.995232,-114.048427 37.000263,-114.048427 36.19515
6	CA	California	MULTIPOLYGON(((-123.233256 42.006186,-122.378853 42.011663,-121.037003 41.995232,-120.001861 41.9952

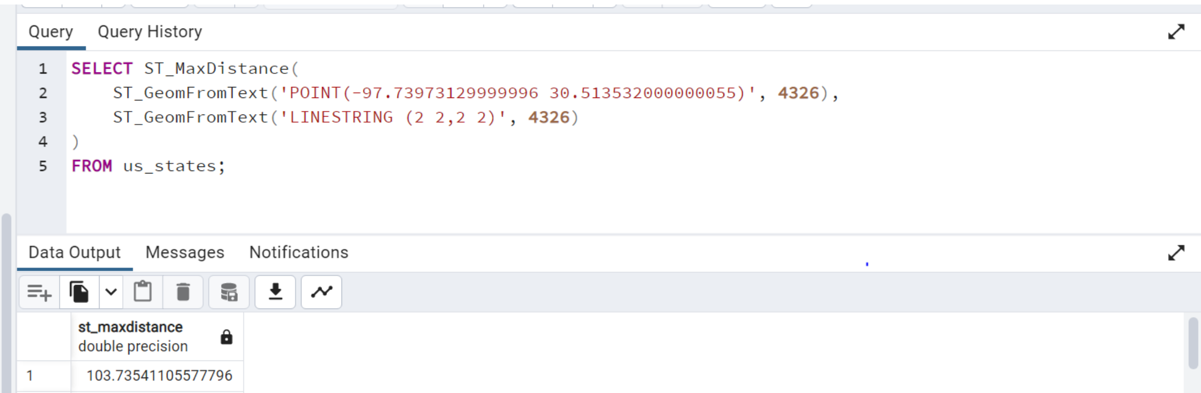
Query Calculates the distance between the two points.

SELECT ST_Distance(
 ST_GeomFromText('MULTIPOLYGON(((-85.79043399999999 31.320266999999998,-85.79032699999999 31.323452,-85.790116 31.330081999999997,-85.79010000000001 31.336275999999998,......)))', 4326),
 ST_GeomFromText(ST_AsText('0106000020E6100000010000000010300....'), 4326)
)
FROM us_counties;



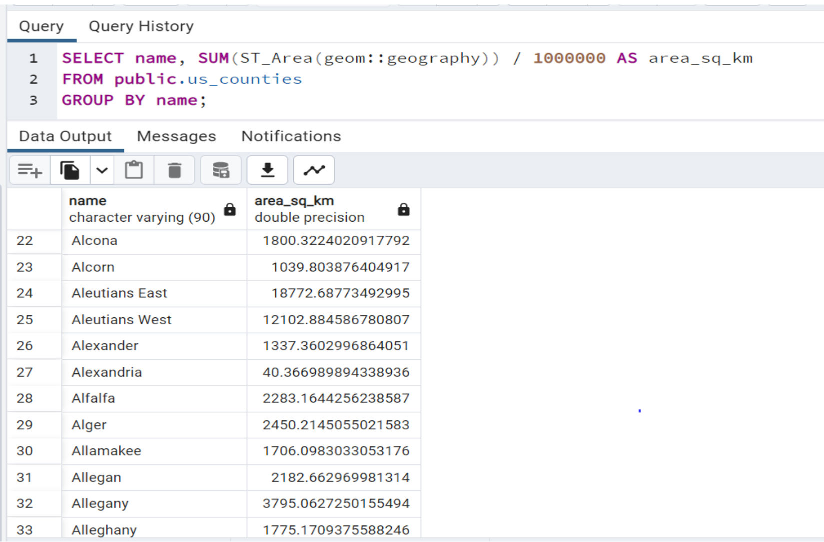
Query calculates the maximum distance between the points.

SELECT ST_MaxDistance(
 ST_GeomFromText('POINT(-97.73973129999996 30.513532000000055)', 4326),
 ST_GeomFromText('LINESTRING (2 2,2 2)', 4326)
)
FROM us_states;



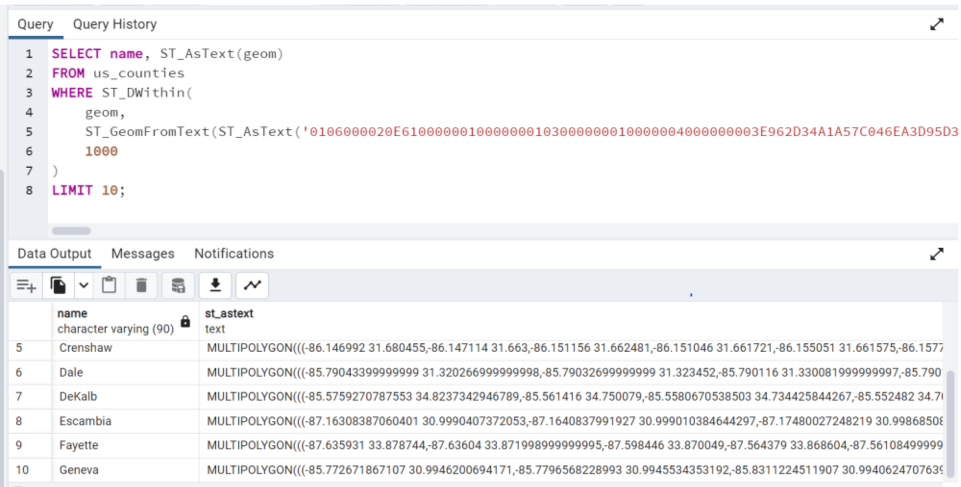
Query gives the total area of interest for each county in square kilometers.

```
SELECT name, SUM(ST_Area(geom::geography)) / 1000000 AS area_sq_km
FROM public.us_counties
GROUP BY name;
```



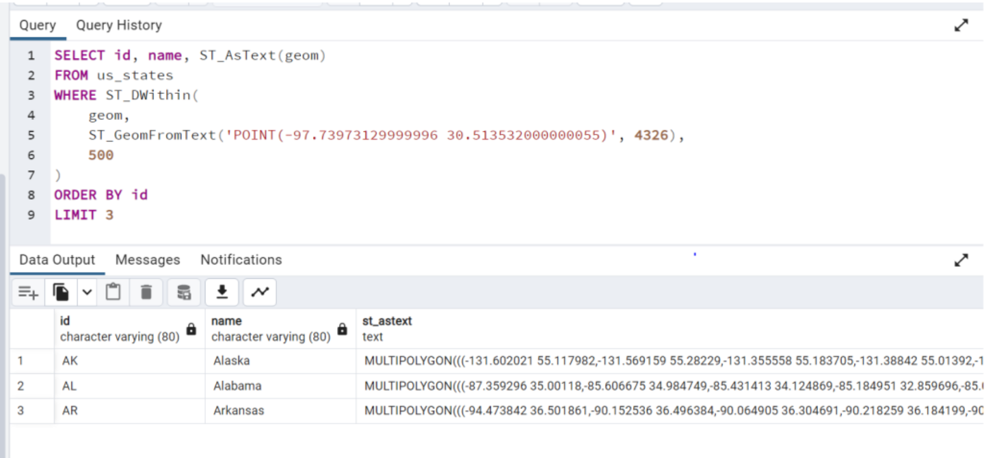
Query returns counties which is less than or equal to 1000 units.

```
SELECT name, ST_AsText(geom)
FROM us_counties
WHERE ST_DWithin(
    geom,
    ST_GeomFromText(ST_AsText('0106000020E610.....'), 4326),
    1000
)
LIMIT 10;
```



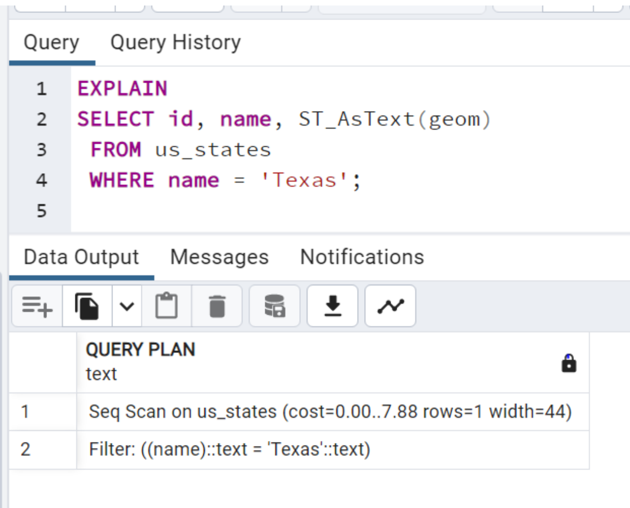
Query returns counties which is less than or equal to 500 units.

```
SELECT id, name, ST_AsText(geom)
FROM us_states
WHERE ST_DWithin(
   geom,
   ST_GeomFromText('POINT(-97.73973129999996 30.513532000000055)', 4326),
   500
) ORDER BY id
LIMIT 3;
```



In PostgreSQL, **EXPLAIN and EXPLAIN ANALYZE** commands are used to analyze the performance of a query. These commands show the execution plan that the PostgreSQL query planner generates for a given query. The execution plan shows how the query will be executed and helps to identify the potential performance issues.

```
EXPLAIN
SELECT id, name, ST_AsText(geom)
 FROM us_states WHERE name = 'Texas';
```



EXPLAIN ANALYZE
SELECT c.gid, c.county, c.name, s.name, ST_AsText(c.geom)
FROM us_counties c, us_states s
WHERE s.id = 'NY' and s.gid = c.state;

Query | Query History

```
EXPLAIN ANALYZE
SELECT c.gid, c.county, c.name, s.name, ST_AsText(c.geom)
    FROM us_counties c, us_states s
    WHERE s.id = 'NY' and s.gid = c.state;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Hash Join (cost=6.64..1554.48 rows=16 width=57) (actual time=1.709..3.759 rows=17 loops=1)
2	Hash Cond: (c.state = s.gid)
3	-> Seq Scan on us_counties c (cost=0.00..1519.21 rows=3221 width=4228) (actual time=0.019..2.077 rows=3221 loops=1)
4	-> Hash (cost=6.63..6.63 rows=1 width=13) (actual time=0.055..0.056 rows=1 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 9kB
6	-> Seq Scan on us_states s (cost=0.00..6.63 rows=1 width=13) (actual time=0.030..0.039 rows=1 loops=1)
7	Filter: ((id)::text = 'NY'::text)
8	Rows Removed by Filter: 49
9	Planning Time: 0.364 ms
10	Execution Time: 3.807 ms

CREATE INDEX IF NOT EXISTS us_counties_name_idx
ON us_counties (name);

Query | Query History

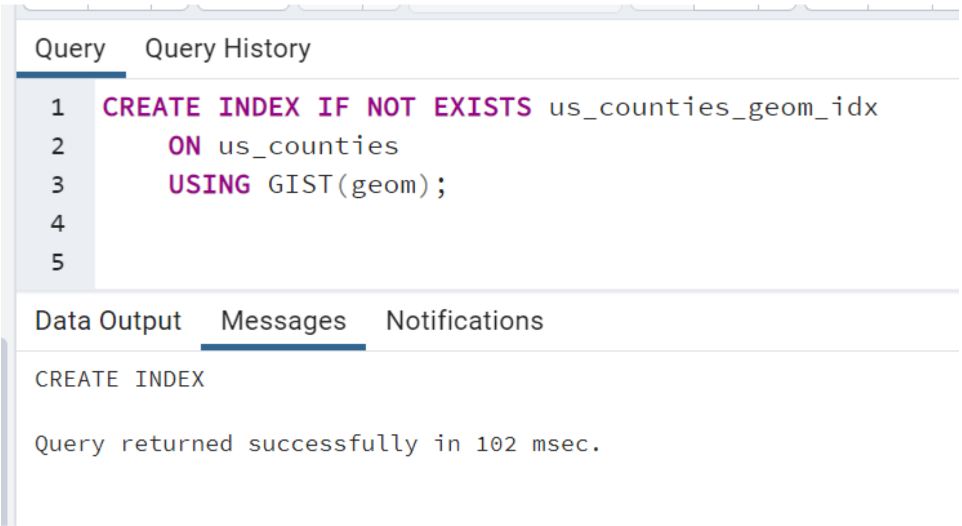
```
CREATE INDEX IF NOT EXISTS us_counties_name_idx
    ON us_counties (name);
```

Data Output | Messages | Notifications

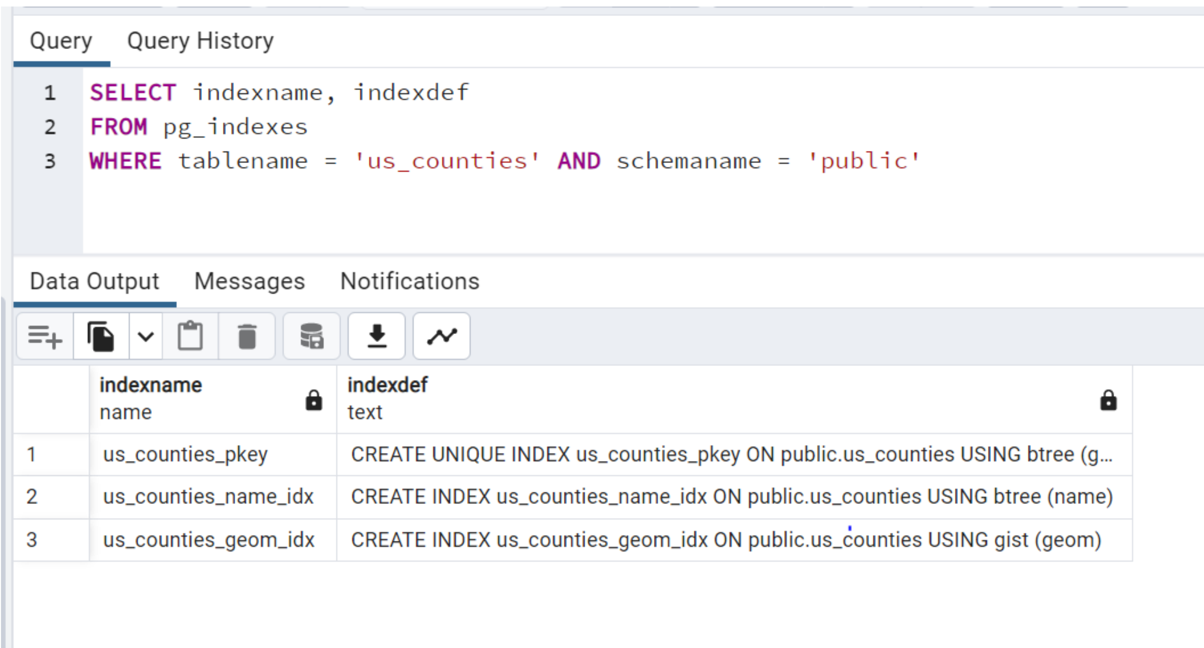
```
CREATE INDEX

Query returned successfully in 183 msec.
```

```
CREATE INDEX IF NOT EXISTS us_counties_geom_idx
    ON us_counties
    USING GIST(geom);
```

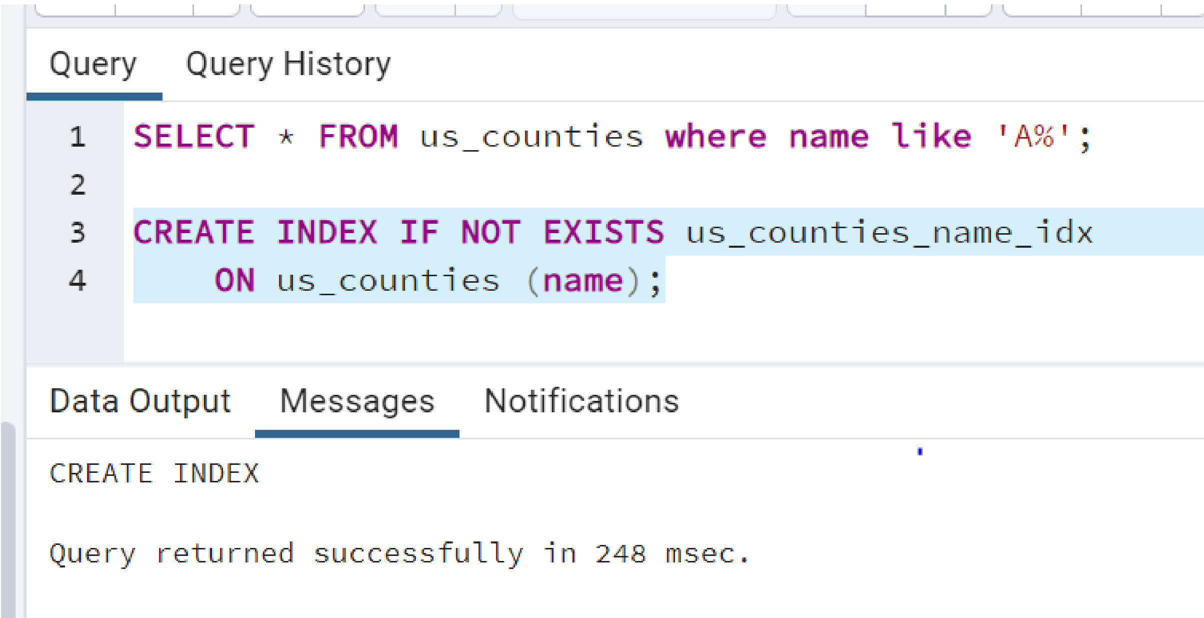



```
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'us_counties' AND schemaname = 'public';
```



N-Optimization by creating index for name.

CREATE INDEX IF NOT EXISTS us_counties_name_idx
ON us_counties (name);



<u>After indexing the name column, the query execution time reduced almost 30 milli seconds.</u>