ARISTOTLE UNIVERSITY OF THESSALONIKI
Department of Computer Science
MSc Data and Web Science

DWS206 - Advanced Topics in Databases
Spring 2023

Spatial Database Project : USA Mapping

Moutsiounas Panagiotis - 153

Spatial Database Project : USA Mapping

Moutsiounas Panagiotis - 153

June 17, 2023

Contents

1 Application Approach

The United States of America is a vast country with a diverse landscape and a complex transportation network. To accurately map this country, it is important to have a robust spatial database that includes a variety of entities such as counties, urban areas, rails, primary roads, and congressional districts. These entities are not only essential for creating detailed maps but are also crucial for various applications such as traffic monitoring, vehicle routing, and urban planning.

In addition to the spatial entities, the database also includes non-spatial entities such as users, traffic incidents, US state vehicles, and maintenance records. These entities provide important information about the usage, maintenance, and safety of the transportation network in the USA.

The data for the spatial tables were downloaded from https://www.census.gov/cgi-bin/geo/shapefiles/index.php and for the rest, created by https://www.mockaroo.com/

Spatial Entities	Non-Spatial Entities
CountiesAndEquivalent	Users
UrbanAreas	Traffic_Incidents
Rails	US_State_Vehicles
PrimaryRoads	Maintenance
CongressionalDistricts	

Table 1: Entities in the Spatial Database for Mapping the USA

2 Entity Relationship Diagram

For this assignment, both the ER and the Relation Diagram were created using **Visual Paradigm**.

Visual Paradigm is a comprehensive software development platform that supports a wide range of modeling notations and languages, including UML, BPMN, SysML, ERD, and DFD. It offers features to support agile development methodologies, collaboration and teamwork, code and documentation generation, and more. Overall, Visual Paradigm is a powerful and versatile tool for software development, providing teams with the tools and features they need to design, model, and develop their software projects efficiently and effectively.

As shown in the diagram below, apart from the entity names and their relations, there are also shown all their respective attributes, just like the relation model. That is a trait of the Visual Paradigm application and is important because from this type of model derives the final physical model (Relational Model).

The entities and their relations (one to many, many to many etc.) are shown in the traditional way. The entity name is added on the line. With the report, there will also be included a .png file for every diagram. That is so they can be most clearly understood and also so that they are zoomable and clear.

After the diagram, we will present the entities and a word about their meaning on this db.

Note that the spatial entities might not be well described as their entire tables were taken as a whole from the web. Any meaning given was assigned to each attribute after research; not documentation was included.

Finally, we will see the relations between the entities.

Table 2: Entities and Attributes

Entity	Attribute	Description
Counties and Equivalent (Spatial)		
	gid	The unique ID of the table.
	statefp	The state code.
	countyfp	The county code.
	countyns	Another county characteristic
	geoid	The id of each geographical location.
	name	The name of each county.
	namelsad	The name of each county and the county's prefix.
	lsad	The county's prefix.
	...	The rest were unidentified.
	geom	The geometry attribute of the entity.
Congressional Districts (Spatial)		
	gid...	Already mentioned attributes will be skipped.
	...	Unidentified.
Rails (Spatial)		
	gid...	Already mentioned attributes will be skipped.
	linearid	The linear id of each rail.
	fullname	The full name of the rail road.
	...	Already mentioned attributes will be skipped.
Primary Roads (Spatial)		
	..	Identical to "Rails"
Urban Areas (Spatial)		
	..	Identical to "Counties and Equivalent" with some unidentified extra attributes.
US State Vehicle		
	comment	This entity is about vehicles registered on the USA Mapping system. Ex. Governn
	Vehicle id	The unique id of each vehicle.
	make	The brand of each vehicle.
	model	The model of each vehicle.
	year	The year the vehicle was created.
	fuel type	The type of the fuel that moves the vehicle.
	safety rating	The safety rating of each vehicle.
	emissions rating	The rating of the emissions of each vehicle.
Maintenance		
	comment	This entity gives information about any maintenance in roads, rails or us state veh
	maintenance id	The unique id of each maintenance.
	type	The maintenance type.
	location	The location.
	time	The time the maintenance occurred.
	description	A description of the maintenance.
Traffic Incidents		
	comment	The entity where the traffic incidents are stored in the DB. Identical to the "Mainte
Users		
	comment	This entity resembles the entitled and authorised personnel that has access to the d
	user id	The unique user id.
	username	The username.
	password	The password.
	email	The email.
	first name	The first name.
	last name	The last name.

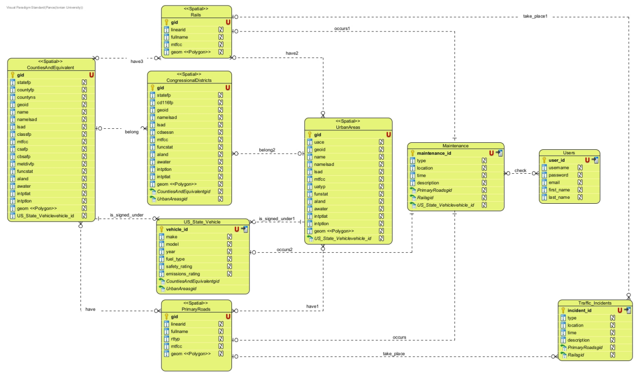

Figure 1: ER diagram

Entity ->	Relation ->	Entity
CountiesAndEquivalent	Many-to-many	Rails
CountiesAndEquivalent	Many-to-many	PrimaryRoads
CountiesAndEquivalent	One-to-many	US State Vehicle
CountiesAndEquivalent	One-to-many	CongressionalDistricts
CongressionalDistricts	Many-to-one	UrbanAreas
Rails	One-to-many	Traffic Incidents
Rails	One-to-one	Maintenance
Rails	Many-to-many	UrbanAreas
PrimaryRoads	Many-to-many	UrbanAreas
PrimaryRoads	One-to-one	Maintenance
PrimaryRoads	One-to-many	Traffic Incidents
US State$_{V}ehicle$	One-to-many	Maintenance
US State$_{V}ehicle$	Many-to-one	UrbanAreas
Maintenance	Many-to-many	Users

Table 3: Entitiy Relationships

3 Relational Model

Here we show the relational model of the database. It was also created in Visual Paradigm and is explained according to the transformation rules below.

Table 4: Transformation to Relation Model.

Entity	Relation	Changes
Counties and Equivalent (Spatial)		
	is_signed_under, US_State_Vehicle	A Foreign key "US_State_Vehiclevehicle_id" is generated.
	belong, CongressionalDistricts	A Foreign key is added to the "CongressionalDistricts" table as an attribute.
	have, Rails, PrimaryRoads	A new table is created containing the primary keys of each of the previous tables.
CongressionalDistricts (Spatial)		
	belong, UrbanAreas	A Foreign key is added to the "CongressionalDistricts" table as an attribute.
US_State_Vehicle		
	is_signed_under, UrbanAreas, CongressionalDistricts	Two Foreign keys are added to the "UrbanAreas" table as an attribute.
UrbanAreas (Spatial)		
	have, Rails, PrimaryRoads	Two new tables are created which contain the primary keys of "UrbanAreas" and "PrimaryRoads"-"Rails" respectively.
Maintenance		
	check, Users	A new table is created which contains the primary keys of both "Maintenance" and "User" tables.
	occurs, Rails, PrimaryRoads, US_State_Vehicle	A new attribute is added on the "Maintenance" table for each of the other tables' primary keys.
Traffic_Incidents		
	take place, Rails, PrimaryRoads	The primary key of each of the referenced tables is added as a foreign key to the "Traffic_Incidents" table.

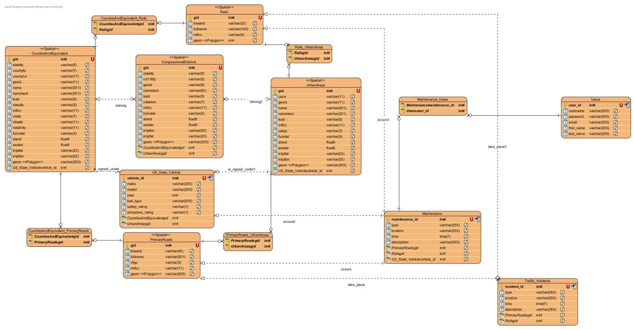

Figure 2: Relation model diagram

4 SQL Application

This project was developed in PostgreSQL. PostgreSQL is a robust and scalable open-source relational database management system that provides enterprise-level performance and reliability. It supports complex queries and large volumes of data and is a popular choice for a wide range of applications. Along with PostgreSQL, PostGIS was used, which is an extension for PostgreSQL that adds support for geographic objects, enabling the storage, querying, and manipulation of spatial data within a PostgreSQL database. It offers spatial indexing, spatial functions and operators, and support for standard spatial data formats like Shapefiles and GeoJSON. The entire project was created throughout pgAdmin. pgAdmin is an open-source administration and management tool for PostgreSQL databases that provides a user-friendly graphical interface for managing PostgreSQL databases. It offers a query tool with syntax highlighting and autocompletion, server monitoring, backup and restore capabilities, and more.

The script to create the database tables and their relations can be found in the **createscript.sql** file.

5 Inserts

The inserts for the spatial data were driven by the census.gov. The data for the rest of the tables were AI-generated using Mockaroo. The insert data can be found in the **inserts** directory.

6 Queries

Below follow the 6 non-spatial queries and 7 out of 9 of spatial queries we were required to complete. There will also be given a screenshot of every query run. The queries can be found in the **queries** directory.

Non spatial queries.

1)The count of the number of incidents where the maintenance was not successful and thus happened more incidents.

```
SELECT inc."location", COUNT(*) as num_incidents
FROM "Spatial_US_Mapping"."Traffic_Incidents" as inc
JOIN "Spatial_US_Mapping"."Maintenance" as main ON
   inc."location" = main."location"
WHERE inc."time" > main."time"
GROUP BY inc."location";
```

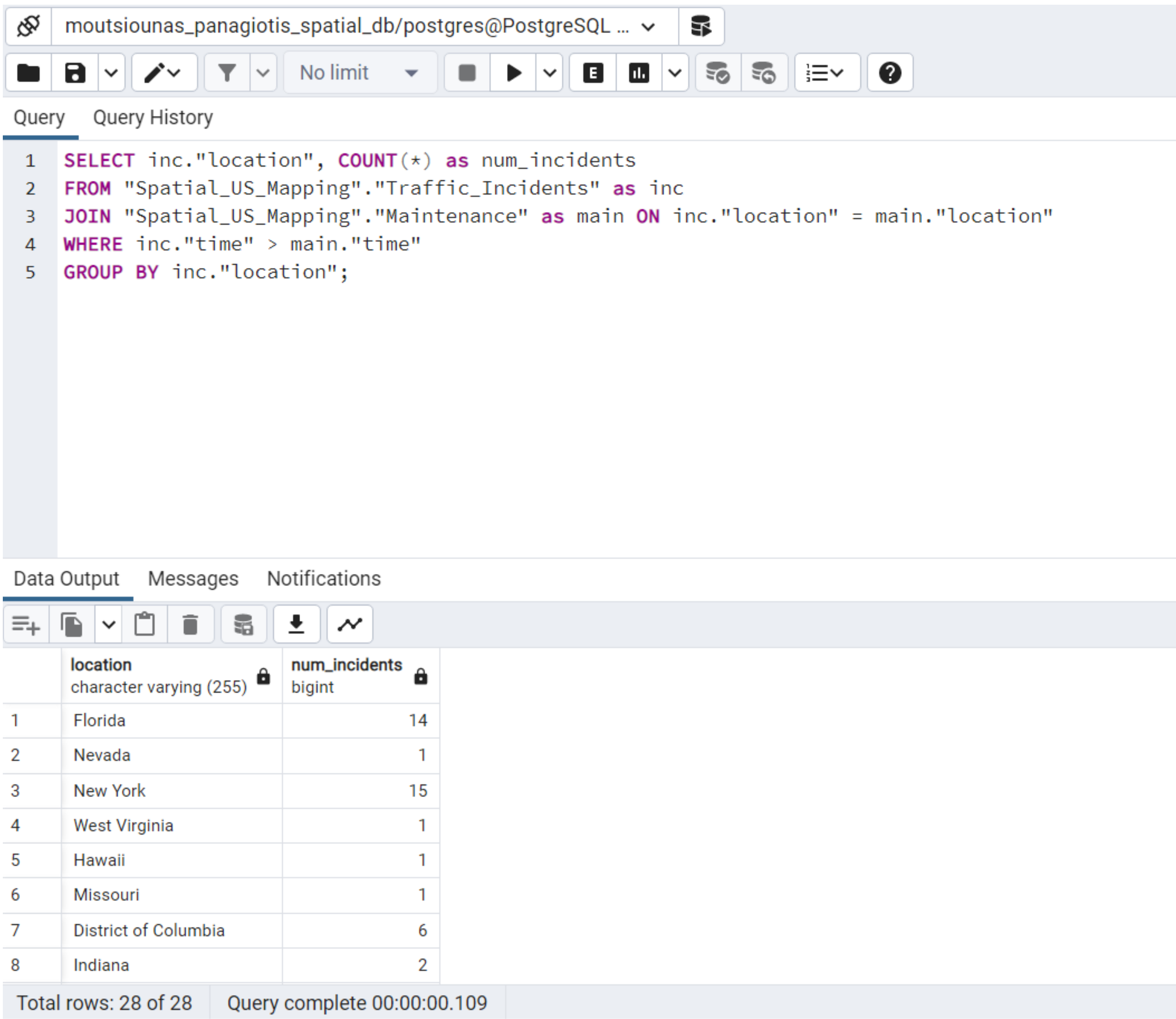

Figure 3: Non-spatial query 1

2)How many incidents occurred past maintenance and how many locations had that many incidents.

```
SELECT num_incidents, COUNT(*) as num_locations
FROM (
    SELECT inc."location", COUNT(*) as num_incidents
    FROM "Spatial_US_Mapping"."Traffic_Incidents" as inc
    JOIN "Spatial_US_Mapping"."Maintenance" as main ON
       inc."location" = main."location"
    WHERE inc."time" > main."time"
    GROUP BY inc."location"
) as subquery
GROUP BY num_incidents
ORDER BY num_incidents DESC;
```

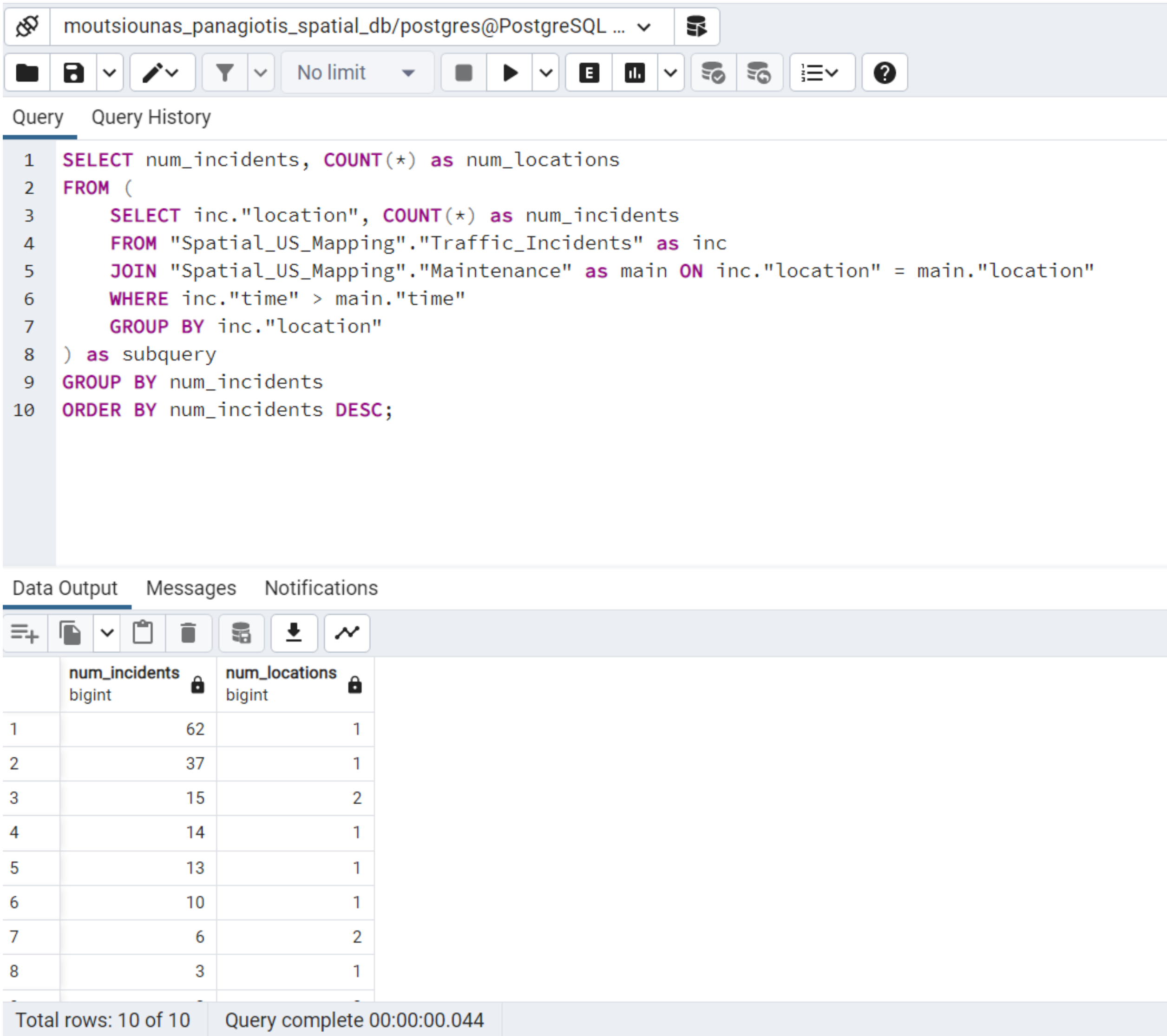

Figure 4: Non-spatial query 2

3)The average accident time.

```sql
SELECT AVG(inc."time") as "Average Accident after Maintenance"
FROM "Spatial_US_Mapping"."Traffic_Incidents" as inc
JOIN "Spatial_US_Mapping"."Maintenance" as main ON
   inc."location" = main."location";
```

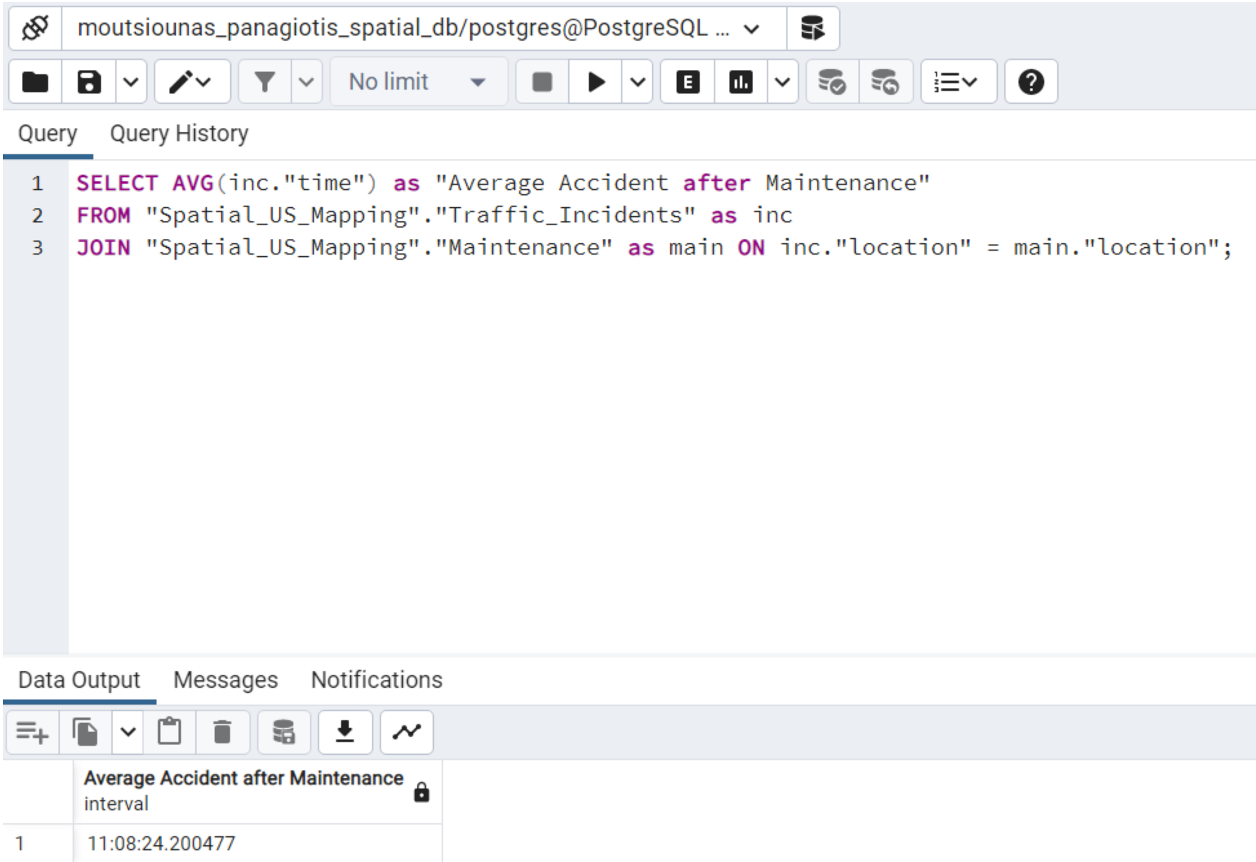

Figure 5: Non-spatial query 3

4)How many diesel cars are to each county, if there are any.

```
SELECT Count(*) as Num_of_Diesels , county."namelsad"
FROM "Spatial_US_Mapping"."US_State_Vehicle" as vehicle
JOIN "Spatial_US_Mapping"."CountiesAndEquivalent" as county ON
   county."gid" = vehicle."countiesandequivalentgid"
WHERE vehicle."fuel_type" = 'Diesel'
GROUP BY county."namelsad"
ORDER BY Num_of_Diesels DESC;
```

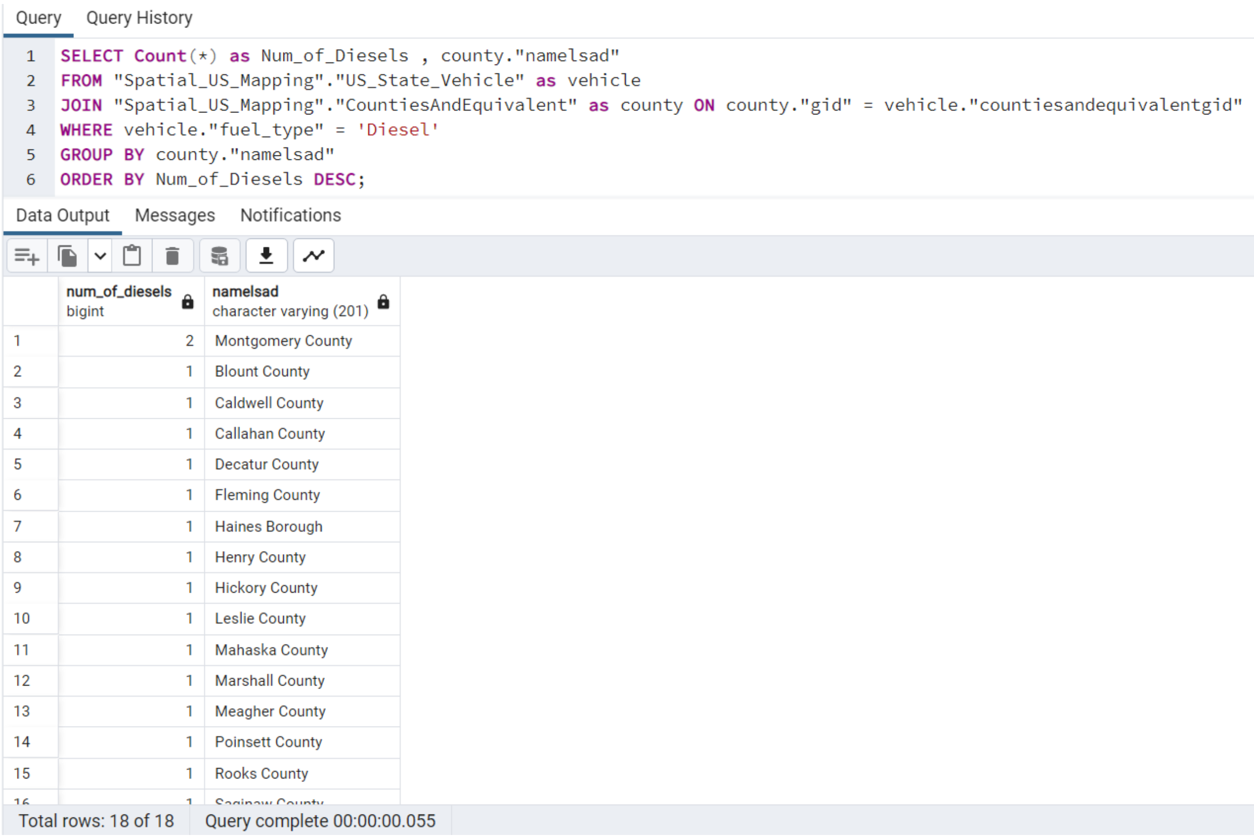

Figure 6: Non-spatial query 4

5)The percentage of accidents per county.

```
SELECT CAST((accidentspercounty*100.0/totalaccidents) AS
   DECIMAL(10,2)) as percentage_of_accidents_per_county, cname
FROM
(SELECT COUNT(*) as AccidentsPerCounty, county."namelsad" as
   cname
FROM "Spatial_US_Mapping"."PrimaryRoads" as pr
JOIN "Spatial_US_Mapping"."Traffic_Incidents" as ti
ON pr."gid" = ti."primaryroadsgid"
JOIN "Spatial_US_Mapping"."CountiesAndEquivalent_PrimaryRoads"
   as copr
ON copr."primaryroadsgid" = pr."gid"
JOIN "Spatial_US_Mapping"."CountiesAndEquivalent" as county
ON county."gid" = copr."countiesandequivalentgid"
GROUP BY cname) as accPC
CROSS JOIN (
    SELECT COUNT(pr."gid") as totalaccidents
    FROM "Spatial_US_Mapping"."PrimaryRoads" as pr
    JOIN "Spatial_US_Mapping"."Traffic_Incidents" as ti
    ON pr."gid" = ti."primaryroadsgid"
    JOIN
       "Spatial_US_Mapping"."CountiesAndEquivalent_PrimaryRoads"
       as copr
    ON copr."primaryroadsgid" = pr."gid"
    JOIN "Spatial_US_Mapping"."CountiesAndEquivalent" as county
    ON county."gid" = copr."countiesandequivalentgid"
) as sub
order by percentage_of_accidents_per_county DESC;
```

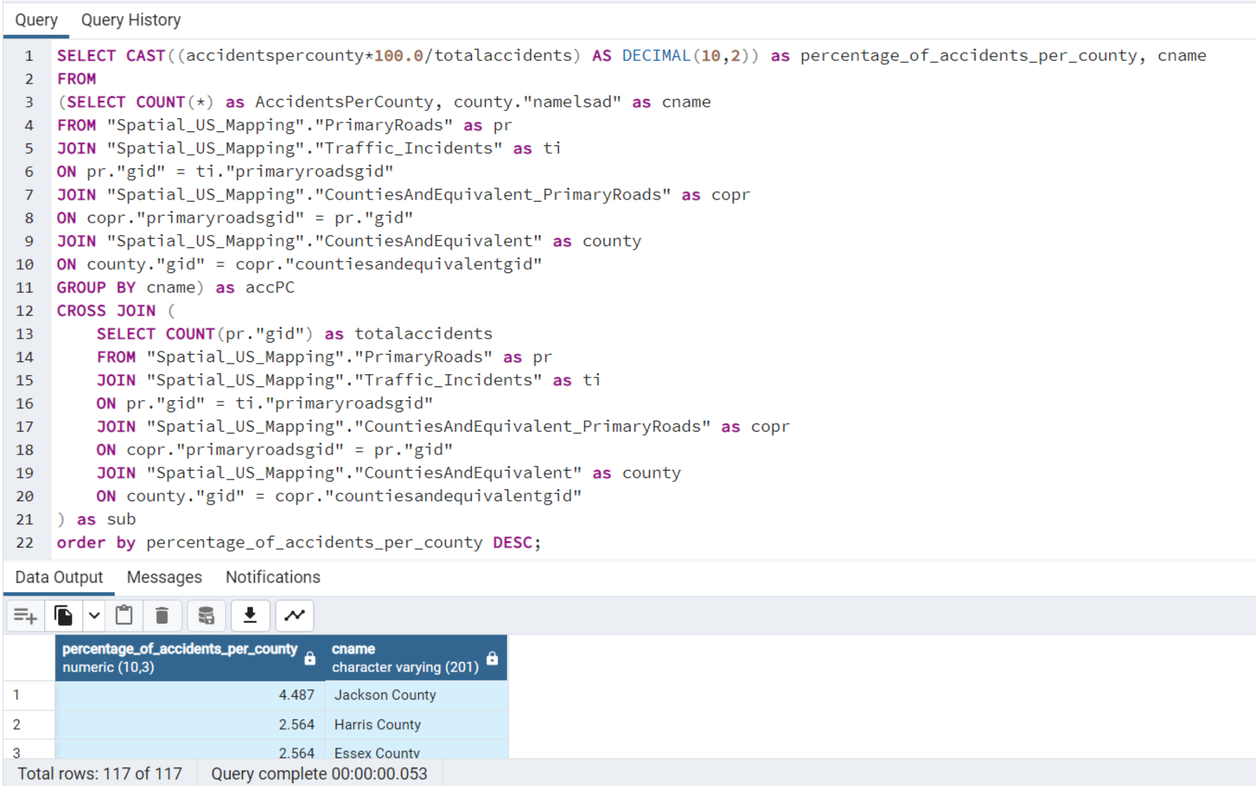

Figure 7: Non-spatial query 5 - 2

6) View that ranks the maintenances based on the type of maintenance that happens on the same road. if a maintenance happens more than three times on the road, it gets marked and added to the view.

```
CREATE VIEW Top_Maintenance_Activities_By_Primary_Road AS
SELECT p."fullname" AS Primary_Road_Name, m."type" AS
   Maintenance_Type, COUNT(*) AS Frequency
FROM "Spatial_US_Mapping"."Maintenance" m
JOIN "Spatial_US_Mapping"."PrimaryRoads" p ON
   m."primaryroadsgid" = p."gid"
GROUP BY p."fullname", m."type"
HAVING COUNT(*) >= 3
ORDER BY p."fullname", COUNT(*) DESC;
```

	percentage_of_accidents_per_county numeric (10,3)	cname character varying (201)
1	4.487	Jackson County
2	2.564	Harris County
3	2.564	Essex County
4	2.564	Jefferson County
5	2.564	Allegan County
6	1.923	Queens County
7	1.923	Union County
8	1.923	Erie County
9	1.282	Garfield County
10	1.282	Mercer County
11	1.282	Washington County
12	1.282	Milwaukee County
13	1.282	Morris County
14	1.282	Wood County
15	1.282	Butler County
16	1.282	Arlington County
17	1.282	Kent County
18	1.282	Duval County
19	1.282	St. Louis city
20	1.282	Tarrant County
21	1.282	Lake County
22	1.282	Cabell County

Figure 8: Non-spatial query 5 - 2

Figure 9: Non-spatial query 6

7) The number of congressional districts per state.

```
SELECT COUNT(*) as Number_of_CongressionalDistricts_per_State,
   dstr."statefp"
FROM "Spatial_US_Mapping"."CountiesAndEquivalent" as county
JOIN "Spatial_US_Mapping"."CongressionalDistricts" as dstr
ON county."statefp" = dstr."statefp"
GROUP BY dstr."statefp"
ORDER BY dstr."statefp" ASC
```

Query | Query History

```sql
SELECT COUNT(*) as Number_of_CongressionalDistricts_per_State, dstr."statefp"
FROM "Spatial_US_Mapping"."CountiesAndEquivalent" as county
JOIN "Spatial_US_Mapping"."CongressionalDistricts" as dstr
ON county."statefp" = dstr."statefp"
GROUP BY dstr."statefp"
ORDER BY dstr."statefp" ASC
```

Data Output | Messages | Notifications

	number_of_congressionaldistricts_per_state bigint	statefp character varying (5)
1	469	01
2	30	02
3	135	04
4	300	05
5	3074	06
6	448	08
7	48	09
8	3	10

Total rows: 56 of 56 | Query complete 00:00:00.093

Figure 10: Non-spatial query 7

Spatial queries.

1) The roads that touches any Urban Areas.

```
SELECT DISTINCT uac20.name20 AS urban_area_name, roads.fullname
   AS road_name
FROM tl_2020_us_primaryroads AS roads
JOIN tl_2020_us_uac20 AS uac20
ON st_touches(st_transform(roads.geom, 4326), uac20.geom)
ORDER BY road_name;
```

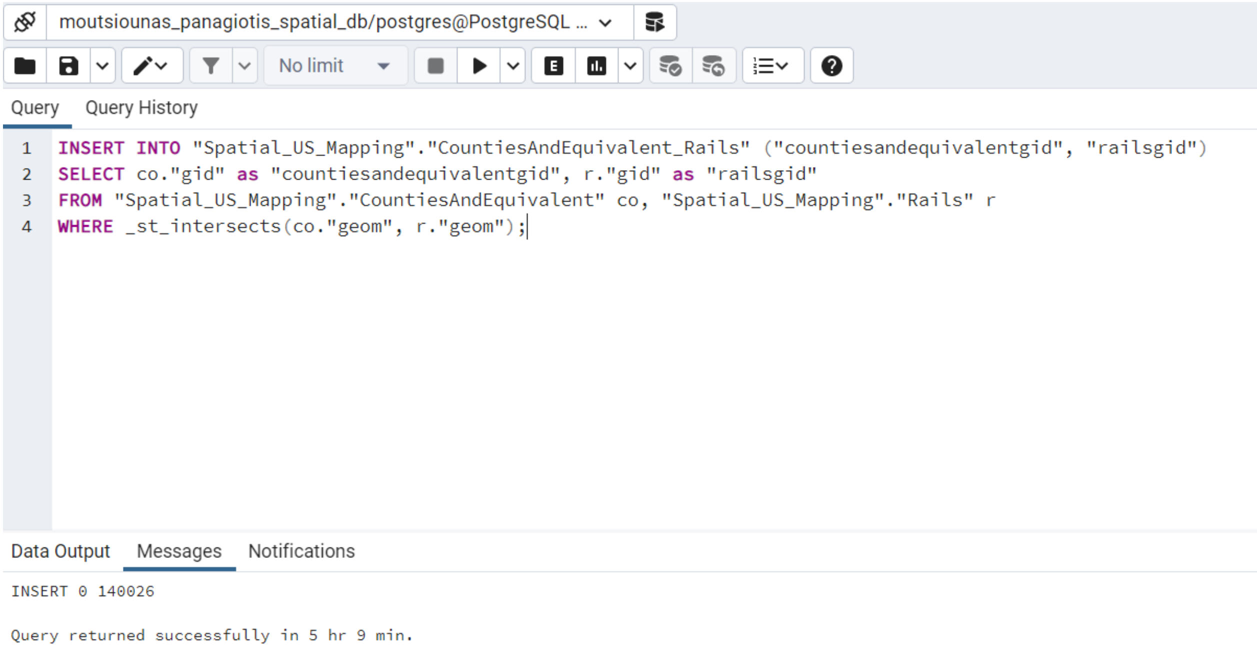

Figure 11: Spatial query 1

2) The roads that connect two Urban Areas.

```
SELECT ua1.name20 AS urban_area_1,r.fullname AS road_name,
   ua2.name20 AS urban_area_2
FROM tl_2020_us_primaryroads AS r
JOIN tl_2020_us_uac20 AS ua1
ON ST_Intersects(ST_Transform(r.geom, 4326), ua1.geom)
JOIN tl_2020_us_uac20 AS ua2
ON ST_Intersects(ST_Transform(r.geom, 4326), ua2.geom) AND
   ua1.geoid20 <> ua2.geoid20;
```

Figure 12: Spatial query 2

3) Fill the many-to-many generated table between ”Rails”and ”UrbanAreas” using geom function.

```
INSERT INTO "Spatial_US_Mapping"."Rails_UrbanAreas" ("railsgid",
   "urbanareasgid")
SELECT r."gid" as "railsgid", ua."gid" as "urbanareasgid"
FROM "Spatial_US_Mapping"."UrbanAreas" ua,
   "Spatial_US_Mapping"."Rails" r
WHERE _st_intersects(ua."geom",r."geom");
```

Figure 13: Spatial query 3

4) Print the total area of every urban area.

```
SELECT st_area(st_transform(ua."geom", 26910)) as emvadon,
   ua."namelsad"
FROM "Spatial_US_Mapping"."UrbanAreas" ua
ORDER BY emvadon DESC
```

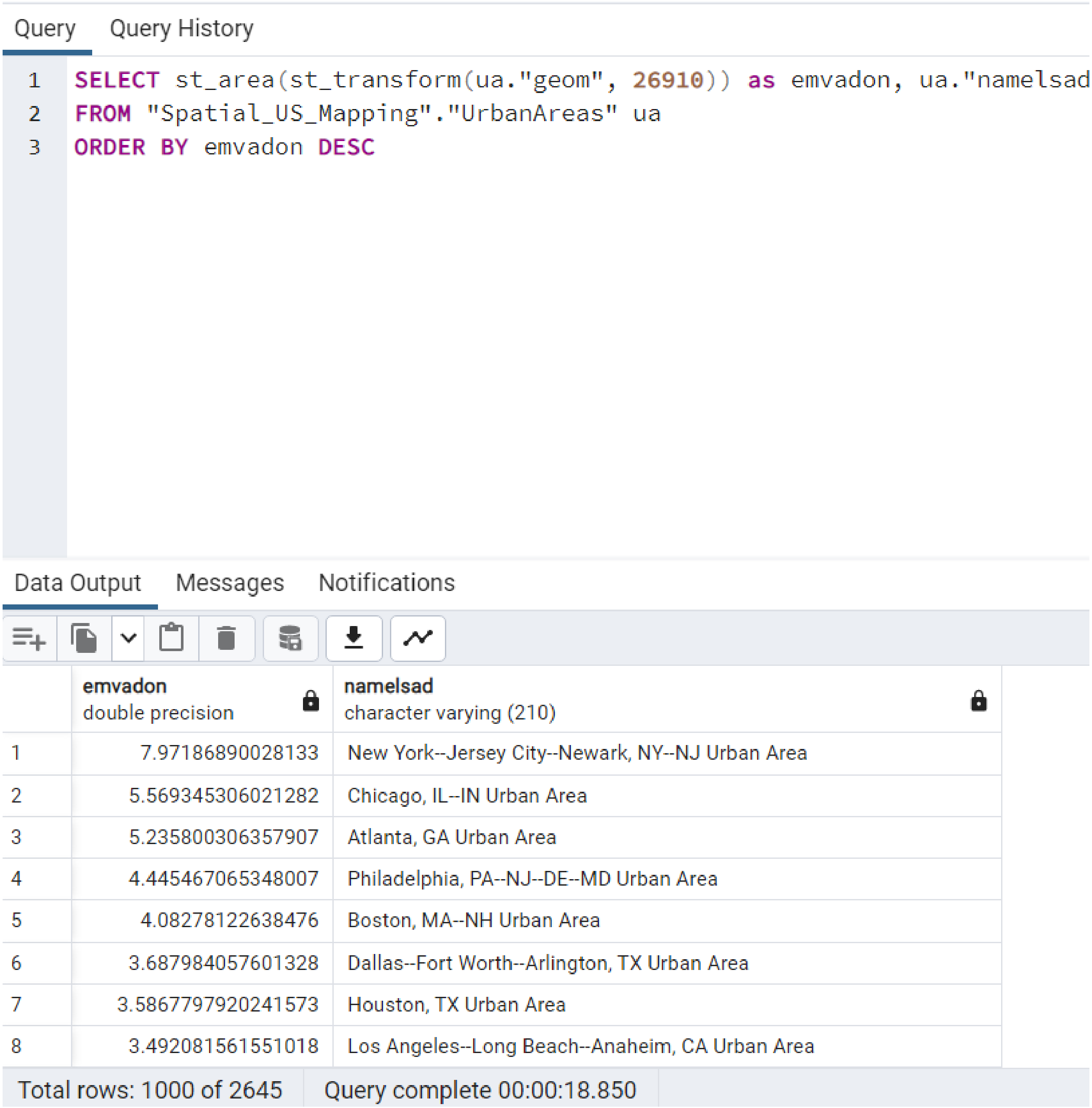

Figure 14: Spatial query 4

5) Get the ids of primary roads that connect urban areas and counties.

```sql
SELECT pr."gid" as RoadIDs_connecting_UAs_and_Counties
FROM "Spatial_US_Mapping"."UrbanAreas" ua
JOIN "Spatial_US_Mapping"."PrimaryRoads_UrbanAreas" prua
ON ua."gid" = prua."urbanareasgid"
JOIN "Spatial_US_Mapping"."CountiesAndEquivalent_PrimaryRoads"
   prc
ON prua."primaryroadsgid" = prc."primaryroadsgid"
JOIN "Spatial_US_Mapping"."PrimaryRoads" pr
ON pr."gid" = prc."primaryroadsgid"
JOIN "Spatial_US_Mapping"."CountiesAndEquivalent" county
ON county."gid" = prc."primaryroadsgid"
WHERE st_intersects(ua."geom", pr."geom") AND
   st_intersects(county."geom", pr."geom")
```

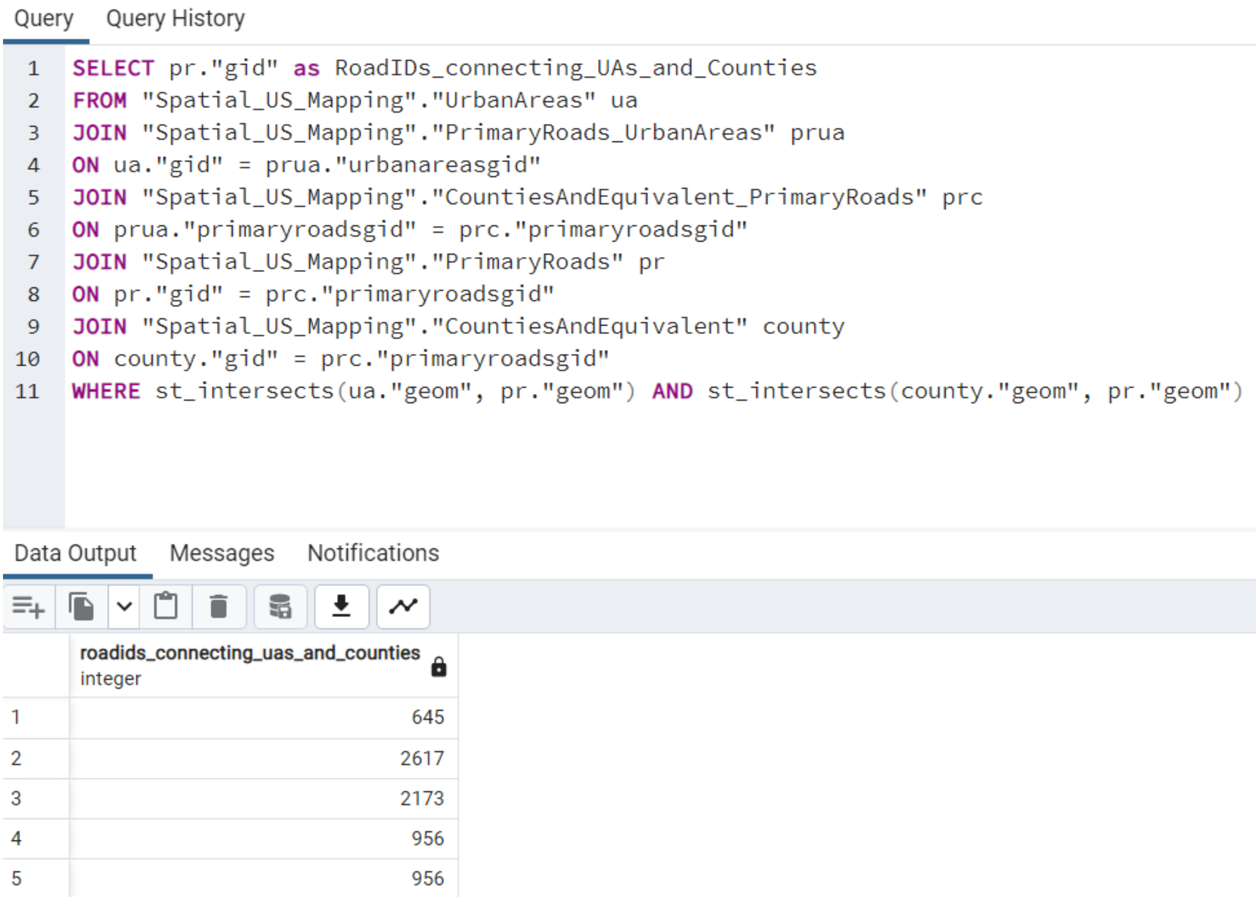

Figure 15: Spatial query 5

6) Return the road that has the shortest distance from each county.

```
SELECT
  c."namelsad",
  r."fullname",
  ST_Distance(c."geom", r."geom") AS distance
FROM
  "Spatial_US_Mapping"."CountiesAndEquivalent" AS c
CROSS JOIN LATERAL (
  SELECT
    r."fullname",
    r."geom"
  FROM
    "Spatial_US_Mapping"."PrimaryRoads" AS r
  JOIN
    "Spatial_US_Mapping"."CountiesAndEquivalent_PrimaryRoads" AS
       cr ON r."gid" = cr."primaryroadsgid"
  WHERE
    cr."countiesandequivalentgid" = c."gid"
  ORDER BY
    c."geom" <-> r."geom"
  LIMIT
    1
) AS r;
```

Query Query History

```sql
SELECT
  c."namelsad",
  r."fullname",
  ST_Distance(c."geom", r."geom") AS distance
FROM
  "Spatial_US_Mapping"."CountiesAndEquivalent" AS c
CROSS JOIN LATERAL (
  SELECT
    r."fullname",
    r."geom"
  FROM
    "Spatial_US_Mapping"."PrimaryRoads" AS r
  JOIN
    "Spatial_US_Mapping"."CountiesAndEquivalent_PrimaryRoads" AS cr ON r."gid" = cr."primaryroadsgid"
  WHERE
```

Data Output Messages Notifications

	namelsad character varying (201)	fullname character varying (201)	distance double precision
1	Lancaster County	I- 80	0
2	Las Piedras Municipio	Pr- 30	0
3	Minnehaha County	I- 229	0
4	Sierra County	I- 80	0
5	Hancock County	State Rte 15	0
6	Beaver County	I- 376	0
7	Jasper County	I- 59	0
8	Chatham County	US Hwy 1	0

Total rows: 1000 of 1730 Query complete 00:03:07.116

Figure 16: Spatial query 6

7) Determine if an entire railroad area is inside a county. return a true or a false.

```
SELECT
  c."namelsad",
  r."fullname",
  ST_Contains(c."geom", r."geom") AS contain
FROM
  "Spatial_US_Mapping"."CountiesAndEquivalent" AS c
CROSS JOIN LATERAL (
  SELECT
    r."fullname",
    r."geom"
  FROM
    "Spatial_US_Mapping"."PrimaryRoads" AS r
  JOIN
    "Spatial_US_Mapping"."CountiesAndEquivalent_PrimaryRoads" AS
       cr ON r."gid" = cr."primaryroadsgid"
  WHERE
    cr."countiesandequivalentgid" = c."gid"
  ORDER BY
    c."geom" <-> r."geom"
) AS r;
```

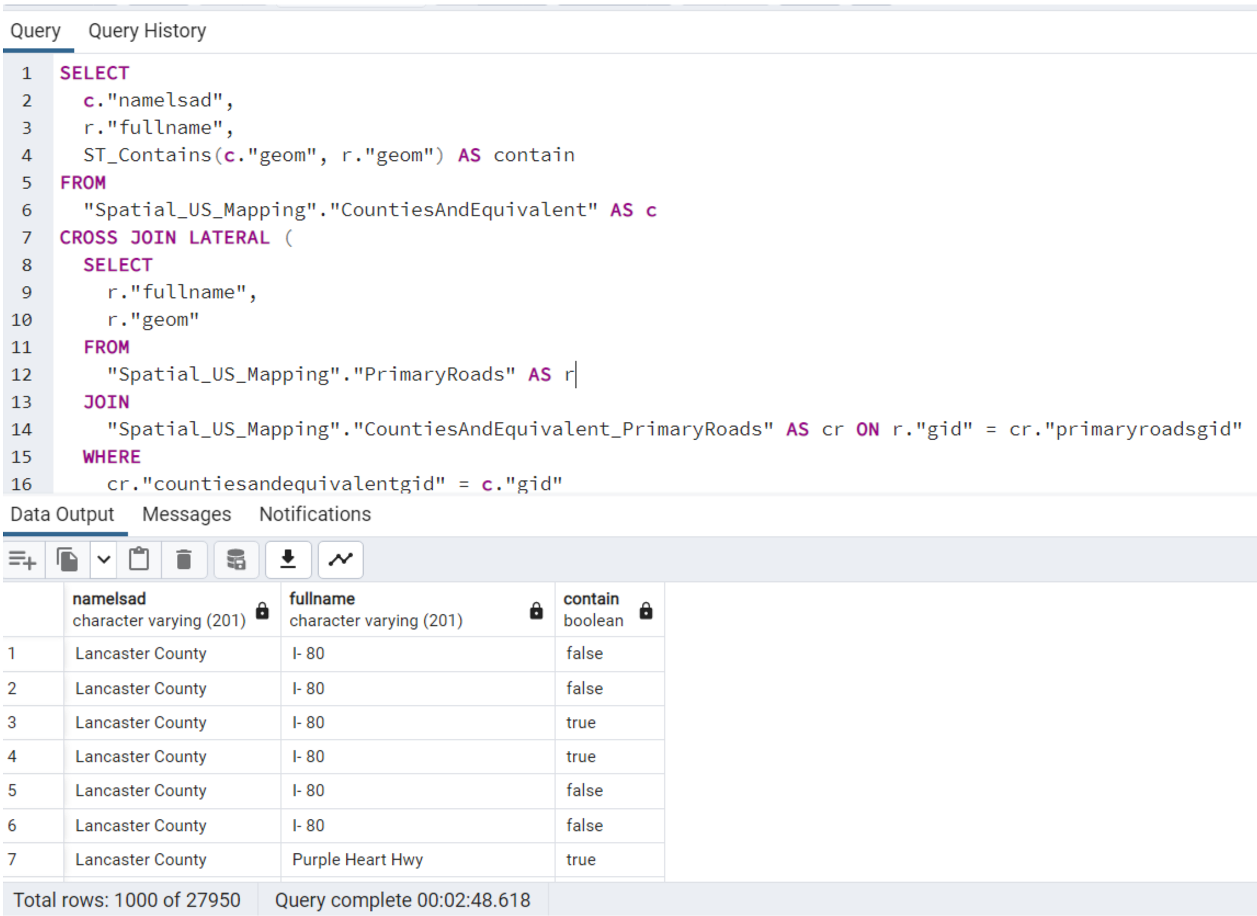

Figure 17: Spatial query 7

7 Conclusion

I want to thank my professor Eleftherios Tiakas for giving me the opportunity to get in touch with spatial data. The knowledge I have acquired will accompany me for on my CV and as a Data Scientist, complex SQL will be one of the most useful skills that I have gotten.