Wednesday, March 9 at 6:00 PM

Where to Park?

Using Python to Map Parking Regulations Signs to the Curb in Near Real-Time

Presented by: Jada Grandchamps, Graduate Student, Geography, Hunter College, CUNY

Visit open-data.nyc to view the full program.

Meet The Team

Maddalena Romano

Analytics and Performance Management

David Arcement

Traffic Control and Engineering

Matthew Garcia

Parking Administration

Suzanne Zopf

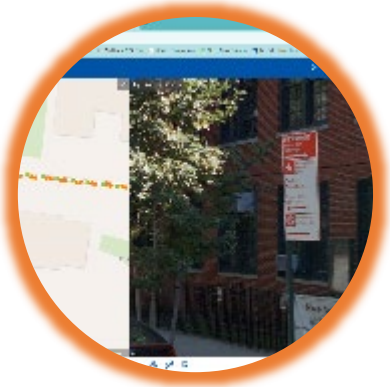
Information Technology

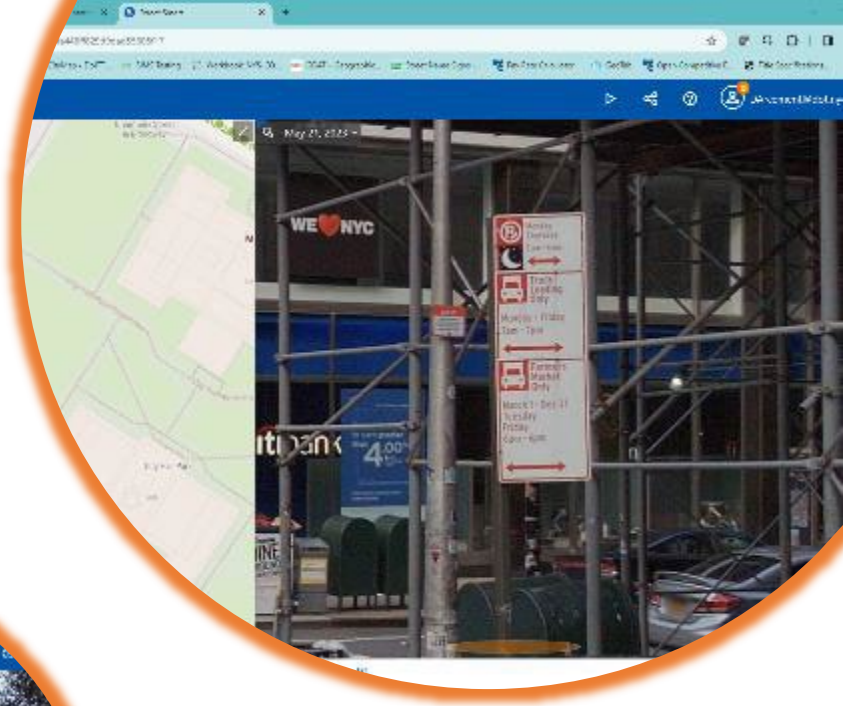
Jada Grandchamps

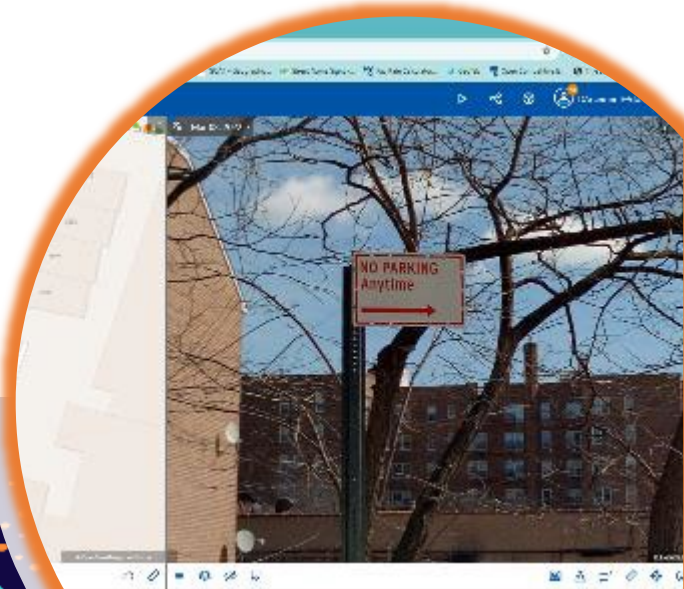
Department of Geography

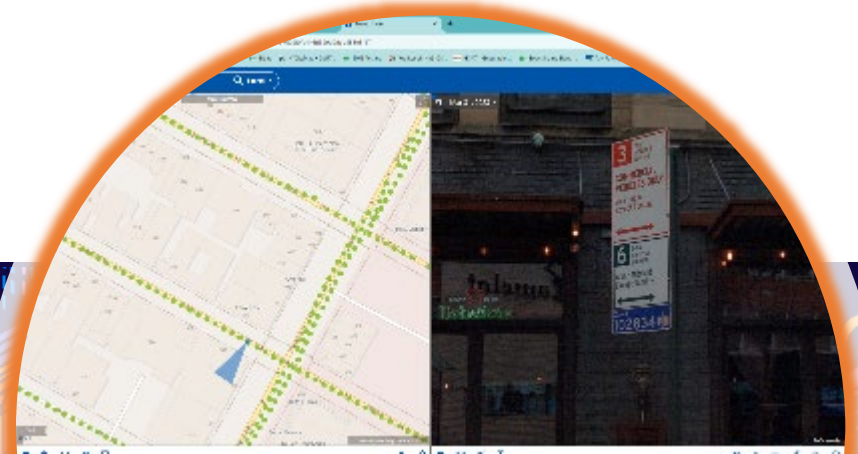
Background

New York City Dept. of Transportation (NYC DOT) is responsible for installing most of the signage in the five boroughs of NYC.

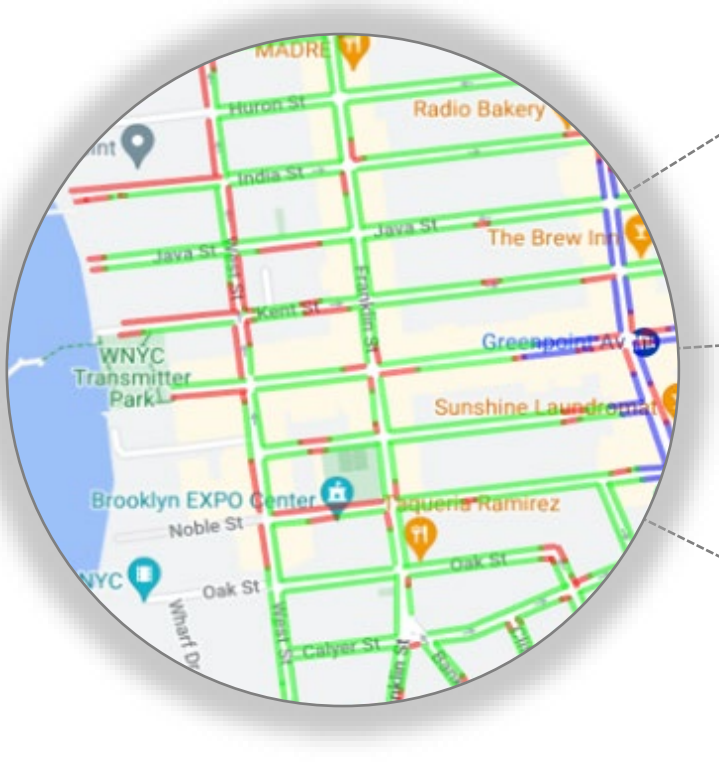








Issues with Other Maps



Coverage

Update Frequency

Categories

Our goal:
A map of all NYC parking regulations that updates in near real time

Scope → Datasets → Software

NYC OpenData

Pavement Edge

Pedestrian Ramps

Parking Regulations

python™

QGIS

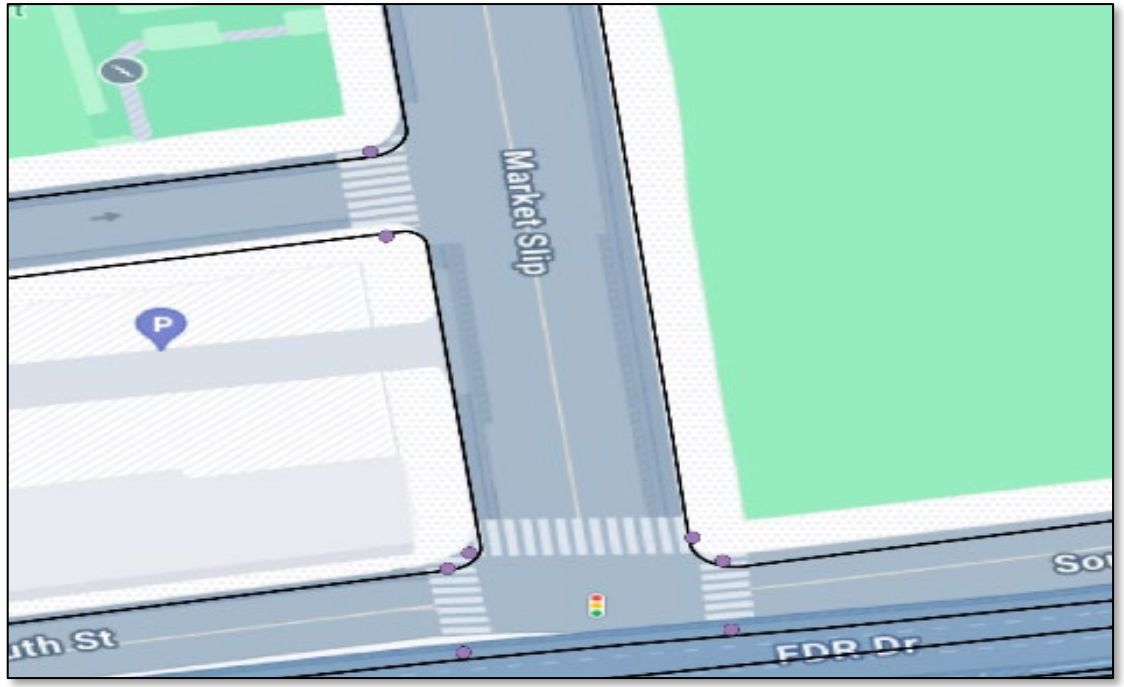
PostgreSQL

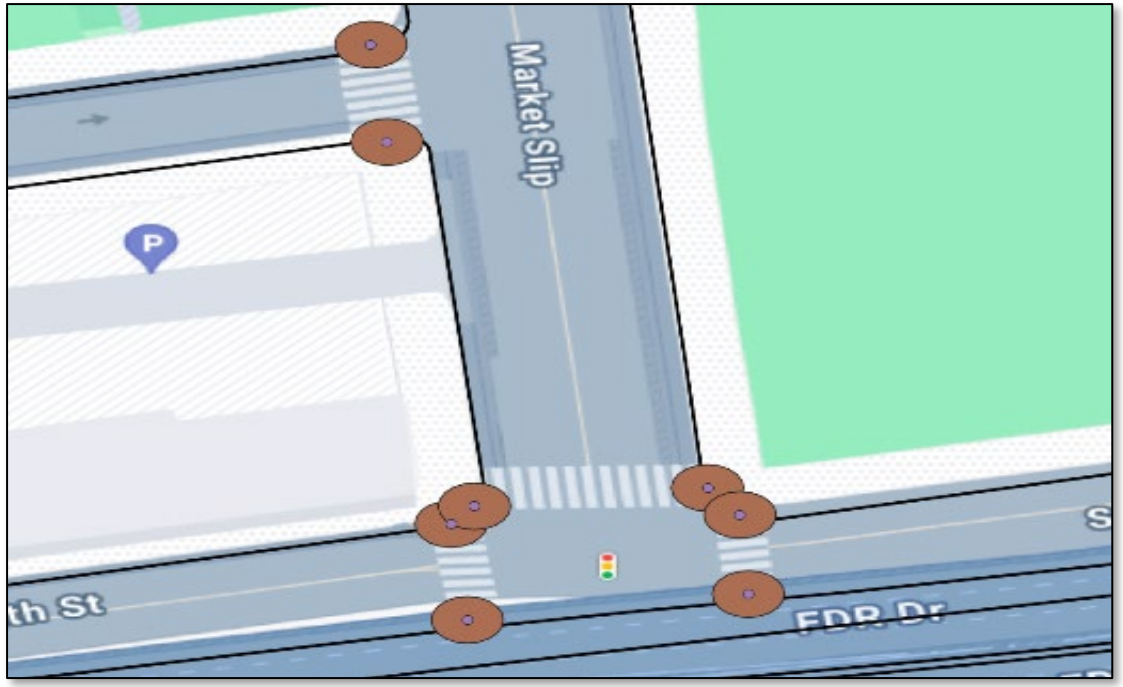
QGIS Component

Methods: Snap Points

```
156  UPDATE "DOT"."ParkingSigns" AS points
157  SET geom = ST_ClosestPoint("DOT".ClippedPave".geom, points.geom)
158  FROM "DOT"."ClippedPave" AS line
159  WHERE ST_DWithin(line.geom, points.geom, 0.001);
```

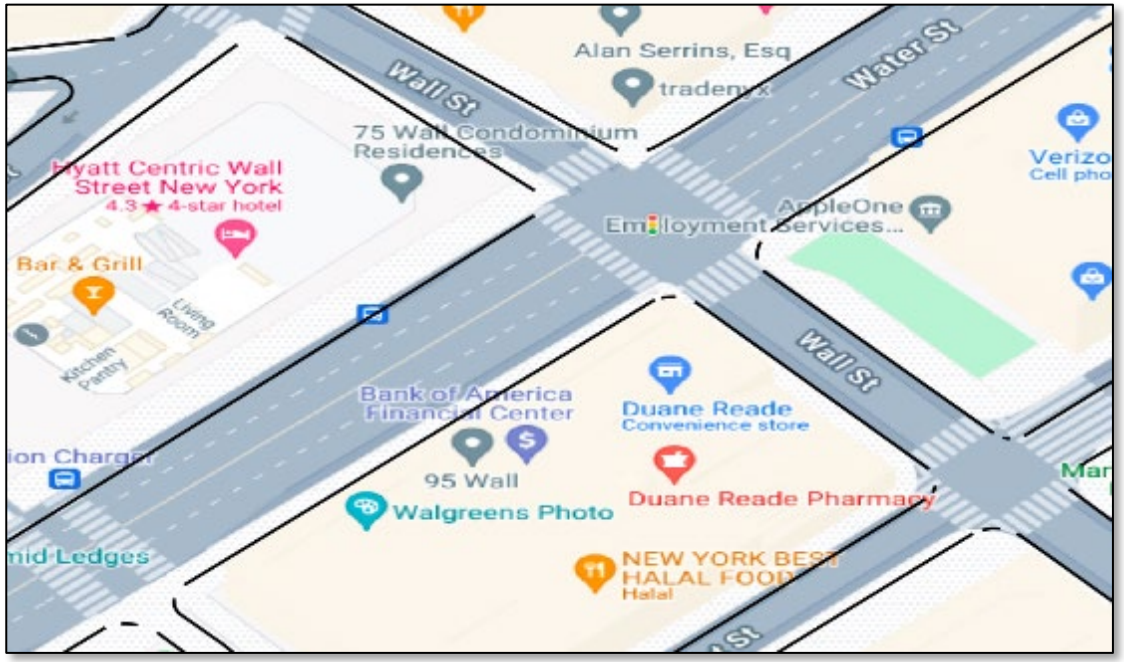
Methods: Buffering






```
169  -- Create a new table to store the buffered points
170  CREATE TABLE buffered_points AS
171  SELECT
172      id, -- Replace with your identifier column
173      ST_Buffer(newgeom, 10) AS buffered_geom
174  FROM
175      "DOT"."PedRamps";
176
```

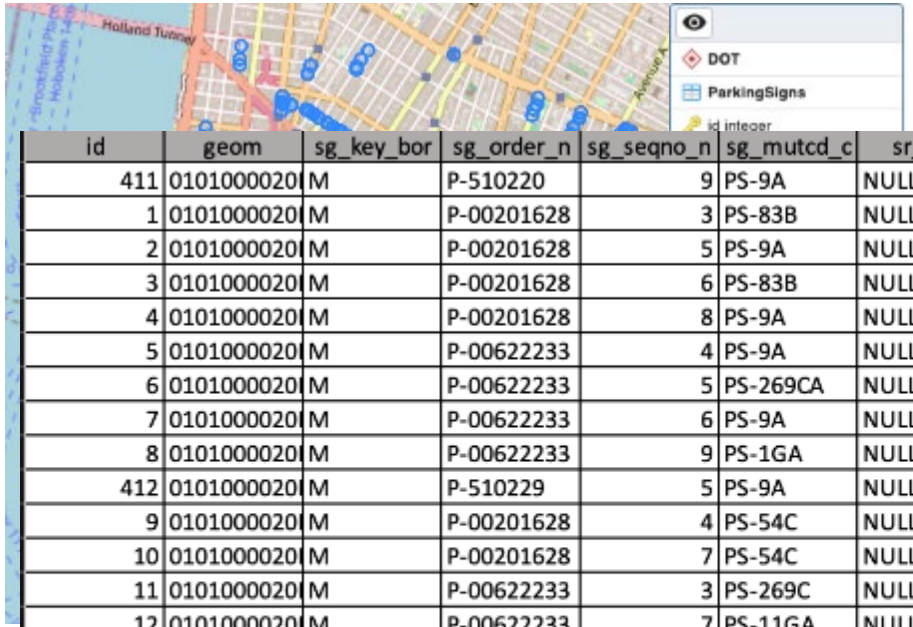
Methods: Clip




```
181  -- Create a new table to store the clipped line segments
182  CREATE TABLE clipped_segments AS
183  SELECT
184      cp.id AS segment_id,
185      bp.id AS point_id,
186      ST_Intersection(cp.newgeom, bp.buffered_geom) AS clipped_geom
187  FROM
188      "DOT"."ClippedPave" cp
189  JOIN
190      buffered_points bp ON ST_Intersects(cp.geom, bp.buffered_geom);
191
```

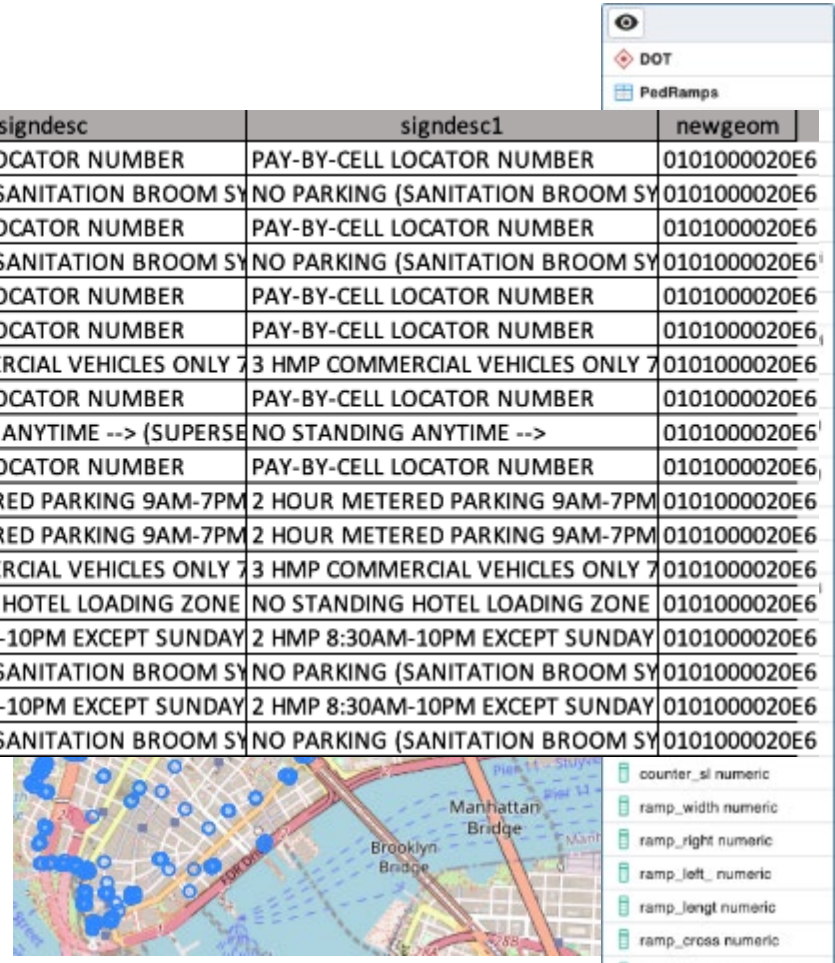
Powered by NYC OpenData

Variables and Assumptions



id	geom	sg_key_bor	sg_order_n	sg_seqno_n	sg_mutcd_c	sr_dist	sg_sign_fc	sg_arrow_d	x	y	signdesc	signdesc1	newgeom
411	0101000020	M	P-510220	9	PS-9A	NULL	NULL	NULL	983165	202507	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
1	0101000020	M	P-00201628	3	PS-83B	NULL	NULL	NULL	986732	200588	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6
2	0101000020	M	P-00201628	5	PS-9A	NULL	NULL	NULL	986732	200588	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
3	0101000020	M	P-00201628	6	PS-83B	NULL	NULL	NULL	986796	200563	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6
4	0101000020	M	P-00201628	8	PS-9A	NULL	NULL	NULL	986796	200563	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
5	0101000020	M	P-00622233	4	PS-9A	NULL	NULL	NULL	984197	201202	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
6	0101000020	M	P-00622233	5	PS-269CA	NULL	NULL	N	984154	201138	3 HMP COMMERCIAL VEHICLES ONLY 7	3 HMP COMMERCIAL VEHICLES ONLY 7	0101000020E6
7	0101000020	M	P-00622233	6	PS-9A	NULL	NULL	NULL	984154	201138	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
8	0101000020	M	P-00622233	9	PS-1GA	NULL	NULL	N	984092	201072	NO STANDING ANYTIME --> (SUPERSE	NO STANDING ANYTIME -->	0101000020E6
412	0101000020	M	P-510229	5	PS-9A	NULL	NULL	NULL	982959	202639	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
9	0101000020	M	P-00201628	4	PS-54C	NULL	NULL	NULL	986732	200588	2 HOUR METERED PARKING 9AM-7PM	2 HOUR METERED PARKING 9AM-7PM	0101000020E6
10	0101000020	M	P-00201628	7	PS-54C	NULL	NULL	NULL	986796	200563	2 HOUR METERED PARKING 9AM-7PM	2 HOUR METERED PARKING 9AM-7PM	0101000020E6
11	0101000020	M	P-00622233	3	PS-269C	NULL	NULL	NULL	984197	201202	3 HMP COMMERCIAL VEHICLES ONLY 7	3 HMP COMMERCIAL VEHICLES ONLY 7	0101000020E6
12	0101000020	M	P-00622233	7	PS-11GA	NULL	NULL	N	984135	201111	NO STANDING HOTEL LOADING ZONE	NO STANDING HOTEL LOADING ZONE	0101000020E6
13	0101000020	M	P-01253794	7	PS-174C	NULL	NULL	NULL	982219	196251	2 HMP 8:30AM-10PM EXCEPT SUNDAY	2 HMP 8:30AM-10PM EXCEPT SUNDAY	0101000020E6
14	0101000020	M	P-01253794	3	PS-167BA	NULL	NULL	N	982309	196333	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6
15	0101000020	M	P-01253794	4	PS-174CA	NULL	NULL	N	982309	196333	2 HMP 8:30AM-10PM EXCEPT SUNDAY	2 HMP 8:30AM-10PM EXCEPT SUNDAY	0101000020E6
16	0101000020	M	P-01253794	6	PS-167B	NULL	NULL	NULL	982219	196251	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6

Pavement Edge



Pedestrian Ramps

Powered by NYC Op

SQL Component

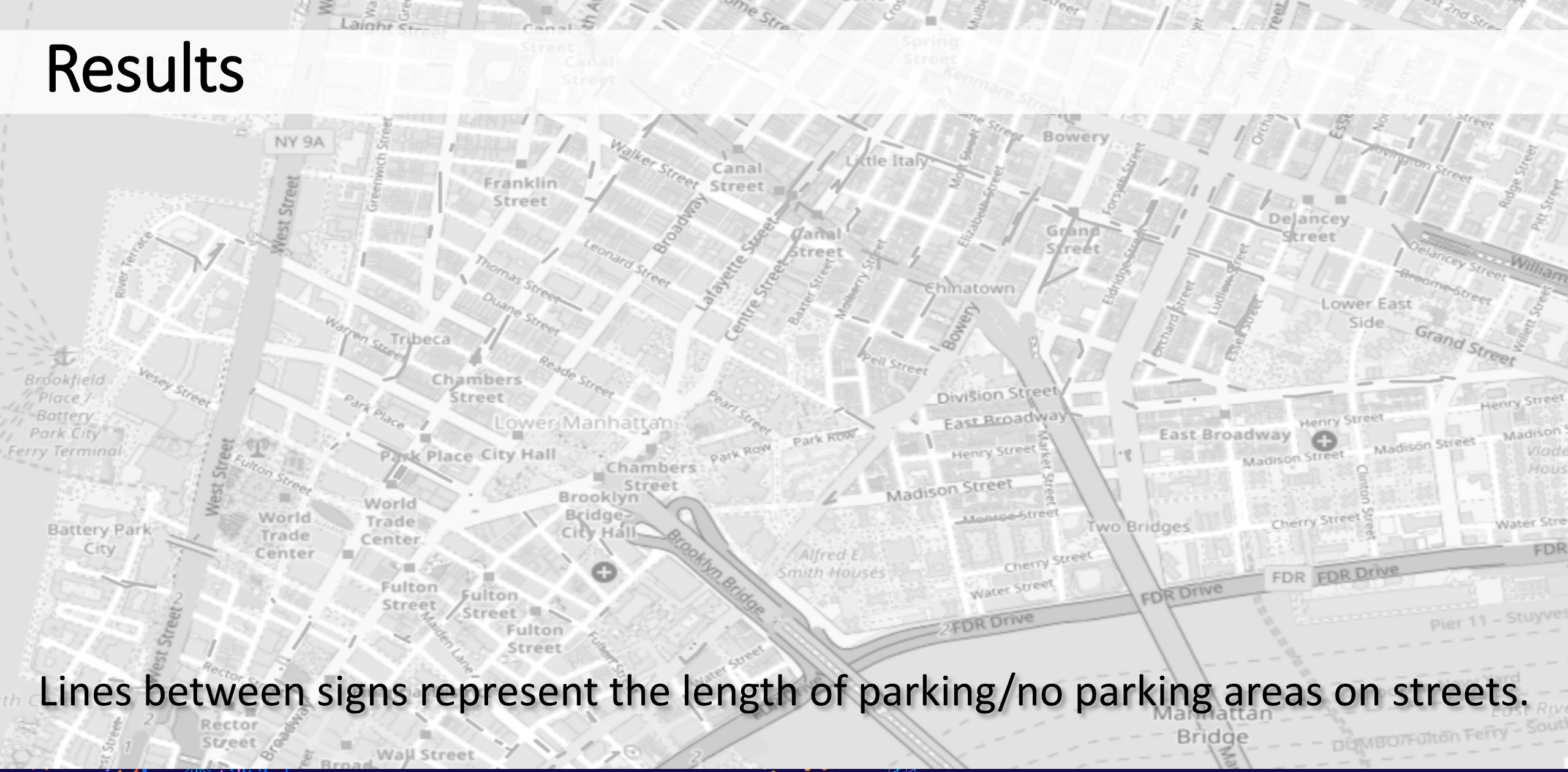
Methods: Connections

```sql
-----------------------------FINAL PROJECT-------------------------

SELECT * FROM "ClippedPave"
SELECT * FROM "CroppedPaveEdge"

ALTER TABLE "ClippedPave"
ADD COLUMN newgeom Geometry(MultiLineString, 4326);

UPDATE "ClippedPave"
SET newgeom = ST_Transform(geom, 4326);

SELECT * FROM "ParkingSigns"

ALTER TABLE "ParkingSigns"
ADD COLUMN newgeom Geometry(Point, 4326);

UPDATE "ParkingSigns"
SET newgeom = ST_Transform(geom, 4326);


--Spatial Index
CREATE INDEX parkingsigns_spatial_index
ON "ParkingSigns"
USING GIST(newgeom);

CREATE INDEX paveedge_spatial_index
ON "ClippedPave"
USING GIST(newgeom);
```

```sql
CREATE TABLE separated_lines2 AS (
    SELECT
        p1.id AS start_point_id,
        p2.id AS end_point_id,
        ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)) AS line_geometry
    FROM
        "ParkingSigns" p1
    JOIN LATERAL (
        SELECT
            p2.id,
            ST_Distance(p1.newgeom, p2.newgeom) AS distance,
            p2.newgeom AS closest_geom,
            MAX(p2.SG_ARROW_D) AS common_direction  -- Use MAX to get the common direction
        FROM
            "ParkingSigns" p2
        WHERE
            p1.id != p2.id
            AND p1.signdesc = p2.signdesc  -- Add condition for matching description
        GROUP BY
            p2.id, p2.newgeom
        ORDER BY
            ST_Distance(p1.newgeom, p2.newgeom)
        LIMIT 1
    ) p2 ON true
```

- Reprojected Geometry Column from ESPG:2263 to 4326
- Created Spatial Index to shorter query run time
- Joined Pavement Edge and Parking Regulations using new geometry column
- Ran a query that took the common direction of one point on the same line segment and connected the points in that same direction

Results

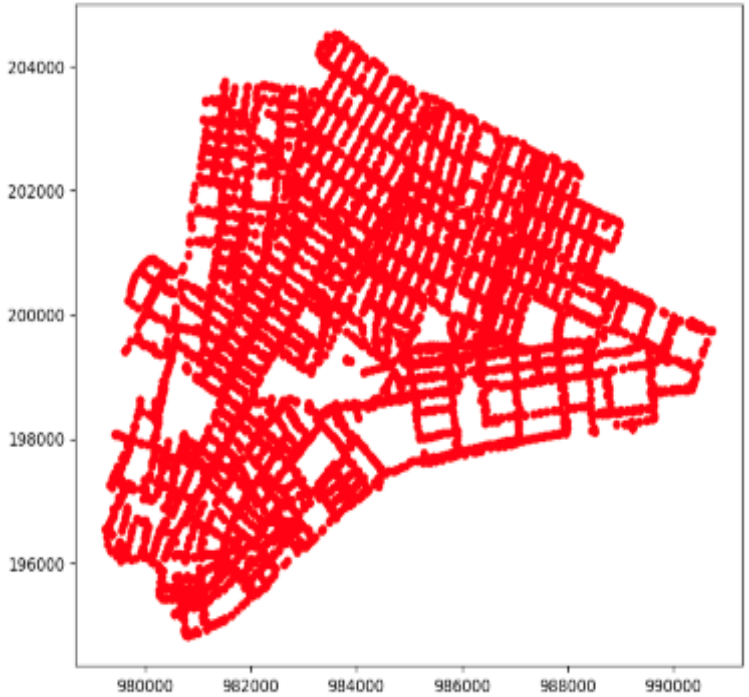


Lines between signs represent the length of parking/no parking areas on streets.

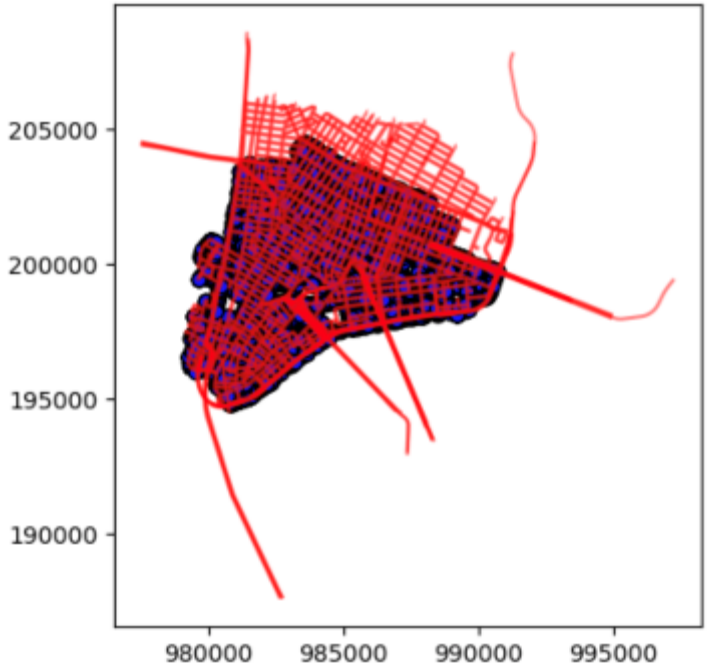
Python Component

Datasets

Pavement Edge



Parking Signs



Parking Signs overlayed on Pavement Edge

Method: Initial

CREATE LINES BETWEEN SIGNS USING PROXIMITY, TYPE, & DIRECTION

STEP 1: IMPORT NECESSCARY PACKAGES

[13]:
```
import pandas as pd
import geopandas as gpd
import gmplot
import matplotlib.pyplot as plt
import plotly.express as px

from shapely.geometry import MultiLineString, LineString, Point
from math import radians, sin, cos, sqrt, atan2
```

STEP 2: IMPORT DATA

[7]:
```
# IMPORT SHAPEFILES
Signs = gpd.read_file('/Users/jadamacharie/Downloads/DOT Internship/CommunityBoard1/SnappedPoints.shp')
print(Signs.columns.tolist())
print(Signs.crs)
# Signs.head()
Signs
```

[10]:
```
Edge = gpd.read_file('/Users/jadamacharie/Downloads/DOT Internship/CommunityBoard1/UseCroppedPaveEdge.shp')
Edge.head()
```

[10]:

	blockf_id	conflated	feat_code	shape_leng	source_id	status	sub_code	geometry
0	122260309.0	1.0	2260.0	298.393698	1.222600e+10	Unchanged	226000.0	LINESTRING (989638.410 198318.832, 989626.845 ...
1	122260709.0	1.0	2260.0	434.117494	1.222601e+10	Unchanged	226000.0	LINESTRING (983445.853 200097.374, 983440.315 ...
2	122260055.0	1.0	2260.0	31.256344	1.222600e+10	Updated	226000.0	LINESTRING (982061.120 202854.831, 982060.617 ...
3	122260516.0	1.0	2260.0	170.491174	1.222601e+10	Unchanged	226000.0	LINESTRING (983230.736 200965.158, 983233.711 ...
4	122260543.0	0.0	2260.0	10.151556	1.222601e+10	Unchanged	226000.0	LINESTRING (986882.476 201679.109, 986883.156 ...

STEP 3: SPATIALY CLEAN DATA

[11]:
```
# #CHANGE PROJECTIONS
Signs.crs
{'init': 'epsg:2263'}
print(Signs.crs)

Edge.crs
{'init': 'epsg:2263'}
Edge.crs
```

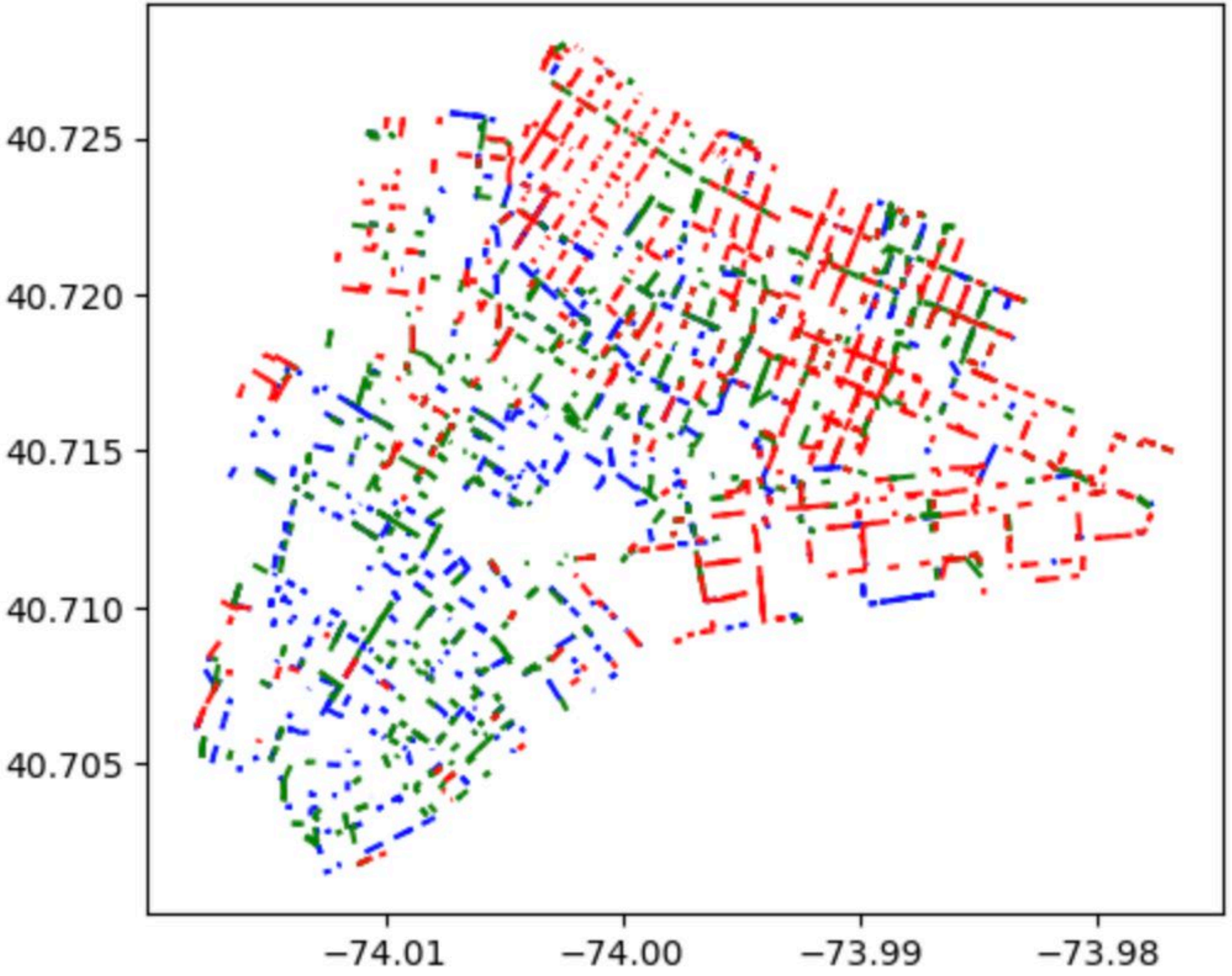
Method: Creating No Parking/ Parking Lengths

STEP 5: CREATE LINES BETWEEN SIGN DATA

[55]:

```python
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, select, text
from geoalchemy2 import Geometry

# Replace 'your_database_url' with the actual URL of your database
database_url = 'postgresql://jadanacharie:Animalplanet8@localhost:5432/postgres'
engine = create_engine(database_url)

# Define the metadata and the table
metadata = MetaData()

separated_lines2 = Table(
    'separated_lines2', metadata,
    Column('start_point_id', Integer),
    Column('end_point_id', Integer),
    Column('line_geometry', Geometry('LINESTRING')),
)

sql_query = """
INSERT INTO "DOT"."separated_lines2" (start_point_id, end_point_id, line_geometry)
SELECT
    p1.id AS start_point_id,
    p2.id AS end_point_id,
    ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)) AS line_geometry
FROM
    "DOT"."ParkingSigns" p1
JOIN LATERAL (
    SELECT
        p2.id,
        ST_Distance(p1.newgeom, p2.newgeom) AS distance,
        p2.newgeom AS closest_geom,
        MAX(p2.SG_ARROW_D) AS common_direction
    FROM
        "DOT"."ParkingSigns" p2
        MAX(p2.SG_ARROW_D) AS common_direction
    FROM
        "DOT"."ParkingSigns" p2
    WHERE
        p1.id != p2.id
        AND p1.signdesc = p2.signdesc
    GROUP BY
        p2.id, p2.newgeom
    ORDER BY
        ST_Distance(p1.newgeom, p2.newgeom)
    LIMIT 1
) p2 ON true
WHERE
    ST_DWithin(p1.newgeom, ST_Transform(p2.closest_geom, 4326), 0.001)
    AND NOT EXISTS (
        SELECT 1
        FROM "DOT"."ClippedPave" s
        WHERE ST_Intersects(ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)), ST_Transform(s.geom, 4326))
    )
    AND NOT EXISTS (
        SELECT 1
        FROM "DOT"."separated_lines2" sl2
        WHERE sl2.start_point_id = p1.id
        AND sl2.end_point_id = p2.id
    )
"""

# Execute the query
with engine.connect() as connection:
    result = connection.execute(text(sql_query))

# Commit the changes
metadata.create_all(engine)
```

Results



Discussion

Limitations:

- Transforming my Parking Signs to line data types
- Negating fire hydrants, Bus stops, loading docks, etc.
- Some of the Pedestrian Ramps are not located on the corners of the streets
- Creating lines between Parking Signs using Pavement Edge
- Converting all Parking Signs to lines
- Clipping Pavement Edge by Sign Points

Next Steps:

- Create web front-end application for data visualization

Thank you for participating in Open Data Week!

jada.macharie96@myhunter.cuny.edu
https://github.com/Jada68/DOT-Parking-Regulations

QUESTIONS?

Join us at more events through Sunday, March 24. View the program at open-data.nyc.