

# Where to Park?

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

Jada Macharie, Graduate Student  
Department of Geography, Hunter College, CUNY

2024 Annual Meeting of the  
Transportation Research Board  
9 January 2024

# Background



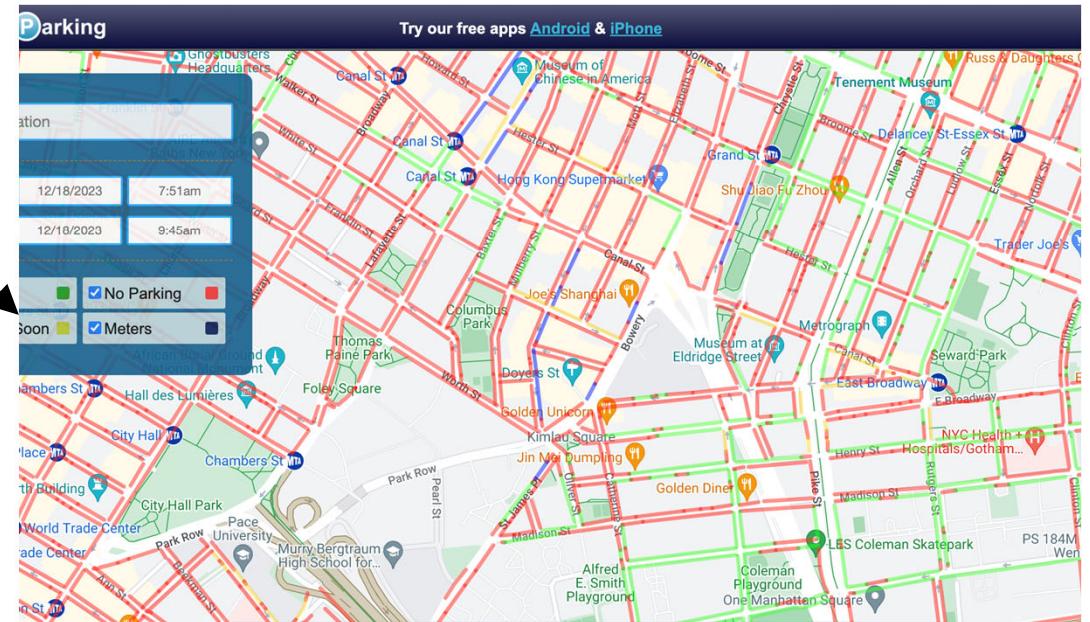
The New York City Department of Transportation is responsible for installing most of the **signage** in the five boroughs of NYC.

<https://data.cityofnewyork.us/Transportation/Parking-Regulation-Locations-and-Signs/xswq-wnv9>

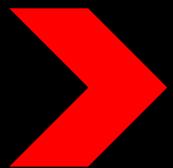




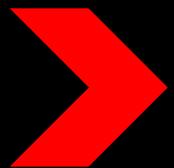
The **data** used to create DOT's interactive map, will be used to a **dynamic map** that is like smooth parking's web map.



Scope



Datasets



Software

CB1, Financial District



NYC Open Data

Pavement Edge  
Pedestrian Ramps  
Parking Regulations



# QGIS COMPONENT

# Methods: Snap Points

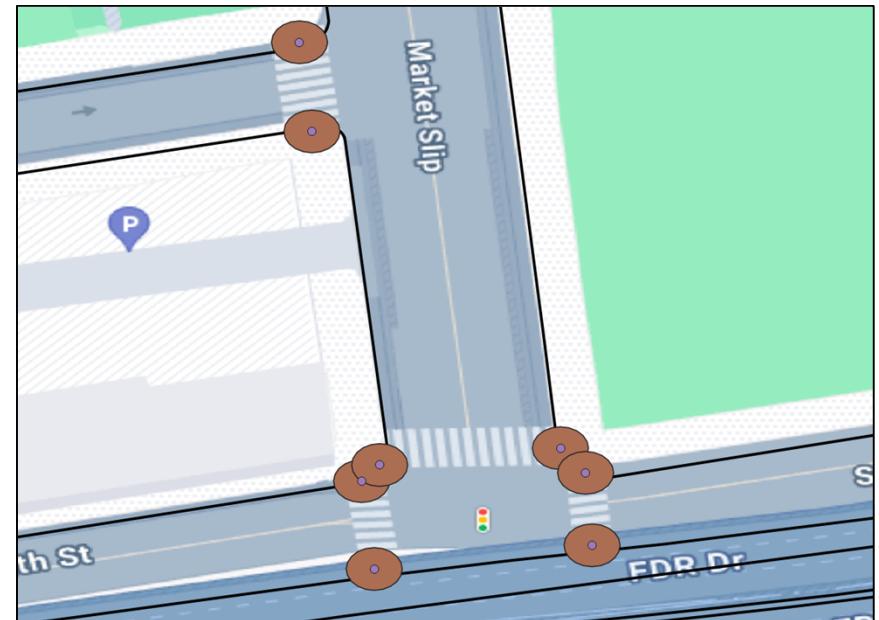
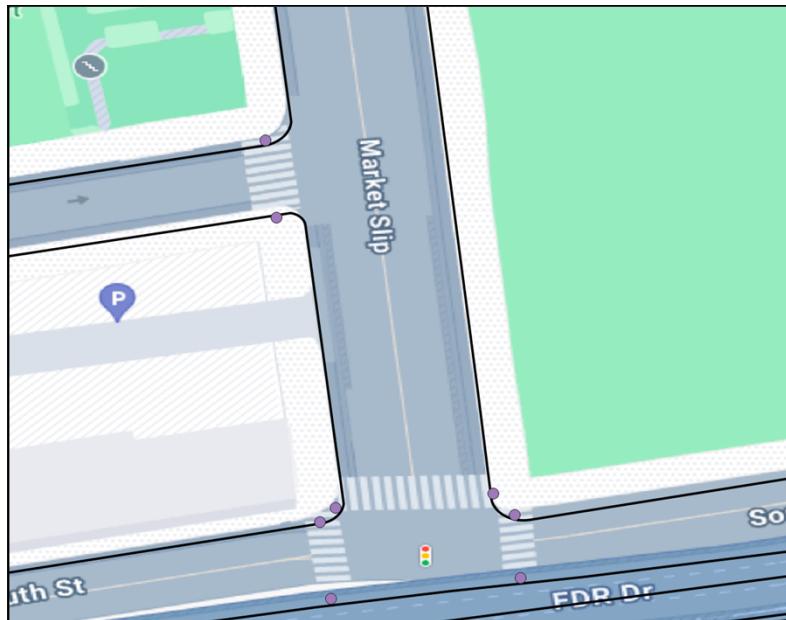


# Methods: Snap Points

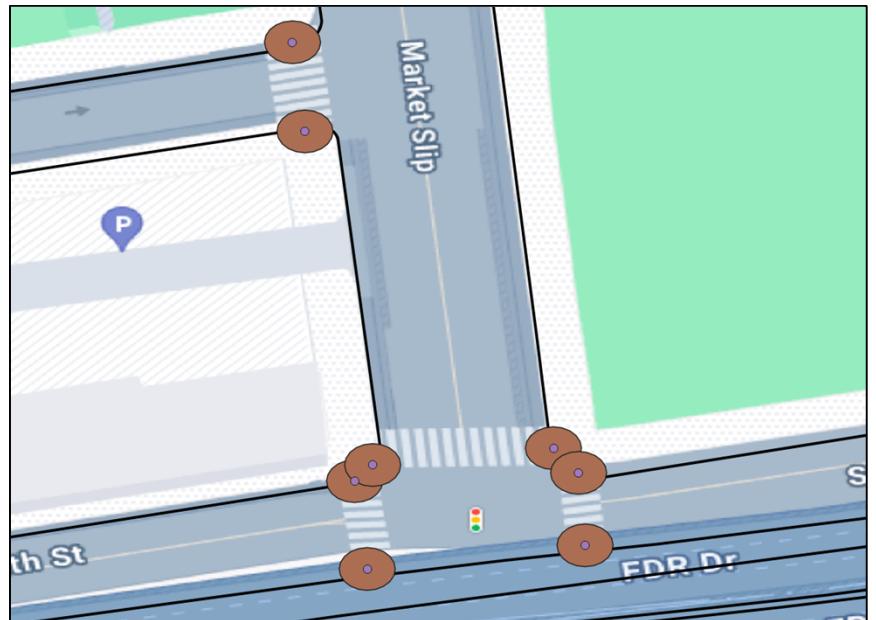
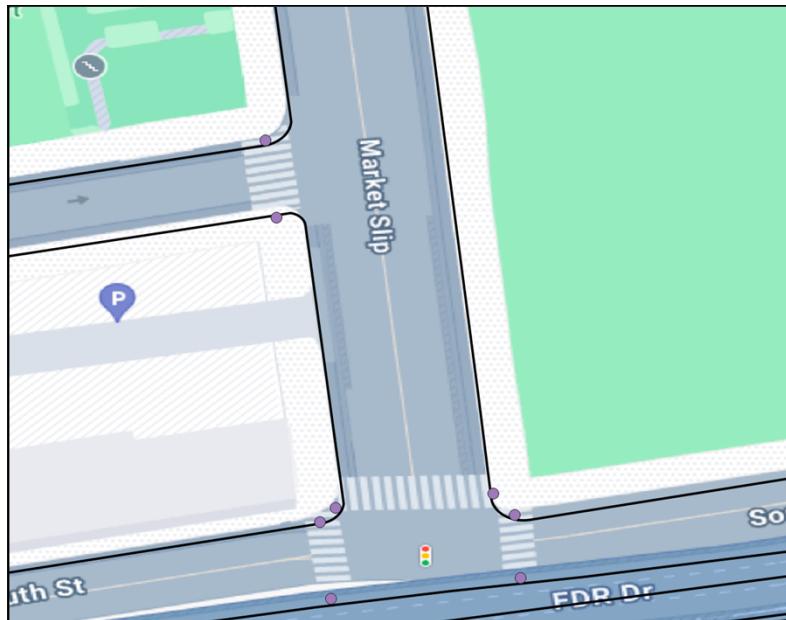


```
156 UPDATE "DOT"."ParkingSigns" AS points  
157 SET geom = ST_ClosestPoint("DOT".ClippedPave".geom, points.geom)  
158 FROM "DOT"."ClippedPave" AS line  
159 WHERE ST_DWithin(line.geom, points.geom, 0.001);  
160
```

# Methods: Buffering

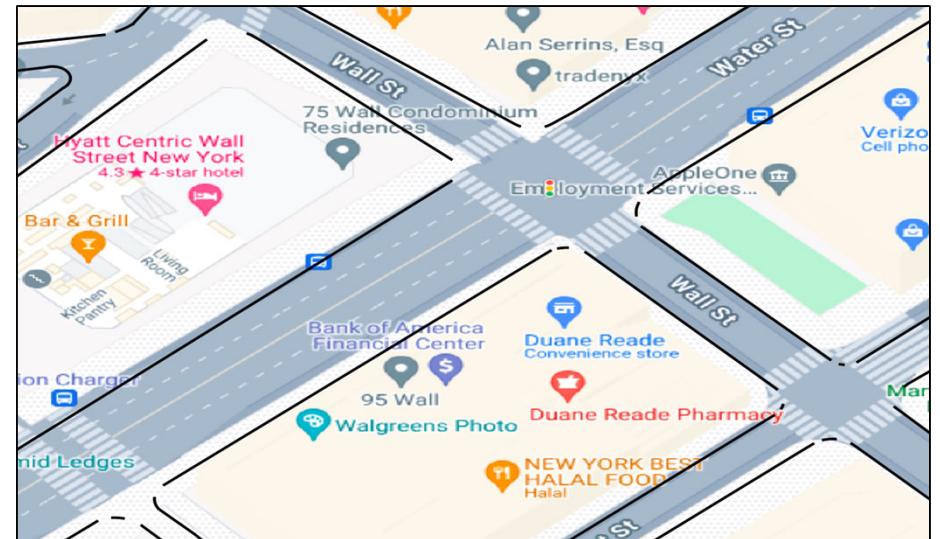
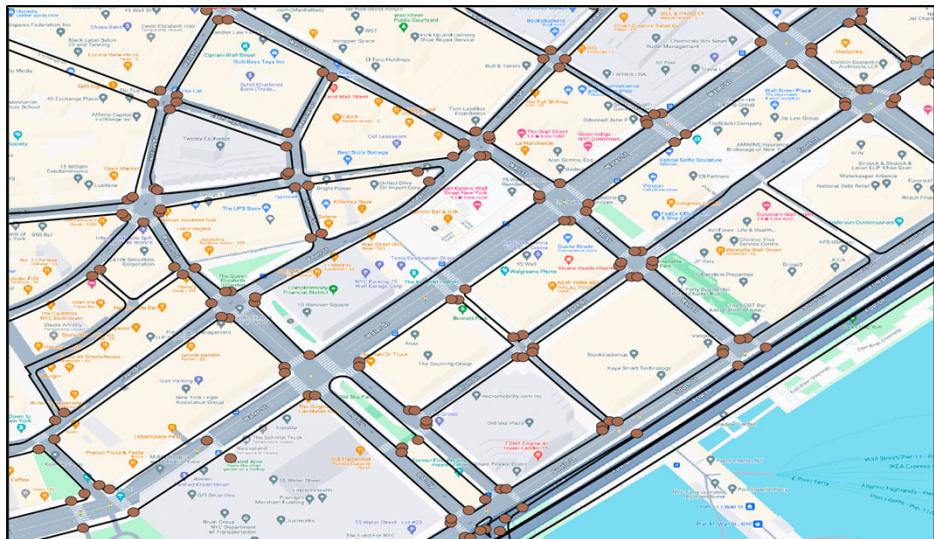


# Methods: Buffering

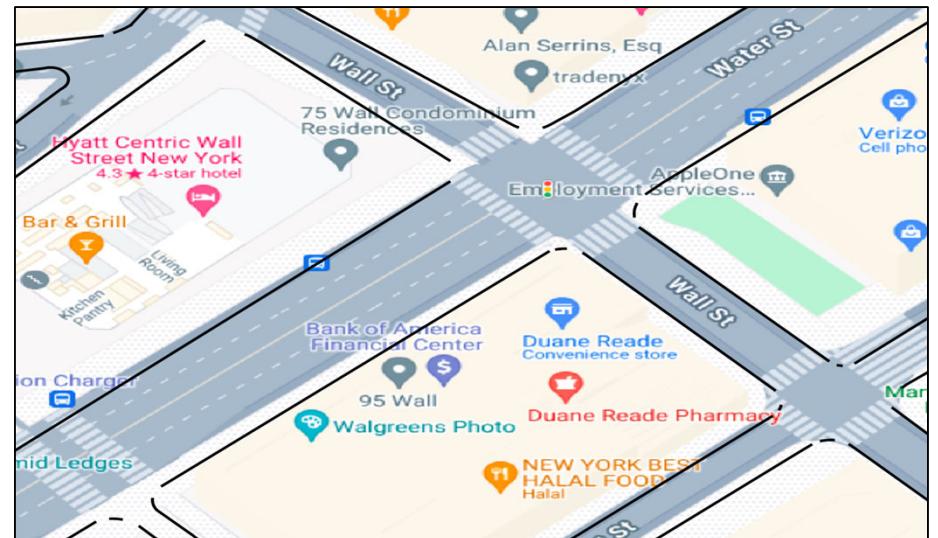
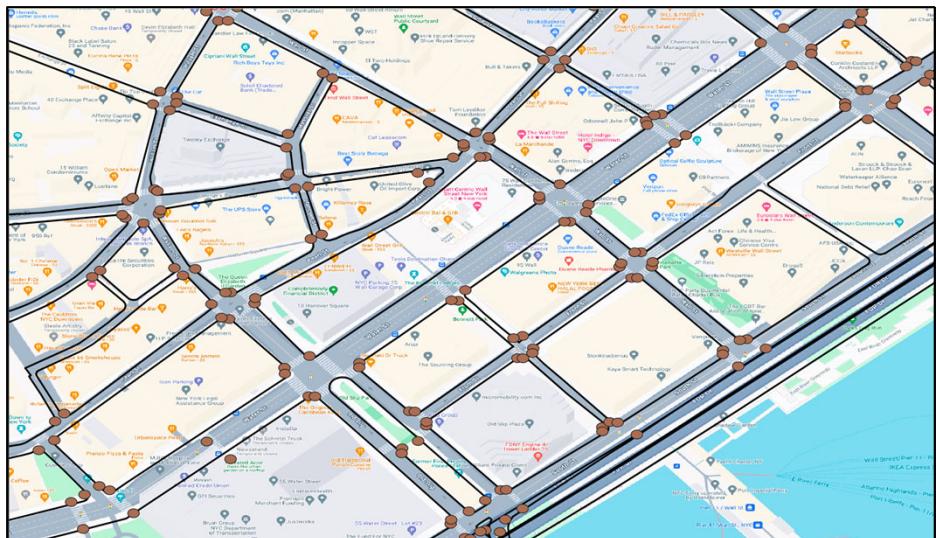


```
169 -- Create a new table to store the buffered points
170 CREATE TABLE buffered_points AS
171 SELECT
172     id, -- Replace with your identifier column
173     ST_Buffer(newgeom, 10) AS buffered_geom
174 FROM
175     "DOT"."PedRamps";
176
```

# Methods: Clip



# Methods: Clip

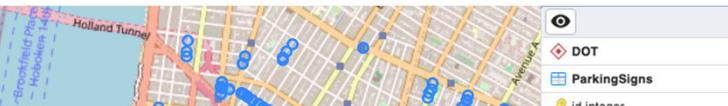


```
181 -- Create a new table to store the clipped line segments
182 CREATE TABLE clipped_segments AS
183 SELECT
184     cp.id AS segment_id,
185     bp.id AS point_id,
186     ST_Intersection(cp.newgeom, bp.buffered_geom) AS clipped_geom
187 FROM
188     "DOT"."ClippedPave" cp
189 JOIN
190     buffered_points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

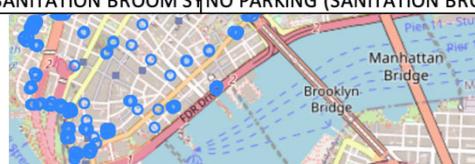
# SQL COMPONENT

# Variables and Assumptions

**Pavement Edge**



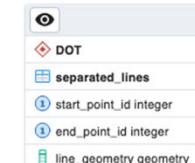
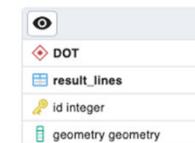
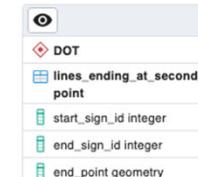
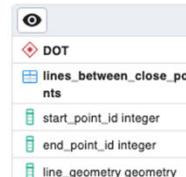
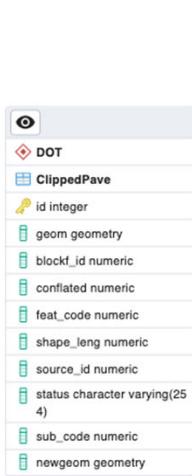
**Pedestrian Ramps**



id	geom	sg_key_bor	sg_order_n	sg_seqno_n	sg_muted_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	3 HMP COMMERCIAL VEHICLES ONLY	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	3 HMP COMMERCIAL VEHICLES ONLY	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

**counter\_sl numeric**  
**ramp\_width numeric**  
**ramp\_right numeric**  
**ramp\_left\_ numeric**  
**ramp\_lengt numeric**  
**ramp\_cross numeric**  
**ponding character varying (254)**  
**obstacles\_ character varying(254)**  
**obstacle\_2 character varying(254)**

# Database Analysis



# Database Analysis

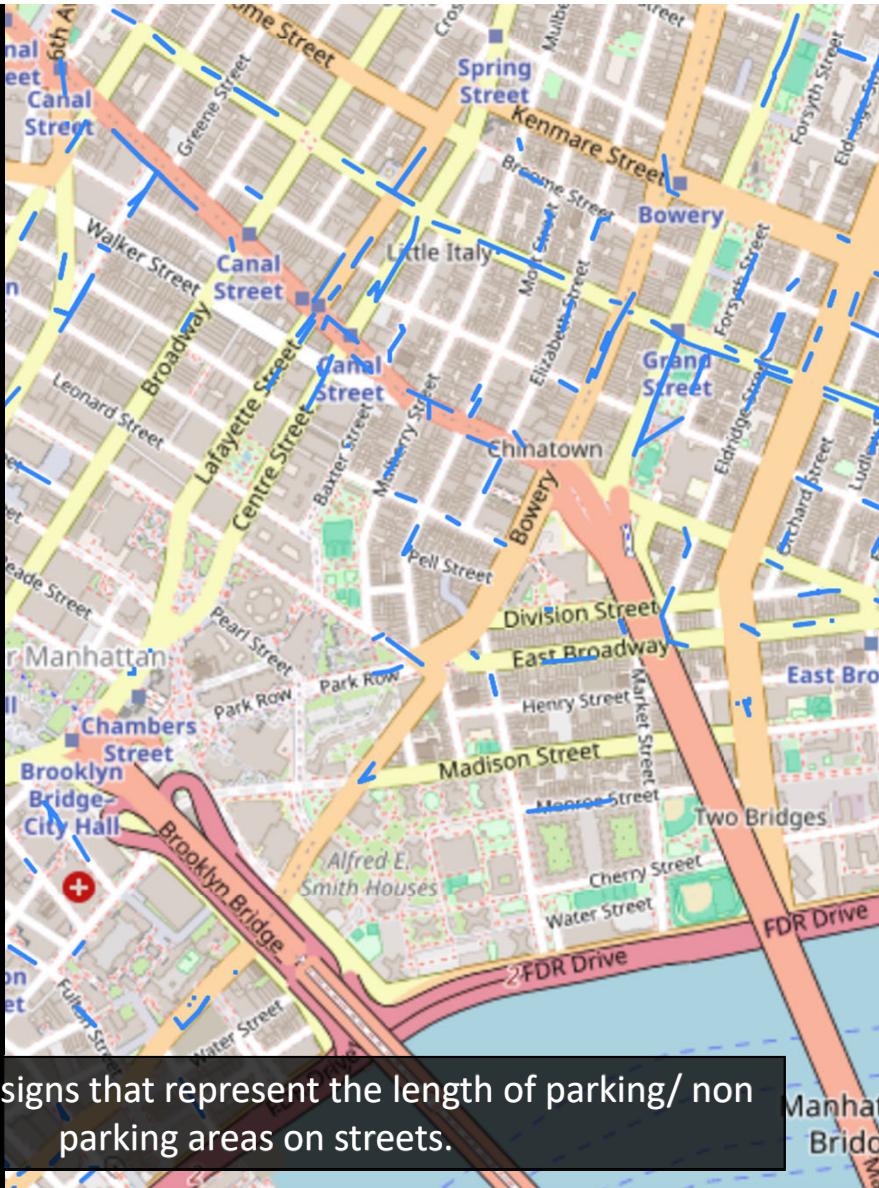
# Methods: Connections

```
1      --FINAL PROJECT--  
2  
3  SELECT * FROM "ClippedPave"  
4  SELECT * FROM "CroppedPaveEdge"  
5  
6  ALTER TABLE "ClippedPave"  
7  ADD COLUMN newgeom Geometry(MultiLineString, 4326);  
8  
9  UPDATE "ClippedPave"  
10 SET newgeom = ST_Transform(geom, 4326);  
11  
12 SELECT * FROM "ParkingSigns"  
13  
14 ALTER TABLE "ParkingSigns"  
15 ADD COLUMN newgeom Geometry(Point, 4326);  
16  
17 UPDATE "ParkingSigns"  
18 SET newgeom = ST_Transform(geom, 4326);  
19  
20  
  
21 --Spatial Index  
22 CREATE INDEX parkingsigns_spatial_index  
23 ON "ParkingSigns"  
24 USING GIST(newgeom);  
25  
26 CREATE INDEX paveedge_spatial_index  
27 ON "ClippedPave"  
28 USING GIST(newgeom);
```

- 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

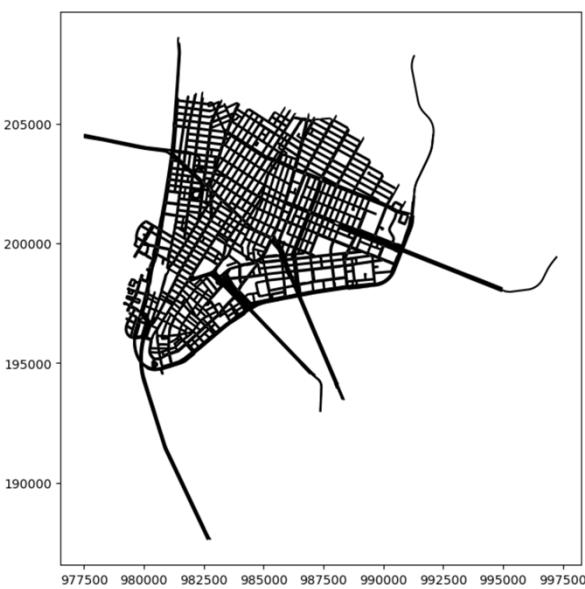
```
107 CREATE TABLE separated_lines2 AS (  
108   SELECT  
109     p1.id AS start_point_id,  
110     p2.id AS end_point_id,  
111     ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)) AS line_geometry  
112   FROM  
113     "ParkingSigns" p1  
114   JOIN LATERAL (  
115     SELECT  
116       p2.id,  
117       ST_Distance(p1.newgeom, p2.newgeom) AS distance,  
118       p2.newgeom AS closest_geom,  
119       MAX(p2.SG_ARROW_D) AS common_direction -- Use MAX to get the common direction  
120   FROM  
121     "ParkingSigns" p2  
122   WHERE  
123     p1.id != p2.id  
124     AND p1.signdesc = p2.signdesc -- Add condition for matching description  
125   GROUP BY  
126     p2.id, p2.newgeom  
127   ORDER BY  
128     ST_Distance(p1.newgeom, p2.newgeom)  
129   LIMIT 1  
130 ) p2 ON true  
131 WHERE  
132   ST_DWithin(p1.newgeom, ST_Transform(p2.closest_geom, 4326), 0.001)  
133   AND NOT EXISTS (  
134     SELECT 1  
135     FROM "ClippedPave" s  
136     WHERE ST_Intersects(ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)), ST_Transform(s.geom, 4326))  
137   );  
138 );  
139
```

# Results



# PYTHON COMPONENT

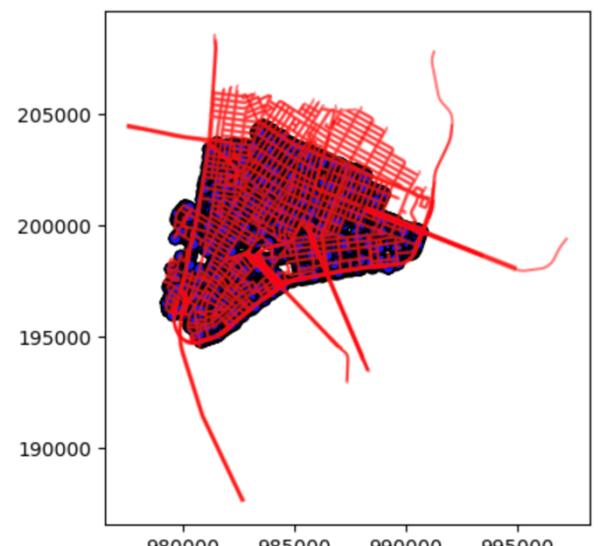
# Datasets



Pavement Edge



Parking Signs



Parking Signs overlayed on  
Pavement Edge

## CREATE LINES BETWEEN SIGNS USING PROXIMITY, TYPE, & DIRECTION

### STEP 1: IMPORT NECESSARY PACKAGES

```
[13]: import pandas as pd
import geopandas as gpd
import gplot
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()
```

	blockf_id	conflated	feat_code	shape_leng	source_id	status	sub_code	geometry
0	122260309.0	1.0	2260.0	296.393696	1.222600e+10	Unchanged	226000.0	LINESTRING (989639.410 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	122260518.0	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 (985882.476 201679.109, 985883.156 ...)

### STEP 3: SPATIALLY CLEAN DATA

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

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

# Method: Initial Steps

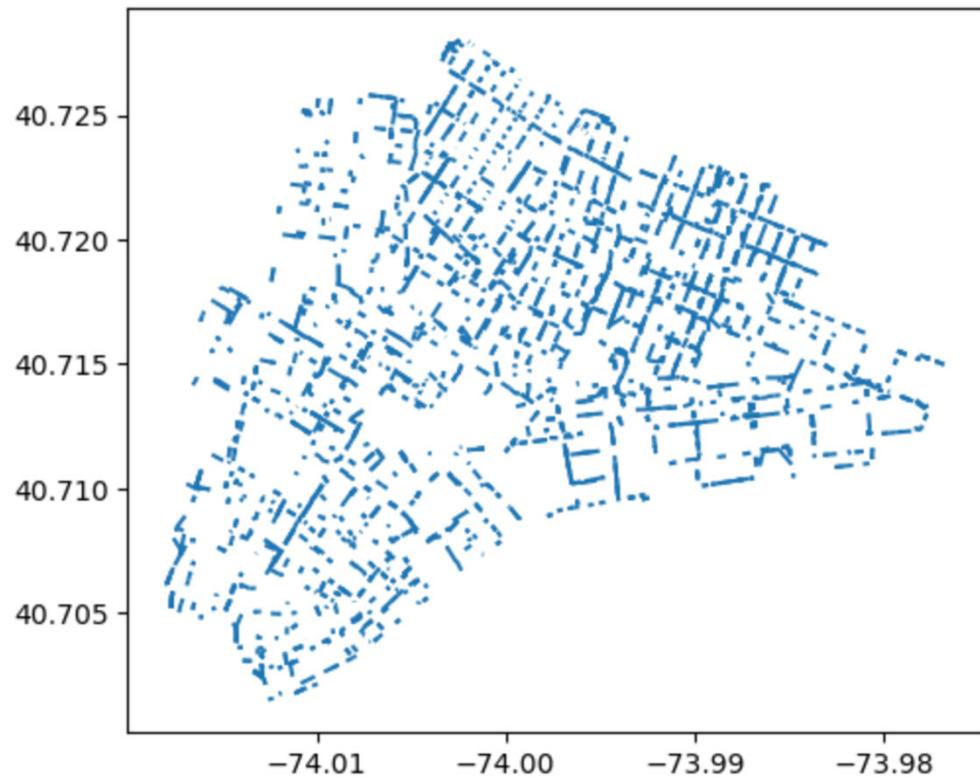
# Method: Creating No Parking/ Parking Lengths Between Signs

## STEP 5: CREATE LINES BETWEEN SIGN DATA

```
[55]: 1 from sqlalchemy import create_engine, MetaData, Table, Column, Integer, select, text
2 from geoalchemy2 import Geometry
3
4 # Replace 'your_database_url' with the actual URL of your database
5 database_url = "REDACTED"
6 engine = create_engine(database_url)
7
8 # Define the metadata and the table
9 metadata = MetaData()
10
11 separated_lines2 = Table(
12     'separated_lines2', metadata,
13     Column('start_point_id', Integer),
14     Column('end_point_id', Integer),
15     Column('line_geometry', Geometry('LINESTRING')),
16 )
17
18 sql_query = """
19 INSERT INTO "DOT"."separated_lines2" (start_point_id, end_point_id, line_geometry)
20 SELECT
21     p1.id AS start_point_id,
22     p2.id AS end_point_id,
23     ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)) AS line_geometry
24 FROM
25     "DOT"."ParkingSigns" p1
26 JOIN LATERAL (
27     SELECT
28         p2.id,
29         ST_Distance(p1.newgeom, p2.newgeom) AS distance,
30         p2.newgeom AS closest_geom,
31         MAX(p2.SG_ARROW_D) AS common_direction
32     FROM
33         "DOT"."ParkingSigns" p2
34
35     WHERE
36         p1.id != p2.id
37         AND p1.signdesc = p2.signdesc
38     GROUP BY
39         p2.id, p2.newgeom
40     ORDER BY
41         ST_Distance(p1.newgeom, p2.newgeom)
42     LIMIT 1
43 ) p2 ON true
44 WHERE
45     ST_DWithin(p1.newgeom, ST_Transform(p2.closest_geom, 4326), 0.001)
46     AND NOT EXISTS (
47         SELECT 1
48         FROM "DOT"."ClippedPave" s
49         WHERE ST_Intersects(ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)), ST_Transform(s.geom, 4326))
50     )
51     AND NOT EXISTS (
52         SELECT 1
53         FROM "DOT"."separated_lines2" s12
54         WHERE s12.start_point_id = p1.id
55             AND s12.end_point_id = p2.id
56     )
57
58 # Execute the query
59 with engine.connect() as connection:
60     result = connection.execute(text(sql_query))
61
62 # Commit the changes
63 metadata.create_all(engine)
```

```
31     MAX(p2.SG_ARROW_D) AS common_direction
32
33     FROM
34         "DOT"."ParkingSigns" p2
35     WHERE
36         p1.id != p2.id
37         AND p1.signdesc = p2.signdesc
38     GROUP BY
39         p2.id, p2.newgeom
40     ORDER BY
41         ST_Distance(p1.newgeom, p2.newgeom)
42     LIMIT 1
43 ) p2 ON true
44 WHERE
45     ST_DWithin(p1.newgeom, ST_Transform(p2.closest_geom, 4326), 0.001)
46     AND NOT EXISTS (
47         SELECT 1
48         FROM "DOT"."ClippedPave" s
49         WHERE ST_Intersects(ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)), ST_Transform(s.geom, 4326))
50     )
51     AND NOT EXISTS (
52         SELECT 1
53         FROM "DOT"."separated_lines2" s12
54         WHERE s12.start_point_id = p1.id
55             AND s12.end_point_id = p2.id
56     )
57
58 # Execute the query
59 with engine.connect() as connection:
60     result = connection.execute(text(sql_query))
61
62 # Commit the changes
63 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

- **What did not work:**

- Creating lines between **Parking Signs** using **Pavement Edge**
- Converting all **Parking Signs** to lines
- Clipping **Pavement Edge** by **Sign Points**

- **Next Steps:**

- Color Code line connections based on regulation types/descriptions



[jada.macharie96@myhunter.cuny.edu](mailto:jada.macharie96@myhunter.cuny.edu)

QUESTIONS?

