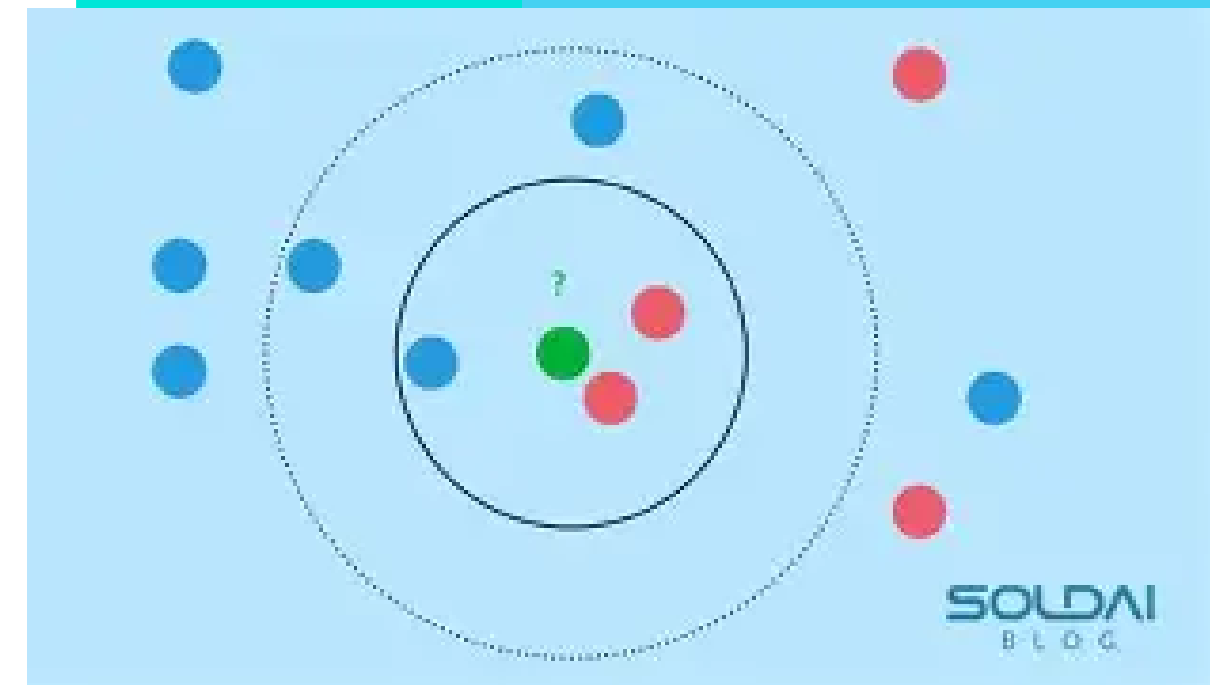


BÚSQUEDA DE VECINOS CERCANOS EN SQL

ST_DISTANCE

<->



Juan Sebastián Rodríguez Ovalle -
20201025127

CONTENIDOS

1. **Introducción**
2. **Metodología en SQL**
 - a. **Ejemplos guía y básicos**
 - b. **Uso de búsqueda de vecinos cercanos con IA**
3. **Modelo Físico, conceptual y lógico**
4. **Ejemplos con BDE Tunjuelito**

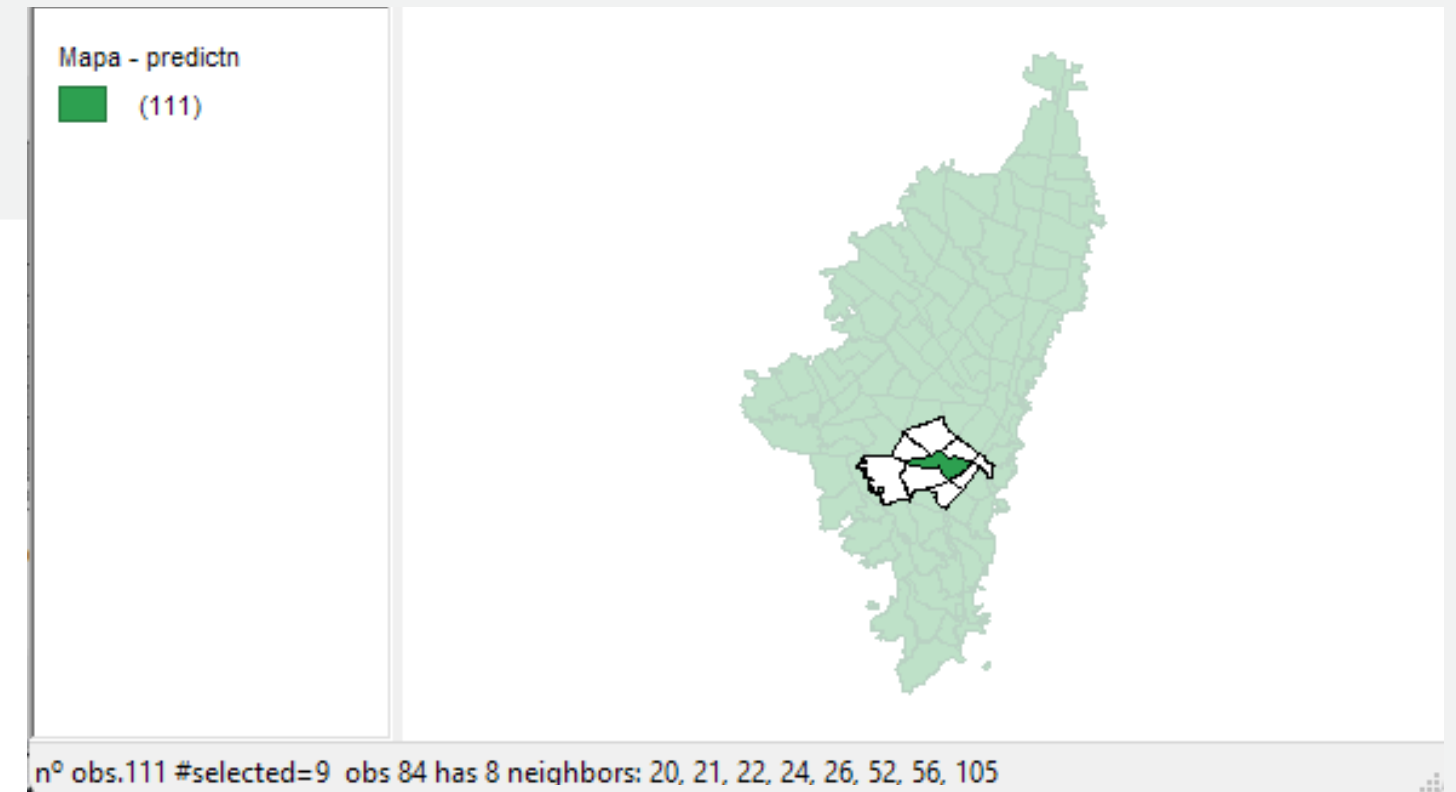
INTRODUCCIÓN

Consiste en encontrar el o los puntos en un conjunto de datos que están más próximos a un objeto de consulta propuesto.

Se determina la **cercanía** mediante una **función de distancia**, como euclidiana, geodésica, etc.

Se utilizan distintos criterios para establecer los vecinos cercanos

- Contigüidad.
- Knn.
- Funciones Kernels.



Pesos de Contigüidad

☒ Contigüidad Reina

☐ Contigüidad Torre

☐ Precisión umbral

☐ Block weights

Orden de contigüidad

☐ Incluir órdenes inferiores

Pesos de Distancias

Variables:

Geometric centroids

Variables

Variable coordenada-X <X-Centroids>

Variable coordenada-Y <Y-Centroids>

Distancia metrica Euclidean Distance

Método:

Banda de Distancia K-vecinos Más Cercanos Kernel

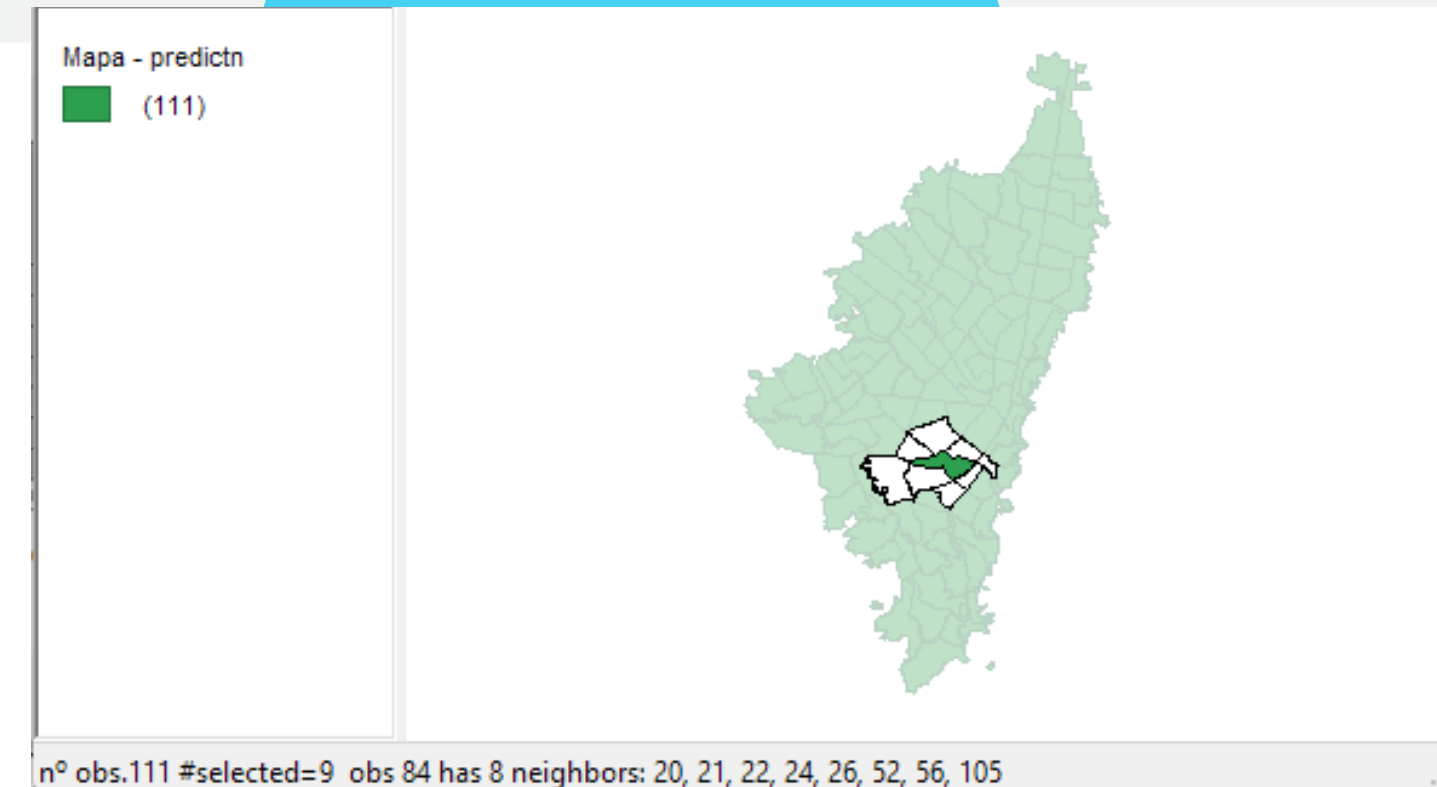
Especificar ancho de banda

☐ ¿Usar distancia inversa? Potencia

INTRODUCCIÓN

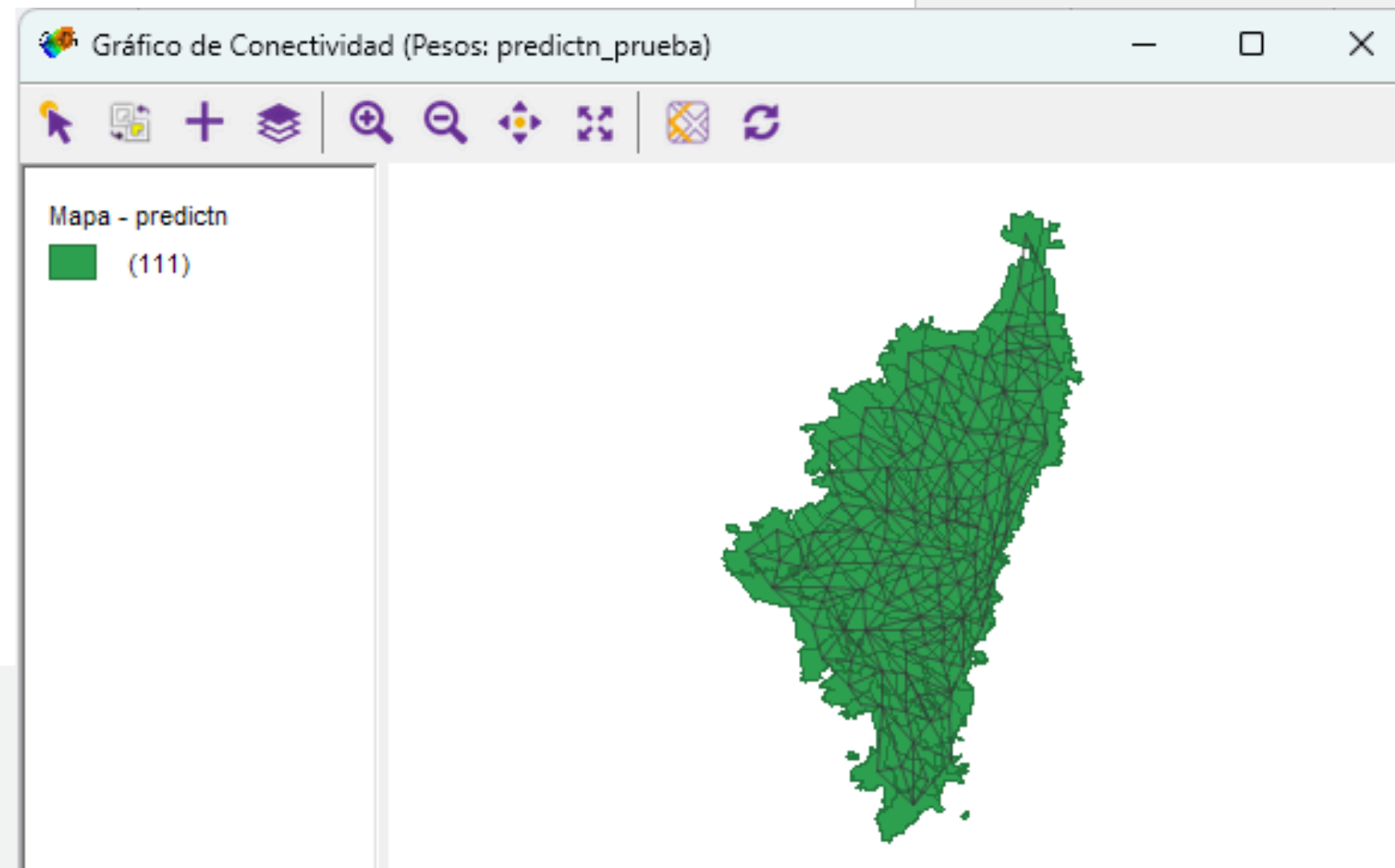
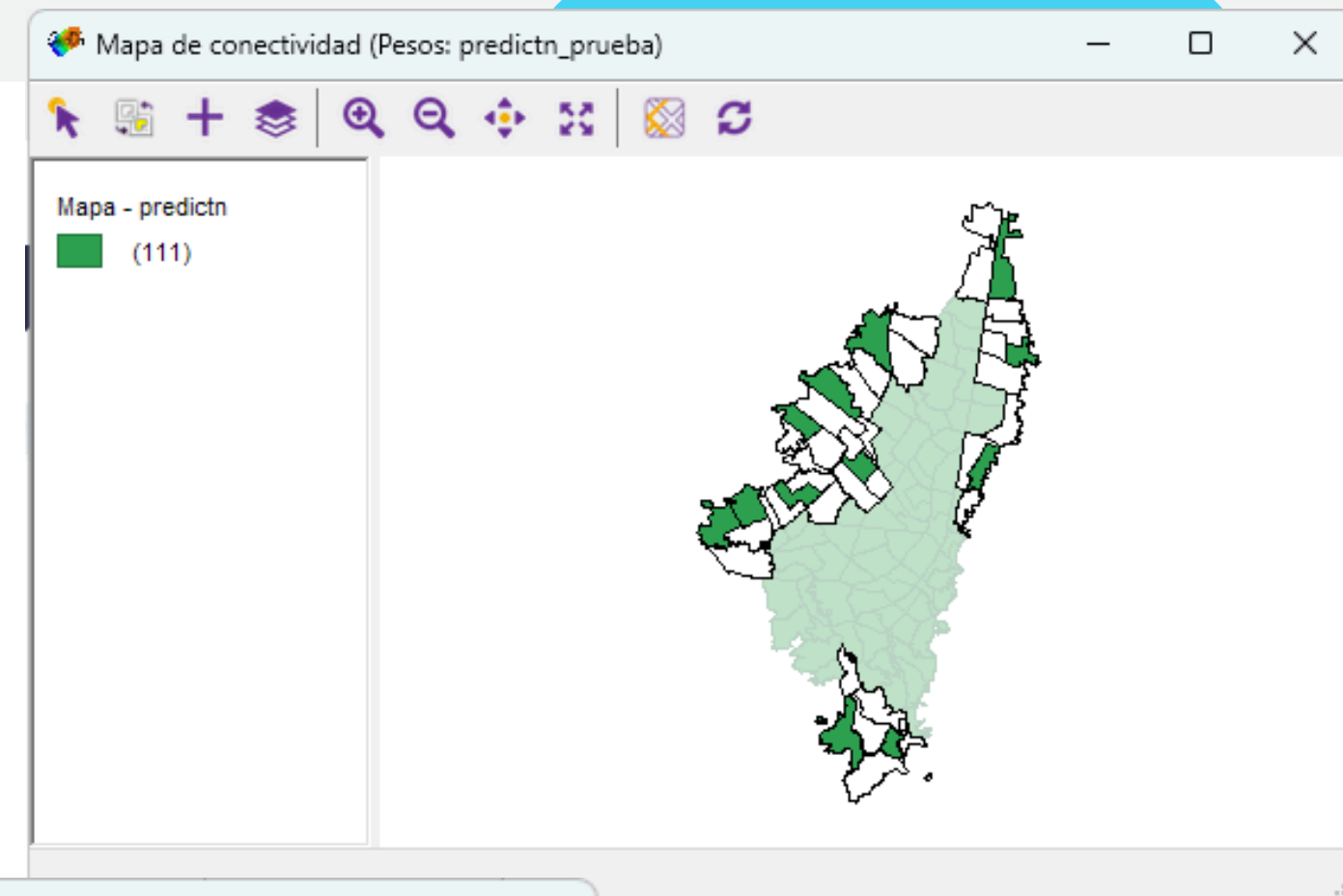
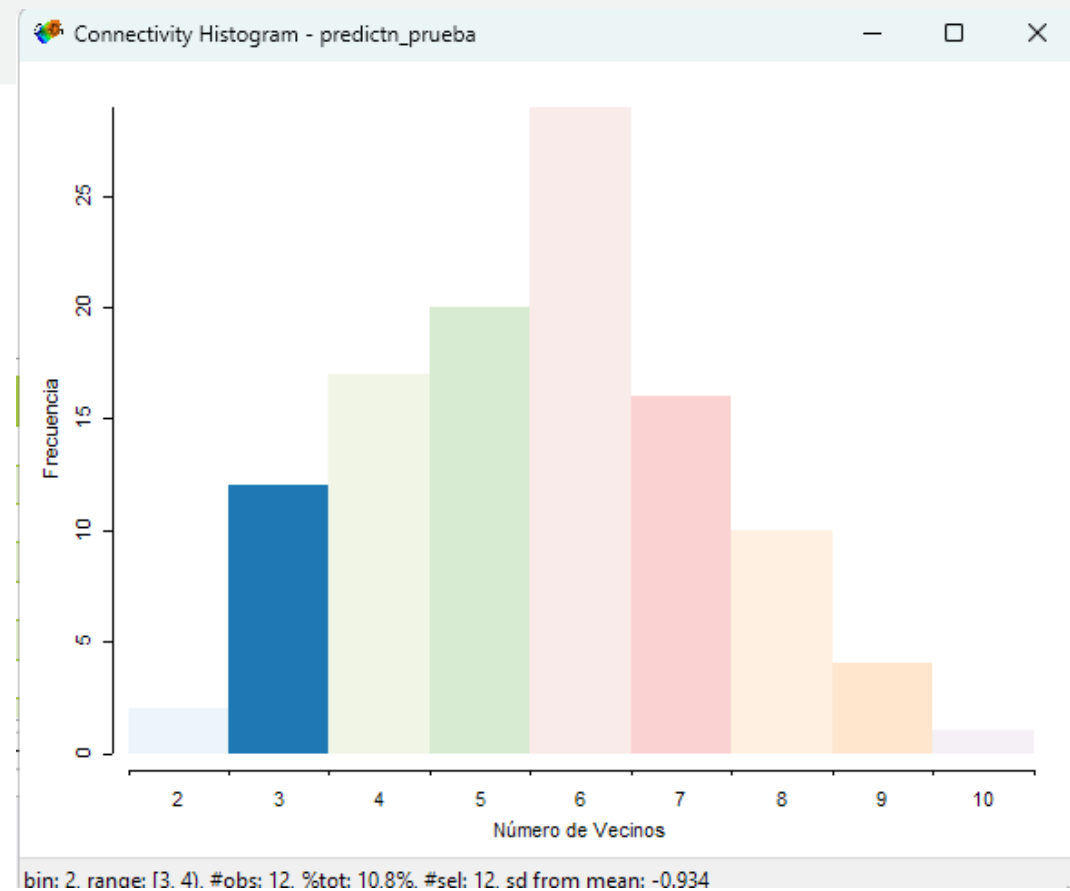
Es un problema fundamental en:

- Ciencia de la computación.
- **Análisis de datos.**
- **Aprendizaje automático.**
- Visión por computadora.
- Sistemas de información geográfico.



	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	1	0
3	1	1	0	0	1	1
4	0	1	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0

INTRODUCCIÓN



METODOLOGÍA EN SQL

1. Representación de datos

1.1 Verificar que tengan características numéricas o datos geométricos.

2. Cálculo de la distancia

2.1. Para datos numéricos

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2.2. Para datos geométricos

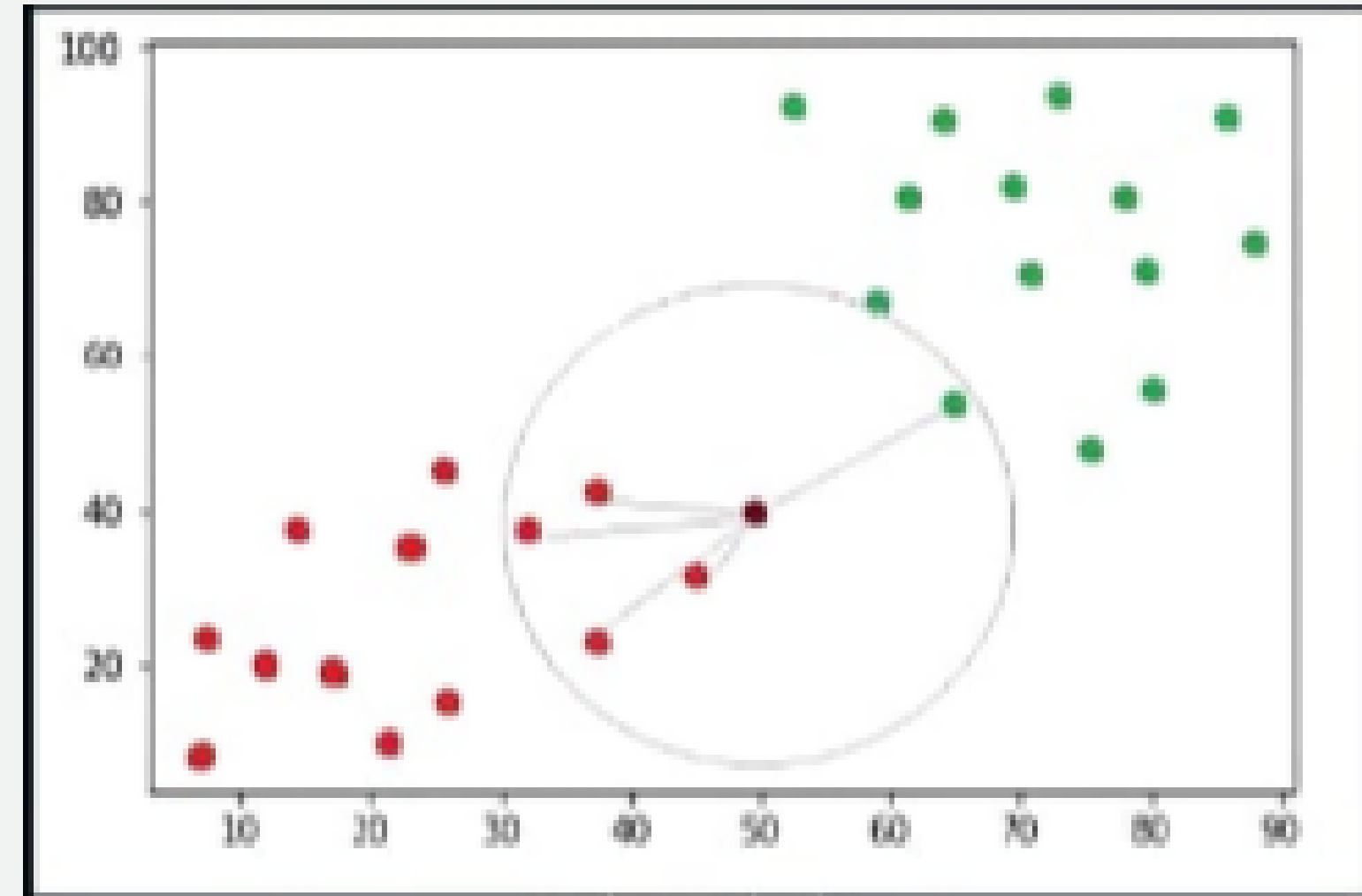
`ST_Distance(geom_a, geom_b)` `geom_a <-> geom_b`

3. Ordenar por distancia

`ORDER BY`
`distancia ASC`

4. Limitar los resultados

`LIMIT K`



Ejemplo básico

```
CREATE TABLE cultivos (  
  id_producto INT,  
  nombre VARCHAR(255),  
  coor_X DECIMAL(10, 2),  
  coor_Y DECIMAL(10, 2),  
  coor_Z DECIMAL(10, 2)  
);
```

```
INSERT INTO cultivos (id_producto, nombre, coor_X, coor_Y, coor_Z) VALUES  
(1, 'Arroz', 5.0, 10.0, 2.0),  
(2, 'Papa', 5.5, 9.5, 2.1),  
(3, 'Maiz', 1.0, 2.0, 1.5),  
(4, 'Platano', 4.8, 10.2, 2.0),  
(5, 'Cereza', 6.0, 11.0, 2.5),  
(6, 'Manzana', 10.0, 15.0, 3.0);
```

```
SELECT coor_X, coor_Y, coor_Z  
FROM cultivos  
WHERE id_producto = 1;
```

```
SELECT  
  p.id_producto,  
  p.nombre,  
  SQRT(  
    POWER(p.coor_X - 5.0, 2) +  
    POWER(p.coor_Y - 10.0, 2) +  
    POWER(p.coor_Z - 2.0, 2)  
  ) AS distancia  
FROM  
  cultivos AS p  
WHERE  
  p.id_producto != 1  
ORDER BY  
  distancia ASC  
LIMIT 3;
```

Ejemplo básico

```
CREATE TABLE cultivos (  
  id_producto INT,  
  nombre VARCHAR(255),  
  coor_X DECIMAL(10, 2),  
  coor_Y DECIMAL(10, 2),  
  coor_Z DECIMAL(10, 2)  
);
```

```
INSERT INTO cultivos (id_producto, nombre, coor_X, coor_Y, coor_Z) VALUES  
(1, 'Arroz', 5.0, 10.0, 2.0),  
(2, 'Papa', 5.5, 9.5, 2.1),  
(3, 'Maiz', 1.0, 2.0, 1.5),  
(4, 'Platano', 4.8, 10.2, 2.0),  
(5, 'Cereza', 6.0, 11.0, 2.5),  
(6, 'Manzana', 10.0, 15.0, 3.0);
```

```
SELECT coor_X, coor_Y, coor_Z  
FROM cultivos  
WHERE id_producto = 1;
```

	coor_x numeric (10,2) 🔒	coor_y numeric (10,2) 🔒	coor_z numeric (10,2) 🔒
1	5.00	10.00	2.00

```
SELECT  
  p.id_producto,  
  p.nombre,  
  SQRT(  
    POWER(p.coor_X - 5.0, 2) +  
    POWER(p.coor_Y - 10.0, 2) +  
    POWER(p.coor_Z - 2.0, 2)  
  ) AS distancia  
FROM  
  cultivos AS p  
WHERE  
  p.id_producto != 1  
ORDER BY  
  distancia ASC  
LIMIT 3;
```

	id_producto integer 🔒	nombre character varying (255) 🔒	distancia numeric 🔒
1	4	Platano	0.28284271247461901
2	2	Papa	0.714142842854285000
3	5	Cereza	1.500000000000000000

Uso de Búsqueda de Vecinos cercanos con IA

```
import pysal as ps
```

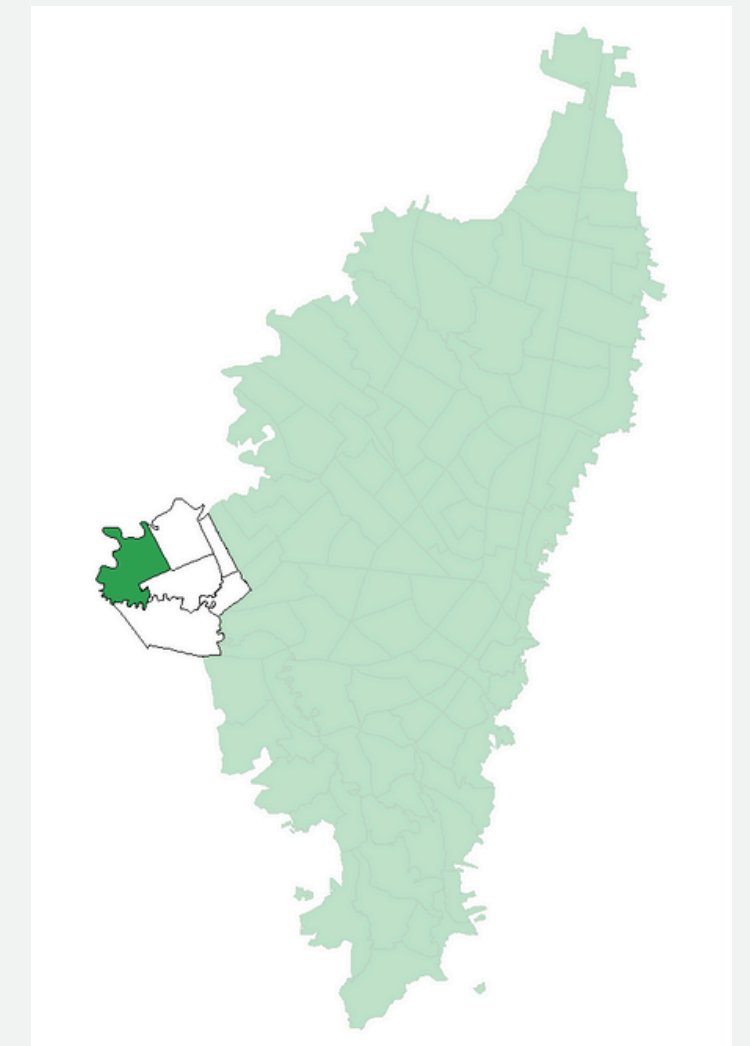
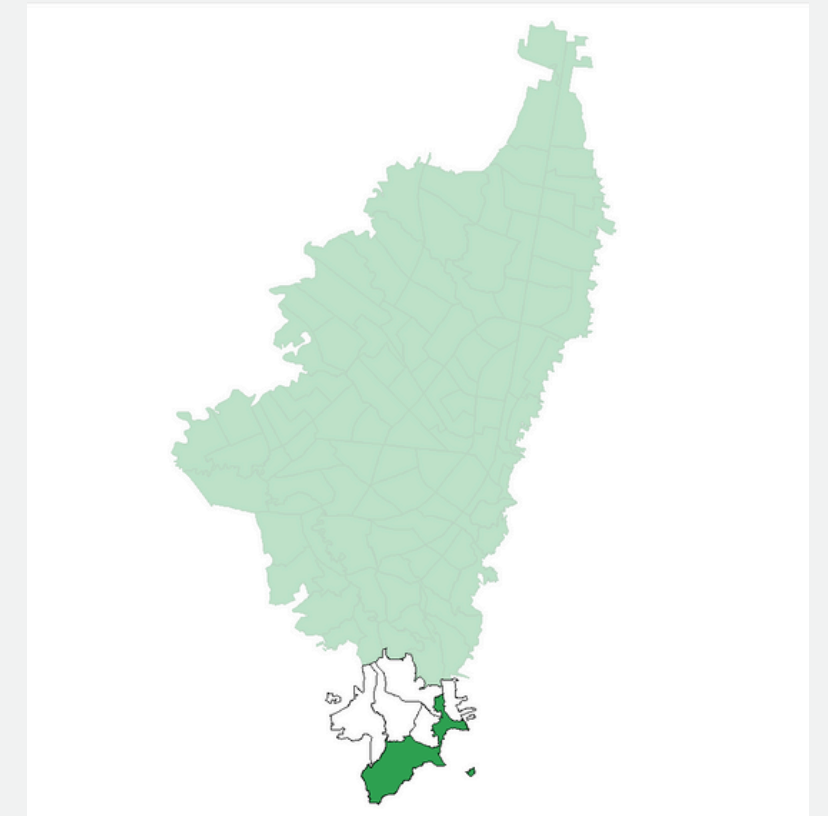
Biblioteca de **análisis espacial en Python** que maneja pesos o vecindades espaciales

```
[ ] knn=weights.KNN.from_dataframe(db,geom_col='geometry',k=5)

A_5nn_sym=make_sym_matrix_boolean_sum(knn.full()[0])
A_tilde_5nn_sym=renormalized_trick_mx(A_5nn_sym)
Laplacian_knn=renormalized_trick_laplacian(A_tilde_5nn_sym)
# check_symmetric(Laplacian_knn)

adj=torch.FloatTensor(Laplacian_knn).to(device)
```

- Aplicamos el criterio de vecindad **K-Vecinos Cercanos (5)**.
- Toma las matrices **asimétricas** y las convierte en **simétricas** (Suma la matriz transpuesta e incluye umbrales).
- Se normaliza para **estabilizar los gradientes** y asegurar que las características de los nodos se difundan correctamente en el grafo (Cada nodo se considera su vecino).
- Matriz fundamental para el análisis espectral de grafos, siendo la base matemática para muchas operaciones de **convolución del grafo**.



Ejemplo 1 guía

¿Cuáles son las 3 calles más cercanas a la estación de metro “Broad St”?

```
-- Get the geometry of Broad St
SELECT ST_AsEWKT(geom, 1)
FROM nyc_subway_stations
WHERE name = 'Broad St';
```

Obtenemos las coordenadas y el SRID de la estación Broad St.

Estas coordenadas nos sirven para hacer otra consulta partiendo de acá

```
SRID=26918;POINT(583571.9 4506714.3)
```

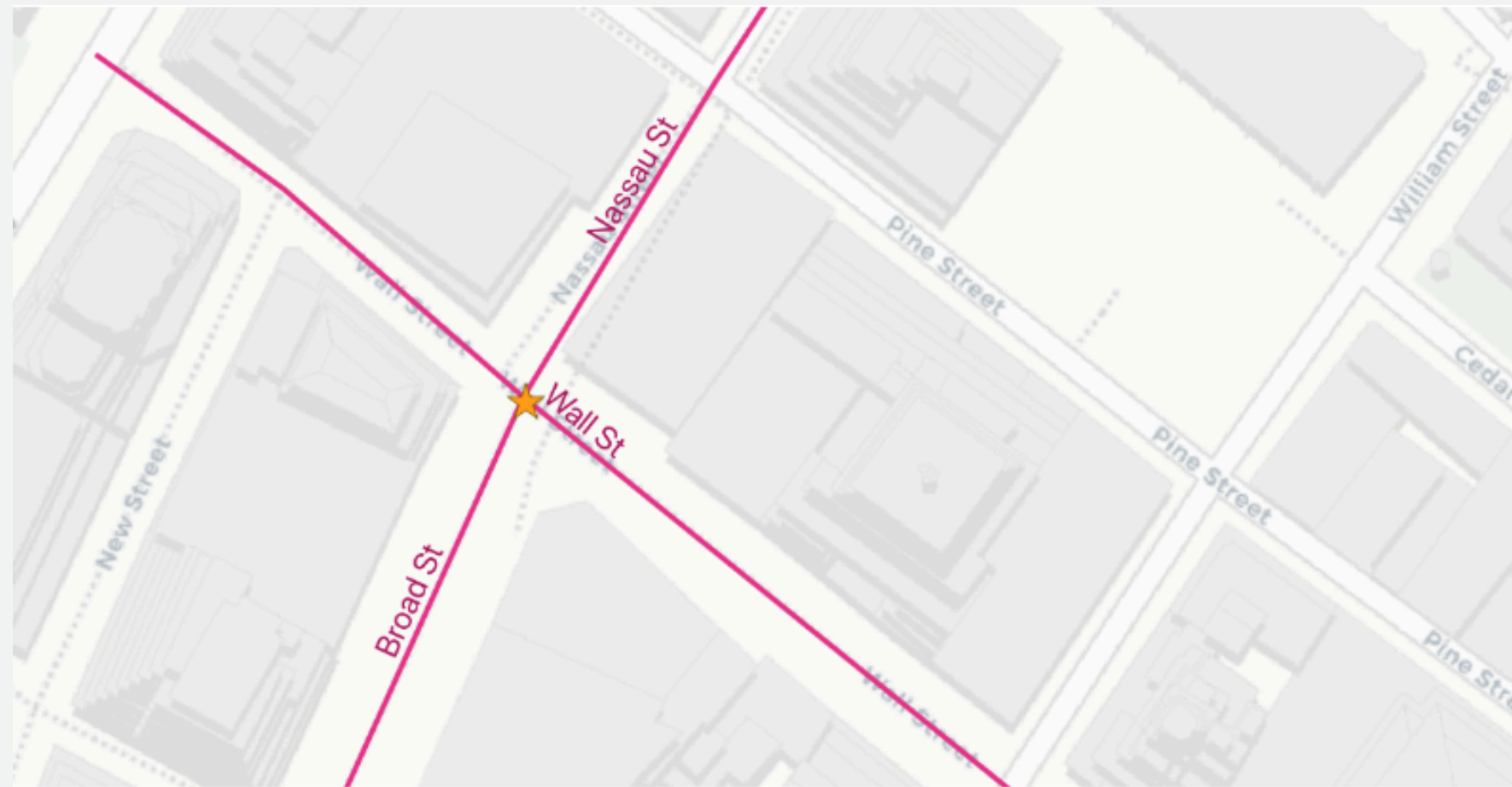
```
-- Plug the geometry into a nearest-neighbor query
SELECT streets.gid, streets.name,
       ST_Transform(streets.geom, 4326),
       streets.geom <-> 'SRID=26918;POINT(583571.9 4506714.3)::geometry AS dist
FROM
  nyc_streets streets
ORDER BY
  dist
LIMIT 3;
```

Aplicamos el operador distancia, ordenamos los datos y los limitamos.

Ejemplo 1 guía

¿Cuáles son las 3 calles más cercanas a la estacion de metro “Broad St”?

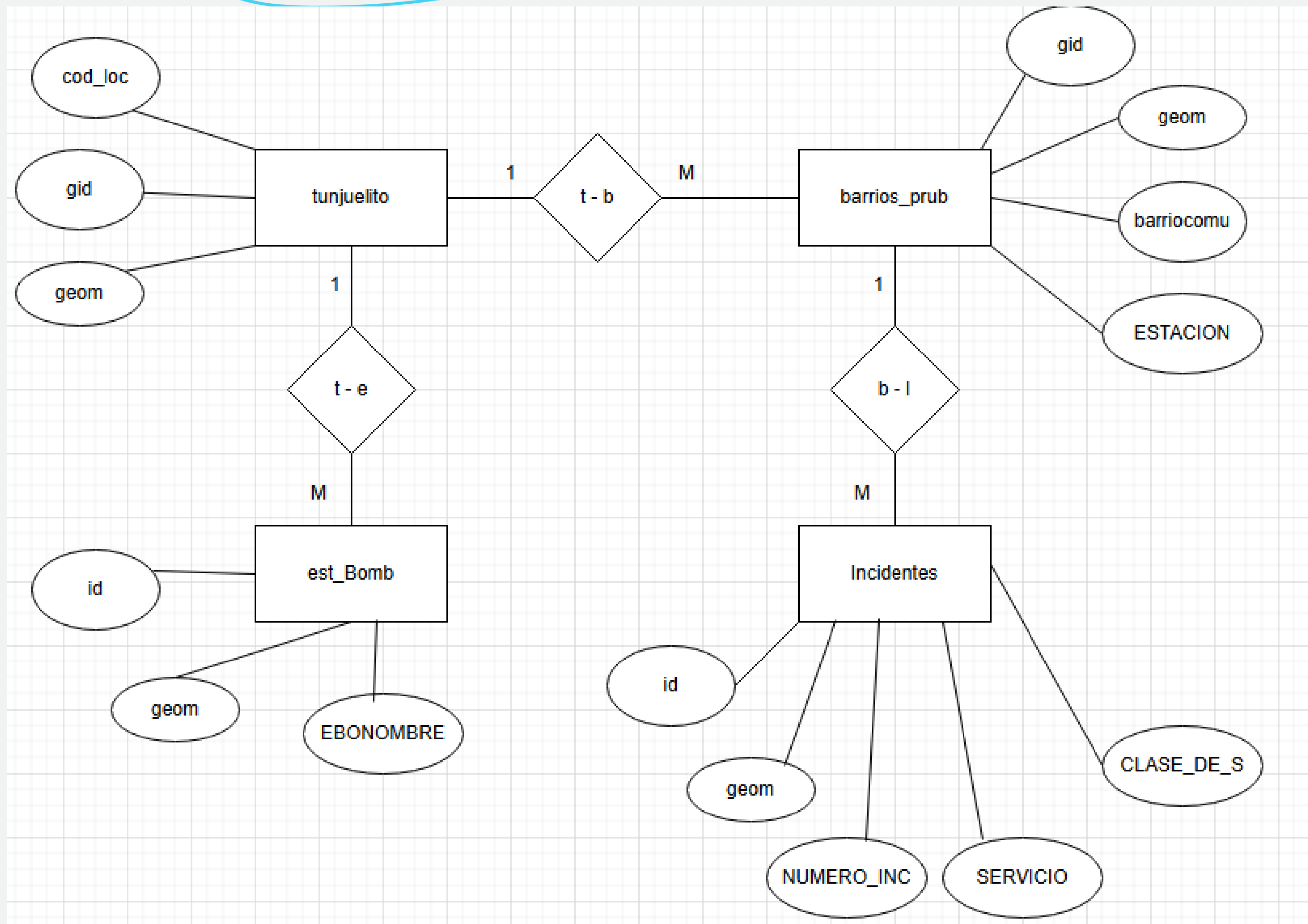
gid	name	dist
17385	Wall St	0.749987508809928
17390	Broad St	0.8836306235191059
17436	Nassau St	1.3368280241070414



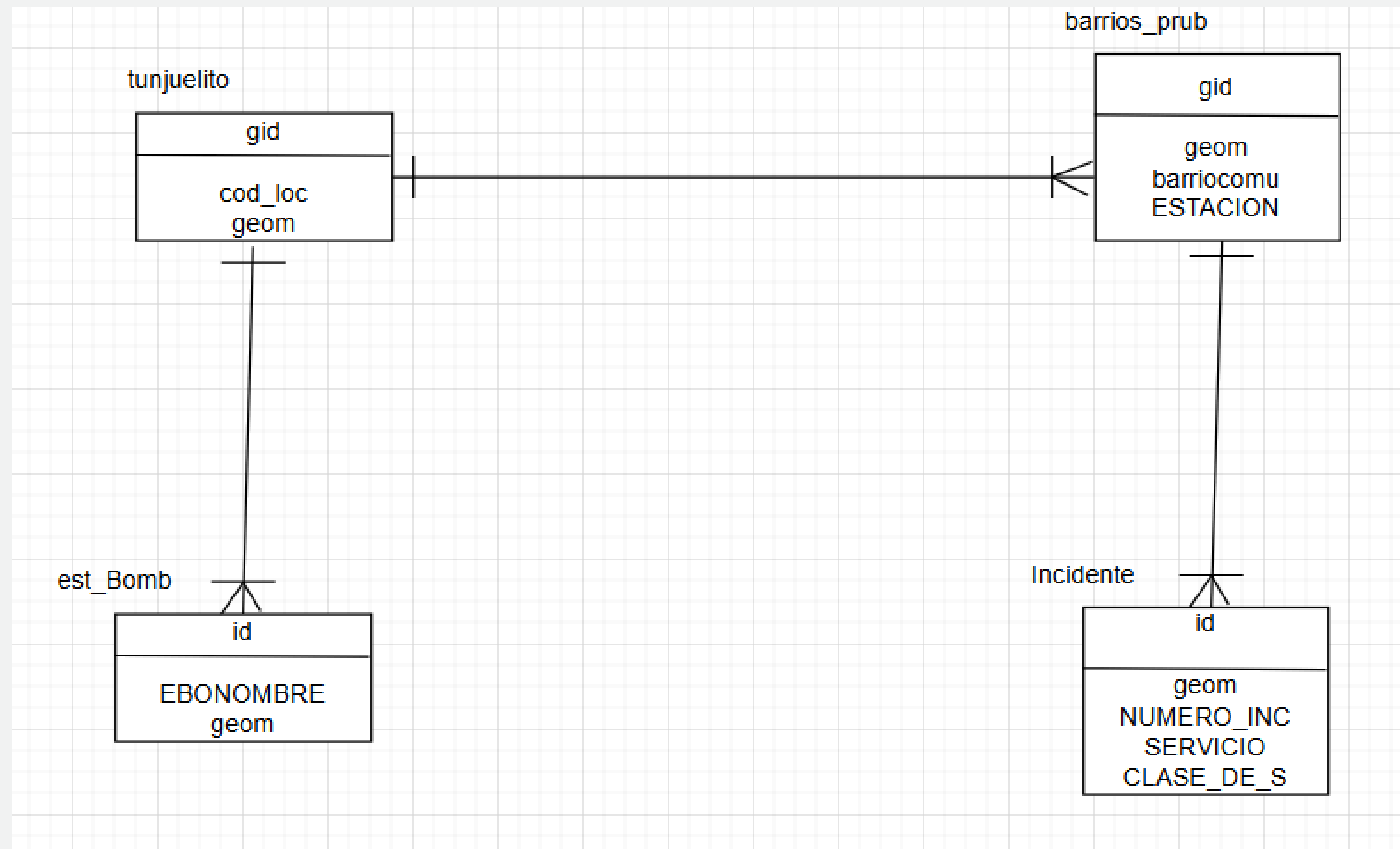
QUERY PLAN

```
Limit (cost=0.28..79.58 rows=3 width=31)
->  Index Scan using nyc_streets_geom_idx on nyc_streets streets
    (cost=0.28..504685.12 rows=19091 width=31)
    Order By:
        (geom <-> '01010000202669000000EEBD4CF27CF2141BC17D69516315141'::geometry)
```

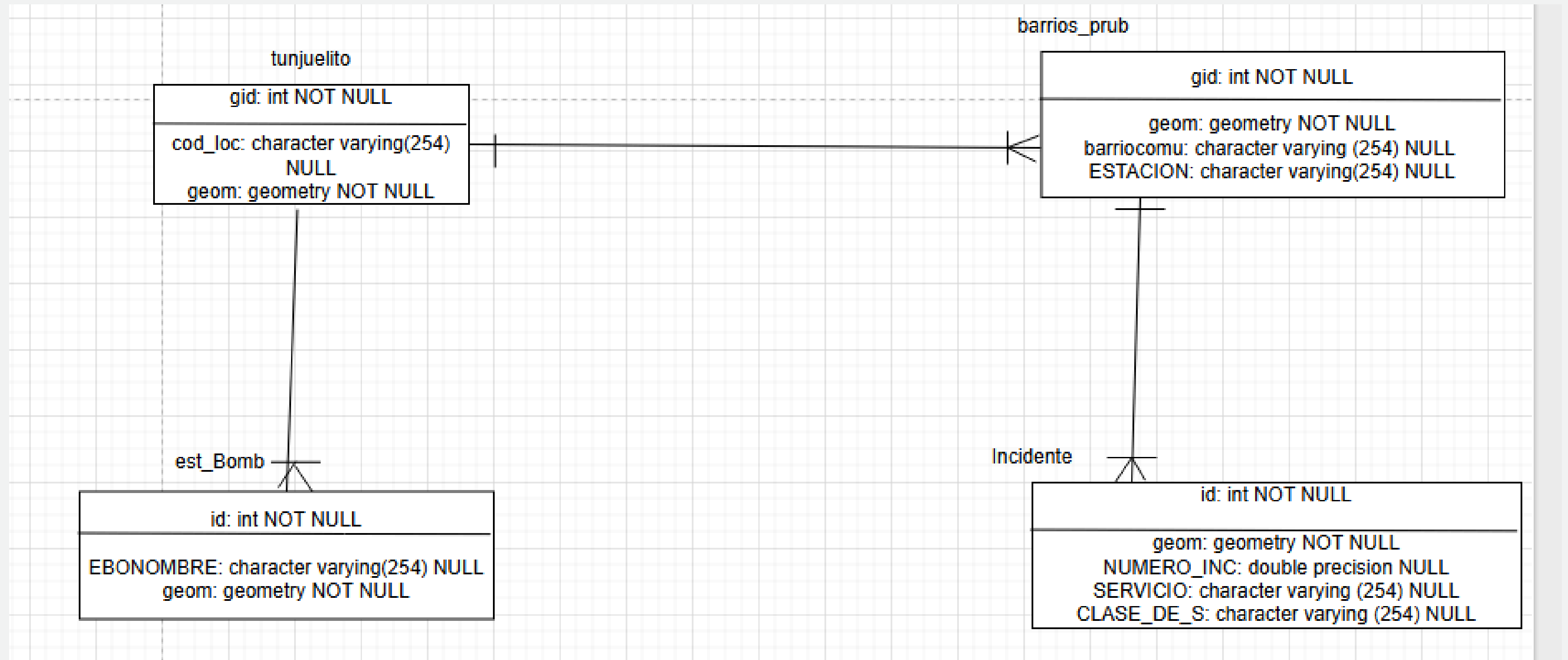
Modelo conceptual



Modelo lógico



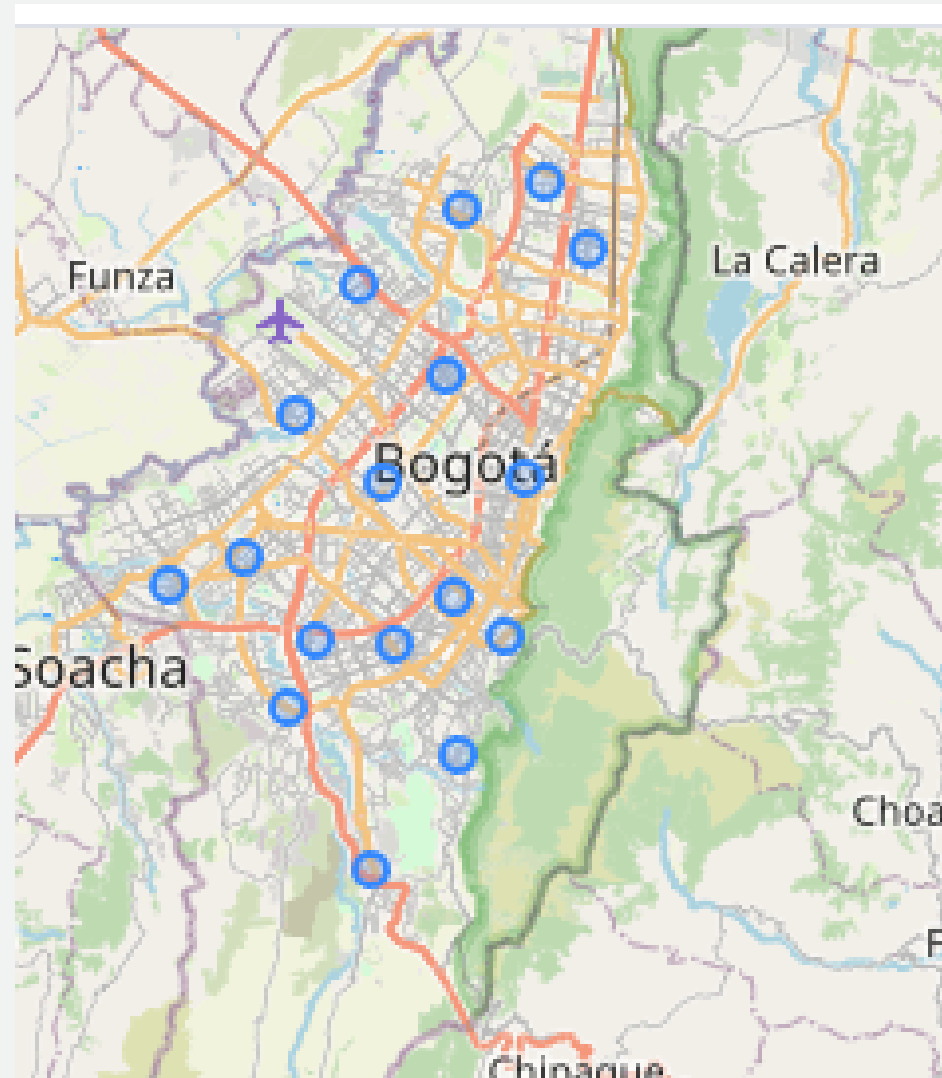
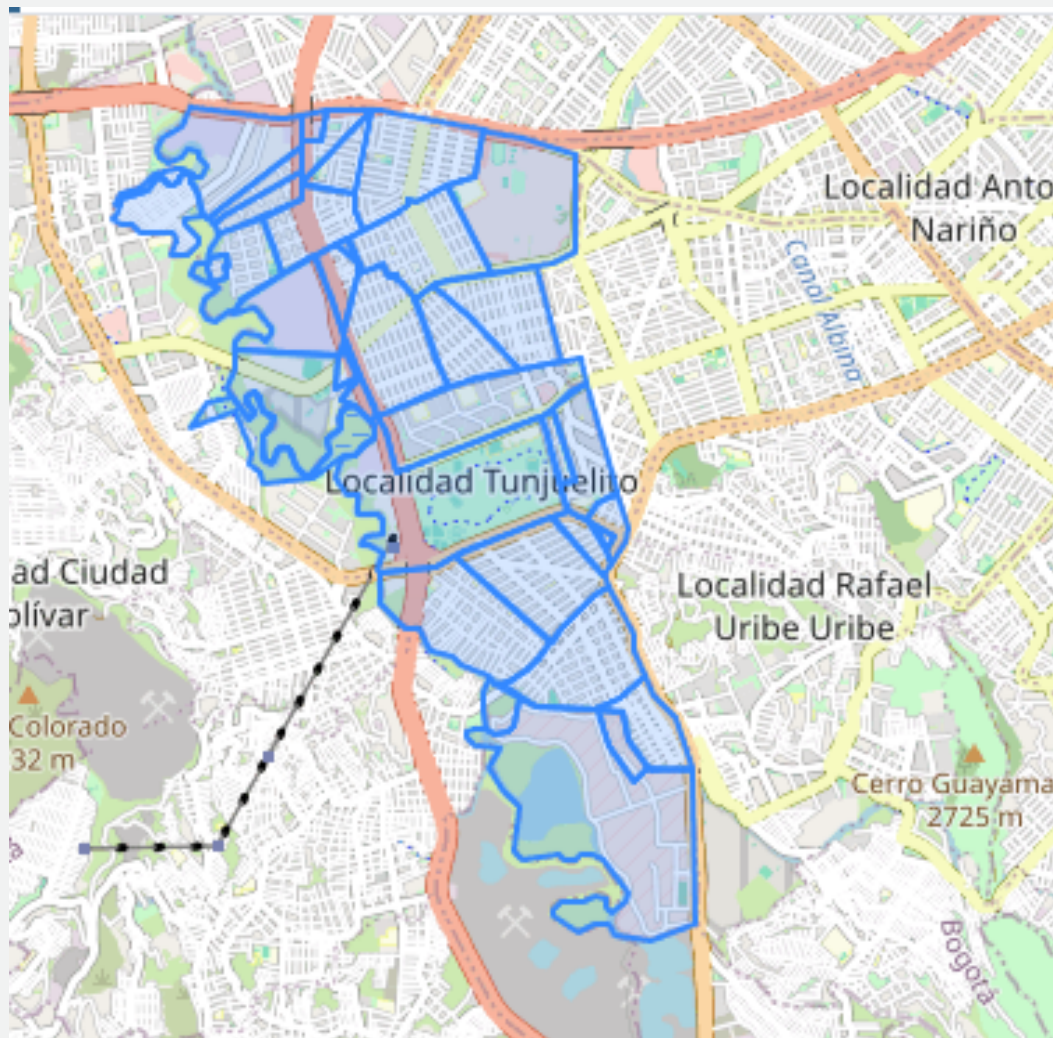
Modelo Físico



Verificación de geometrías

```
--Verificar la geometría de las capas
```

```
SELECT gid, barriocomu AS nombre, geom FROM barrios_prub;  
SELECT id, "EBONOMBRE", ST_AsText(geom),geom FROM public."est_Bomb";  
SELECT id, geom FROM public."Incidentes";
```



3D geometries not rendered.

Verificación de geometrías

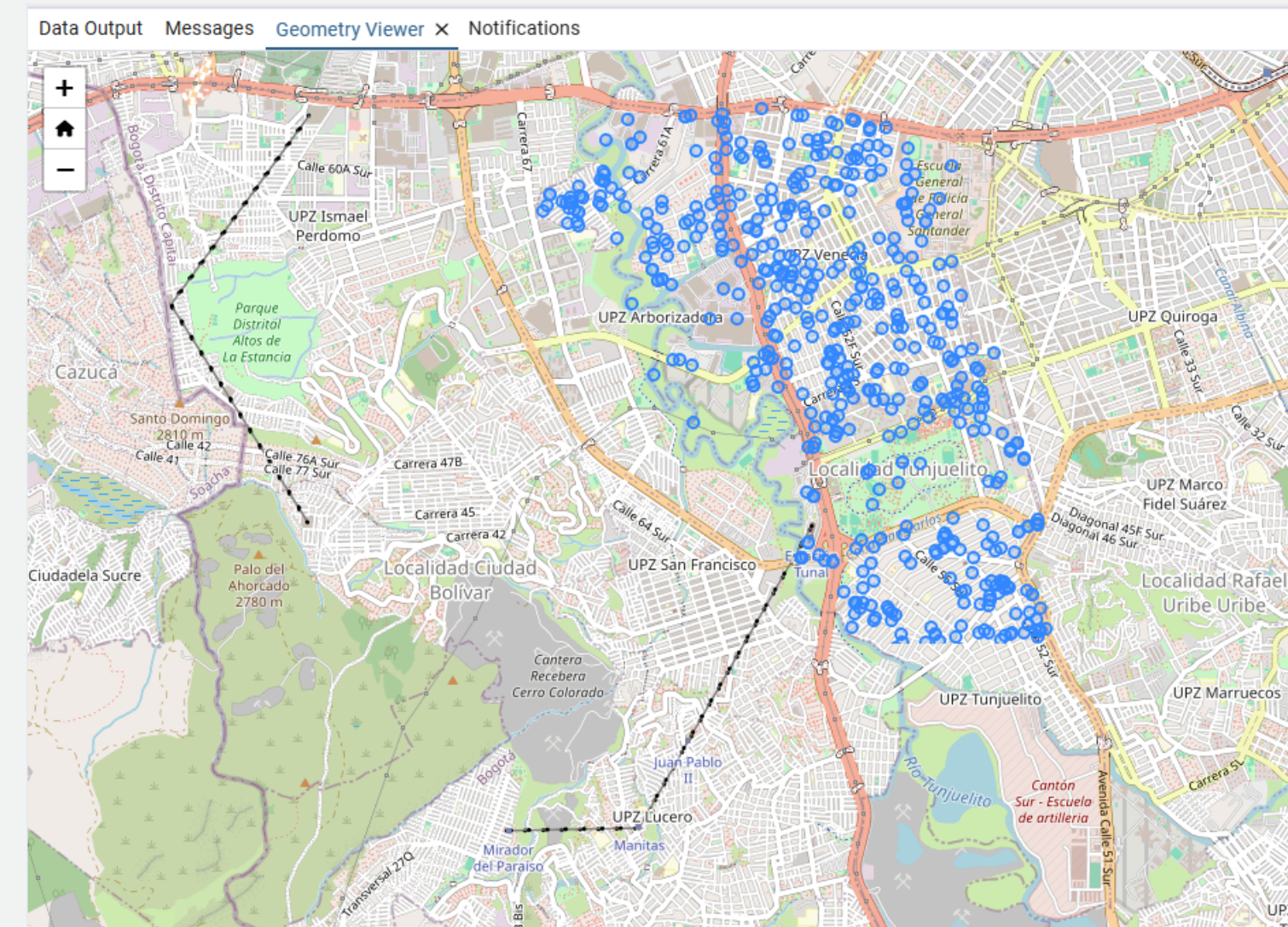
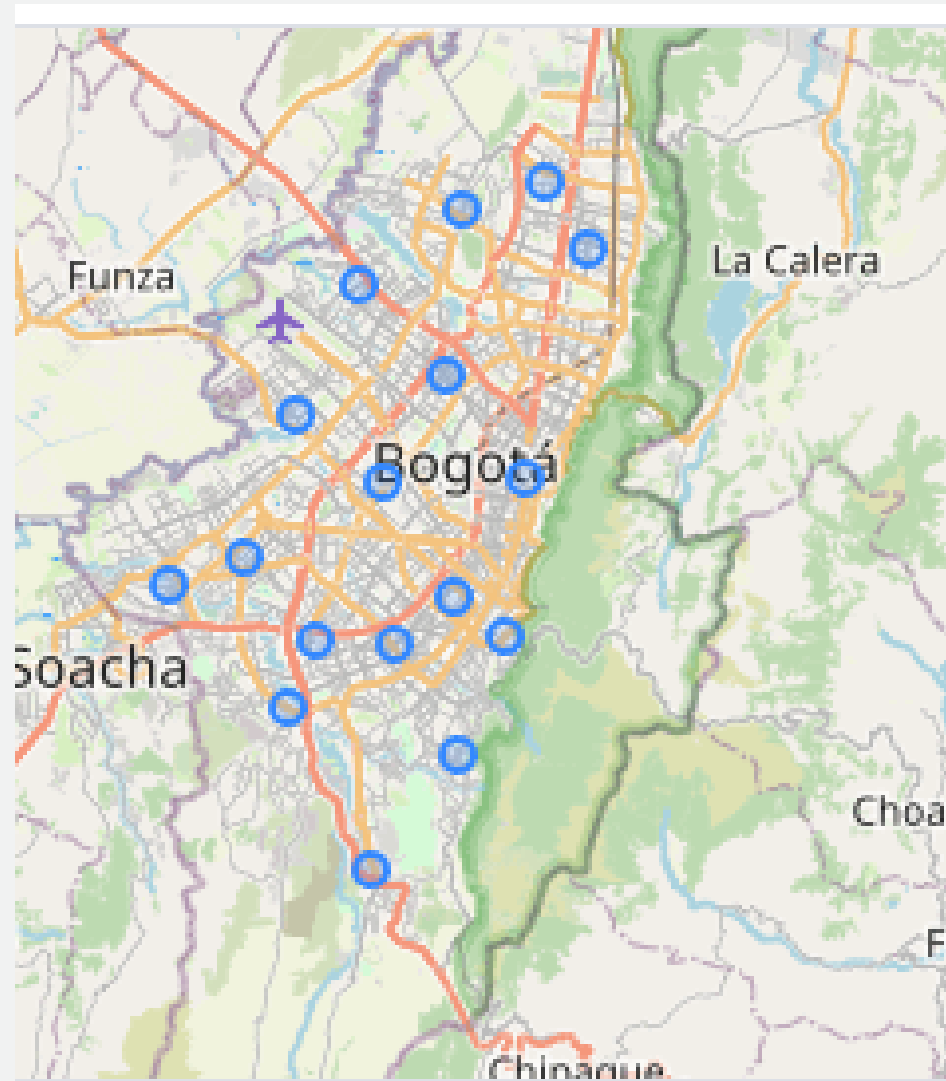
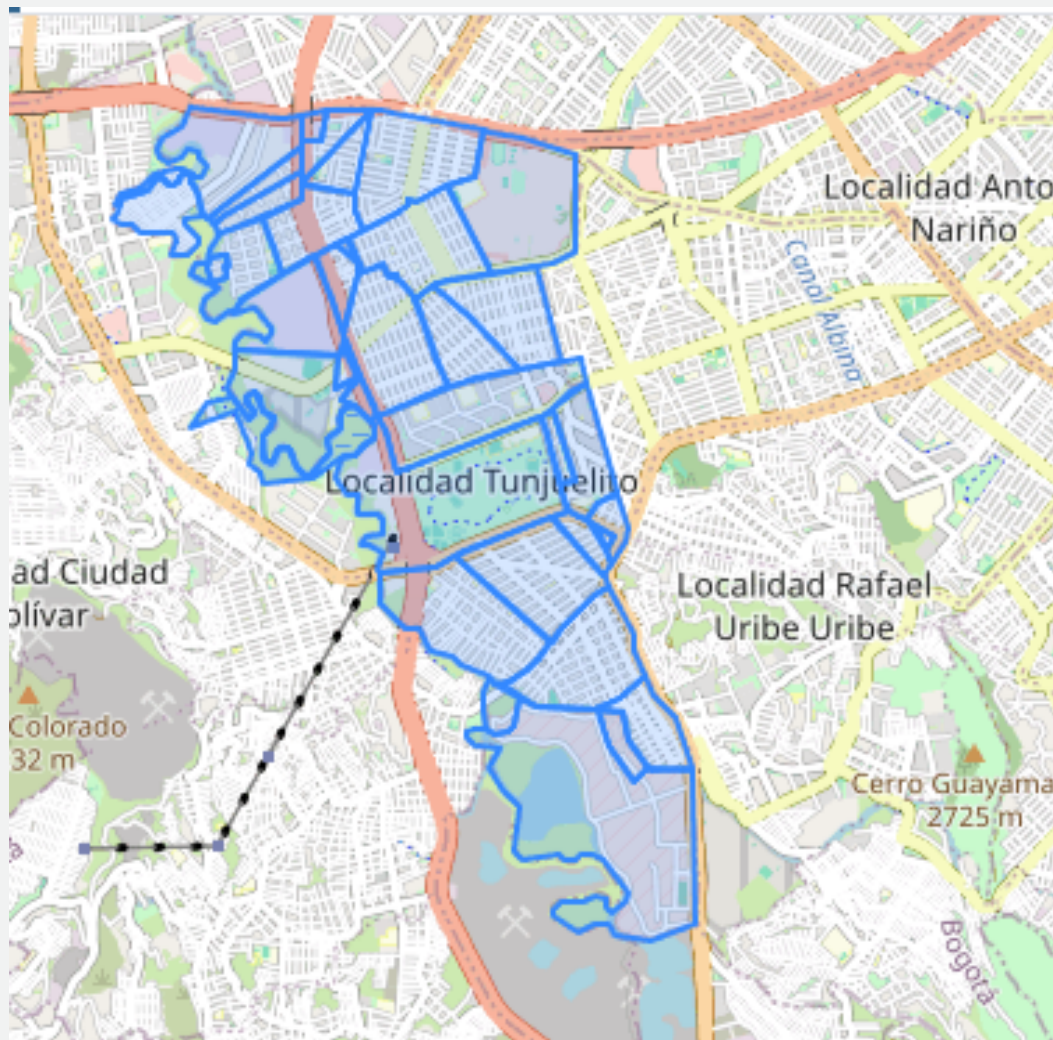
--Verificar la geometría de las capas

```
SELECT gid, barriocomu AS nombre, geom FROM barrios_prub;
```

```
SELECT id, "EBONOMBRE", ST_AsText(geom), geom FROM public."est_Bomb";
```

```
SELECT id, geom FROM public."Incidentes";
```

```
SELECT id, geom, ST_Force2D(geom) AS geometria_2d FROM public."Incidentes";
```



¿Cuáles son las coordenadas de la estación de bomberos de venecia?

EBONOMBRE character varying (50)
B- 7 ESTACION FERIAS
B- 9 ESTACION BELLAVISTA
B-10 ESTACION MARICHUELA
B-11 ESTACION CANDELARIA
B-12 ESTACION SUBA
B-13 ESTACION CAOPOS SALAZAR
B-15 ESTACION GARCES NAVAS
B-16 ESTACION VENECIA

```
SELECT ST_AsEWKT(geom, 15) AS Coordenadas
FROM public."est_Bomb" WHERE "EBONOMBRE" = 'B-16 ESTACION VENECIA';
```

coordenadas text	
1	SRID=4326;POINT(-74.13587046443098 4.591289109879961)

Ejercicio práctico 1

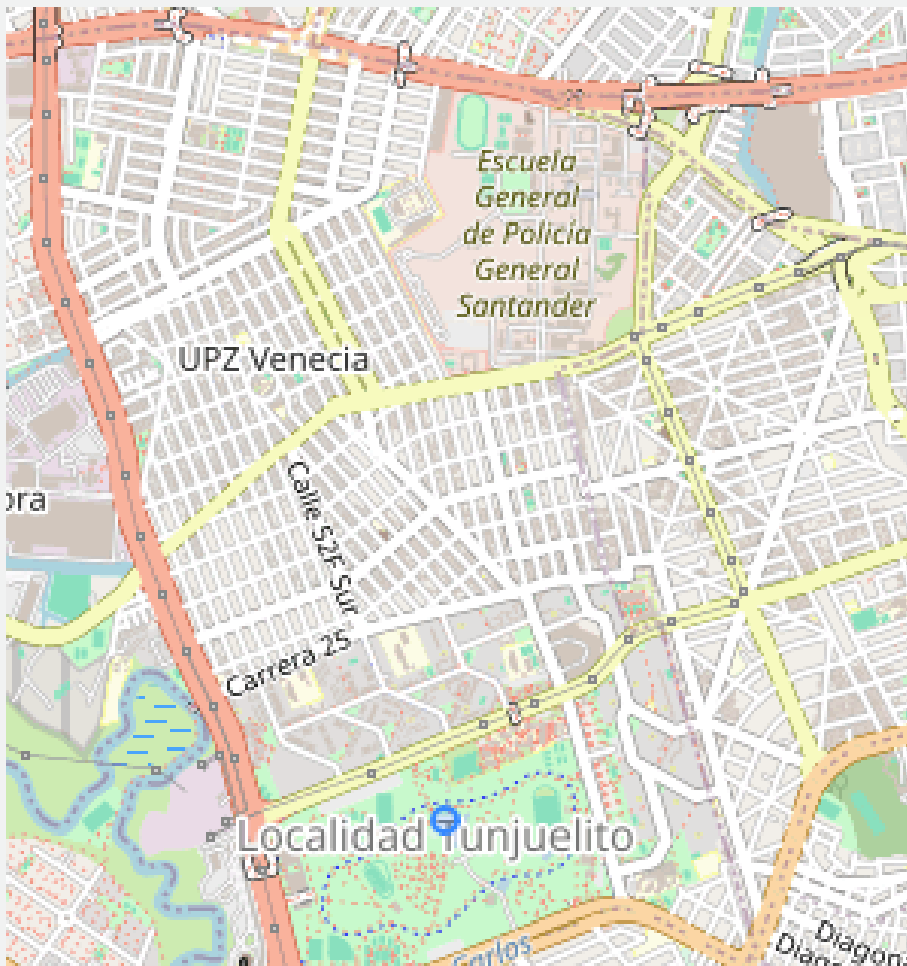
```
--Número incidente: 67213, Incidente emergencia en potencia - Bloqueo de salida edificaciones  
-- ¿Cuáles son las coordenadas del incidente?
```

Ejercicio práctico 1

--Número incidente: 67213, Incidente emergencia en potencia - Bloqueo de salida edificaciones
-- ¿Cuáles son las coordenadas del incidente?

```
SELECT id, ST_AsEWKT(geom, 15), ST_Force2D(geom) AS geometria_2d
FROM "Incidentes" WHERE "NUMERO_INC"='67213';
```

	id [PK] integer	st_asewkt text	geometria_2d geometry
1	399	SRID=4326;POINT(-74.13403500131993 4.573161920978619 0)	0101000020E61000000FCC8A07948852C0AFA967F5EA4A1240






Ejercicio práctico 1

- `SRID=4326;POINT(-74.13403500131993 4.573161920978619 0)`
- ¿Cuáles son las estaciones de bomberos más cercanas?

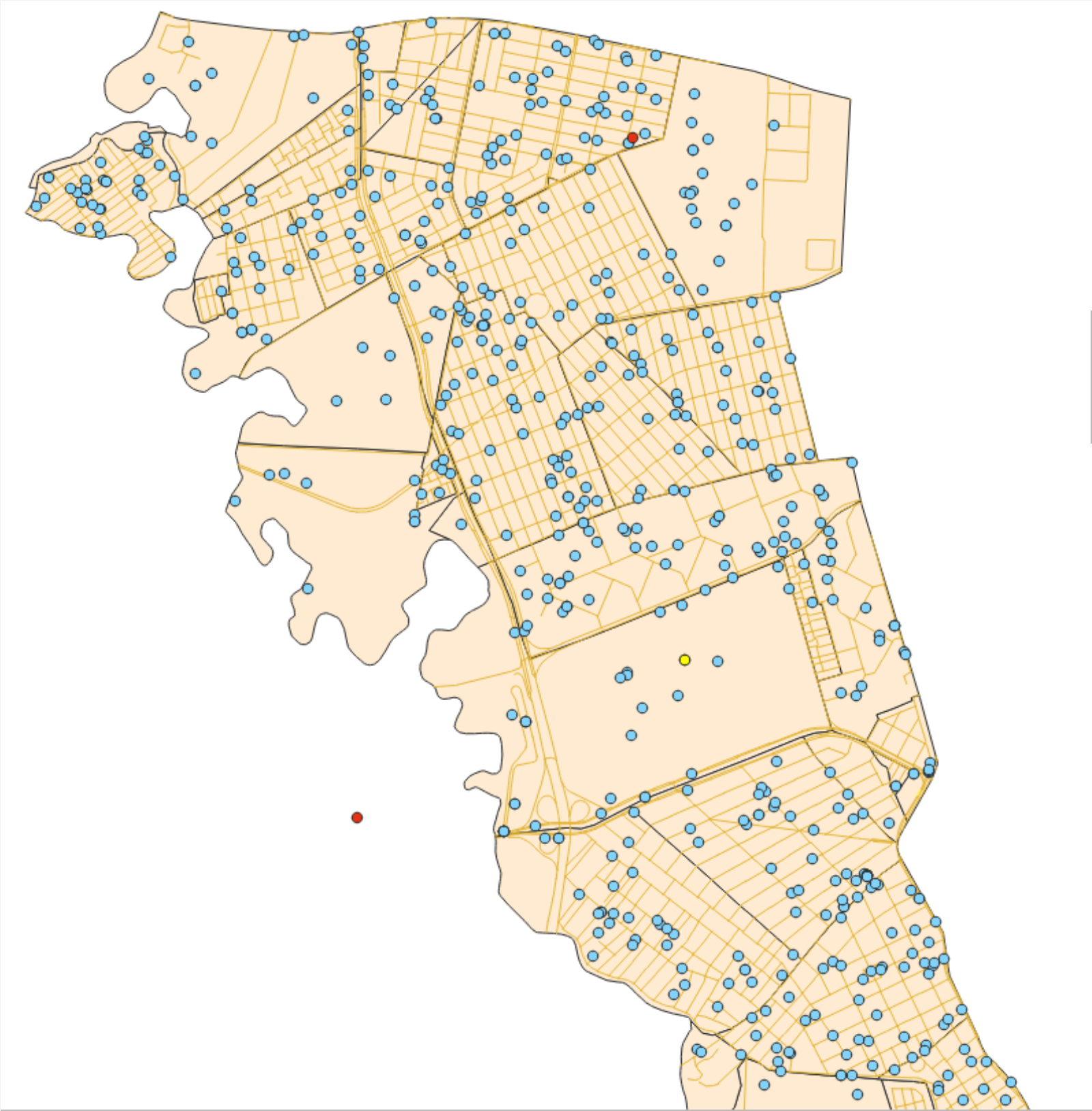
Ejercicio práctico 1

- SRID=4326;POINT(-74.13403500131993 4.573161920978619 0)
- ¿Cuáles son las estaciones de bomberos más cercanas?

```
SELECT "est_Bomb".id, "est_Bomb"."EBONOMBRE",  
       "est_Bomb".geom <-> 'SRID=4326;POINT(-74.13403500131993 4.573161920978619 0) '::geography AS dist  
FROM "est_Bomb" ORDER BY dist LIMIT 2
```

	id [PK] integer 	EBONOMBRE character varying (50) 	dist double precision 
1	9	B-11 ESTACION CANDELARIA	1403.0399075774462
2	13	B-16 ESTACION VENECIA	2025.8950150639535

Realidad



Dada la jurisdicción de la estación de Venecia y la cercanía de la estación Candelaria
¿Cuál creen que atendió el incidente presentado en el Parque el Tunal?

399

26/07/2024

67213,00000000...

B-16 VENECIA

15. INCIDENTE EMERGENCIA EN POTENCIA

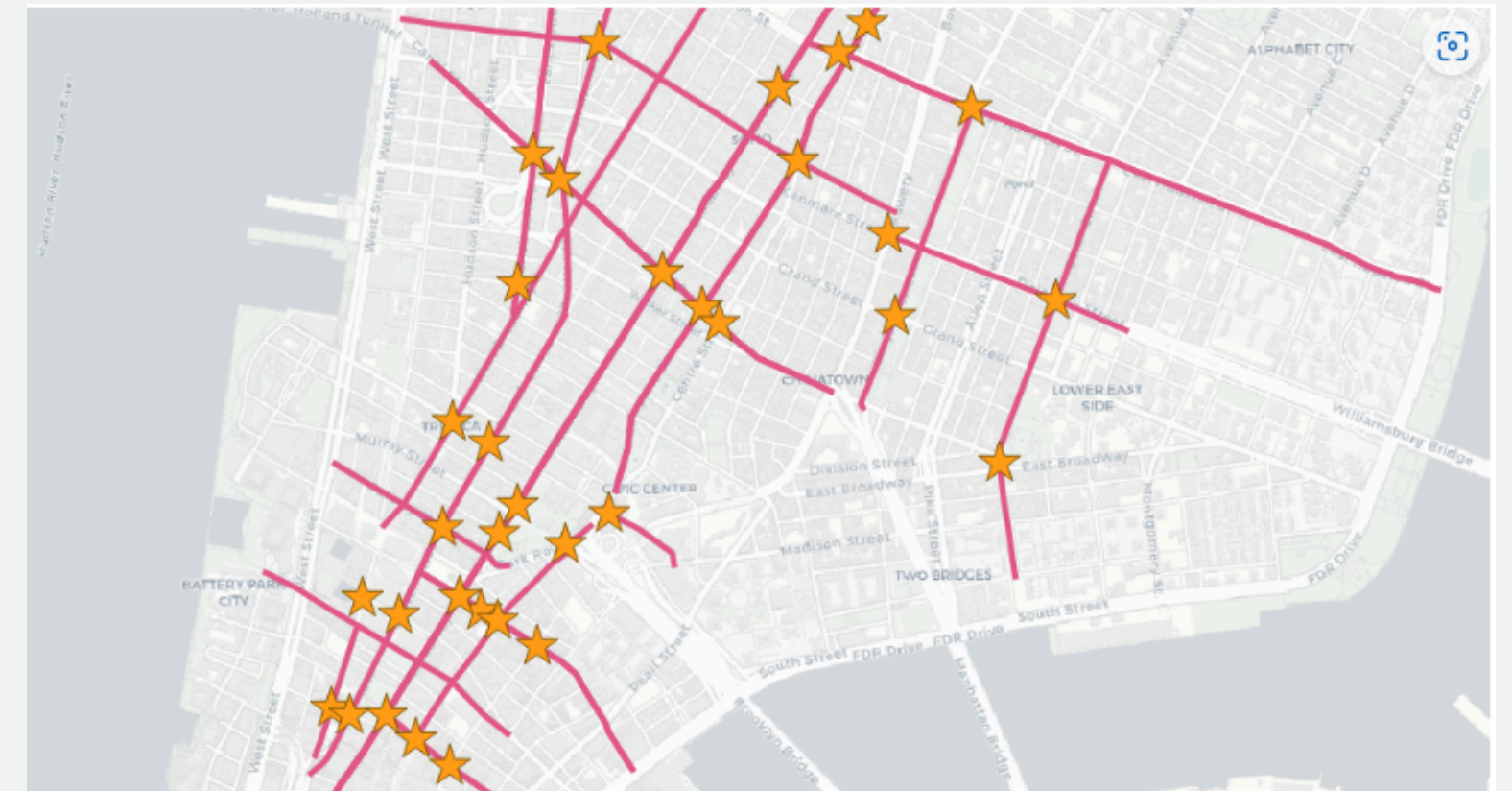
15.2 ABRIR EDIFICACIONES POR RIESGOS EN POTENCIA

Ejemplo 2 guía

¿Cuál es la calle más cercana a cada estación del metro?

De la forma anterior nos demoraríamos muchísimo buscando la calle más cercana para cada estación del metro, por tanto, se hace uso del **cross lateral join**.

```
SELECT subways.gid AS subway_gid,  
       subways.name AS subway,  
       streets.name AS street,  
       streets.gid AS street_gid,  
       streets.geom::geometry(MultiLinestring, 26918) AS street_geom,  
       streets.dist  
FROM nyc_subway_stations subways  
CROSS JOIN LATERAL (  
  SELECT streets.name, streets.geom, streets.gid, streets.geom <-> subways.geom AS dist  
  FROM nyc_streets AS streets  
  ORDER BY dist  
  LIMIT 1  
) streets;
```



Vecinos cercanos uno a muchos

Ejercicio práctico 1

¿Para cada uno de los incidentes qué estación de bomberos lo atendió y cuál era realmente la más cercana?

```
SELECT
    i.id AS incidente_id,
    i."ESTACION" AS estacion_que_atendio,
    eb_cercana.id AS id_estacion_mas_cercana,
    eb_cercana."EBONOMBRE" AS estacion_mas_cercana,
    ST_Distance(i.geom::geography, eb_cercana.geom::geography) AS cercana_distancia
FROM
    "Incidentes" i
CROSS JOIN LATERAL (
    SELECT
        eb.id,
        eb."EBONOMBRE",
        eb.geom,
        eb.geom <-> i.geom AS distancia_para_ordenar
    FROM "est_Bomb" eb
    ORDER BY
        distancia_para_ordenar
    LIMIT 1
) AS eb_cercana;
```


Ejercicio práctico 1

¿Para cada uno de los incidentes qué estación de bomberos lo atendió y cuál era realmente la más cercana?

	incidente_id integer	estacion_que_atendio character varying (254)	id_estacion_mas_cercana integer	estacion_mas_cercana character varying (50)	cercana_distancia double precision
1 ●	1	B-3 RESTREPO	13	B-16 ESTACION VENECIA	327.77006954
2	2	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	3000.58448888
3 ●	3	B-10 MARICHUELA	9	B-11 ESTACION CANDELARIA	3366.43683184
4	4	B-16 VENECIA	13	B-16 ESTACION VENECIA	996.41217266
5	5	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	2212.28984768
6	6	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	1387.56554727
7	7	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	1854.84527645
8 ●	8	B-10 MARICHUELA	9	B-11 ESTACION CANDELARIA	3396.6223084
9 ●	9	B-10 MARICHUELA	9	B-11 ESTACION CANDELARIA	3249.38593176
10	10	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	1900.52652593
11	11	B-16 VENECIA	13	B-16 ESTACION VENECIA	287.48242759
12	12	B-16 VENECIA	13	B-16 ESTACION VENECIA	1867.54111465
13	13	B-16 VENECIA	13	B-16 ESTACION VENECIA	1103.07813605
14	14	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	1903.19979587
15 ●	15	B-3 RESTREPO	13	B-16 ESTACION VENECIA	868.12489569
16	16	B-16 VENECIA	13	B-16 ESTACION VENECIA	808.83825707
17	17	B-16 VENECIA	13	B-16 ESTACION VENECIA	1321.70426449
18	18	B-16 VENECIA	13	B-16 ESTACION VENECIA	1223.9371598
19	19	B-16 VENECIA	13	B-16 ESTACION VENECIA	1826.84479454
20	20	B-16 VENECIA	13	B-16 ESTACION VENECIA	698.17857551
21 ●	21	B-3 RESTREPO	9	B-11 ESTACION CANDELARIA	1056.03504623
22	22	B-11 CANDELARIA	9	B-11 ESTACION CANDELARIA	1055.09970041