Séance 5 - Correction TP - PostGIS 1

0. Création de la BD et activation de PostGIS

On créé la BD :

```
CREATE DATABASE paris;
```

On s'y connecte puis on active l'extension PostGIS :

```
CREATE EXTENSION postgis;
```

1. Import des données

```
shp2pgsql -s 2154 -c -D -I QUARTIER_DE_PARIS/QUARTIER_DE_PARIS.shp quartier
 ↪  | psql -U postgres -d paris

shp2pgsql -s 2154 -c -D -I
 ↪  RECENSEMENT_IRIS_LOGEMENT/RECENSEMENT_IRIS_LOGEMENT.shp iris | psql -U
 ↪  postgres -d paris

shp2pgsql -s 2154 -c -D -I
 ↪  STATION_DE_TRANSPORT_EN_COMMUN/STATION_DE_TRANSPORT_EN_COMMUN_METRO.shp
 ↪  station_metro | psql -U postgres -d paris

shp2pgsql -s 2154 -c -D -I
 ↪  LIGNE_DE_TRANSPORT_EN_COMMUN/LIGNE_DE_TRANSPORT_EN_COMMUN_METRO.shp
 ↪  ligne_metro | psql -U postgres -d paris
```

2.

2.1

```
SELECT gid, l_qu, ST_Area(geom)
FROM quartier
ORDER BY ST_Area(geom) ASC
LIMIT 1;
```

```
 gid |  l_qu   |      st_area
-----+---------+--------------------
  40 | Gaillon | 188012.21033959283
(1 row)
```

ou

```
SELECT gid, l_qu
FROM quartier
WHERE ST_Area(geom) = (
    SELECT MIN(st_area(geom))
    FROM quartier
);
```

```
 gid |  l_qu
-----+---------
  40 | Gaillon
(1 row)
```

2.2

```
SELECT sum(ST_AREA(geom)) / count(*) as superficie_moyenne FROM quartier;
```

```
 superficie_moyenne
--------------------
 1317159.6567839629
(1 row)
```

2.3

```
SELECT st_y(geom) as N, l_station
FROM station_metro
ORDER BY N DESC
LIMIT 1;
```

```
         n          |      l_station
--------------------+----------------------
 6866680.820796113 | PORTE DE LA VILLETTE
(1 row)
```

2.4

```
SELECT iris.nb_log, iris.l_ir
FROM iris WHERE nb_log is not null
ORDER BY nb_log DESC LIMIT 1;
```

```
        nb_log         |   l_ir
-----------------------+-----------
 3492.996020479019990 | Necker 10
(1 row)
```

2.5

```sql
SELECT nb_logvac / nb_log * 100 as pct_logvac, l_ir
FROM iris
WHERE
    nb_log is not null AND
    nb_logvac > 0
ORDER BY pct_logvac DESC
LIMIT 5;
```

```
       pct_logvac         |          l_ir
--------------------------+----------------------
 57.28655827377580082800 | Bois de Boulogne
 41.18053160611318657400 | Bois de Vincennes
 40.07478137354589811200 | Bois de Boulogne
 27.53026839721719714800 | Champs Elysées 1
 22.58062375320266718900 | Grandes Carrières 29
(5 rows)
```

2.6

```sql
SELECT l_qu
FROM quartier, ligne_metro
WHERE
    ST_Intersects(quartier.geom, ligne_metro.geom)
    AND ligne_metro.l_ligne = '14'
    AND ligne_metro.n_annee = 2020;
```

```
     l_qu
---------------
 Epinettes
 Batignolles
 Epinettes
 Madeleine
 Europe
 Saint-Georges
(6 rows)
```

2.7

```sql
SELECT DISTINCT l_qu
FROM quartier, ligne_metro
WHERE
    ST_Intersects(quartier.geom, ligne_metro.geom)
    AND ligne_metro.n_annee = 1900;
```

```
          l_qu
------------------------
 Arsenal
 Bel-Air
 Chaillot
 Champs-Elysées
 Charonne
 Faubourg-du-Roule
 Halles
 Madeleine
 Palais-Royal
 Picpus
 Place-Vendôme
 Porte-Dauphine
 Quinze-Vingts
 Sainte-Marguerite
 Saint-Gervais
 Saint-Merri
 St-Germain-l'Auxerrois
 Ternes
(18 rows)
```

2.8

```sql
SELECT SUM(nb_log) as nb_log_qu, n_qu, l_qu
FROM iris
JOIN quartier ON iris.n_qu = quartier.c_qu
WHERE
    nb_log is not null
GROUP BY n_qu, l_qu
ORDER BY nb_log_qu DESC
LIMIT 5;
```

```
       nb_log_qu       | n_qu |       l_qu
```

```
----------------------+------+------------------
 54798.438080529620495 |   57 | Saint-Lambert
 46551.049197472311302 |   69 | Grandes-Carrières
 45497.048181544429895 |   70 | Clignancourt
 42708.567181889555352 |   61 | Auteuil
 39612.830953169907684 |   50 | Gare
(5 rows)
```

2.9

```
SELECT SUM(nb_logvac) / SUM(nb_log) * 100 as pct_logvac, n_qu, l_qu
FROM iris
JOIN quartier ON iris.n_qu = quartier.c_qu
WHERE
    nb_log is not null AND
    nb_logvac > 0
GROUP BY n_qu, l_qu
ORDER BY pct_logvac DESC
LIMIT 3;
```

```
       pct_logvac        | n_qu |         l_qu
-------------------------+------+----------------------
 19.661975835232839305 |   29 | Champs-Elysées
 14.882263696543069075 |    8 | Bonne-Nouvelle
 13.816837573493285709 |   37 | Saint-Vincent-de-Paul
(3 rows)
```

ou

```
SELECT SUM(nb_logvac) / SUM(nb_log) * 100 as pct_logvac, n_qu, l_qu
FROM iris, quartier
WHERE
    ST_Contains(quartier.geom, ST_Centroid(iris.geom)) AND
    nb_log is not null AND
    nb_logvac > 0
GROUP BY n_qu, l_qu
ORDER BY pct_logvac DESC
LIMIT 3;
```

2.10

```
SELECT a.l_qu, SUM(ST_Length(ST_Intersection(a.geom, b.geom)))
FROM quartier a, ligne_metro b
```

```
WHERE ST_Intersects(a.geom, b.geom)
GROUP BY a.l_qu
ORDER BY a.l_qu;
```

```
          l_qu          |        sum
------------------------+--------------------
 Amérique               |   4114.10932358217
 Archives               |  217.8932326380176
 Arsenal                |  2806.613028937873
 Arts-et-Metiers        |   2535.63270282871
 Auteuil                |  5716.645529028718
 Batignolles            | 1930.4501089712453
 Bel-Air                |  984.1403649443128
 Belleville             |  769.0751307886278
 Bercy                  |  3390.969797282655
 Bonne-Nouvelle         | 1687.4768615961834
 Chaillot               |  3642.203444506908
 Champs-Elysées         |  4147.121297262528
 Charonne               | 2408.2925834461353
 Chaussée-d'Antin       |  3487.713867428912
 Clignancourt           | 2840.8865108826403
```

2.11

On va ici utiliser un buffer de 10m afin de récupérer les stations qui ne sont pas exactement sur le tracé de la ligne 14 :

```
SELECT DISTINCT m.l_station
FROM station_metro as m, ligne_metro as l
WHERE ST_Intersects(ST_Buffer(l.geom, 10), m.geom) AND l.l_ligne = '14';
```

```
            l_station
---------------------------------
 BERCY
 BIBLIOTHEQUE FRANCOIS MITTERRAND
 CHATELET
 COUR SAINT-EMILION
 MADELEINE
 OLYMPIADES
 PONT CARDINET
 PORTE DE CLICHY
 PYRAMIDES
```

```
 SAINT-LAZARE
(10 rows)
```

3

3.1

```sql
CREATE TABLE arrondissement
AS
SELECT c_ar as code_arrondissement, ST_Union(geom) AS geom
FROM quartier
GROUP BY code_arrondissement;

ALTER TABLE arrondissement ADD PRIMARY KEY (code_arrondissement);

SELECT Populate_Geometry_Columns('public.arrondissement'::regclass);
```

3.2

```sql
SELECT
    q.c_ar as code_arrondissement,
    count(m.gid) as nb_station
FROM quartier as q, station_metro as m
WHERE ST_Intersects(q.geom, m.geom)
GROUP BY q.c_ar
ORDER BY q.c_ar;
```

```
 code_arrondissement | nb_station
---------------------+------------
                   1 |          9
                   2 |          5
                   3 |          5
                   4 |          7
                   5 |          6
                   6 |          9
                   7 |         10
                   8 |         15
                   9 |         11
                  10 |         14
                  11 |         18
                  12 |         17
```

```
                  13 |          17
                  14 |          12
                  15 |          21
                  16 |          20
                  17 |          13
                  18 |          13
                  19 |          14
                  20 |          11
(20 rows)
```

3.3

On cherche la station de métro qui à la plus grande distance la séparant de sa station la plus proche :

```
SELECT a.l_station
FROM station_metro a
CROSS JOIN LATERAL (
  SELECT b.l_station, b.geom
  FROM station_metro b
  WHERE a.l_station <> b.l_station
  ORDER BY a.geom <-> b.geom ASC LIMIT 1
) AS b
ORDER BY a.geom <-> b.geom DESC LIMIT 1;
```

```
      l_station
----------------------
 CHATEAU DE VINCENNES
(1 row)
```

3.4

On cherche la paire de stations de métro séparée par la plus petite distance :

```
SELECT a.l_station, b.l_station, st_distance(a.geom, b.geom) AS distance
FROM station_metro a
CROSS JOIN LATERAL (
  SELECT b.l_station, b.geom
  FROM station_metro b
  WHERE a.l_station <> b.l_station
  ORDER BY st_distance(a.geom, b.geom) ASC LIMIT 1
```

```
) AS b
ORDER BY st_distance(a.geom, b.geom) ASC LIMIT 1;

       l_station        | l_station |      distance
------------------------+-----------+--------------------
 SAINT-GERMAIN DES PRES | MABILLON  | 156.76913554296698
(1 row)
```

4.

- Il faudrait faire une carte **choroplèthe** (en aplats de couleurs) puisque la variable “part de logements vacants” est une variable *quantitative relative*.
- On pourrait créer, dans PostGIS, une table `quartier_log_vac` contenant la géométrie des quartiers, leurs labels et la part de logements vacants dans chacun d’entre eux :

```
CREATE TABLE quartier_log_vac
AS
SELECT n_qu, SUM(nb_logvac) / SUM(nb_log) * 100 as pct_logvac, l_qu,
 ↪  quartier.geom as geom
FROM iris
JOIN quartier ON iris.n_qu = quartier.c_qu
WHERE
    nb_log is not null AND
    nb_logvac > 0
GROUP BY n_qu, l_qu, quartier.geom
ORDER BY pct_logvac DESC;

ALTER TABLE quartier_log_vac ADD PRIMARY KEY (n_qu);

SELECT Populate_Geometry_Columns('public.quartier_log_vac'::regclass);
```

- Dans **QGIS** on va ensuite aller dans les options de symbologie, et demander une représentation *“graduée”* (*discrétisation des données en un nombre de classes donné*) :