TP SQL — la base `education.db` — Corrigé

Écoles, collèges, lycées, spécialités et communes de France : la base `education.db` est décrite en détail dans `README_bases.md`. Ce qu'il faut savoir pour ce TP :

- **aucune valeur `NULL`** ; dans les tables d'effectifs, l'absence de ligne signifie un effectif nul ;
- les **codes** (`code_insee`, `code_commune`, `dep_code`, `uai`…) sont des **chaînes de caractères** : on écrit `dep_code = '78'`, avec des apostrophes ;
- les établissements sont **datés** : la clé primaire de `ecoles`, `colleges` et `lycees` est le couple `(uai, rentree)`.

Rappels hors programme : `NULL`, `LEFT JOIN`, `WITH`, `IN`/`NOT IN`, `EXISTS`, `GROUP_CONCAT`.

Tout ce TP se traite avec `SELECT … FROM … JOIN … ON … WHERE … GROUP BY … HAVING … ORDER BY … LIMIT/OFFSET`, les sous-requêtes et `UNION`/`INTERSECT`/`EXCEPT`.

Chargement de la base

```
from sql_fonctions import GestionnaireSQL

db = GestionnaireSQL("education.db")
db.liste_tables()
```

```
name
-----------------
communes
ecoles
colleges
effectifs_colleges
lycees
effectifs_lycees
specialites
combinaisons
cpge
```

```
# Les schémas des tables utilisées dans ce TP.
for table in ["communes", "ecoles", "colleges", "effectifs_colleges",
              "lycees", "effectifs_lycees", "specialites", "cpge"]:
    print(f"{table} : {db.schema('SELECT * FROM ' + table)}")
```

```
communes : ['code_insee', 'nom_standard', 'nom_sans_pronom', 'nom_a',
    'nom_de', 'nom_sans_accent', 'nom_majuscule', 'reg_code', 'reg_nom',
    'dep_code', 'dep_nom', 'epci_code', 'epci_nom', 'code_postal',
    'academie_code', 'academie_nom', 'population', 'superficie_km2',
    'altitude_moyenne', 'altitude_min', 'altitude_max',
    'latitude_mairie', 'longitude_mairie', 'latitude_centre',
    'longitude_centre', 'grille_densite', 'grille_densite_texte',
    'gentile', 'url_wikipedia']
ecoles : ['uai', 'rentree', 'denomination', 'patronyme', 'secteur',
    'code_commune', 'rep', 'rep_plus', 'nb_classes', 'pre_elementaire',
    'elementaire', 'ulis']
colleges : ['uai', 'rentree', 'denomination', 'patronyme', 'secteur',
    'code_commune', 'rep', 'rep_plus']
```

```
effectifs_colleges : ['uai', 'rentree', 'niveau', 'filles', 'garcons']
lycees : ['uai', 'rentree', 'denomination', 'patronyme', 'secteur',
    'code_commune']
effectifs_lycees : ['uai', 'rentree', 'niveau', 'serie', 'filles',
    'garcons']
specialites : ['uai', 'rentree', 'niveau', 'code_specialite',
    'specialite', 'filles', 'garcons']
cpge : ['rentree', 'ministere', 'filiere', 'annee_etude',
    'garcons_public', 'filles_public', 'garcons_prive', 'filles_prive']
```

Exécuter une requête dans ce TP

Placer ce notebook, `education.db` et `sql_fonctions.py` dans le même dossier, puis exécuter les deux cellules de chargement ci-dessus avec **Maj + Entrée**. Dans une cellule de code, écrire une requête SQL entre les triples guillemets de `db.requete("""…""")`, puis exécuter la cellule : les noms des colonnes et les lignes du résultat s'affichent en dessous. Les triples guillemets permettent d'écrire la requête sur plusieurs lignes ; les valeurs textuelles SQL restent entre apostrophes. Par exemple :

```
db.requete("""
SELECT nom_standard
FROM communes
LIMIT 5
""")
```

Modifier la requête et réexécuter la cellule pour essayer une autre solution. Pour la partie graphique, `db.liste("""…""")` renvoie une liste de tuples à conserver dans une variable, tandis que `db.schema("""…""")` renvoie les noms des colonnes. Après un redémarrage du noyau, réexécuter les cellules de chargement.

1. Lire le schéma

Question 1. Sans écrire de requête :

1. Quelle est la clé primaire de `communes` ? Celle de `lycees` ? Pourquoi `uai` seul ne suffit-il pas ?
2. Quelle colonne de `lycees` est une clé étrangère, et vers quelle colonne de quelle table ?
3. Donner le type d'association ($1-1$, $1-*$ ou $*-*$) entre : `communes` et `lycees` ; `lycees` et `effectifs_lycees` ; les lycées et les spécialités.

Réponse.

1. `code_insee` ; le couple `(uai, rentree)` : un même établissement apparaît une fois **par rentrée**, `uai` seul n'est donc pas unique dans la table.
2. `code_commune` référence `communes(code_insee)` — les deux colonnes ont le **même contenu mais pas le même nom**.
3. `communes–lycees` : $1-*$ (une commune a plusieurs lycées, un lycée une seule commune) ; `lycees–effectifs_lycees` : $1-*$; lycées–spécialités : $*-*$, et c'est précisément le rôle de la **table d'association** `specialites`, dont chaque ligne relie un lycée (clé étrangère `(uai, rentree)`) à une spécialité (`code_specialite`).

2. Une seule table

Question 2 *(projection, sélection, tri)*. Le nom et la population des communes des Yvelines (`dep_code = '78'`) de plus de 20 000 habitants, par population décroissante.

```
db.requete("""
SELECT nom_standard, population
FROM communes
WHERE dep_code = '78' AND population > 20000  -- '78' entre apostrophes : dep_code
 ↪est une chaîne
ORDER BY population DESC
""")
```

```
nom_standard              population
-----------------------------------
Versailles                     84095
Sartrouville                   52763
Saint-Germain-en-Laye          45931
Mantes-la-Jolie                43526
Poissy                         40983
Conflans-Sainte-Honorine       36958
Trappes                        34689
Les Mureaux                    34632
Houilles                       33983
Montigny-le-Bretonneux         32465
Plaisir                        31811
Le Chesnay-Rocquencourt        30689
Chatou                         30598
Guyancourt                     29778
Rambouillet                    27724
Élancourt                      26365
Maisons-Laffitte               23093
Vélizy-Villacoublay            23011
Mantes-la-Ville                22332
Achères                        22241
Saint-Cyr-l'École              21268
Carrières-sous-Poissy          20825
Maurepas                       20629
La Celle-Saint-Cloud           20460
```

Question 3 *(`DISTINCT`)*.

1. Comparer le nombre de lignes de `colleges` et son nombre d'établissements distincts. Expliquer l'écart.
2. Combien de communes distinctes comptent au moins un collège REP+ à la rentrée 2025 ?

```
db.requete("""
SELECT COUNT(*) AS lignes, COUNT(DISTINCT uai) AS etablissements
FROM colleges
""")
# 48 676 lignes pour 7 029 collèges : une ligne par établissement ET par rentrée
 ↪(2019-2025).
```

```
lignes  etablissements
--------------------
 48676            7029
```

```
db.requete("""
SELECT COUNT(DISTINCT code_commune) AS nb_communes
FROM colleges
WHERE rentree = 2025 AND rep_plus = TRUE
""")
# « au moins un » se traduit par DISTINCT : une commune à deux collèges REP+ ne␣
 ↪compte qu'une fois.
# rep_plus est un booléen : on le compare à TRUE (l'écriture = 1 est acceptée).
```

```
nb_communes
-----------
        198
```

Question 4 *(`ORDER BY`, `LIMIT`, `OFFSET`)*. Les trois communes les plus peuplées de France, puis les communes des rangs 4 à 10 (nom et population).

```
db.requete("""
SELECT nom_standard, population
FROM communes
ORDER BY population DESC
LIMIT 3
""")

db.requete("""
SELECT nom_standard, population
FROM communes
ORDER BY population DESC
LIMIT 7 OFFSET 3
""")
# LIMIT sans ORDER BY renverrait des lignes non déterminées,
# et LIMIT tranche arbitrairement entre ex æquo.
```

```
nom_standard  population
------------------------
Paris            2103778
Marseille         886040
Lyon              519127
nom_standard  population
------------------------
Toulouse          514819
Nice              357737
Nantes            327734
Montpellier       310240
Strasbourg        293771
Bordeaux          267991
Lille             238246
```

Question 5 *(renommage `AS`)*. Les dix communes les plus denses parmi celles de plus de 10 000 habitants (nom et densité en habitants/km²).

```
db.requete("""
SELECT nom_standard, population / superficie_km2 AS densite
FROM communes
WHERE population > 10000       -- l'alias n'existe pas encore ici :
ORDER BY densite DESC          -- ... mais il existe déjà là !
LIMIT 10
""")
# Ordre d'exécution : FROM -> WHERE -> SELECT (l'alias naît ici) -> ORDER BY ->␣
 ↪LIMIT.
```

```
nom_standard                     densite
---------------------------------------
Levallois-Perret       28137.19008264463
Vincennes              25364.736842105263
Le Pré-Saint-Gervais   24275.714285714286
Saint-Mandé            23154.945054945056
Montrouge               22378.74396135266
Clichy                  20912.33766233766
Paris                    19954.2634923646
Courbevoie             19928.365384615383
Asnières-sur-Seine     19449.482401656314
Boulogne-Billancourt   19321.266233766233
```

Agrégats

Question 6 *(agrégats simples)*. En une seule requête : le nombre de communes des Yvelines, leur population totale, leur superficie totale et l'altitude du plus haut sommet du département.

```
db.requete("""
SELECT COUNT(*) AS nb_communes, SUM(population) AS nb_habitants,
       SUM(superficie_km2) AS km2, MAX(altitude_max) AS sommet
FROM communes
WHERE dep_code = '78'
""")
```

```
nb_communes  nb_habitants      km2  sommet
-----------------------------------------
        259       1485086  2305.68     201
```

Question 7 *(`GROUP BY`)*. Le nombre de communes et la population totale de chaque région, par population décroissante (nom de région, nombre de communes, population de la région).

```
db.requete("""
SELECT reg_nom, COUNT(*) AS nb_communes, SUM(population) AS nb_habitants
FROM communes
GROUP BY reg_nom
ORDER BY nb_habitants DESC
""")
```

```
reg_nom                     nb_communes  nb_habitants
-------------------------------------------------
Île-de-France                      1266      12463067
Auvergne-Rhône-Alpes               4019       8199817
```

```
Nouvelle-Aquitaine                4293       6150451
Occitanie                         4445       6123113
Hauts-de-France                   3781       5992194
Grand Est                         5115       5563378
Provence-Alpes-Côte d'Azur         946       5218960
Pays de la Loire                  1225       3888744
Bretagne                          1202       3449370
Normandie                         2643       3343946
Bourgogne-Franche-Comté           3685       2802670
Centre-Val de Loire               1754       2587031
La Réunion                          24        889679
Guadeloupe                          32        384160
Martinique                          34        360630
Corse                              360        355486
Guyane                              22        293996
Mayotte                             17        256518
```

Question 8 *(`WHERE` et `HAVING`)*. Les académies comptant au moins cinq communes de plus de 50 000 habitants, avec ce nombre, par ordre décroissant.

```
db.requete("""
SELECT academie_nom, COUNT(*) AS nb
FROM communes
WHERE population > 50000   -- filtre les LIGNES, avant regroupement
GROUP BY academie_nom
HAVING COUNT(*) >= 5       -- filtre les GROUPES, après agrégation
ORDER BY nb DESC
""")
```

```
academie_nom  nb
----------------
Créteil       23
Versailles    21
Nice           9
Nantes         7
Lille          6
La Réunion     6
Rennes         5
Montpellier    5
Lyon           5
Bordeaux       5
```

Question 9 *(division entière !)*. La proportion de filles en terminale générale (série `G`) en France à la rentrée 2025.

```
db.requete("""
SELECT SUM(filles) / SUM(filles + garcons) AS proportion
FROM effectifs_lycees
WHERE rentree = 2025 AND niveau = 'terminale' AND serie = 'G'
""")
# 0 ! Les effectifs sont entiers : la division est ENTIÈRE.
```

```
proportion
----------
         0
```

```
db.requete("""
SELECT 1.0 * SUM(filles) / SUM(filles + garcons) AS proportion
FROM effectifs_lycees
WHERE rentree = 2025 AND niveau = 'terminale' AND serie = 'G'
""")
# Multiplier par 1.0 force le calcul en flottant.
```

```
       proportion
-----------------
0.5562758311351282
```

Sous-requêtes

Question 10 *(sous-requête scalaire)*. La commune la plus peuplée des Yvelines, de deux façons : avec `ORDER BY … LIMIT 1`, puis avec une sous-requête scalaire. Quelle version garde les ex æquo ? Pourquoi `SELECT nom_standard, MAX(population)` serait-il incorrect ?

On affichera (nom, population).

```
db.requete("""
SELECT nom_standard, population
FROM communes
WHERE dep_code = '78'
ORDER BY population DESC
LIMIT 1
""")

db.requete("""
SELECT nom_standard, population
FROM communes
WHERE dep_code = '78'
  AND population = (SELECT MAX(population) FROM communes WHERE dep_code = '78')
""")
# La sous-requête scalaire garde les ex æquo ; LIMIT 1 en écarterait.
# SELECT nom_standard, MAX(population) est incorrect : une colonne non agrégée
# n'a pas de sens à côté d'un agrégat (même si certains logiciels l'acceptent).
```

```
nom_standard  population
------------------------
Versailles         84095
nom_standard  population
------------------------
Versailles         84095
```

Question 11 *(rang)*. Le rang de Versailles au classement des communes françaises par population décroissante.

```
db.requete("""
SELECT 1 + COUNT(*) AS rang
FROM communes
WHERE population > (SELECT population FROM communes WHERE nom_standard =␣
 ↪'Versailles')
""")
# Rang = 1 + nombre de communes strictement plus peuplées.
```

```
# La sous-requête doit être SCALAIRE : on a vérifié qu'une seule commune s'appelle␣
 ↪Versailles.
```

```
rang
----
  56
```

Question 12 *(agrégat d'agrégat)*. Le nombre moyen d'écoles par commune à la rentrée 2025 (parmi les communes qui en ont au moins une).

```
db.requete("""
SELECT AVG(nb) AS ecoles_par_commune
FROM (SELECT COUNT(*) AS nb
      FROM ecoles
      WHERE rentree = 2025
      GROUP BY code_commune)
""")
# AVG(COUNT(*)) n'existe pas : un agrégat d'agrégat exige une sous-requête dans le␣
 ↪FROM.
```

```
ecoles_par_commune
------------------
2.2388224559381453
```

Question 13 *(compter des groupes)*. Dans un département, appelons *doublon* un ensemble de communes ayant exactement la même population. Les départements comptant au moins cent doublons, avec ce nombre, par nombre décroissant.

```
db.requete("""
SELECT dep_nom, COUNT(*) AS nb_doublons
FROM (SELECT dep_nom, population
      FROM communes
      GROUP BY dep_nom, population
      HAVING COUNT(*) > 1)
GROUP BY dep_nom
HAVING COUNT(*) >= 100
ORDER BY nb_doublons DESC
""")
# Deux GROUP BY empilés et deux HAVING : celui du DEDANS ne garde que les
# groupes d'au moins deux communes (les doublons), celui du DEHORS ne garde que
# les départements qui en comptent au moins cent.
```

```
dep_nom             nb_doublons
------------------------------
Aisne                       172
Somme                       170
Côte-d'Or                   163
Pas-de-Calais               153
Marne                       128
Haute-Saône                 120
Meuse                       119
Seine-Maritime              114
Oise                        113
Moselle                     112
Jura                        106
```

```
Meurthe-et-Moselle          103
Hautes-Pyrénées             103
Gers                        103
Doubs                       103
Eure                        100
```

3. Jointures

Toute cette partie se joue dans l'univers des lycées, avec des tables introduites une à une : d'abord le couple `lycees–communes`, puis `effectifs_lycees`, et enfin la table d'association `specialites`.

3.1 Un premier couple : `lycees` et `communes`

Question 14 *(jointure simple)*. La dénomination, le patronyme et le secteur des lycées de Versailles à la rentrée 2025.

```
db.requete("""
SELECT L.denomination, L.patronyme, L.secteur
FROM (lycees AS L JOIN communes AS C ON L.code_commune = C.code_insee)
WHERE C.nom_standard = 'Versailles' AND L.rentree = 2025
""")
# La condition de jointure apparie la clé étrangère code_commune à la clé
# primaire code_insee : deux colonnes de même contenu... mais pas de même nom.
```

```
denomination                  patronyme                 secteur
------------------------------------------------------------------
LYCEE D'ENSEIGNEMENT GENERAL  HOCHE                     PUBLIC
LYCEE GENERAL ET TECHNOLOGIQUE  LA BRUYERE              PUBLIC
LYCEE POLYVALENT              JULES FERRY               PUBLIC
LYCEE GEN.ET TECHNOL.PRIVE    LES CHATAIGNIERS          PRIVE
LYCEE POLYVALENT PRIVE        ST VINCENT DE PAUL        PRIVE
LYCEE GEN.ET TECHNOL.PRIVE    SAINT JEAN ET HULST       PRIVE
LYCEE GEN.ET TECHNOL.PRIVE    NOTRE DAME DU GRANDCHAMP  PRIVE
LYCEE GENERAL ET TECHNOLOGIQUE  MARIE CURIE             PUBLIC
```

Question 15 *(désigner par la clé, pas par le nom)*. Combien la commune nommée Saint-Denis compte-t-elle de lycées à la rentrée 2025 ? La réponse obtenue est-elle correcte ?

```
db.requete("""
SELECT COUNT(*) AS nb_lycees
FROM (lycees AS L JOIN communes AS C ON L.code_commune = C.code_insee)
WHERE L.rentree = 2025 AND C.nom_standard = 'Saint-Denis'
""")
# 11 lycées ? Non : un NOM ne désigne pas une commune.
```

```
nb_lycees
---------
       11
```

```
db.requete("""
SELECT C.nom_standard, C.dep_nom, COUNT(*) AS nb_lycees
FROM (lycees AS L JOIN communes AS C ON L.code_commune = C.code_insee)
WHERE L.rentree = 2025 AND C.nom_standard = 'Saint-Denis'
GROUP BY C.code_insee, C.nom_standard, C.dep_nom
```

```
""")
# Le 11 fusionnait DEUX communes : Saint-Denis (93) et Saint-Denis (974).
# On regroupe par la CLÉ PRIMAIRE (en répétant dans le GROUP BY les colonnes
# affichées) : 1 446 noms de communes sont partagés, seul le code identifie.
```

```
nom_standard  dep_nom            nb_lycees
----------------------------------------
Saint-Denis   Seine-Saint-Denis          5
Saint-Denis   La Réunion                 6
```

3.2 Un deuxième couple : `lycees` et `effectifs_lycees`

Question 16 *(condition de jointure composée)*. Le patronyme et l'effectif de terminale G des lycées de Versailles à la rentrée 2025, par effectif décroissant. Le code INSEE de Versailles, lu dans `communes` à la question 14, est `'78646'`.

```
db.requete("""
SELECT L.patronyme, E.filles + E.garcons AS effectif
FROM (lycees AS L JOIN effectifs_lycees AS E
      ON L.uai = E.uai AND L.rentree = E.rentree)
WHERE L.code_commune = '78646' AND L.rentree = 2025
  AND E.niveau = 'terminale' AND E.serie = 'G'
ORDER BY effectif DESC
""")
# Les établissements sont datés : la jointure porte sur uai ET rentree.
# En oubliant la seconde égalité, chaque lycée serait apparié aux effectifs
# de TOUTES ses rentrées.
```

```
patronyme                 effectif
---------------------------------
HOCHE                          377
LA BRUYERE                     376
SAINT JEAN ET HULST            314
NOTRE DAME DU GRANDCHAMP       254
MARIE CURIE                    236
JULES FERRY                    193
LES CHATAIGNIERS                27
```

Question 17 *(joindre avant d'agréger)*. Le plus grand effectif de terminale G d'un lycée privé à la rentrée 2025.

```
db.requete("""
SELECT MAX(E.filles + E.garcons) AS record
FROM (lycees AS L JOIN effectifs_lycees AS E
      ON L.uai = E.uai AND L.rentree = E.rentree)
WHERE L.secteur = 'PRIVE' AND E.rentree = 2025
  AND E.niveau = 'terminale' AND E.serie = 'G'
""")
# Le secteur est dans lycees, l'effectif dans effectifs_lycees : il faut
# JOINDRE avant d'agréger. Une sous-requête scalaire ne peut pas remplacer la
# jointure : il y a 852 lycées privés (comparer au « record de France », X 2019).
```

```
record
------
   495
```

Question 18 *(portée d'une sous-requête scalaire)*. Le ou les lycées privés atteignant ce record en 2025 (patronyme, effectif). Que renvoie la requête si la sous-requête calcule le maximum sur tous les lycées ?

```
db.requete("""
SELECT L.patronyme, E.filles + E.garcons AS effectif
FROM (lycees AS L JOIN effectifs_lycees AS E
      ON L.uai = E.uai AND L.rentree = E.rentree)
WHERE L.secteur = 'PRIVE' AND E.rentree = 2025
  AND E.niveau = 'terminale' AND E.serie = 'G'
  AND E.filles + E.garcons = (SELECT MAX(filles + garcons)
                              FROM effectifs_lycees
                              WHERE rentree = 2025
                                AND niveau = 'terminale' AND serie = 'G')
""")
# Résultat vide : le maximum TOUS SECTEURS (658 élèves) est atteint par un
# lycée public. La sous-requête a une portée trop large.
```

```
patronyme  effectif
------------------
Résultat vide.
```

```
db.requete("""
SELECT L.patronyme, E.filles + E.garcons AS effectif
FROM (lycees AS L JOIN effectifs_lycees AS E
      ON L.uai = E.uai AND L.rentree = E.rentree)
WHERE L.secteur = 'PRIVE' AND E.rentree = 2025
  AND E.niveau = 'terminale' AND E.serie = 'G'
  AND E.filles + E.garcons =
      (SELECT MAX(E2.filles + E2.garcons)
       FROM (lycees AS L2 JOIN effectifs_lycees AS E2
             ON L2.uai = E2.uai AND L2.rentree = E2.rentree)
       WHERE L2.secteur = 'PRIVE' AND E2.rentree = 2025
         AND E2.niveau = 'terminale' AND E2.serie = 'G')
""")
# La sous-requête doit refaire la MÊME jointure et le MÊME filtre : son maximum
# porte alors sur les seuls lycées privés (cf. CCMP 2022 Q8, 2024 Q9).
```

```
patronyme          effectif
---------------------------
SAINTE MARIE LYON       495
```

3.3 La même table des deux côtés : autojointures

Question 19 *(autojointure)*. Les paires de lycées distincts de Versailles à la rentrée 2025. Combien de lignes obtiendrait-on avec `<>` au lieu de `<` ? Et sans aucune condition sur les `uai` ?

```
db.requete("""
SELECT L1.patronyme, L2.patronyme
FROM (lycees AS L1 JOIN lycees AS L2
      ON L1.code_commune = L2.code_commune AND L1.rentree = L2.rentree)
WHERE L1.rentree = 2025 AND L1.code_commune = '78646'
  AND L1.uai < L2.uai
```

```
ORDER BY L1.patronyme, L2.patronyme
""")
# Versailles a 8 lycées. Avec L1.uai < L2.uai : 28 paires.
# Avec <> : 56 lignes, chaque paire apparaît DEUX FOIS (cf. CCMP 2021 Q4).
# Sans condition : 64 lignes, chaque lycée est apparié à lui-même.
```

```
patronyme           patronyme
------------------------------------------
HOCHE               JULES FERRY
HOCHE               LA BRUYERE
HOCHE               LES CHATAIGNIERS
HOCHE               MARIE CURIE
HOCHE               NOTRE DAME DU GRANDCHAMP
HOCHE               SAINT JEAN ET HULST
HOCHE               ST VINCENT DE PAUL
JULES FERRY         LES CHATAIGNIERS
JULES FERRY         MARIE CURIE
JULES FERRY         NOTRE DAME DU GRANDCHAMP
JULES FERRY         SAINT JEAN ET HULST
JULES FERRY         ST VINCENT DE PAUL
LA BRUYERE          JULES FERRY
LA BRUYERE          LES CHATAIGNIERS
LA BRUYERE          MARIE CURIE
LA BRUYERE          NOTRE DAME DU GRANDCHAMP
LA BRUYERE          SAINT JEAN ET HULST
LA BRUYERE          ST VINCENT DE PAUL
LES CHATAIGNIERS    NOTRE DAME DU GRANDCHAMP
LES CHATAIGNIERS    SAINT JEAN ET HULST
LES CHATAIGNIERS    ST VINCENT DE PAUL
MARIE CURIE         LES CHATAIGNIERS
MARIE CURIE         NOTRE DAME DU GRANDCHAMP
MARIE CURIE         SAINT JEAN ET HULST
MARIE CURIE         ST VINCENT DE PAUL
SAINT JEAN ET HULST NOTRE DAME DU GRANDCHAMP
ST VINCENT DE PAUL  NOTRE DAME DU GRANDCHAMP
ST VINCENT DE PAUL  SAINT JEAN ET HULST
```

Question 20 *(autojointure temporelle)*. Pour chaque lycée de Versailles, l'effectif de terminale G à la rentrée 2019 et à la rentrée 2025, et la variation entre les deux, par variation décroissante.

```
db.requete("""
SELECT L.patronyme,
       E19.filles + E19.garcons AS effectif_2019,
       E25.filles + E25.garcons AS effectif_2025,
       (E25.filles + E25.garcons) - (E19.filles + E19.garcons) AS variation
FROM ((lycees AS L JOIN effectifs_lycees AS E25
       ON L.uai = E25.uai AND L.rentree = E25.rentree)
      JOIN effectifs_lycees AS E19
      ON E19.uai = E25.uai AND E19.niveau = E25.niveau AND E19.serie = E25.serie)
WHERE L.rentree = 2025 AND E19.rentree = 2019
  AND E25.niveau = 'terminale' AND E25.serie = 'G'
  AND L.code_commune = '78646'
ORDER BY variation DESC
```

```
""")
# Deux copies de effectifs_lycees, une par rentrée : on joint la table à
# elle-même sur (uai, niveau, serie) et chaque copie reçoit sa propre rentrée.
# Un lycée sans terminale G à l'une des deux rentrées disparaît du résultat.
```

```
patronyme                 effectif_2019  effectif_2025  variation
------------------------------------------------------------------
SAINT JEAN ET HULST                 307            314          7
NOTRE DAME DU GRANDCHAMP            254            254          0
MARIE CURIE                         246            236        -10
HOCHE                               397            377        -20
LA BRUYERE                          401            376        -25
JULES FERRY                         242            193        -49
```

3.4 Trois tables et plus

Question 21 *(jointure triple, agrégat)*. Les dix académies dont l'effectif total de terminale G est le plus élevé à la rentrée 2025 (nom, effectif).

```
db.requete("""
SELECT C.academie_nom, SUM(E.filles + E.garcons) AS effectif
FROM ((effectifs_lycees AS E JOIN lycees AS L
       ON E.uai = L.uai AND E.rentree = L.rentree)
      JOIN communes AS C ON L.code_commune = C.code_insee)
WHERE E.rentree = 2025 AND E.niveau = 'terminale' AND E.serie = 'G'
GROUP BY C.academie_nom
ORDER BY effectif DESC
LIMIT 10
""")
# Trois tables, mais rien de neuf : l'effectif est dans effectifs_lycees,
# l'académie dans communes, et lycees fait le pont entre les deux.
```

```
academie_nom   effectif
----------------------
Versailles        41509
Créteil           27218
Lille             22026
Nantes            20826
Grenoble          19135
Lyon              19027
Bordeaux          18350
Rennes            17897
Aix-Marseille     16971
Normandie         16790
```

Question 22 *(maximum par groupe)*. Pour chaque académie, le lycée dont l'effectif de terminale G est le plus élevé à la rentrée 2025 (académie, patronyme, effectif), par effectif décroissant. Pourquoi la sous-requête scalaire de la question 18 ne suffit-elle plus ici ?

```
db.requete("""
SELECT C.academie_nom, L.patronyme, E.filles + E.garcons AS effectif
FROM (((effectifs_lycees AS E JOIN lycees AS L
        ON E.uai = L.uai AND E.rentree = L.rentree)
```

```
       JOIN communes AS C ON L.code_commune = C.code_insee)
      JOIN (SELECT C2.academie_nom AS academie,
                   MAX(E2.filles + E2.garcons) AS record
            FROM ((effectifs_lycees AS E2 JOIN lycees AS L2
                   ON E2.uai = L2.uai AND E2.rentree = L2.rentree)
                  JOIN communes AS C2 ON L2.code_commune = C2.code_insee)
            WHERE E2.rentree = 2025 AND E2.niveau = 'terminale' AND E2.serie = 'G'
            GROUP BY C2.academie_nom) AS M
      ON C.academie_nom = M.academie AND E.filles + E.garcons = M.record)
WHERE E.rentree = 2025 AND E.niveau = 'terminale' AND E.serie = 'G'
ORDER BY effectif DESC
""")
# Une sous-requête SCALAIRE ne renvoie QU'UNE valeur : elle donnerait le record
# de France (658, question 18), et non celui de chaque académie. Il en faudrait
# une PAR académie.
# On les calcule donc toutes d'un coup par un GROUP BY : la sous-requête du FROM
# produit la TABLE (academie, record), qu'on JOINT au résultat de la question 21
# sur l'académie ET sur l'effectif. Ne survivent que les lignes atteignant le
# record de LEUR académie : 30 lignes, une par académie (les ex æquo seraient
# tous conservés, comme à la question 10).
```

```
academie_nom      patronyme                    effectif
-------------------------------------------------------
Versailles        ROSA PARKS                        658
Lyon              INTERNATIONAL                     649
Nice              AUGUSTE RENOIR                    491
Toulouse          SAINT-SERNIN                      462
Rennes            ST PAUL                           460
Grenoble          CLAUDE LOUIS BERTHOLLET           445
Montpellier       FRANCOIS ARAGO                    442
Aix-Marseille     GEORGES DUBY                      434
Clermont-Ferrand  JEANNE D'ARC                      430
Bordeaux          FRANCOIS MAURIAC                  425
Nancy-Metz        HENRI POINCARE                    415
Normandie         MALHERBE                          406
Nantes            SAINT BENOIT                      401
Mayotte           YOUNOUSSA BAMANA                  380
Poitiers          JEAN DAUTET                       380
Orléans-Tours     GRANDMONT                         379
Lille             INTERNATIONAL MONTEBELLO          374
Paris             HELENE BOUCHER                    368
Corse             GIOCANTE DE CASABIANCA            360
Créteil           WOLFGANG AMADEUS MOZART           349
Amiens            PIERRE D AILLY                    348
Strasbourg        JEAN MERMOZ                       347
Dijon             CATHERINE ET RAYMOND JANOT        338
Limoges           D ARSONVAL                        325
Besançon          XAVIER MARMIER                    311
Reims             JEAN JAURES                       310
Guadeloupe        BAIMBRIDGE                        281
Guyane            LEON-GONTRAN DAMAS                274
La Réunion        EVARISTE DE PARNY                 249
Martinique        BELLEVUE                          249
```

Question 23 *(deux jointures triples)*. Les dix académies où la proportion d'élèves de terminale inscrits en voie générale (série `G`) est la plus élevée à la rentrée 2025 (nom, proportion).

```
db.requete("""
SELECT T1.academie, 1.0 * T1.effectif / T2.effectif AS proportion_TG
FROM ((SELECT C.academie_nom AS academie,
              SUM(E.filles + E.garcons) AS effectif
       FROM ((effectifs_lycees AS E JOIN lycees AS L
              ON E.uai = L.uai AND E.rentree = L.rentree)
             JOIN communes AS C ON L.code_commune = C.code_insee)
       WHERE E.rentree = 2025 AND E.niveau = 'terminale' AND E.serie = 'G'
       GROUP BY C.academie_nom) AS T1
      JOIN
      (SELECT C.academie_nom AS academie,
              SUM(E.filles + E.garcons) AS effectif
       FROM ((effectifs_lycees AS E JOIN lycees AS L
              ON E.uai = L.uai AND E.rentree = L.rentree)
             JOIN communes AS C ON L.code_commune = C.code_insee)
       WHERE E.rentree = 2025 AND E.niveau = 'terminale'
       GROUP BY C.academie_nom) AS T2
      ON T1.academie = T2.academie)
ORDER BY proportion_TG DESC
LIMIT 10
""")
# Le numérateur et le dénominateur portent sur des ENSEMBLES DE LIGNES
# DIFFÉRENTS (la terminale G, toute la terminale) : un seul GROUP BY ne peut
# pas les produire ensemble. On calcule donc chaque total dans sa propre
# sous-requête du FROM, puis on les lie académie par académie.
```

```
academie               proportion_TG
-----------------------------------
Paris             0.8300464967131633
Nice              0.7724336865868189
Corse             0.7682983682983683
Bordeaux          0.7581391505536275
Versailles        0.7501807272464397
Strasbourg         0.742875788557755
Poitiers          0.7420810626702997
Clermont-Ferrand  0.7409080104587592
Nantes            0.7374384759746468
Reims             0.7364384797564001
```

Question 24 *(table d'association : traduire un « et »)*. Les lycées de l'académie de Versailles proposant, en terminale à la rentrée 2025, à la fois « Numérique et sciences informatiques » et « Arts plastiques » (patronyme du lycée, nom de la commune). Pourquoi la condition `specialite = 'Numérique et sciences informatiques' AND specialite = 'Arts plastiques'` renvoie-t-elle une table vide ?

```
db.requete("""
SELECT L.patronyme
FROM ((specialites AS S JOIN lycees AS L ON S.uai = L.uai AND S.rentree = L.rentree)
      JOIN communes AS C ON L.code_commune = C.code_insee)
WHERE S.rentree = 2025 AND S.niveau = 'terminale' AND C.academie_nom = 'Versailles'
```

```
  AND S.specialite = 'Numérique et sciences informatiques'
  AND S.specialite = 'Arts plastiques'
""")
# Vide : une MÊME ligne de specialites ne peut pas porter deux valeurs à la fois.
```

```
patronyme
---------
Résultat vide.
```

```
db.requete("""
SELECT L.patronyme, C.nom_standard
FROM (((specialites AS S1 JOIN specialites AS S2
        ON S1.uai = S2.uai AND S1.rentree = S2.rentree AND S1.niveau = S2.niveau)
       JOIN lycees AS L ON S1.uai = L.uai AND S1.rentree = L.rentree)
      JOIN communes AS C ON L.code_commune = C.code_insee)
WHERE S1.rentree = 2025 AND S1.niveau = 'terminale' AND C.academie_nom =␣
 ↪'Versailles'
  AND S1.specialite = 'Numérique et sciences informatiques'
  AND S2.specialite = 'Arts plastiques'
ORDER BY C.nom_standard
""")
# Le « et » se traduit en joignant DEUX FOIS la table d'association : une copie
# par spécialité exigée (cf. X 2017, points et ensembles). La jointure de S1
# avec S2 est aussi une autojointure : tout de cette partie se retrouve ici.
```

```
patronyme                      nom_standard
-------------------------------------------------------
SAINTE MARIE LA CROIX          Antony
EDMOND MICHELET                Arpajon
NOTRE-DAME                     Bourg-la-Reine
SAINT PIERRE                   Brunoy
FRANCO ALLEMAND                Buc
EMMANUEL MOUNIER               Châtenay-Malabry
GUY DE MAUPASSANT              Colombes
JULES FERRY                    Conflans-Sainte-Honorine
ROBERT DOISNEAU                Corbeil-Essonnes
ECOLE EUROPEENNE DE PARIS LA D Courbevoie
PAUL LAPIE                     Courbevoie
NIKOLA TESLA                   Dourdan
RENE CASSIN                    Gonesse
MONTESQUIEU                    Herblay-sur-Seine
FRAGONARD                      L'Isle-Adam
JEAN MONNET                    La Queue-les-Yvelines
L'ESSOURIAU                    Les Ulis
ST EXUPERY                     Mantes-la-Jolie
NOTRE-DAME DE BURY             Margency
PARC DE VILGENIS               Massy
FUSTEL DE COULANGES            Massy
ROSA PARKS                     Montgeron
JEAN-JACQUES ROUSSEAU          Montmorency
COURS SECONDAIRE               Orsay
LE CORBUSIER                   Poissy
CAMILLE PISSARRO               Pontoise
LOUIS BASCAN                   Rambouillet
```

```
RICHELIEU                      Rueil-Malmaison
MANSART                        Saint-Cyr-l'École
ALBERT EINSTEIN                Sainte-Geneviève-des-Bois
JEAN-BAPTISTE COROT            Savigny-sur-Orge
MARIE CURIE                    Sceaux
JEAN-PIERRE VERNANT            Sèvres
CAMILLE CLAUDEL                Vauréal
HOCHE                          Versailles
LA BRUYERE                     Versailles
NOTRE DAME DE SION             Évry-Courcouronnes
```

4. Ensembles et antijointure

Question 25 *(INTERSECT)*. Les codes INSEE des communes des Yvelines ayant au moins un lycée à la rentrée 2025.

```
db.requete("""
SELECT code_insee FROM communes WHERE dep_code = '78'
INTERSECT
SELECT code_commune FROM lycees WHERE rentree = 2025
""")
# Les deux requêtes renvoient une colonne de MÊME TYPE (un code de commune)
# mais de noms différents : c'est la position qui compte, pas le nom.
# Variante : jointure + SELECT DISTINCT.
```

```
code_insee
----------
78005
78029
78117
78124
78126
78158
78172
78297
78335
78354
78358
78361
78372
78383
78401
78423
78440
78490
78498
78501
78513
78517
78545
78551
78586
78621
78642
```

```
78646
78650
78674
78683
```

Question 26 *(EXCEPT et antijointure)*.

1. Combien de communes de plus de 10 000 habitants n'ont aucun lycée à la rentrée 2025 ?
2. Donner le nom, le département et la population des dix plus peuplées d'entre elles.

```
db.requete("""
SELECT COUNT(*) AS nb_communes
FROM (SELECT code_insee FROM communes WHERE population > 10000
      EXCEPT
      SELECT code_commune FROM lycees WHERE rentree = 2025)
""")
```

```
nb_communes
-----------
        334
```

```
db.requete("""
SELECT C.nom_standard, C.dep_nom, C.population
FROM (communes AS C
      JOIN (SELECT code_insee FROM communes WHERE population > 10000
            EXCEPT
            SELECT code_commune FROM lycees WHERE rentree = 2025) AS S
      ON C.code_insee = S.code_insee)
ORDER BY C.population DESC
LIMIT 10
""")
# EXCEPT ne renvoie que la colonne comparée : pour retrouver nom et population,
# on REJOINT le résultat à communes. C'est l'antijointure du programme
# (NOT IN est hors programme).
```

```
nom_standard            dep_nom          population
--------------------------------------------------
Villenave-d'Ornon       Gironde               42545
Le Cannet               Alpes-Maritimes       41938
Six-Fours-les-Plages    Var                   37109
Châtillon               Hauts-de-Seine        36705
Schiltigheim            Bas-Rhin              34708
Houilles                Yvelines              33983
Villiers-sur-Marne      Val-de-Marne          33162
Saint-Médard-en-Jalles  Gironde              32910
Saint-Laurent-du-Var    Alpes-Maritimes       32172
Koungou                 Mayotte               32156
```

Question 27 *(distance)*. La commune des Yvelines, autre que Versailles, dont le centre est le plus proche de celui de Versailles (au sens de la distance euclidienne sur les coordonnées `latitude_centre`, `longitude_centre`).

```
db.requete("""
SELECT nom_standard
```

```
FROM communes
WHERE dep_code = '78' AND nom_standard <> 'Versailles'
ORDER BY
    (latitude_centre - (SELECT latitude_centre FROM communes
                        WHERE nom_standard = 'Versailles'))
  * (latitude_centre - (SELECT latitude_centre FROM communes
                        WHERE nom_standard = 'Versailles'))
  + (longitude_centre - (SELECT longitude_centre FROM communes
                         WHERE nom_standard = 'Versailles'))
  * (longitude_centre - (SELECT longitude_centre FROM communes
                         WHERE nom_standard = 'Versailles'))
LIMIT 1
""")
# Minimiser la distance = minimiser son carré : inutile de calculer une racine
# (et l'écart longitude/latitude n'est qu'approximativement euclidien).
```

```
nom_standard
----------------------
Le Chesnay-Rocquencourt
```

5. Synthèse et compléments

Question 28 *(la requête complète)*. Pour chaque académie, le nombre de lycées publics de la rentrée 2025 situés dans une commune de moins de 20 000 habitants ; ne garder que les académies en comptant au moins vingt, par nombre décroissant de lycées.

```
db.requete("""
SELECT K.academie_nom, COUNT(*) AS nb_lycees                    -- 4. projection
FROM (lycees AS L JOIN communes AS K
      ON L.code_commune = K.code_insee)                         -- 1. jointure
WHERE L.rentree = 2025 AND L.secteur = 'PUBLIC'
  AND K.population < 20000                                      -- 2. lignes
GROUP BY K.academie_nom                                         -- 3. groupes
HAVING COUNT(*) >= 20                                           -- 3'. filtre des␣
 ↪groupes
ORDER BY nb_lycees DESC                                         -- 5. tri
""")
```

```
academie_nom      nb_lycees
--------------------------
Normandie                52
Toulouse                 46
Grenoble                 45
Bordeaux                 36
Nancy-Metz               34
Lille                    33
Versailles               32
Rennes                   31
Nantes                   31
Créteil                  31
Montpellier              27
Strasbourg               25
Lyon                     25
Clermont-Ferrand         22
```

```
Orléans-Tours            21
Dijon                    21
Poitiers                 20
```

Question 29 *(bonus, une table pour finir)*. Pour chaque rentrée, le nombre total d'étudiants en CPGE et la proportion de filles. Attention à la division…

```
db.requete("""
SELECT rentree,
       SUM(garcons_public + filles_public + garcons_prive + filles_prive) AS total,
       1.0 * SUM(filles_public + filles_prive)
           / SUM(garcons_public + filles_public + garcons_prive + filles_prive)
       AS part_filles
FROM cpge
GROUP BY rentree
ORDER BY rentree
""")
```

```
rentree  total          part_filles
---------------------------------
   2009  81135   0.4268441486411536
   2010  79874  0.41943561108746275
   2011  80411  0.41892278419619206
   2012  82221   0.4206711180841877
   2013  83520   0.4202586206896552
   2014  84151  0.41850958396216326
   2015  85938   0.4207568246875655
   2016  86473  0.42575139060747286
   2017  86478  0.42793542866393763
   2018  85121  0.42555891025716336
   2019  85070   0.4224520982720113
   2020  84903  0.41852466932852783
```

Question 30 *(bonus, retour sur la question 15)*. Plusieurs communes peuvent porter le même nom. On s'intéresse ici aux couples de communes homonymes.

1. Combien y a-t-il de couples de communes distinctes portant le même nom ?
2. En donner les vingt premiers par ordre alphabétique (nom, et les deux départements).

```
db.requete("""
SELECT COUNT(*) AS nb_couples
FROM (communes AS C1 JOIN communes AS C2 ON C1.nom_standard = C2.nom_standard)
WHERE C1.code_insee < C2.code_insee
""")
# 3 893 couples pour 1 446 noms : certains noms sont portés par plus de deux
# communes, et n communes homonymes fournissent n(n-1)/2 couples.
```

```
nb_couples
----------
      3893
```

```
db.requete("""
SELECT C1.nom_standard, C1.dep_nom AS departement_1, C2.dep_nom AS departement_2
FROM (communes AS C1 JOIN communes AS C2 ON C1.nom_standard = C2.nom_standard)
```

```
WHERE C1.code_insee < C2.code_insee
ORDER BY C1.nom_standard
LIMIT 20
""")
# L'autojointure de la question 19 au service de la question 15.
# On compare les CLÉS PRIMAIRES et non les dep_code : avec la condition
# C1.dep_code < C2.dep_code, deux homonymes d'un MÊME département
# s'échapperaient (il n'y en a aucun ici, mais la requête serait fausse).
```

```
nom_standard  departement_1            departement_2
-----------------------------------------------------
Abancourt     Nord                     Oise
Aboncourt     Meurthe-et-Moselle       Moselle
Abzac         Charente                 Gironde
Achères       Cher                     Yvelines
Aiglun        Alpes-de-Haute-Provence  Alpes-Maritimes
Aigremont     Gard                     Haute-Marne
Aigremont     Gard                     Yvelines
Aigremont     Gard                     Yonne
Aigremont     Haute-Marne              Yvelines
Aigremont     Haute-Marne              Yonne
Aigremont     Yvelines                 Yonne
Aigueperse    Puy-de-Dôme              Rhône
Aigues-Vives  Ariège                   Aude
Aigues-Vives  Ariège                   Gard
Aigues-Vives  Ariège                   Hérault
Aigues-Vives  Aude                     Gard
Aigues-Vives  Aude                     Hérault
Aigues-Vives  Gard                     Hérault
Ainvelle      Haute-Saône              Vosges
Alaincourt    Aisne                    Haute-Saône
```

6. Utilisation des données *(bonus)*

Cette partie ne sera pas traitée en TP. Elle montre ce que l'on fait des résultats d'une requête une fois revenu dans Python : le SQL fait le calcul, matplotlib le donne à voir.

Importation de matplotlib et de la fonction $\log_{10}$

```
import matplotlib.pyplot as plt
from math import log10
```

On écrit les requêtes en langage SQL comme précédemment, mais on utilise la fonction `db.liste` pour récupérer la liste des données demandées (sans le schéma de la table).

Création d'une carte

Question 31 *(du SQL au graphique)*

1. Récupérer dans une liste nommée `V` la longitude et la latitude du centre de chaque commune de France métropolitaine, ainsi que sa densité (hab/km^2).
2. Stocker dans `x` la liste des longitudes, dans `y` celle des latitudes et dans `L_d` les images des densités par $t \mapsto \log_{10}(t+1)$.

3. À l'aide d'une requête SQL, récupérer dans `D` la densité maximale des communes de France métropolitaine.

```
V=db.liste("""
SELECT longitude_centre, latitude_centre, population/superficie_km2 FROM communes
WHERE dep_code<'96'
""")
```

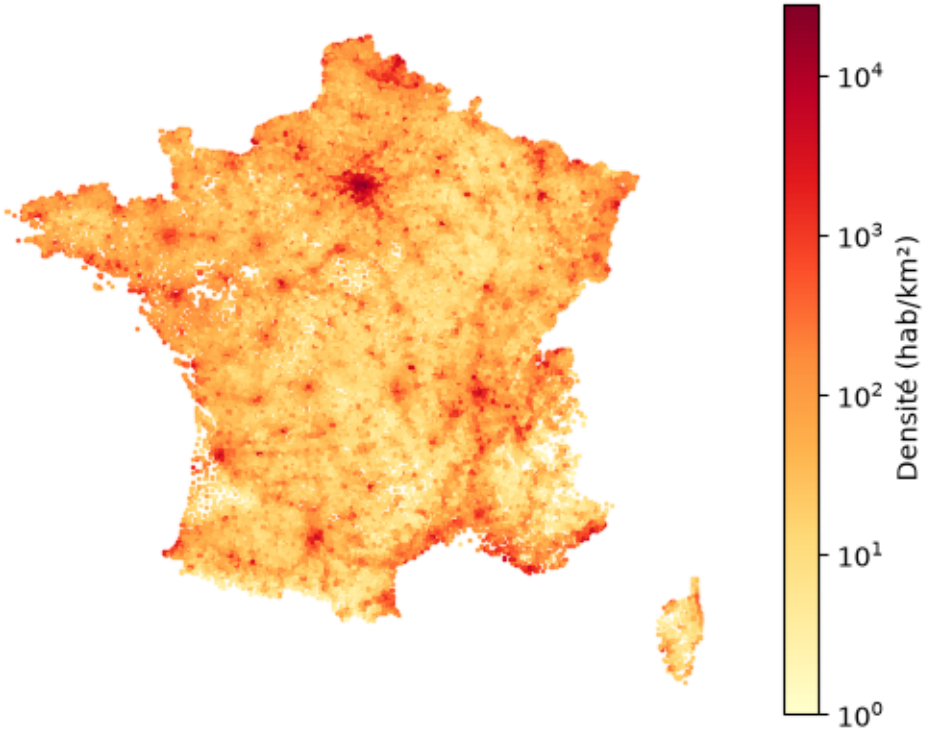
```
x = [t[0] for t in V]
y = [t[1] for t in V]
L_d = [log10(t[2]+1) for t in V]
```

```
D=db.liste("""
SELECT MAX(population/superficie_km2) FROM communes
WHERE dep_code<'96'
""")[0][0]

print(D)
```

```
28137.19008264463
```

Une fois `x`, `y`, `L_d` et `D` calculés, exécuter la cellule suivante pour tracer la carte.

```
fig, ax = plt.subplots()

# Tracé
points = ax.scatter(x, y, c=L_d, cmap="YlOrRd", vmin=0, vmax=log10(D+1),␣
 ↪marker="s", s=0.75)

# Création de la légende
cbar = fig.colorbar(points, label="Densité (hab/km²)")

# Graduations entières de 0 à la valeur maximale possible
ticks = list(range(0, int(log10(D+1)) + 1))

# Étiquettes en puissances de 10 correspondantes
labels = [f"$10^{k}$" for k in ticks]

cbar.set_ticks(ticks)
cbar.set_ticklabels(labels)

ax.set_axis_off()
```

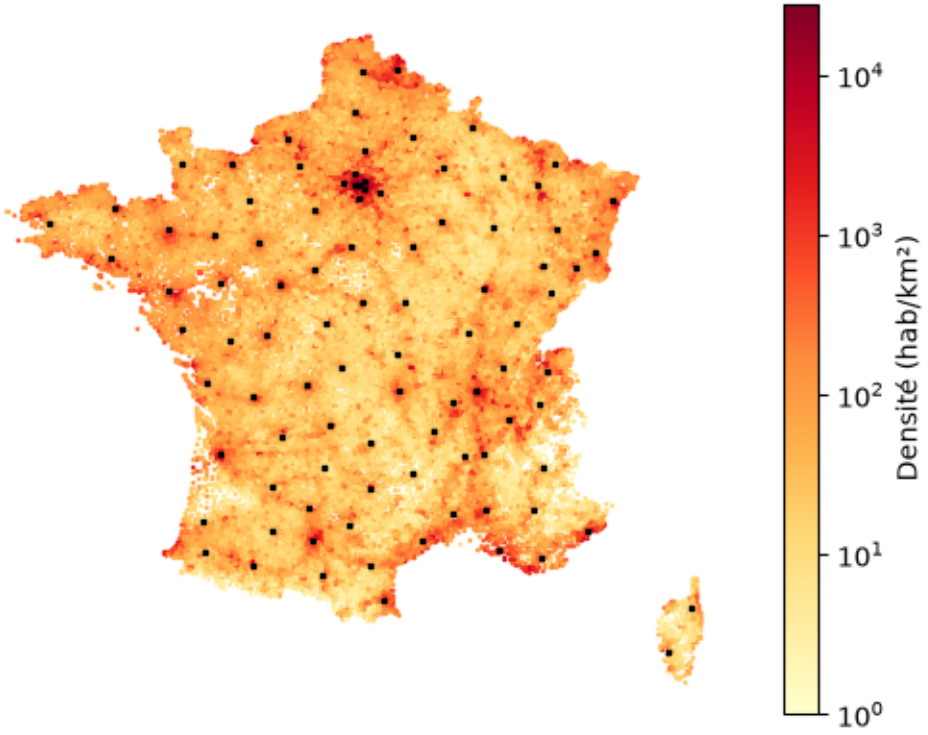


4. Récupérer dans une liste nommée `G` les centres de gravité démographiques (barycentres des centres des communes, pondérés par leur population) des départements de France métropolitaine (`dep_code` strictement inférieur à `'96'` : c'est une chaîne de caractères).

```
G=db.liste("""
SELECT SUM(longitude_centre*population)/SUM(population),
    SUM(latitude_centre*population)/SUM(population)
FROM communes
WHERE dep_code<'96'
GROUP BY dep_code
""")

xG=[t[0] for t in G]
yG=[t[1] for t in G]
```

Une fois `G` calculée, extraire les longitudes dans `xG` et les latitudes dans `yG`, puis exécuter la cellule suivante pour ajouter ces points à la carte.

```
ax.scatter(xG, yG, marker="s", s=0.75,c='k')
fig
```


10^4
10^3
10^2
10^1
10^0
Densité (hab/km²)