

TD04 : Communes et départements

Fusionner, contrôler et interpréter

Anderson VIL | L3 Économie | 90 minutes

Analyse des données | CY Cergy Paris Université

Année universitaire 2026–2027

La question économique du TD04

La question. Quelle part de la population de son département chaque commune représente-t-elle ? Pour la calculer, il faut **rapprocher deux tableaux** — c'est l'objet de ce TD.

Pour chaque ligne, il faut connaître **la population locale** et **la population départementale**. Ces deux informations viennent de feuilles différentes.

$$\text{part (\%)} = 100 \times \frac{\text{population municipale locale}}{\text{population municipale du département}}$$

Observation, variable et unité statistique

Mot	Dans ce TD
Observation	Une ligne du tableau chargé
Variable	Une colonne : population, nom, code. . .
Unité statistique	L'objet décrit : commune ou arrondissement municipal
Identifiant	Un code qui distingue cet objet
Clé de fusion	La colonne commune aux deux tableaux

Deux niveaux. Une ligne de **Départements** décrit un département. Une ligne de **Communes** décrit généralement une commune. Le cas des arrondissements municipaux reviendra en Q10.

Un identifiant peut ressembler à un nombre

"01"

Code de l'Ain : le zéro fait partie du code.

"2A" et "2B"

Codes des deux départements corses.

Un code sert à reconnaître une entité.

Convertir "01" en entier donne 1. Il faut préserver les identifiants comme du texte et employer le même codage des deux côtés.

Un mini-exemple dans Python

```
import pandas as pd
g = pd.DataFrame({
    "commune": ["A", "B", "C", "D"],
    "dep": ["01", "01", "02", "99"],
    "pop": [200, 300, 400, 100]})
d = pd.DataFrame({
    "dep": ["01", "02", "03"],
    "nom": ["Nord", "Sud", "Est"],
    "pop_dep": [1000, 2000, 500]})
display(g)
display(d)
```

`pd.DataFrame` construit un tableau. Chaque liste fournit les valeurs d'une colonne. Ici, les codes sont entre guillemets.

Résultat du code python : les deux tableaux avant fusion

Gauche : *g*

Commune	dep	pop
A	01	200
B	01	300
C	02	400
D	99	100

Droite : *d*

dep	Nom	pop_dep
01	Nord	1000
02	Sud	2000
03	Est	500

Plusieurs-à-un : A et B partagent le code 01. À droite, chaque code apparaît une seule fois. 99 manque à droite et 03 n'a aucune commune à gauche.

Fusionner et empiler : deux opérations

Fusionner

Rapprocher les lignes sur une clé et ajouter des informations.

Exemple : donner à A la population de Nord.

Empiler

Ajouter sous un tableau d'autres lignes de même nature.

Exemple : communes d'une année puis d'une autre.

```
g.merge(d, on="dep", how="left")  
# Pour empiler des tableaux de meme structure :  
pd.concat([tableau_1, tableau_2], ignore_index=True)
```

La fusion du TD conserve une unité locale à gauche. Empiler communes et départements mélangerait deux niveaux dans les mêmes lignes.

Fusion interne : prédire les lignes conservées

```
interne = g.merge(d, on="dep", how="inner")
```

Pause recherche : Quelles communes trouvent une correspondance ? Combien de lignes restera-t-il ? Que devient D ?

`on="dep"` compare les valeurs de la colonne `dep`. `how="inner"` conserve seulement les correspondances.

Fusion interne : trois lignes après fusion

Commune	dep	pop	nom	pop_dep
A	01	200	Nord	1000
B	01	300	Nord	1000
C	02	400	Sud	2000

4 lignes avant, 3 après. D a disparu, car 99 n'a aucun partenaire. Le département 03 ne figure pas dans la sortie.

La disparition de D ne déclenche aucune erreur. Compter les lignes signale la perte, puis l'inspection des clés l'explique.

Fusion à gauche : D reste dans le tableau

```
gauche = g.merge(d, on="dep", how="left")
```

Commune	dep	pop	nom	pop_dep
A	01	200	Nord	1000
B	01	300	Nord	1000
C	02	400	Sud	2000
D	99	100	NaN	NaN

4 lignes avant, 4 après, dans cet exemple. La population de D reste connue. Seules les informations attendues de droite manquent. NaN ne signifie pas zéro.

L'indicateur rend la correspondance visible

```
gauche = g.merge(d, on="dep", how="left", indicator=True)
```

Commune	dep	nom	_merge
A	01	Nord	both
B	01	Nord	both
C	02	Sud	both
D	99	manquant	left_only

`both` : une ligne trouve une clé des deux côtés.

`left_only` : une ligne de gauche ne trouve aucune clé à droite.

Une clé répétée à droite : prédire l'effet

dep	nom	pop_dep
01	Nord	1000
02	Sud	2000
03	Est	500
01	Nord	1000

Droite anormale : le code 01 apparaît deux fois.

A trouve deux partenaires. B aussi.

```
d_double = pd.concat([d, d.iloc[[0]]], ignore_index=True)
bizarre = g.merge(d_double, on="dep", how="left")
```

Pause recherche : Combien de lignes après cette fusion ? Quelles communes se répètent ?

Une fusion à gauche peut augmenter l'effectif

Commune	dep	pop	nom	pop_dep
A	01	200	Nord	1000
A	01	200	Nord	1000
B	01	300	Nord	1000
B	01	300	Nord	1000
C	02	400	Sud	2000
D	99	100	NaN	NaN

4 lignes avant, 6 après. La fusion produit une ligne par partenaire. Additionner ensuite la population compterait A et B deux fois.

Contrôle. La conservation des lignes de gauche n'implique pas la conservation du nombre de lignes.

La relation plusieurs-à-un devient un contrôle

```
g.merge(d_double, on="dep", how="left", validate="m:1")
```

Sortie exécutée : MergeError

Merge keys are not unique in right dataset

La clé 01 est répétée à droite.

m autorise plusieurs lignes à gauche pour la même clé.

1 exige une seule ligne à droite par clé.

Contrôle. Vérifier l'unicité à droite avant la fusion : `d["dep"].is_unique`. Ici, avec `d_double`, le résultat est `False`.

Une part dépend de son dénominateur

Commune	Population locale	Population département	Part
A	200	1000	20 %
B	300	1000	30 %
C	400	2000	20 %

$$100 \times \frac{300}{1000} = 30 \% \quad 100 \times \frac{400}{2000} = 20 \%$$

Pause recherche : B est moins peuplée que C. Pourquoi sa part est-elle pourtant plus élevée ?

La population du département de B est plus petite. Le rapport mesure le poids relatif de la commune, pas sa seule taille.

Ressources et prise en main

TD04 : Communes et départements

La page du cours : énoncé et notebook

TD 4

Communes et départements

énoncé

notebook

[Page du cours](#) **Énoncé TD04**

Ouvrir le bloc **TD 4 — Communes et départements**. Garder l'énoncé à côté du notebook.

GitHub : la cellule de chargement

```
# TODO : completez le parametre qui saute les lignes de titre  
com = pd.read_excel(URL, sheet_name="Communes", header=_____)  
dep = pd.read_excel(URL, sheet_name="Départements", header=_____)
```

Zone à compléter : les deux paramètres header. Remplacer les blancs par la ligne de l'en-tête, puis exécuter la cellule.

Le notebook : téléchargement et cellules à compléter

[Notebook étudiant sur GitHub](#)

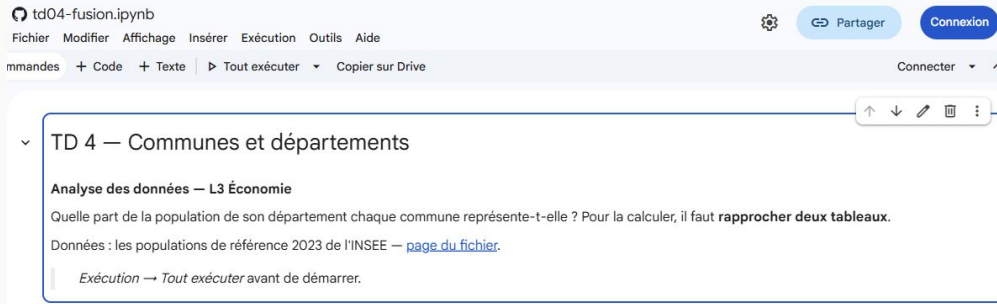
[Télécharger le fichier .ipynb](#)

Les blancs _____ se remplacent par les paramètres demandés. La prévisualisation GitHub affiche les cellules sans les exécuter.

Une consigne à adapter au notebook incomplet

« Tout exécuter avant de démarrer » n'est pas recommandé. car le notebook contient des blancs. Compléter et tester d'abord les cellules une par une.

Colab : ouvrir et conserver une copie personnelle



Ouverture directe du TD04 dans Colab

Se connecter, puis choisir **Copier sur Drive** ou **Fichier / Enregistrer une copie dans Drive**.

Importer, exécuter une cellule, lire la sortie

Depuis le fichier téléchargé

1. Ouvrir **Colab** et se connecter.
2. Choisir **Fichier / Importer le notebook**, onglet **Importer**.
3. Sélectionner `td04-fusion.ipynb`.
4. Enregistrer une copie personnelle.

Pendant le TD

1. Lire la question.
2. Remplir les blancs.
3. Cliquer sur le triangle à gauche de la cellule, ou Maj+Entrée.
4. Lire le tableau ou le nombre sous la cellule.

Contrôle. À la fin, redémarrer la session puis choisir **Exécution / Tout exécuter**. Corriger la première erreur avant de poursuivre.

La source INSEE et le fichier Excel

France hors Mayotte		
(xlsx, 2 Mo) 	(csv, 1008 Ko) 	(pdf, 12 Mo) 

2023 : référence statistique. **2025** : limites territoriales.
2026 : entrée en vigueur au 1er janvier.

Page INSEE et documentation **Classeur ensemble.xlsx**

Regarder le **bouton xlsx à gauche**. Le champ affiché est **France hors Mayotte**.

Partie 1 — Charger : Q1–Q2

TD04 : Communes et départements

Q1–Q2. Chargement

1. Charger

Les deux feuilles s'appellent `Communes` et `Départements`. Ouvrez d'abord le fichier dans un tableur pour voir à quoi il ressemble.

Question 1. Chargez les deux feuilles avec `header=7`. Pourquoi faut-il ce paramètre ?

Question 2. Combien de lignes contient chaque feuille ? De quoi une ligne est-elle la description, dans l'une et dans l'autre ?

Pause recherche : Dans quelle ligne du classeur les vrais noms de colonnes commencent-ils ?
Combien de lignes et de colonnes voyez-vous après chargement ?

Q1. Repérer les noms des variables

Ligne	C	F	G	H	J
1	Populations de référence des communes en vigueur au 1er janvier 2026				
2	Mise à jour : décembre 2025				
3	en habitant				
4	Champ : France métropolitaine, départements d'outre-mer de la Guadeloupe, de la Guyane, de la Martinique et de La Réunion, limites territoriales en vigueur au 1er janvier 2025				
5	Date de référence statistique : 1er janvier 2023				
6	Source : Insee, Recensement de la population 2023				
7					
8	Code département	Code commune	Nom de la commune	Population municipale	Population totale
9	01	001	L'Abergement-Clémenciat	860	876
10	01	002	L'Abergement-de-Varey	270	276
11	01	004	Ambérieu-en-Bugey	15934	16339

Q1. Huitième ligne du classeur, index 7 pour pandas

La ligne **8 du classeur** contient les noms des variables. pandas compte à partir de zéro : ligne 1 = index 0, donc ligne 8 = **index 7**.

Ligne du classeur	Index pandas
1	0
7 (vide)	6
8 (noms des variables)	7
9 (première observation)	8

`header=7` choisit la ligne des noms de colonnes. La première observation sera ensuite à l'index 0 du tableau chargé.

Q1. Lire les deux feuilles

```
import pandas as pd
import numpy as np

URL = ("https://www.insee.fr/fr/statistiques/fichier/"
      "8680726/ensemble.xlsx")
com = pd.read_excel(URL, sheet_name="Communes", header=7)
dep = pd.read_excel(URL, sheet_name="Départements", header=7)
```

Q1. Les paramètres de lecture

Paramètre	Rôle
URL	Adresse du classeur à lire
sheet_name	Nom exact de la feuille
header=7	Ligne des noms de colonnes, numérotation dès 0

`com` et `dep` mémorisent les deux tableaux. Sans le bon en-tête, le titre devient un nom de colonne et d'autres colonnes s'appellent `Unnamed`.

Q2. Taille des tableaux et unité des lignes

```
print(com.shape, dep.shape)
print(len(com), len(dep))
```

SORTIE RÉELLE avant de créer CODGEO

(34904, 10) (100, 9)

34904 100

shape donne (**nombre de lignes, nombre de colonnes**). **len** donne le nombre de lignes.

Communes : 34 904 unités locales, dont les arrondissements municipaux de Paris, Marseille et Lyon.

Départements : 100 départements.

Q2. Les premières communes

```
com[["Code département", "Nom de la commune",  
     "Population municipale"]].head(3)
```

Département	Nom de la commune	Pop. municipale
01	L' Abergement-Clémenciat	860
01	L' Abergement-de-Varey	270
01	Ambérieu-en-Bugey	15934

Q2. Les premiers départements

```
dep[["Code département", "Nom du département",  
     "Population municipale"]].head(3)
```

Département	Nom du département	Pop. municipale
01	Ain	679344
02	Aisne	523342
03	Allier	333298

Le champ et les deux populations

Population municipale

Personnes ayant leur résidence habituelle dans la commune. C'est la mesure utilisée au numérateur et au dénominateur.

Population totale

Population municipale + population comptée à part. Exemple : $860 + 16 = 876$. Elle peut inclure des doubles comptes entre communes.

Le champ général du classeur est la métropole et les quatre DOM hors Mayotte. La feuille Communes inclut aussi **quatre lignes de collectivités d'outre-mer**, que la fusion révélera.

Source : Classeur INSEE, lignes de métadonnées et observations. [Définitions INSEE](#), onglet Documentation.

Partie 2 — La clé : Q3–Q5

TD04 : Communes et départements

Q3–Q5. La clé

2. La clé

Question 3. Quel est le type de `Code département` ? Et de `Code commune` ? Que vaut le code commune de L'Abergement-Clémenciat ?

Question 4. L'INSEE identifie une commune par le code département suivi d'un code commune **sur trois chiffres**. Construisez cet identifiant dans une colonne `CODGE0`. Vérifiez : L'Abergement-Clémenciat doit valoir 01001.

Question 5. Tous vos `CODGE0` font-ils cinq caractères ? Comptez-les par longueur. Lesquels font exception, et pourquoi ?

Pause recherche : Quel type pandas donne-t-il à chacun des deux codes ? Quelle valeur lit-il pour la première commune

Q3. Types stockés et sens des codes

```
print(com[["Code département", "Code commune"]].dtypes)
com[["Code département", "Code commune",
     "Nom de la commune"]].head(3)
```

Variable	Sortie exécutée	Exemple
Code département	str	"01", "2A", "971"
Code commune	int64	1, 2, 4

Q3. Le code de L'Abergement-Clémenciat

Code département	Code commune	Nom de la commune
01	1	L'Abergement-Clémenciat
01	2	L'Abergement-de-Varey
01	3	Ambérieu-en-Bugey

Code commune est un entier : L'Abergement-Clémenciat porte le code 1, et non 001. Les zéros de tête ont disparu.

Q4. Construire un code : texte et zéros initiaux

```
nombres = com["Code commune"]  
texte = nombres.astype(str)  
trois_chiffres = texte.str.zfill(3)
```

Operation	Première ligne
Code commune	1 (entier)
astype(str)	"1" (texte)
str.zfill(3)	"001" (trois caractères)

`astype(str)` transforme les valeurs en texte. `str` donne accès aux opérations sur le texte.
`zfill(3)` ajoute des zéros à gauche jusqu'à trois caractères.

Q4. Concaténer et vérifier 01001

```
com["CODGEO"] = (com["Code département"]  
                 + com["Code commune"].astype(str).str.zfill(3))  
  
com.loc[com["Nom de la commune"] == "L' Abergement-Clémenciat",  
        "CODGEO"]
```

SORTIE RÉELLE sortie exécutée, première observation

0 01001

Name: CODGEO, dtype: str

Le signe + colle deux textes : "01" + "001" donne "01001". `loc[condition, colonne]` garde les lignes qui satisfont la condition puis affiche la colonne demandée.

Q4. Deux codes avec deux fonctions différentes

CODGEO

Identifie une unité locale.

Exemples : 01001, 2A004.

Deux communes d'un département ont deux CODGEO différents.

Code département

Sert de clé de fusion du TD.

Exemples : 01, 2A.

Plusieurs communes partagent cette valeur.

On construit **CODGEO** pour apprendre à traiter les identifiants. La fusion communes-départements utilise **Code département**.

Q5. Compter les longueurs avant d'expliquer

```
longueurs = com["CODGEO"].str.len()  
print(longueurs.value_counts())
```

SORTIE RÉELLE sortie exécutée

Longueur	Effectif
5	34 788
6	116

Pause recherche : Quelles lignes ont six caractères ? Quel code département ont-elles ? Pourquoi la construction précédente échoue-t-elle ?

Q5. Filtrer les codes de six caractères

```
six = com["CODGEO"].str.len() == 6  
com.loc[six, ["Code département", "Code commune", "CODGEO"]].head(3)  
com.loc[six, "Code département"].value_counts().sort_index()
```

Code département	Nombre de lignes
971	32
972	34
973	22
974	24
975	2
977	1
978	1

Q5. Les exceptions ultramarines

Cas	Construction provisoire	Longueur
Métropole	$01 + 001 = 01001$	5
Corse	$2A + 004 = 2A004$	5
Miquelon-Langlade	$975 + 501 = 975501$	6

Les sept codes ultramarins observés ont trois caractères. Ajouter encore trois caractères produit six caractères. Le code officiel de Miquelon-Langlade est 97501.

La correction sera traitée en **Q13 facultative**. L'erreur de CODGEO n'empêche pas la fusion du TD, puisque la clé utilisée est **Code département**.

Partie 3 — Fusionner : Q6–Q8

TD04 : Communes et départements

Q6–Q8. Fusionnement

3. Fusionner

Question 6. Fusionnez les communes avec les départements, sur Code département, avec `how="left"` et `indicator=True`. Combien de lignes avant, combien après ? Pourquoi est-ce le même nombre ?

Question 7. Combien de communes ne trouvent pas leur département ? Lesquelles ?

Question 8. Cherchez ce que désignent leurs codes. Pourquoi ces communes n'ont-elles pas de département ?

Pause recherche : Quel tableau doit rester à gauche ? Quelles colonnes faut-il prendre à droite ? Quelle clé partagent les deux feuilles ?

Q6. Sélectionner les informations de droite

```
colonnes_dep = ["Code département",  
                "Nom du département",  
                "Population municipale"]  
droite = dep[colonnes_dep]
```

Colonne retenue	Utilité
Code département	Clé commune aux deux tableaux
Nom du département	Interpréter la correspondance
Population municipale	Dénominateur de la part

`dep[...]` passe une **liste de colonnes** à pandas. Cette sélection ajoute uniquement les informations nécessaires.

Q6. Le code de fusion, paramètre par paramètre

```
f = com.merge(  
    dep[["Code département", "Nom du département",  
        "Population municipale"]],  
    on="Code département",  
    how="left",  
    indicator=True,  
    suffixes=("_com", "_dep"))
```

`com` : tableau de gauche. L'argument de `merge` : tableau de droite.
`on` : colonne comparée. `how="left"` : conserver les observations de gauche.
`indicator=True` : ajouter le diagnostic `_merge`.

Q6. Pourquoi deux suffixes ?

Avant la fusion	Après la fusion
com : Population municipale	Population municipale_com
dep : Population municipale	Population municipale_dep

Les deux colonnes ont le même nom. `suffixes=("_com", "_dep")` précise leur origine : le premier suffixe appartient à gauche, le second à droite.

Q6. Deux populations, deux niveaux

Commune	Pop. municipale_com	Pop. municipale_dep
L'Abergement-Clémenciat	860	679344
L'Abergement-de-Varey	270	679344
Ambérieu-en-Bugey	15934	679344

Le dénominateur départemental se répète sur toutes ses communes. Il faut le lire pour chaque ligne, pas additionner cette colonne fusionnée.

Q6. Même effectif : une propriété à vérifier

```
print("lignes avant :", len(com), "| apres :", len(f))  
print(dep["Code département"].is_unique)
```

SORTIE RÉELLE sorties exécutées

lignes avant : 34904 | apres : 34904
True

La fusion à gauche conserve chaque ligne locale. L'unicité de la clé à droite lui donne **au plus un partenaire**, donc aucune multiplication de lignes.

Contrôle. Le même effectif ne prouve pas que toutes les lignes sont appariées. Il faut lire `_merge`.

Q7. Lire l'indicateur avant de filtrer

```
print(f["_merge"].value_counts())
```

SORTIE RÉELLE sortie exécutée

both	34 900
left_only	4
right_only	0

Pause recherche : Comment afficher les quatre lignes non appariées ? Quel opérateur compare la valeur de chaque ligne à left_only ?

Contrôle. $34\,900 + 4 = 34\,904$. La catégorie right_only a un effectif nul car la fusion est à gauche.

Q7. Le filtre des observations non appariées

```
sans_dep = f["_merge"] == "left_only"  
f.loc[sans_dep, ["Code département", "Code commune",  
                "Nom de la commune"]]
```

`sans_dep` est une condition ligne par ligne. `loc` filtre les lignes puis sélectionne les colonnes.

Q7. Les quatre lignes non appariées

```
_merge
both          34900
left_only      4
right_only     0
Name: count, dtype: int64
```

	Code département	Code commune	Nom de la commune
34900	975	501	Miquelon-Langlade
34901	975	502	Saint-Pierre
34902	977	701	Saint-Barthélemy
34903	978	801	Saint-Martin

Q8. Trois collectivités, quatre communes

Code du fichier	Collectivité	Commune(s)
975	Saint-Pierre-et-Miquelon	Miquelon-Langlade, Saint-Pierre
977	Saint-Barthélemy	Saint-Barthélemy
978	Saint-Martin	Saint-Martin

Ces collectivités d'outre-mer ont un statut distinct de celui d'un département. Les codes existent dans le référentiel des collectivités d'outre-mer, tandis que la feuille Départements ne contient aucun partenaire.

Le nom **Code département** est celui de la colonne Excel. Pour ces lignes, il représente le code de la **collectivité**.

Source : [INSEE, COG 2025](#), tables v_comer_2025 et v_commune_comer_2025. Codes, noms et rattachements contrôlés.

Q8. Une absence de correspondance peut informer

Interprétation. Ici, les quatre absences correspondent à des collectivités d'outre-mer présentes dans la feuille Communes et absentes de la feuille Départements.

Conséquence pour la part. La population départementale manque. La part reste NaN pour ces quatre lignes. Remplacer le dénominateur manquant par zéro ou inventer un département serait incorrect.

```
print(f.loc[sans_dep, "Population municipale_dep"].isna().all())
```

Contrôle. Sortie exécutée : True. Une fusion inner aurait réduit le tableau à 34 900 lignes.

Partie 4 — Interpréter : Q9–Q10

TD04 : Communes et départements

Q9–Q10. Interprétation

4. Interpréter

Question 9. Calculez la part de chaque commune dans la population municipale de son département. Quelle commune pèse le plus lourd ? Le résultat vous surprend-il ?

Question 10. Ni Paris, ni Marseille, ni Lyon n'apparaissent en tête. Cherchez-les : pourquoi ?

Pause recherche : Quelles colonnes vont au numérateur et au dénominateur ? Pourquoi une grande ville pourrait-elle être absente du haut du classement ?

Q9. Calcul de la part ligne par ligne

```
population_locale = f["Population municipale_com"]  
population_departement = f["Population municipale_dep"]  
f["part"] = 100 * population_locale / population_departement
```

$$\text{part (\%)} = 100 \times \frac{\text{population municipale de l'unité locale}}{\text{population municipale du département}}$$

Chaque division utilise les deux populations **de la même ligne**. La colonne **part** s'exprime en pourcentage. Pour les quatre lignes sans partenaire, elle vaut NaN.

Contrôle. Quatre parts manquantes, aucun dénominateur apparié nul et des parts entre 0 et 100.

Q9. Trier : trois opérations successives

```
classement = f.sort_values("part", ascending=False)
colonnes = ["Nom de la commune", "Nom du département", "part"]
classement[colonnes].head(5)
```

Opération	Lecture
sort_values("part")	Réordonner les lignes selon la part
ascending=False	Du plus grand au plus petit
[colonnes]	Garder les trois colonnes à lire
head(5)	Afficher les cinq premières lignes

Les valeurs manquantes vont en fin de classement par défaut. Le classement porte ici sur les unités de la feuille, pas uniquement sur des villes entières.

Q9. Ajaccio en tête du classement des lignes

	Nom de la commune	Nom du département	part
6579	Ajaccio	Corse-du-Sud	45.345977
11789	Toulouse	Haute-Garonne	34.986762
33144	Limoges	Haute-Vienne	34.820067
34198	Belfort	Territoire de Belfort	32.734662
1953	Nice	Alpes-Maritimes	31.702525

Lecture à l'arrondi : Ajaccio 45,3 %, Toulouse 35,0 %, Limoges 34,8 %, Belfort 32,7 %, Nice 31,7 %.

Le classement compare des parts dans des départements différents. Le premier rang ne signifie pas « ville la plus peuplée de France ».

Q9. Le dénominateur explique le résultat d'Ajaccio

Commune	Population locale	Population département	Part
Ajaccio	76320	168306	45,35 %
Toulouse	514819	1471468	34,99 %

$$100 \times \frac{76\,320}{168\,306} = 45,346 \%$$

Ajaccio représente près de la moitié de la population municipale de la Corse-du-Sud. Toulouse compte bien davantage d'habitants, mais son département est beaucoup plus peuplé.

Contrôle. Lire et vérifier les deux populations, au même millésime, avant d'interpréter une part.

Q10. Rechercher Paris, Marseille et Lyon

```
for ville in ["Paris", "Marseille", "Lyon"]:  
    trouve = f["Nom de la commune"].str.contains(  
        ville, regex=False, na=False)  
    print(ville, trouve.sum())  
    print(f.loc[trouve, ["Nom de la commune", "part"]].head(3))
```

`str.contains` teste la présence d'un texte à l'intérieur du nom. `regex=False` traite le texte littéralement. `na=False` classe un nom manquant comme non trouvé.

Pause recherche : Quels noms apparaissent ? Décrivent-ils la ville entière ou seulement une partie ?

Q10. Le fichier présente des arrondissements municipaux

Ville	Lignes d'arrondissements	Exemples de codes
Paris	20	75101 à 75120
Marseille	16	13201 à 13216
Lyon	9	69381 à 69389

Paris, Marseille et Lyon sont des communes. **La ville entière n'est pas représentée par une seule ligne dans cette feuille.** Chacun de ses arrondissements municipaux possède sa propre ligne.

On trouve par exemple Marseille 1er Arrondissement, et aucun nom exactement égal à Marseille. Une recherche par égalité du nom de la ville renvoie donc zéro ligne.

Source : Feuille Communes et [COG 2025](#), TYPECOM=ARM, commune parente dans COMPARENT.

Q10. Une ligne d'arrondissement pèse une fraction de la ville

Unité de la feuille	Population	Département	Part
Marseille 1er	37599	Bouches-du-Rhône	1,80 %
Marseille 13e	95112	Bouches-du-Rhône	4,56 %
Marseille 16e	15854	Bouches-du-Rhône	0,76 %

Ces lignes ne mesurent pas la population de Marseille entière. Dans cette feuille, les parts des seize arrondissements de Marseille vont de **0,76 % à 4,56 %**.

Contrôle. Définir l'unité statistique avant le classement. Une comparaison de villes exige que toutes les lignes représentent des villes entières.

Q10. Retrouver les villes entières : principe de l'agrégation

1. Identifier uniquement les arrondissements de la ville.
2. Additionner leurs populations municipales.
3. Diviser ce total par **une seule** population départementale.

Ville entière	Arrondissements	Population	Part du département
Marseille	16	886040	42,44 %
Lyon	9	519127	27,11 %
Paris	20	2103778	100,00 %

Paris correspond ici à la totalité de la population municipale du département de Paris. L'agrégation modifie donc le classement par rapport aux seules lignes du fichier.

Source : Calculs exécutés en regroupant les lignes ARM selon COMPARENT, COG 2025. Code détaillé en annexe.

Interprétation

« Dans le fichier INSEE des populations de référence 2023, en géographie au 1er janvier 2025, Ajaccio compte 76 320 habitants en population municipale, contre 168 306 pour la Corse-du-Sud. Sa part est donc 45,35 %. Ce rang compare les unités présentes dans la feuille, qui incluent des arrondissements municipaux pour trois grandes villes. »

Contrôle. La phrase indique la mesure, le millésime, le numérateur, le dénominateur et l'unité de la ligne.

Synthèse du parcours principal

TD04 : Communes et départements

Les erreurs fréquentes et leurs symptômes

Erreur	Symptôme	Contrôle
header omis	Noms de colonnes Unnamed	Lire les premières lignes
Zéros initiaux perdus	011 au lieu de 01001	Types, zfill et référence
Clé répétée à droite	Plus de lignes après fusion	is_unique et validate
Fusion inner non contrôlée	4 lignes supprimées	Compter, lire _merge
NaN remplacé par zéro	Part sans sens	Inspecter les non-appariées
Unités locales confondues	Rang de ville mal interprété	Lire les noms et le COG

Une opération peut produire une sortie sans erreur et répondre à une autre question que celle du TD.

Les trois contrôles essentiels

```
print(len(com), len(f))
print(f["_merge"].value_counts())
f.loc[f["_merge"] == "left_only",
      ["Code département", "Nom de la commune"]]
print(dep["Code département"].is_unique)
```

Effectifs : 34 904 avant et après.

Appariement : 34 900 both, 4 left_only.

Inspection : quatre communes de collectivités d'outre-mer.

Unicité à droite : True.

Vérification finale du notebook étudiant

Avant « Tout exécuter »

Compléter tous les blancs du parcours principal, tester les cellules et écrire les réponses Q1–Q10.

À la fin

Redémarrer la session puis choisir **Exécution** / **Tout exécuter**. Lire les sorties du haut vers le bas. Enregistrer la copie personnelle.

Q11–Q13 sont facultatives. Sans Q13, les 116 codes provisoires à six caractères restent visibles, mais la fusion sur **Code département** fonctionne. Après Q13, vérifier aussi les codes contre le COG.

Approfondissements facultatifs : Q11–Q13

TD04 : Communes et départements

Q11–Q13. Approfondissements

5. Pour ceux qui ont terminé

Question 11. [facultatif] Cherchez « Marseille » avec `str.contains`. Qu'obtenez-vous, en plus des arrondissements ? Que conclure sur la recherche par nom ?

Question 12. [facultatif] Ajoutez `validate="m:1"` à votre fusion. Que vérifie ce paramètre ? Que se passerait-il si la feuille des départements contenait deux fois le même code ?

Question 13. [facultatif] Corrigez la construction de `CODGEO` pour l'outre-mer, et vérifiez que tous les codes font désormais cinq caractères.

Q11. Chercher les noms contenant Marseille

FACULTATIF

```
marseille = f["Nom de la commune"].str.contains(  
    "Marseille", regex=False, na=False)  
f.loc[marseille, ["Nom de la commune", "Code département", "part"]]
```

regex=False cherche le texte littéral. na=False écarte les noms manquants. Prédire les lignes sélectionnées avant d'exécuter.

Q11. Une recherche par sous-chaîne produit des faux positifs

FACULTATIF

Résultat	Code département
16 arrondissements de Marseille	13
Marseillette	11
Marseilles-lès-Aubigny	18
Marseille-en-Beauvaisis	60

19 lignes au total. contains reconnaît une suite de lettres, pas une appartenance à la ville.
Marseillette est une commune de l'Aude.

Source : Calculs exécutés sur le classeur INSEE, populations de référence 2023, le 06/10/2026.

Q11. Une sélection explicite des arrondissements

FACULTATIF

```
nom = f["Nom de la commune"]
marseille_arm = nom.str.fullmatch(
    r"Marseille \d+(er|e) Arrondissement", na=False)
print(marseille_arm.sum())
```

SORTIE RÉELLE sortie exécutée

16

`fullmatch` exige que **tout le nom** suive le motif. `\d+` représente des chiffres et `(er|e)` les terminaisons observées. Cette expression dépend de la convention des noms du fichier.

Q12. Ajouter validate à la fusion du TD

FACULTATIF

```
f_sure = com.merge(  
    dep[["Code département", "Nom du département",  
        "Population municipale"]],  
    on="Code département", how="left",  
    indicator=True, suffixes=("_com", "_dep"),  
    validate="m:1")  
print(len(f_sure))
```

SORTIE RÉELLE sortie exécutée

34904

m:1 vérifie l'unicité de la clé de droite. Cela autorise plusieurs communes par département, mais un seul département par valeur de clé.

Q12. Une clé dupliquée provoque une erreur explicite

FACULTATIF

```
dep_double = pd.concat([dep, dep.iloc[[0]]], ignore_index=True)
com.merge(dep_double, on="Code département",
          how="left", validate="m:1")
```

Sortie exécutée : MergeError.

Le code 01 n'est plus unique à droite. La fusion s'arrête.

Q12. Inspecter la clé répétée

FACULTATIF

```
doublons = dep_double["Code département"].duplicated(keep=False)
dep_double.loc[doublons, ["Code département", "Nom du département"]]
```

Contrôle. Afficher les deux lignes de l'Ain, puis revenir à la source. Ne pas supprimer arbitrairement une ligne sans comprendre la duplication.

Code département	Nom du département
01	Ain
01	Ain

Source : Calculs exécutés sur le classeur INSEE, populations de référence 2023, le 06/10/2026.

Q13. Une règle particulière au classeur ultramarin

FACULTATIF

Unité	Code territoire	Code local brut	CODGEO officiel
Miquelon-Langlade	975	501	97501
Saint-Pierre	975	502	97502
Saint-Barthélemy	977	701	97701
Saint-Martin	978	801	97801

Dans ce classeur, le code local sur trois chiffres reprend déjà le dernier chiffre du code territoire. Miquelon-Langlade : $97 + 501 = 97501$, plutôt que $975 + 501$.

Cette règle dépend de la convention du fichier INSEE utilisé. Dans un autre fichier, il faut relire la structure des codes avant de la réutiliser.

Source : [COG 2025](#), table v_commune_comer_2025, et classeur INSEE.

Q13. Corriger avec une condition ligne par ligne

FACULTATIF

```
d = com["Code département"]
n = com["Code commune"].astype(str).str.zfill(3)

com["CODGEO"] = np.where(
    d.str.len() == 3,
    d.str[:2] + n,
    d + n)
print(com["CODGEO"].str.len().value_counts())
```

Q13. Lire la condition et contrôler le format

FACULTATIF

`np.where(condition, si vrai, si faux)` choisit le code à chaque ligne.
`d.str[:2]` garde les deux premiers caractères.

SORTIE RÉELLE sortie exécutée

5 34904

Contrôle. Tous les codes ont cinq caractères. Cela contrôle leur format, pas encore leur identité.

Q13. Comparer à une référence officielle

FACULTATIF

```
cog = pd.read_csv("donnees/v_commune_2025.csv", dtype=str)
comer = pd.read_csv("donnees/v_commune_comer_2025.csv", dtype=str)
refs = pd.concat([
    cog.loc[cog["TYPECOM"].isin(["COM", "ARM"]),
            ["COM", "DEP", "LIBELLE"]],
    comer.rename(columns={"COM_COMER": "COM", "COMER": "DEP"})
               [["COM", "DEP", "LIBELLE"]], ignore_index=True)
```

Le COG 2025 correspond à la géographie du classeur. `dtype=str` préserve les codes. COM et ARM désignent communes et arrondissements municipaux. L'autre table couvre les collectivités d'outre-mer.

Source : [Page officielle COG 2025 et téléchargement des CSV](#). Copies des CSV dans l'archive.

Q13. Format correct et identité vérifiée

FACULTATIF

```
audit = com.merge(refs, left_on="CODGEO", right_on="COM",  
                  how="left", indicator=True, validate="1:1")  
print(audit["_merge"].value_counts())  
print((audit["Code département"] != audit["DEP"]).sum())
```

SORTIE RÉELLE sorties exécutées

```
both 34904 ; left_only 0 ; right_only 0  
0 rattachement discordant.
```

Q13. Ce que prouve le référentiel officiel

FACULTATIF

Les 34 904 codes existent dans le référentiel 2025 et leur rattachement correspond. Le libellé officiel et l'unité sont disponibles pour inspecter les cas connus : 01001, 97501, etc.

Une longueur de cinq caractères est nécessaire. Un code à cinq caractères inventé, périmé ou associé à une mauvaise commune resterait possible sans référence.

Annexes de secours et sources

TD04 : Communes et départements

Annexe. Agréger avec la commune parente du COG

```
f_geo = f.merge(refs_arm, left_on="CODGEO", right_on="COM",
               how="left", validate="1:1")
arm = f_geo.loc[f_geo["TYPECOM"] == "ARM"]
villes = arm.groupby("COMPARENT").agg(
    population=("Population municipale_com", "sum"),
    departement=("Population municipale_dep", "first"))
villes["part"] = 100 * villes["population"] / villes["departement"]
```

`refs_arm` conserve COM, TYPECOM et COMPARENT du COG. Le notebook complémentaire crée ce tableau et refait `f` après la correction de CODGEO.

`sum` additionne les populations locales. `first` prend une seule population départementale, après avoir vérifié que tous les arrondissements d'une ville ont le même département.

Annexe. Sources statistiques et techniques

Source	Contenu
INSEE, populations 2023 ensemble.xlsx	Classeur, champ, définitions, dates Feuilles Communes et Départements
INSEE, COG 2025	Codes, types d'unités et communes parentes
Documentation pandas merge	how, indicator, suffixes et validate