

## ▼ PROJET 5 -DATA ANALYST -

# Créez et utilisez une base de données immobilière avec SQL

**Remarque :**

Pour des raisons de lisibilité des parties ont été masquées. Elles peuvent être démasquées directement sur le Notebook .ipynb

Remarque :

Dans ma fonction `Description_Fichier_Tableau` : Je pourrai l'améliorer pour avoir une colonne avec les valeurs du type `' '` ou `' '`. Il faudrait que je regarde les expressions régulières pour identifier le nombre de valeur de ce type avec la bonne regex par colonne, et afficher cette valeur.

```
import re
```

- RECHERCHE pour une fonction avec les valeurs blanches

[ ] ↳ 11 cellules masquées

## ✓ I. IMPORTATION DES MODULES UTILES

```
#Importation de la librairie Numpy
import numpy as np
```

```
#Importation de la librairie Pandas
import pandas as pd
```

## ✓ II. IMPORTATION DES FICHIERS

```
donnees_communes = pd.read_excel('donnees_communes.xlsx')
```

```
referentiel_geographique = pd.read_excel('fr-esr-referentiel-geographique.xlsx')
```

```
valeurs_foncières = pd.read_excel('Valeurs-foncières.xlsx')
```

### ✓ III. ETUDE DES FICHIERS

```
def Description_Fichier(FICHIER):
```

```
#Consulter le nombre de colonnes
print("Le tableau comporte ",FICHIER.shape[1]," colonne(s) : ")
```

```
for elt in FICHIER.columns :
    print(elt)
```

```
print("\n")
```

```
#le type de données et Le nombre de valeurs présentes dans chacune des colonnes
for elt in FICHIER.columns :
    print("La colonne ",elt," (est de type ",FICHIER[elt].dtypes," ) et contient :")
    print("\t", len(FICHIER[elt])," lignes")
    print("\t", FICHIER[elt].notna().sum()," valeur(s) non-vide(s)")
    print("\t", FICHIER[elt].isna().sum()," valeur(s) vide(s)")
    print ("\t", len(FICHIER[elt].value_counts()), " valeur(s) distincte(s)." )
    print("\n")
```

```
def Description_Fichier_Tableau(FICHIER):
```

[illegible]

```
#remplissage des lignes
for elt in FICHIER.columns :
    Tab_Descriptif.loc[elt,:] = [FICHIER[elt].dtypes, len(FICHIER[elt]) ,
                                FICHIER[elt].notna().sum() ,FICHIER[elt].isna().sum(),
                                len(FICHIER[elt].unique()) ]

#print(Tab_Descriptif)
return Tab_Descriptif
```

## ➤ RECHERCHE pour une fonction avec les valeurs blanches

[ ] 1, 6 cellules masquées

## ✓ RECHERCHE pour une fonction avec les valeurs blanches

```
def Description_Fichier_Tableau_avc_ValBlanches(FICHIER):
    re2 = r"\A[\s]*$"

    #création d'un DataFrame qui va contenir les colonnes du Fichier en Index,
    #et la description des lignes dans les colonnes
    Tab_Descriptif = pd.DataFrame(index = FICHIER.columns,
                                  columns=['Type', 'Nb lignes',
                                           'Valeurs non-vides', 'Valeurs vides',
                                           'Valeurs distinctes', 'Valeurs blanches'])

    #remplissage des lignes
    for elt in FICHIER.columns :
        col_string = (FICHIER[elt]).astype('string')
        Tab_Descriptif.loc[elt,:] = [FICHIER[elt].dtypes, len(FICHIER[elt]) ,
                                    FICHIER[elt].notna().sum() ,FICHIER[elt].isna().sum(),
                                    len(FICHIER[elt].unique()), col_string.str.match(re2).sum() ]

    #print(Tab_Descriptif)
    return Tab_Descriptif
```

## ✓ A. FICHIER donnees communes

donnees\_communes.head()

	CODREG	CODDEP	CODARR	CODCAN	CODCOM	COM	PMUN	PCAP	PTOT
0	84	01	2.0	08	1	L'Abergement-Clémenciat	779	19	798
1	84	01	1.0	01	2	L'Abergement-de-Varey	256	1	257
2	84	01	1.0	01	4	Ambérieu-en-Bugey	14134	380	14514
3	84	01	2.0	22	5	Ambérieux-en-Dombes	1751	25	1776
4	84	01	1.0	04	6	Ambléon	112	6	118

donnees\_communes.shape

(34991, 9)

donnees\_communes.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34991 entries, 0 to 34990
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0   CODREG      34991 non-null  int64
1   CODDEP      34991 non-null  object
2   CODARR      34990 non-null  float64
3   CODCAN      34990 non-null  object
4   CODCOM      34991 non-null  int64
5   COM         34991 non-null  object
6   PMUN        34991 non-null  int64
7   PCAP        34991 non-null  int64
8   PTOT        34991 non-null  int64
dtypes: float64(1), int64(5), object(3)
memory usage: 2.4+ MB
```

Description\_Fichier\_Tableau(donnees\_communes).T



	CODREG	CODDEP	CODARR	CODCAN	CODCOM	COM	PMUN	PCAP	PTOT
Type	int64	object	float64	object	int64	object	int64	int64	int64
Nb lignes	34991	34991	34991	34991	34991	34991	34991	34991	34991
Valeurs non-vides	34991	34991	34990	34990	34991	34991	34991	34991	34991
Valeurs vides	0	0	1	1	0	0	0	0	0
Valeurs distinctes	17	100	10	60	908	32732	5900	719	5931

Description\_Fichier\_Tableau\_avc\_ValBlanches(donnees\_communes)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
CODREG	int64	34991	34991	0	17	0
CODDEP	object	34991	34991	0	100	0
CODARR	float64	34991	34990	1	10	0
CODCAN	object	34991	34990	1	60	76
CODCOM	int64	34991	34991	0	908	0
COM	object	34991	34991	0	32732	0
PMUN	int64	34991	34991	0	5900	0
PCAP	int64	34991	34991	0	719	0
PTOT	int64	34991	34991	0	5931	0

## ▼ B. FICHIER referentiel\_geographique

referentiel\_geographique.head()



	regrgp_nom	reg_nom	reg_nom_old	aca_nom	dep_nom	com_code	com_code1	com_code2	com_id	com_nom_maj_court	...	fd_id	fr_id
0	Province	Auvergne-Rhône-Alpes	Rhône-Alpes	Lyon	Ain	01001	1001	1001	C01001	L ABERGEMENT CLEMENCIAT	...	FD111	FR11
1	Province	Auvergne-Rhône-Alpes	Rhône-Alpes	Lyon	Ain	01002	1002	1002	C01002	L ABERGEMENT DE VAREY	...	FD111	FR11
2	Province	Auvergne-Rhône-Alpes	Rhône-Alpes	Lyon	Ain	01003	1003	1003	C01003	AMAREINS	...	FD111	FR11
3	Province	Auvergne-Rhône-Alpes	Rhône-Alpes	Lyon	Ain	01004	1004	1004	C01004	AMBERIEU EN BUGEY	...	FD111	FR11

referentiel\_geographique.shape



(38916, 37)

referentiel\_geographique.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38916 entries, 0 to 38915
Data columns (total 37 columns):
#   Column                Non-Null Count  Dtype
---  -
0   regrgp_nom            38916 non-null  object
1   reg_nom               38916 non-null  object
2   reg_nom_old           38916 non-null  object
3   aca_nom               38916 non-null  object
4   dep_nom               38916 non-null  object
5   com_code              38916 non-null  object
6   com_code1             38916 non-null  object
7   com_code2             38916 non-null  object
8   com_id               38916 non-null  object
9   com_nom_maj_court     38916 non-null  object
10  com_nom_maj           38916 non-null  object
11  com_nom               38916 non-null  object
12  uu_code               7625 non-null   object
13  uu_id                 38916 non-null  object
14  uucr_id               38916 non-null  object
15  uucr_nom              38916 non-null  object
16  ze_id                 38916 non-null  object
```

```

17 dep_code      38916 non-null object
18 dep_id        38916 non-null object
19 dep_nom_num    38916 non-null object
20 dep_num_nom    38916 non-null object
21 aca_code       38916 non-null int64
22 aca_id         38916 non-null object
23 reg_code       38916 non-null int64
24 reg_id         38916 non-null object
25 reg_code_old   38916 non-null int64
26 reg_id_old     38916 non-null object
27 fd_id         38916 non-null object
28 fr_id         38916 non-null object
29 fe_id         38916 non-null object
30 uu_id_99       38916 non-null object
31 au_code        21444 non-null object
32 au_id          38916 non-null object
33 auc_id         38916 non-null object
34 auc_nom        38916 non-null object
35 uu_id_10       38916 non-null object
36 geolocalisation 36745 non-null object
dtypes: int64(3), object(34)
memory usage: 11.0+ MB

```

Description\_Fichier\_Tableau(referentiel\_geographique).T

↻

	regrgp_nom	reg_nom	reg_nom_old	aca_nom	dep_nom	com_code	com_code1	com_code2	com_id	com_nom_maj_court	...	fd_id
Type	object	object	object	object	object	object	object	object	object	object	...	object
Nb lignes	38916	38916	38916	38916	38916	38916	38916	38916	38916	38916	...	38916
Valeurs non-vides	38916	38916	38916	38916	38916	38916	38916	38916	38916	38916	...	38916
Valeurs vides	0	0	0	0	0	0	0	0	0	0	...	0

◀ ▶

Description\_Fichier\_Tableau\_avc\_ValBlanches(referentiel\_geographique)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
<b>regrgp_nom</b>	object	38916	38916	0	3	0
<b>reg_nom</b>	object	38916	38916	0	19	0
<b>reg_nom_old</b>	object	38916	38916	0	28	0
<b>aca_nom</b>	object	38916	38916	0	37	0
<b>dep_nom</b>	object	38916	38916	0	109	0
<b>com_code</b>	object	38916	38916	0	38916	0
<b>com_code1</b>	object	38916	38916	0	38891	0
<b>com_code2</b>	object	38916	38916	0	38871	0
<b>com_id</b>	object	38916	38916	0	38916	0
<b>com_nom_maj_court</b>	object	38916	38916	0	35572	0
<b>com_nom_maj</b>	object	38916	38916	0	35525	0
<b>com_nom</b>	object	38916	38916	0	35872	0
<b>uu_code</b>	object	38916	7625	31291	2468	0
<b>uu_id</b>	object	38916	38916	0	2468	0
<b>uucr_id</b>	object	38916	38916	0	29946	0
<b>uucr_nom</b>	object	38916	38916	0	31370	0
<b>ze_id</b>	object	38916	38916	0	324	0
<b>dep_code</b>	object	38916	38916	0	109	0
<b>dep_id</b>	object	38916	38916	0	109	0
<b>dep_nom_num</b>	object	38916	38916	0	109	0
<b>dep_num_nom</b>	object	38916	38916	0	109	0
<b>aca_code</b>	int64	38916	38916	0	37	0
<b>aca_id</b>	object	38916	38916	0	37	0
<b>reg_code</b>	int64	38916	38916	0	19	0
<b>reg_id</b>	object	38916	38916	0	19	0
<b>reg_code_old</b>	int64	38916	38916	0	28	0
<b>reg_id_old</b>	object	38916	38916	0	28	0
<b>fd_id</b>	object	38916	38916	0	3	0
<b>fr_id</b>	object	38916	38916	0	2	0
<b>fe_id</b>	object	38916	38916	0	1	0
<b>uu_id_99</b>	object	38916	38916	0	2054	0
<b>au_code</b>	object	38916	21444	17472	799	0
<b>au_id</b>	object	38916	38916	0	801	0
<b>auc_id</b>	object	38916	38916	0	18185	0
<b>auc_nom</b>	object	38916	38916	0	17523	0
<b>uu_id_10</b>	object	38916	38916	0	2294	0
<b>geolocalisation</b>	object	38916	36745	2171	36746	0

### ✓ C. FICHIER valeurs\_foncieres

```
valeurs_foncieres.head()
```

	Code service CH	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Date mutation	Nature mutation	...	Nombre de lots	Code type local	Type local
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	2	2	Apparter
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	2	2	Apparter
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	1	2	Apparter
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	1	2	Apparter
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	1	2	Apparter

5 rows × 46 columns

valeurs\_foncieres.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34169 entries, 0 to 34168
Data columns (total 46 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Code service CH                           0 non-null      float64
1   Reference document                         0 non-null      float64
2   1 Articles CGI                            0 non-null      float64
3   2 Articles CGI                            0 non-null      float64
4   3 Articles CGI                            0 non-null      float64
5   4 Articles CGI                            0 non-null      float64
6   5 Articles CGI                            0 non-null      float64
7   No disposition                             34169 non-null  int64
8   Date mutation                             34169 non-null  datetime64[ns]
9   Nature mutation                           34169 non-null  object
10  Valeur fonciere                           34151 non-null  float64
11  No voie                                   34036 non-null  float64
12  B/T/Q                                    2174 non-null   object
13  Code type de voie                         34169 non-null  int64
14  Type de voie                             33229 non-null  object
15  Code voie                                34169 non-null  object
16  Voie                                     34169 non-null  object
17  Code ID commune                           34169 non-null  int64
18  Code postal                              34168 non-null  float64
19  Commune                                  34169 non-null  object
20  Code departement                          34169 non-null  object
21  Code commune                             34169 non-null  int64
22  Préfixe de section                       1143 non-null   float64
23  Section                                  34168 non-null  object
24  No plan                                   34169 non-null  int64
25  No Volume                                0 non-null      float64
26  1er lot                                   34169 non-null  object
27  Surface Carrez du 1er lot                 34169 non-null  float64
28  2eme lot                                  0 non-null      float64
29  Surface Carrez du 2eme lot                 0 non-null      float64
30  3eme lot                                  0 non-null      float64
31  Surface Carrez du 3eme lot                 0 non-null      float64
32  4eme lot                                  0 non-null      float64
33  Surface Carrez du 4eme lot                 0 non-null      float64
34  5eme lot                                  0 non-null      float64
35  Surface Carrez du 5eme lot                 0 non-null      float64
36  Nombre de lots                           34169 non-null  int64
37  Code type local                           34169 non-null  int64
38  Type local                               34169 non-null  object
39  Identifiant local                         0 non-null      float64
40  Surface reelle bati                       34169 non-null  int64
41  Nombre pieces principales                 34169 non-null  int64
42  Nature culture                           253 non-null    object
43  Nature culture speciale                   8 non-null      object
44  Surface terrain                           253 non-null    float64
45  Nom de l'acquereur                       34169 non-null  object
dtypes: datetime64[ns](1), float64(23), int64(9), object(13)
memory usage: 12.0+ MB

```

Description\_Fichier\_Tableau(valeurs\_foncieres).T



	Code service CH	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Date mutation	Nature mutation	...	Nombre de lots	Cc ty loc
Type	float64	float64	float64	float64	float64	float64	float64	int64	datetime64[ns]	object	...	int64	in
Nb lignes	34169	34169	34169	34169	34169	34169	34169	34169	34169	34169	...	34169	341
Valeurs non- vides	0	0	0	0	0	0	0	34169	34169	34169	...	34169	341
Valeurs vides	34169	34169	34169	34169	34169	34169	34169	0	0	0	...	0	
Valeurs distinctes	1	1	1	1	1	1	1	3	158	1	...	14	

Description\_Fichier\_Tableau\_avc\_ValBlanches(valeurs\_foncières)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
Code service CH	float64	34169	0	34169	1	0
Reference document	float64	34169	0	34169	1	0
1 Articles CGI	float64	34169	0	34169	1	0
2 Articles CGI	float64	34169	0	34169	1	0
3 Articles CGI	float64	34169	0	34169	1	0
4 Articles CGI	float64	34169	0	34169	1	0
5 Articles CGI	float64	34169	0	34169	1	0
No disposition	int64	34169	34169	0	3	0
Date mutation	datetime64[ns]	34169	34169	0	158	0
Nature mutation	object	34169	34169	0	1	0
Valeur fonciere	float64	34169	34151	18	9681	0
No voie	float64	34169	34036	133	1469	0
B/T/Q	object	34169	2174	31995	25	0
Code type de voie	int64	34169	34169	0	80	0
Type de voie	object	34169	33229	940	80	0
Code voie	object	34169	34169	0	5765	0
Voie	object	34169	34169	0	14133	0
Code ID commune	int64	34169	34169	0	3215	0
Code postal	float64	34169	34168	1	2450	0
Commune	object	34169	34169	0	3110	0
Code departement	object	34169	34169	0	96	0
Code commune	int64	34169	34169	0	666	0
Préfixe de section	float64	34169	1143	33026	159	0
Section	object	34169	34168	1	464	0
No plan	int64	34169	34169	0	1861	0
No Volume	float64	34169	0	34169	1	0
1er lot	object	34169	34169	0	1555	0
Surface Carrez du 1er lot	float64	34169	34169	0	10398	0
2eme lot	float64	34169	0	34169	1	0
Surface Carrez du 2eme lot	float64	34169	0	34169	1	0
3eme lot	float64	34169	0	34169	1	0
Surface Carrez du 3eme lot	float64	34169	0	34169	1	0
4eme lot	float64	34169	0	34169	1	0
Surface Carrez du 4eme lot	float64	34169	0	34169	1	0
5eme lot	float64	34169	0	34169	1	0
Surface Carrez du 5eme lot	float64	34169	0	34169	1	0
Nombre de lots	int64	34169	34169	0	14	0
Code type local	int64	34169	34169	0	2	0
Type local	object	34169	34169	0	2	0
Identifiant local	float64	34169	0	34169	1	0
Surface reelle bati	int64	34169	34169	0	256	0
Nombre pieces principales	int64	34169	34169	0	12	0
Nature culture	object	34169	253	33916	8	0
Nature culture speciale	object	34169	8	34161	5	0
Surface terrain	float64	34169	253	33916	201	0
Nom de l'acquireur	object	34169	34169	0	11112	0

✓ ANALYSE DE VALEURS ABERRANTES :



```
#valeurs_foncieres.describe(include = np.number)
```

```
#Reduisons à l'utile
```

```
(valeurs_foncieres.describe(include = np.number) ).loc[['count','mean','min','max'], ['Surface Carrez du 1er lot','Surface reelle bati',
```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere
<b>count</b>	34169.000000	34169.000000	34169.000000	253.000000	3.415100e+04
<b>mean</b>	57.644635	56.724458	2.616026	355.529644	2.528471e+05
<b>min</b>	0.400000	1.000000	0.000000	1.000000	5.375000e+02
<b>max</b>	5153.000000	379.000000	11.000000	2670.000000	9.000000e+06

Il y a quelques valeurs étranges minimum :

Surf Carrez =0,4m2 Surface reelle bati =1, Nbre pieces principales = 0 ;

Ca doit plus être un cajibi qu'un appartement ou une maison !!!

A priori cette ligen serait à retirer

```
valeurs_foncieres.loc[ valeurs_foncieres['Nombre pieces principales'] == 0, :['Surface terrain']
```



	Code service CH	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Date mutation	Nature mutation	...	Surface Carrez du 5eme lot	Nombre de lots
540	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-07	Vente	...	NaN	1
576	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-07	Vente	...	NaN	1
868	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-08	Vente	...	NaN	5
3663	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-22	Vente	...	NaN	1
4309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-24	Vente	...	NaN	1
4556	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-27	Vente	...	NaN	1
5357	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-30	Vente	...	NaN	1
6130	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-31	Vente	...	NaN	1
6668	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-04	Vente	...	NaN	1
7089	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-06	Vente	...	NaN	1
8456	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-12	Vente	...	NaN	1
10962	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-24	Vente	...	NaN	1
11340	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-26	Vente	...	NaN	1
12822	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-03-03	Vente	...	NaN	1
15311	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-03-16	Vente	...	NaN	1
16025	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-03-20	Vente	...	NaN	1
16235	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-03-24	Vente	...	NaN	1
17052	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-04-06	Vente	...	NaN	1
18932	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-04-30	Vente	...	NaN	1
19133	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-04-30	Vente	...	NaN	1
20095	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-05-11	Vente	...	NaN	1
21599	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-05-15	Vente	...	NaN	2
22380	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-05-19	Vente	...	NaN	1
25396	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-05-28	Vente	...	NaN	1
26903	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-02	Vente	...	NaN	2
26982	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-02	Vente	...	NaN	1
29068	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-10	Vente	...	NaN	1
30078	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-13	Vente	...	NaN	1
31191	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-18	Vente	...	NaN	2
31221	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-18	Vente	...	NaN	2
									2020-06-				

32353	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-24	Vente	...	NaN	1
32416	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-24	Vente	...	NaN	1
32681	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-25	Vente	...	NaN	1

pour Nombres de pièces il semble que ce soit plus une case qui n'a pas été remplie car la Surface réelle est plsu grande.

(valeurs\_foncieres.loc[ valeurs\_foncieres['Surface reelle bati'] <= 4, :] )[['Surface Carrez du 1er lot','Surface reelle bati', 'Nombre

↔

	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
22380	2.36	2	0	NaN	116500.0	Appartement
30429	9.62	1	1	NaN	175000.0	Appartement

Ces 2 valeurs semblent abérrentes, en effet :

- pour la première : les données sont cohérentes, mais il ne semble pas s'agir d'un appartement plutôt d'un local avec 2,36 m2
- pour la seconde : les données sont incohérentes : avoir 9,62 m2 n'est pas cohérent avec le fait d'avoir 1m2 bati.

Je fais donc le choix de supprimer ces valeurs.  
CELA SERA FAIT DANS LE NETTOYAGE DES FICHIERS

Commencez à coder ou à [générer](#) avec l'IA.

Commencez à coder ou à [générer](#) avec l'IA.

▼ IV. NETTOYAGE DES FICHIERS

Commencez à coder ou à [générer](#) avec l'IA.

▼ A. Nettotage Fichier donnees communes

Commencez à coder ou à [générer](#) avec l'IA.

Commencez à coder ou à [générer](#) avec l'IA.

▼ B. Nettoyage FICHIER referentiel\_geographique

Commencez à coder ou à [générer](#) avec l'IA.

Commencez à coder ou à [générer](#) avec l'IA.

▼ C. Nettoyage FICHIER valeurs\_foncieres

Description\_Fichier\_Tableau(valeurs\_foncieres)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Code service CH	float64	34169	0	34169	1
Reference document	float64	34169	0	34169	1
1 Articles CGI	float64	34169	0	34169	1
2 Articles CGI	float64	34169	0	34169	1
3 Articles CGI	float64	34169	0	34169	1
4 Articles CGI	float64	34169	0	34169	1
5 Articles CGI	float64	34169	0	34169	1
No disposition	int64	34169	34169	0	3
Date mutation	datetime64[ns]	34169	34169	0	158
Nature mutation	object	34169	34169	0	1
Valeur fonciere	float64	34169	34151	18	9681
No voie	float64	34169	34036	133	1469
B/T/Q	object	34169	2174	31995	25
Code type de voie	int64	34169	34169	0	80
Type de voie	object	34169	33229	940	80
Code voie	object	34169	34169	0	5765
Voie	object	34169	34169	0	14133
Code ID commune	int64	34169	34169	0	3215
Code postal	float64	34169	34168	1	2450
Commune	object	34169	34169	0	3110
Code departement	object	34169	34169	0	96
Code commune	int64	34169	34169	0	666
Préfixe de section	float64	34169	1143	33026	159
Section	object	34169	34168	1	464
No plan	int64	34169	34169	0	1861
No Volume	float64	34169	0	34169	1
1er lot	object	34169	34169	0	1555
Surface Carrez du 1er lot	float64	34169	34169	0	10398
2eme lot	float64	34169	0	34169	1
Surface Carrez du 2eme lot	float64	34169	0	34169	1
3eme lot	float64	34169	0	34169	1
Surface Carrez du 3eme lot	float64	34169	0	34169	1
4eme lot	float64	34169	0	34169	1
Surface Carrez du 4eme lot	float64	34169	0	34169	1
5eme lot	float64	34169	0	34169	1
Surface Carrez du 5eme lot	float64	34169	0	34169	1
Nombre de lots	int64	34169	34169	0	14
Code type local	int64	34169	34169	0	2
Type local	object	34169	34169	0	2
Identifiant local	float64	34169	0	34169	1
Surface reelle bati	int64	34169	34169	0	256
Nombre pieces principales	int64	34169	34169	0	12
Nature culture	object	34169	253	33916	8
Nature culture speciale	object	34169	8	34161	5
Surface terrain	float64	34169	253	33916	201
Nom de l'acquereur	object	34169	34169	0	11112

CREONS un fichier valeurs\_foncieres\_nettoye\_1 avec toutes les colonnes mais que l'on nettoiera

```
valeurs_foncieries_nettoyee_1 = valeurs_foncieries.copy(deep=True)
```

ANONYMYSONS TOUT DE SUITE

```
del valeurs_foncieries_nettoyee_1["Nom de l'acquéreur"]
```

REDUISONS AUX COLONNES UTILES

[ ] 7 cellules masquées

1. Analysons le Code postal manquant


ANALYSONS LE CODE POSTAL ABSENT

```
valeurs_foncieries_analyse_1A.loc[(valeurs_foncieries_analyse_1A['Code postal']).isna(),:]
```

No voie	Type de voie	Voie	B/T/Q	Code postal	Surface Carrez du 1er lot	Type local	Surface reelle bati	Nombre pieces principales	Surface terrain	Code departement	Code commune	Date mutation

Regardons dans le fichier d'origine à l'aide également du 'Code ID commune' :

```
valeurs_foncieries.loc[((valeurs_foncieries['Code departement'])=='2A') & ((valeurs_foncieries['Code commune'])==4),['Code postal','Code de  
']
```



	Code postal	Code departement	Code commune	Code ID commune
908	20090.0	2A	4	618
909	20000.0	2A	4	616
1607	20090.0	2A	4	618
1608	20090.0	2A	4	618
1856	20090.0	2A	4	618
...	...	...	...	...
30751	20090.0	2A	4	618
30752	20000.0	2A	4	616
30753	20000.0	2A	4	616
31977	20000.0	2A	4	616
32242	20090.0	2A	4	618

104 rows x 4 columns

```
df_2 = valeurs_foncieries.loc[((valeurs_foncieries['Code departement'])=='2A') & ((valeurs_foncieries['Code commune'])==4),['Code postal',  
]
```

```
df_3=df_2.drop_duplicates()  
df_3
```

	Code postal	Code departement	Code commune	Code ID commune
908	20090.0	2A	4	618
909	20000.0	2A	4	616
26834	NaN	2A	4	658

Il semble qu'il y ait une erreur sur l'index 26834 : codepostal = 20090.0 et code ID commune=618; car c'est la seule valeur ainsi.  
NETTOYONS le fichier valeurs\_foncieries\_nettoyee\_1 avec cette information

=>CORRECTION : REMPLISSONS LE CODE POSTAL ABSENT

[ ] 4 cellules masquées

## 2. Analysons les Valeurs foncières manquantes

### ANALYSE DE VALEUR FONCIERES MANQUANTES

Réduisons les colonnes à celles qui vont nous servir pour analyser (avant de nettoyer les colonnes utiles) dans valeurs\_foncières\_analyse\_2A

```
valeurs_foncières_analyse_2A = valeurs_foncières_nettoye_1[ ['No voie', 'Type de voie', 'Voie', 'B/T/Q', 'Code postal', 'Surface Carrez', 'Surface terrain', 'Code département', 'Code commune', 'Date mutation', 'Valeur foncière']]
```

	No voie	Type de voie	Voie	B/T/Q	Code postal	Surface Carrez du 1er lot	Type local	Surface reelle bati	Nombre pieces principales	Surface terrain	Code département	Code commune	Date mutation	Valeur foncière
0	347.0	RUE	DU CHATEAU	NaN	1170.0	48.22	Appartement	48	3	NaN	01	103	2020-01-02	16500
1	4.0	BD	EDOUARD BAUDOUIN	NaN	6160.0	39.11	Appartement	40	1	NaN	06	4	2020-01-02	35568
2	20.0	RUE	MARCEAU	B	6000.0	80.25	Appartement	82	3	NaN	06	88	2020-01-02	22950

Description\_Fichier\_Tableau(valeurs\_foncières\_analyse\_2A)

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
No voie	float64	34169	34036	133	1469
Type de voie	object	34169	33229	940	80
Voie	object	34169	34169	0	14133
B/T/Q	object	34169	2174	31995	25
Code postal	float64	34169	34169	0	2449
Surface Carrez du 1er lot	float64	34169	34169	0	10398
Type local	object	34169	34169	0	2
Surface réelle bati	int64	34169	34169	0	256
Nombre pièces principales	int64	34169	34169	0	12
Surface terrain	float64	34169	253	33916	201
Code département	object	34169	34169	0	96
Code commune	int64	34169	34169	0	666
Date mutation	datetime64[ns]	34169	34169	0	158
Valeur foncière	float64	34169	34151	18	9681


Des ventes sans prix cela paraît incohérent.

La vente a-t-elle réellement eu lieu ?

Après une observation dans le fichier Excel, les Valeur foncière nulle n'ont pas l'air de se rapprocher d'autres ventes en doublons. En l'absence d'information, nous allons supprimer ces 18 lignes sur 34169, le phénomène n'est pas massif, cela ne va pas amputer le fichier d'une quantité importante de données, mais peut éviter le fait de faire baisser artificiellement le prix moyen de certaines régions etc...


### =>CORRECTION : SUPPRIMONS LES 18 lignes avec des Valeurs foncières vides

```
valeurs_foncières_nettoye_1.loc[(valeurs_foncières_nettoye_1['Valeur foncière']).isna(), :]
```




	Code service CH	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Date mutation	Nature mutation	...	Surface Carrez du 5eme lot	Nombre de lots	1
1676	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-01-13	Vente	...	NaN	1	
3620	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-22	Vente	...	NaN	1	
3918	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-23	Vente	...	NaN	1	
5518	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-30	Vente	...	NaN	1	
8280	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-12	Vente	...	NaN	1	
11345	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-02-26	Vente	...	NaN	2	
11516	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-02-26	Vente	...	NaN	1	
18482	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-04-24	Vente	...	NaN	1	
19394	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-05-04	Vente	...	NaN	2	
26724	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-05-30	Vente	...	NaN	3	
27910	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-05	Vente	...	NaN	1	
28962	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-09	Vente	...	NaN	2	
29552	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-06-11	Vente	...	NaN	1	
29722	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-12	Vente	...	NaN	1	
31141	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-06-18	Vente	...	NaN	2	
31699	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-06-20	Vente	...	NaN	2	
32503	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2020-06-25	Vente	...	NaN	1	
34010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-06-30	Vente	...	NaN	1	

```
( valeurs_foncieres_nettoye_1.loc[(valeurs_foncieres_nettoye_1['Valeur fonciere']).isna(), :]).index
```

 Index([ 1676, 3620, 3918, 5518, 8280, 11345, 11516, 18482, 19394, 26724, 27910, 28962, 29552, 29722, 31141, 31699, 32503, 34010], dtype='int64')

liste\_col\_utiles\_VF

 ['No voie',  
'Type de voie',  
'Voie',  
'B/T/Q',  
'Code postal',  
'Surface Carrez du 1er lot',  
'Type local',  
'Surface reelle bati',  
'Nombre pieces principales',  
'Surface terrain',  
'Code departement',  
'Code commune',  
'Date mutation',  
'Valeur fonciere']

```
valeurs_foncieres_nettoye_1.loc[(valeurs_foncieres_nettoye_1['Valeur fonciere']).isna(), liste_col_utiles_VF]
```

	No voie	Type de voie	Voie	B/T/Q	Code postal	Surface Carrez du 1er lot	Type local	Surface reelle bati	Nombre pieces principales	Surface terrain	Code departement	Code commune	Date mutation
1676	194.0	RUE	DE RIVOLI	B	75001.0	197.92	Appartement	225	6	NaN	75	101	2020-01-15
3620	23.0	RUE	STE CROIX BRETONNERIE	NaN	75004.0	21.00	Appartement	23	1	NaN	75	104	2020-01-25
3918	15.0	QUAI	DE LA SOMME	NaN	76200.0	34.64	Appartement	42	2	NaN	76	217	2020-01-25
5518	13.0	RUE	DE LA PORTE NEUVE	NaN	62200.0	40.02	Appartement	43	2	NaN	62	160	2020-01-30
8280	5360.0	ESP	DESPIERRE	NaN	5560.0	17.34	Appartement	17	1	NaN	05	177	2020-02-15
11345	74.0	RUE	ARTHUR LAMENDIN	NaN	62400.0	35.47	Appartement	37	2	NaN	62	119	2020-02-20
11516	12.0	AV	PABLO PICASSO	NaN	93420.0	46.95	Appartement	63	2	NaN	93	78	2020-02-20
18482	3.0	RUE	DANTON	NaN	84800.0	72.00	Appartement	91	4	NaN	84	54	2020-04-25
19394	25.0	RUE	DES PARAMIDEAUX	NaN	6110.0	52.02	Appartement	56	2	NaN	06	30	2020-05-05
26724	5421.0	RES	DU PARC 3	NaN	20167.0	45.64	Appartement	46	2	NaN	2A	271	2020-05-30
27910	155.0	RUE	DES FAUVETTES	NaN	16600.0	85.22	Maison	82	5	NaN	16	291	2020-06-05
28062	2.0	RUE	DE LA	NaN	04120.0	21.61	Appartement	22	1	NaN	04	52	2020-06-15

```
(valeurs_foncieres_nettoyee_1.loc[(valeurs_foncieres_nettoyee_1['Valeur fonciere']).isna(), 'Code departement']).value_counts()
```

```
Code departement
75      3
62      2
93      2
76      1
05      1
84      1
06      1
2A      1
16      1
94      1
44      1
28      1
19      1
83      1
Name: count, dtype: int64
```

```
liste_1 = ( valeurs_foncieres_nettoyee_1.loc[(valeurs_foncieres_nettoyee_1['Valeur fonciere']).isna(), :] ).index
valeurs_foncieres_nettoyee_1.drop(index=liste_1, inplace =True)
```

```
Description_Fichier_Tableau(pd.DataFrame(valeurs_foncieres_nettoyee_1['Valeur fonciere']))
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Valeur fonciere	float64	34151	34151	0	9680

C'est bon je n'ai plus de Valeur fonciere vide dans valeurs\_foncieres\_nettoyee\_1

**=>CORRECTION : SUPPRESSION DE 18 lignes avec des Valeurs foncières vides**

### 3. Revenons sur les valeurs abérrantes

#### ANALYSE DE VALEURS ABERRANTES :

```
#valeurs_foncieres_nettoyee_1.describe(include = np.number)
```

```
valeurs_foncieres_nettoyee_1.columns
```



```

Index(['Code service CH', 'Reference document', '1 Articles CGI',
      '2 Articles CGI', '3 Articles CGI', '4 Articles CGI', '5 Articles CGI',
      'No disposition', 'Date mutation', 'Nature mutation', 'Valeur fonciere',
      'No voie', 'B/T/Q', 'Code type de voie', 'Type de voie', 'Code voie',
      'Voie', 'Code ID commune', 'Code postal', 'Commune', 'Code departement',
      'Code commune', 'Préfixe de section', 'Section', 'No plan', 'No Volume',
      '1er lot', 'Surface Carrez du 1er lot', '2eme lot',
      'Surface Carrez du 2eme lot', '3eme lot', 'Surface Carrez du 3eme lot',
      '4eme lot', 'Surface Carrez du 4eme lot', '5eme lot',
      'Surface Carrez du 5eme lot', 'Nombre de lots', 'Code type local',
      'Type local', 'Identifiant local', 'Surface reelle bati',
      'Nombre pieces principales', 'Nature culture',
      'Nature culture speciale', 'Surface terrain'],
      dtype='object')

#Reduisons à l'utile
liste_attributs_surface = ['Surface Carrez du 1er lot', 'Surface reelle bati',
                          'Nombre pieces principales', 'Surface terrain',
                          'Valeur fonciere']
(valeurs_foncieres_nettoye_1.describe(include = np.number) ).loc[['count', 'mean', 'min', 'max'], liste_attributs_surface]

```

	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere
<b>count</b>	34151.000000	34151.000000	34151.000000	253.000000	3.415100e+04
<b>mean</b>	57.645079	56.722644	2.616029	355.529644	2.528471e+05
<b>min</b>	0.400000	1.000000	0.000000	1.000000	5.375000e+02
<b>max</b>	5153.000000	379.000000	11.000000	2670.000000	9.000000e+06

Il y a quelques valeurs étranges minimum :

Surf Carrez =0,4m2 Surface reelle bati =1, Nbre pieces principales = 0 ;

Ca doit plus être un cajibi qu'un appartement ou une maison !!!

A priori cette ligne serait à retirer

#### ✓ a. ETUDE DU NOMBRE DE PIECES PRINCIPALES =0

```

valeurs_foncieres_nettoye_1.loc[ valeurs_foncieres_nettoye_1['Nombre pieces principales'] == 0, liste_attributs_surface]

```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere
540	33.46	27	0	NaN	12000.0
576	18.00	19	0	NaN	80750.0
868	29.67	121	0	NaN	355000.0
3663	25.12	15	0	NaN	97200.0
4309	15.75	14	0	NaN	42500.0
4556	15.05	13	0	NaN	61000.0
5357	8.50	10	0	NaN	91000.0
6130	25.85	26	0	NaN	150600.0
6668	14.54	15	0	NaN	70000.0
7089	19.30	20	0	NaN	76500.0
8456	19.05	20	0	NaN	185000.0
10962	12.59	12	0	NaN	82500.0
11340	65.36	67	0	NaN	161510.0
12822	13.77	16	0	NaN	145000.0
15311	41.78	41	0	NaN	89000.0
16025	14.20	13	0	NaN	52500.0
16235	47.45	60	0	NaN	232000.0
17052	20.95	23	0	NaN	273200.0
18932	24.74	31	0	NaN	125000.0
19133	30.35	28	0	NaN	140000.0
20095	14.50	15	0	NaN	62000.0
21599	15.40	23	0	NaN	189350.0
22380	2.36	2	0	NaN	116500.0
25396	27.42	30	0	NaN	88000.0
26903	49.03	55	0	NaN	123000.0
26982	10.75	12	0	NaN	134000.0
29068	27.90	31	0	NaN	81859.0
30078	10.14	23	0	NaN	100000.0
31191	32.75	34	0	NaN	75000.0
31221	55.54	55	0	NaN	45000.0
32353	25.70	25	0	NaN	215000.0
32416	9.91	10	0	NaN	43000.0
32681	14.29	17	0	NaN	59280.0

pour Nombres de pièces il semble que ce soit plus une case qui n'a pas été systématiquement remplie car la Surface réelle est plus grande que zéro.

**=> PAS DE MODIFICATION pour NOMBRE DE PIECES ==0**

**=> PAS DE MODIFICATION pour NOMBRE DE PIECES ==0**

#### ✓ b. ETUDE des Faibles surfaces baties

```
liste_attributs_surface_2 = ['Surface Carrez du 1er lot','Surface reelle bati',  
                             'Nombre pieces principales', 'Surface terrain',  
                             'Valeur fonciere','Type local']  
(valeurs_foncieres_nettoye_1.loc[ valeurs_foncieres_nettoye_1['Surface reelle bati'] <= 4, :]) [liste_attributs_surface_2]
```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
22380	2.36	2	0	NaN	116500.0	Appartement
30429	9.62	1	1	NaN	175000.0	Appartement

Ces 2 valeurs semblent aberrantes, en effet :

- pour la première : les données sont cohérentes, mais il ne semble pas s'agir d'un appartement plutôt d'un local avec 2,36 m2
- pour la seconde : les données sont incohérentes : avoir 9,62 m2 n'est pas cohérent avec le fait d'avoir 1m2 bati.

Je fais donc le choix de supprimer ces valeurs.

#### ✓ =>CORRECTION : SUPPRIMONS les 2 lignes avec des surfaces réelles baties <=4m2

```
valeurs_foncières_nettoyé_1.shape
```

```
(34151, 45)
```

```
df_4 = (valeurs_foncières_nettoyé_1.loc[ valeurs_foncières_nettoyé_1['Surface réelle bati'] <= 4, :]) [liste_attributs_surface_2]  
liste_index_aberrant_2 = df_4.index
```

```
valeurs_foncières_nettoyé_1.drop(index=liste_index_aberrant_2, inplace=True )
```

```
valeurs_foncières_nettoyé_1.shape
```

```
(34149, 45)
```

REFAISONS LE POINT SUR LES VALEURS ABERRANTES :

```
#Reduisons à l'utile  
liste_attributs_surface = ['Surface Carrez du 1er lot','Surface réelle bati',  
                           'Nombre pièces principales', 'Surface terrain',  
                           'Valeur foncière']  
(valeurs_foncières_nettoyé_1.describe(include = np.number) ).loc[['count','mean','min','max'], liste_attributs_surface]
```



	Surface Carrez du 1er lot	Surface réelle bati	Nombre pièces principales	Surface terrain	Valeur foncière
count	34149.000000	34149.000000	34149.000000	253.000000	3.414900e+04
mean	57.648104	56.725878	2.616153	355.529644	2.528534e+05
min	0.400000	5.000000	0.000000	1.000000	5.375000e+02
max	5153.000000	379.000000	11.000000	2670.000000	9.000000e+06

#### =>CORRECTION : SUPPRESSION de 2 lignes avec des surfaces réelles baties <=4m2

#### ✓ c. ETUDE des Faibles Surface Carrez

```
liste_attributs_surface_2 = ['Surface Carrez du 1er lot','Surface réelle bati',  
                             'Nombre pièces principales', 'Surface terrain',  
                             'Valeur foncière','Type local']  
(valeurs_foncières_nettoyé_1.loc[ valeurs_foncières_nettoyé_1['Surface Carrez du 1er lot'] <= 4, :]) [liste_attributs_surface_2]
```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
1999	3.62	17	2	NaN	40500.0	Appartement
3337	3.30	29	2	NaN	83000.0	Appartement
4782	3.27	84	4	NaN	288000.0	Appartement
5363	4.00	39	2	NaN	164800.0	Appartement
7532	1.76	65	3	NaN	73000.0	Maison
12393	3.45	90	4	NaN	905000.0	Appartement
14202	3.85	7	1	NaN	94750.0	Appartement
17234	2.10	42	3	NaN	537.5	Appartement
18520	0.40	26	1	NaN	50740.0	Appartement
18812	1.31	107	4	NaN	289900.0	Appartement
19668	0.50	18	1	NaN	43500.0	Appartement
19897	1.43	80	3	NaN	3000.0	Appartement
20558	3.00	94	3	NaN	84000.0	Appartement
20687	3.37	19	1	NaN	166440.0	Appartement
21030	0.50	14	1	NaN	336740.0	Appartement
22984	2.00	45	2	NaN	82000.0	Appartement
28116	3.64	5	1	NaN	15000.0	Appartement
28346	1.36	13	1	NaN	57000.0	Appartement
32635	3.45	74	3	NaN	315000.0	Appartement

On a toujours plus de 5 m2 au sol : ça pourrait être un studio, une chambre de bonne !?

=> PAS DE MODIFICATION pour Faibles Surface Carrez

#### ✓ d. ETUDE des Faibles Valeurs foncieres

```
liste_attributs_surface_2 = ['Surface Carrez du 1er lot','Surface reelle bati',  
                             'Nombre pieces principales', 'Surface terrain',  
                             'Valeur fonciere','Type local']  
(valeurs_foncieres_nettoye_1.loc[ valeurs_foncieres_nettoye_1['Valeur fonciere'] <= 10_000, :]) [liste_attributs_surface_2]
```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
1213	47.05	45	3	NaN	9000.00	Appartement
1536	62.74	62	4	NaN	10000.00	Appartement
2404	45.50	45	2	NaN	9000.00	Appartement
4754	10.60	46	3	NaN	7000.00	Appartement
4865	12.40	21	1	NaN	3000.00	Appartement
5472	23.24	23	1	NaN	1021.05	Appartement
6370	23.93	23	1	NaN	8000.00	Appartement
6880	23.27	23	1	NaN	6000.00	Appartement
11982	23.20	22	1	NaN	6500.00	Appartement
12024	162.12	110	5	NaN	8000.00	Appartement
12531	46.95	47	2	NaN	5000.00	Appartement
12773	36.80	32	2	NaN	10000.00	Appartement
14563	24.20	54	3	NaN	10000.00	Appartement
16003	20.50	58	1	NaN	5000.00	Appartement
16922	18.50	19	1	NaN	10000.00	Appartement
17234	2.10	42	3	NaN	537.50	Appartement
17759	62.19	60	3	NaN	10000.00	Appartement
18558	23.52	20	1	NaN	10000.00	Appartement
19897	1.43	80	3	NaN	3000.00	Appartement
20576	15.00	57	3	NaN	2200.00	Appartement
22966	17.72	19	1	NaN	9000.00	Appartement
23892	23.16	22	1	NaN	10000.00	Appartement
24378	21.79	25	2	NaN	8000.00	Appartement
25858	12.86	10	1	NaN	3000.00	Appartement
27628	15.79	17	1	NaN	6000.00	Appartement
27630	42.00	35	1	NaN	9500.00	Appartement
28108	8.78	17	1	NaN	10000.00	Appartement
28409	40.17	22	1	NaN	9000.00	Appartement
31575	25.47	26	1	NaN	5000.00	Appartement
31974	129.67	130	6	NaN	6000.00	Appartement
33764	23.40	23	1	NaN	6500.00	Appartement

Les surfaces Carrez correspondent à des surfaces avec une hauteur minimum sous plafond.

pour un appartement sous comble il se peut qu'une grande partie ne soit pas comptée en surface Carrez.

Je fais donc le choix de conserver ces données sachant qu'elles correspondent à des Surfaces réelles baties plus grande : au moins 5 m2.

Il existe quelques valeurs étonnantes,notammennt pour les montants inférieurs à 2500 euros

```
liste_attributs_surface_2 = ['Surface Carrez du 1er lot','Surface reelle bati',
                             'Nombre pieces principales', 'Surface terrain',
                             'Valeur fonciere','Type local']
(valeurs_fonciere_nettoye_1.loc[ valeurs_fonciere_nettoye_1['Valeur fonciere'] <= 2_500, :]) [liste_attributs_surface_2]
```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
5472	23.24	23	1	NaN	1021.05	Appartement
17234	2.10	42	3	NaN	537.50	Appartement
20576	15.00	57	3	NaN	2200.00	Appartement

Les prix semblent incohérents avec les Surfaces. Problème de virgule dans la valeur Foncière ?

chiffre entrée sans prendre en comptes les centimes ?

Ces lignes pourraient aussi bien être retirées que gardées.

En effet, un état très dégradés du bien, une hauteur sous plafond faible, une mauvais situation du bien limitant son utilisation, ou d'autres

éléments pourraient jsutifier d'une très faible valeur.  
Cette possibilité mériterait de les conserver.  
Cependant la valeur est probablement erronée.

Cependant elles pourraient peser sur une moyenne localement dans un département avec peu de vente.  
Et étant en faible nombre les retirer n'influencerait pas la moyenne nationale.

> => JE FAIS DONC LE CHOIX RETIRER ces valeurs avec un montant <=2\_500 euros

[ ] ↳ 6 cellules masquées

CORRECTION : SUPPRESSION de 3 lignes avec des valeur foncières <=2\_500 euros

▼ e. ETUDE des Grandes Surface Carrez

```
liste_attributs_surface_2 = ['Surface Carrez du 1er lot','Surface reelle bati',
                             'Nombre pieces principales', 'Surface terrain',
                             'Valeur fonciere','Type local']
(valeurs_foncieres_nettoye_1.loc[ valeurs_foncieres_nettoye_1['Surface Carrez du 1er lot'] >= 500, :]) [liste_attributs_surface_2]
```

	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
2354	1291.44	100	3	NaN	700000.0	Appartement
7010	5153.00	53	3	NaN	265000.0	Appartement
7630	1483.78	45	2	NaN	4098416.2	Appartement
10202	4936.00	52	3	NaN	223645.0	Appartement
10713	2910.92	35	1	NaN	2500000.0	Appartement
11615	742.17	40	2	NaN	1683000.0	Maison
12078	509.85	19	1	NaN	285000.0	Appartement
14352	815.00	81	3	NaN	154000.0	Appartement
16356	595.00	241	8	NaN	7200000.0	Appartement
18172	570.00	110	6	NaN	95000.0	Maison
23022	558.87	348	9	NaN	1608880.0	Appartement

Il semble qu'il y ait des incohérences, notamment lorsque le facteur entre surface relle bati et Surface Carrez est incohérent.  
Je ne peux pas corriger cela.  
Je pourrai retirer ces lignes, cependant rien ne m'assure qu'il n'y ait pas d'erreurs sur les surfaces plus petites également.  
Je préfère donc ne pas retirer ces lignes pour le moment.  
J'ai conscience que la Surface Carrez est peu fiable en cas de demande de calcul.

=> JE FAIS DONC LE CHOIX (dans un premier de temps) de ne pas RETIRER ces valeurs

=> PAS DE MODIFICATION pour Grandes Surfaces Carrez

▼ f. ETUDE des Fortes Valeurs foncières

```
liste_attributs_surface_2 = ['Surface Carrez du 1er lot','Surface reelle bati',
                             'Nombre pieces principales', 'Surface terrain',
                             'Valeur fonciere','Type local']
(valeurs_foncieres_nettoye_1.loc[ valeurs_foncieres_nettoye_1['Valeur fonciere'] >= 5_000_000, :]) [liste_attributs_surface_2]
```

	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere	Type local
409	360.95	357	8	NaN	7420000.0	Appartement
1923	122.56	310	7	NaN	7050000.0	Appartement
3624	20.55	289	7	NaN	8577713.0	Appartement
4233	211.07	205	7	NaN	5054500.0	Appartement
5260	64.00	62	3	NaN	8600000.0	Appartement
7601	42.77	42	2	NaN	7620000.0	Appartement
9987	253.30	200	5	NaN	7600000.0	Appartement
11361	78.46	100	4	NaN	6454000.0	Appartement
12167	170.82	172	4	NaN	5600000.0	Appartement
13554	205.08	215	4	NaN	6000000.0	Appartement
16213	157.00	150	4	NaN	5500000.0	Appartement
16356	595.00	241	8	NaN	7200000.0	Appartement
17057	328.35	346	9	NaN	5550000.0	Appartement
17822	139.90	143	4	NaN	7535000.0	Appartement
19160	79.38	76	3	NaN	6600000.0	Appartement
27681	317.05	287	6	NaN	6500000.0	Appartement
30602	9.10	10	1	NaN	9000000.0	Appartement

La seule valeur clairement abérrante est celle de 9 millions pour 9 m2.

Cependant il est difficile de juger le prix d'un bien.

EN effet cette chambre de bonne par exemple a-t-elle été habité par quelqu'un de célèbre et se trouve à proximité de la tour Eiffel !!! ?

Difficile de juger.

Je vais garder ce bien en tête et voir si il impacte mes analyses.

valeurs\_foncieres\_nettoye\_1.columns

```
Index(['Code service CH', 'Reference document', '1 Articles CGI',
      '2 Articles CGI', '3 Articles CGI', '4 Articles CGI', '5 Articles CGI',
      'No disposition', 'Date mutation', 'Nature mutation', 'Valeur fonciere',
      'No voie', 'B/T/Q', 'Code type de voie', 'Type de voie', 'Code voie',
      'Voie', 'Code ID commune', 'Code postal', 'Commune', 'Code departement',
      'Code commune', 'Préfixe de section', 'Section', 'No plan', 'No Volume',
      '1er lot', 'Surface Carrez du 1er lot', '2eme lot',
      'Surface Carrez du 2eme lot', '3eme lot', 'Surface Carrez du 3eme lot',
      '4eme lot', 'Surface Carrez du 4eme lot', '5eme lot',
      'Surface Carrez du 5eme lot', 'Nombre de lots', 'Code type local',
      'Type local', 'Identifiant local', 'Surface reelle bati',
      'Nombre pieces principales', 'Nature culture',
      'Nature culture speciale', 'Surface terrain'],
      dtype='object')
```

#pour l'ensemble de ses informations :

```
(valeurs_foncieres_nettoye_1.loc[ valeurs_foncieres_nettoye_1['Valeur fonciere'] >= 9_000_000, :]).loc[:, 'Date mutation': 'Surface Carrez du 1er lot']
```

Date mutation	Nature mutation	Valeur fonciere	No voie	B/T/Q	Code type de voie	Type de voie	Code voie	Voie	Code ID commune	Code postal	Commune	Code departement	Code commune	Préfixe de section

IL s'agit en effet d'un arrondissement de Paris où les prix sont très élevés.

Il est donc difficile de juger s'il s'agit d'une valeur abérrante ou non.


=> JE FAIS DONC LE CHOIX (dans un premier de temps) de ne pas RETIRER cette valeur

=> PAS DE MODIFICATION pour Fortes valeurs fonciere


g. Fichier Valeurs fonciere APRES NETTOYAGE

Faisons le bilan de notre fichier nettoyé valeurs\_foncieres\_nettoye\_1 .  
On le rappelle : il contient les colonnes mais avec un correctif sur les lignes ou valeurs abérrantes.


```
valeurs_foncieres_nettoye_1.head()
```



	Code service CH	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Date mutation	Nature mutation	...	Surface Carrez du 5eme lot	Nombre de lots	Code type local
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	2	2
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	2	2
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	1	2
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	1	2
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	1	2



```
liste_col_utiles_VF
```



```
['No voie',  
'Type de voie',  
'Voie',  
'B/T/Q',  
'Code postal',  
'Surface Carrez du 1er lot',  
'Type local',  
'Surface reelle bati',  
'Nombre pieces principales',  
'Surface terrain',  
'Code departement',  
'Code commune',  
'Date mutation',  
'Valeur fonciere']
```

```
#Description_Fichier_Tableau(valeurs_foncieres_nettoye_1)  
Description_Fichier_Tableau_avc_ValBlanches(valeurs_foncieres_nettoye_1)
```





	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
Code service CH	float64	34146	0	34146	1	0
Reference document	float64	34146	0	34146	1	0
1 Articles CGI	float64	34146	0	34146	1	0
2 Articles CGI	float64	34146	0	34146	1	0
3 Articles CGI	float64	34146	0	34146	1	0
4 Articles CGI	float64	34146	0	34146	1	0
5 Articles CGI	float64	34146	0	34146	1	0
No disposition	int64	34146	34146	0	3	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Nature mutation	object	34146	34146	0	1	0
Valeur fonciere	float64	34146	34146	0	9677	0
No voie	float64	34146	34013	133	1468	0
B/T/Q	object	34146	2173	31973	25	0
Code type de voie	int64	34146	34146	0	80	0
Type de voie	object	34146	33206	940	80	0
Code voie	object	34146	34146	0	5765	0
Voie	object	34146	34146	0	14130	0
Code ID commune	int64	34146	34146	0	3215	0
Code postal	float64	34146	34146	0	2449	0
Commune	object	34146	34146	0	3110	0
Code departement	object	34146	34146	0	96	0
Code commune	int64	34146	34146	0	666	0
Préfixe de section	float64	34146	1143	33003	159	0
Section	object	34146	34145	1	464	0
No plan	int64	34146	34146	0	1861	0
No Volume	float64	34146	0	34146	1	0
1er lot	object	34146	34146	0	1555	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
2eme lot	float64	34146	0	34146	1	0
Surface Carrez du 2eme lot	float64	34146	0	34146	1	0
3eme lot	float64	34146	0	34146	1	0
Surface Carrez du 3eme lot	float64	34146	0	34146	1	0
4eme lot	float64	34146	0	34146	1	0
Surface Carrez du 4eme lot	float64	34146	0	34146	1	0
5eme lot	float64	34146	0	34146	1	0
Surface Carrez du 5eme lot	float64	34146	0	34146	1	0
Nombre de lots	int64	34146	34146	0	14	0
Code type local	int64	34146	34146	0	2	0
Type local	object	34146	34146	0	2	0
Identifiant local	float64	34146	0	34146	1	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Nature culture	object	34146	253	33893	8	0
Nature culture speciale	object	34146	8	34138	5	0
Surface terrain	float64	34146	253	33893	201	0

Description\_Fichier\_Tableau\_avc\_ValBlanches(valeurs\_foncieries\_nettoye\_1[liste\_col\_utiles\_VF])



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
No voie	float64	34146	34013	133	1468	0
Type de voie	object	34146	33206	940	80	0
Voie	object	34146	34146	0	14130	0
B/T/Q	object	34146	2173	31973	25	0
Code postal	float64	34146	34146	0	2449	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
Type local	object	34146	34146	0	2	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Surface terrain	float64	34146	253	33893	201	0
Code departement	object	34146	34146	0	96	0
Code commune	int64	34146	34146	0	666	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Valeur fonciere	float64	34146	34146	0	9677	0

ATTENTION :

Les codes postaux sont en float mais il n'y a plus de valeur vide dans 'Code postal' qui aurait bloqué le as\_type.

Les Numéro de voie sont aussi en float.

IL FAUDRA CHANGER CELA.

ANALYSE DE VALEURS ABERRANTES :

liste\_attributs\_surface



```
['Surface Carrez du 1er lot',
'Surface reelle bati',
'Nombre pieces principales',
'Surface terrain',
'Valeur fonciere']
```

#Reduisons à l'utile

```
(valeurs_fonciere_nettoyee_1.describe(include = np.number) ).loc[['count','mean','min','max'], liste_attributs_surface]
```



	Surface Carrez du 1er lot	Surface reelle bati	Nombre pieces principales	Surface terrain	Valeur fonciere
count	34146.000000	34146.000000	34146.000000	253.000000	3.414600e+04
mean	57.651988	56.727289	2.616178	355.529644	2.528755e+05
min	0.400000	5.000000	0.000000	1.000000	3.000000e+03
max	5153.000000	379.000000	11.000000	2670.000000	9.000000e+06

### 3. Retypage de colonnes

```
Description_Fichier_Tableau_avc_ValBlanches(valeurs_fonciere_nettoyee_1[liste_col_utiles_VF])
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
No voie	float64	34146	34013	133	1468	0
Type de voie	object	34146	33206	940	80	0
Voie	object	34146	34146	0	14130	0
B/T/Q	object	34146	2173	31973	25	0
Code postal	float64	34146	34146	0	2449	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
Type local	object	34146	34146	0	2	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Surface terrain	float64	34146	253	33893	201	0
Code departement	object	34146	34146	0	96	0
Code commune	int64	34146	34146	0	666	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Valeur fonciere	float64	34146	34146	0	9677	0

```
valeurs_fonciere_nettoye_1_retype_1 = valeurs_fonciere_nettoye_1.copy(deep=True)
```

```
(valeurs_fonciere_nettoye_1_retype_1[liste_col_utiles_VF]).dtypes
```

```

No voie                float64
Type de voie           object
Voie                   object
B/T/Q                 object
Code postal            float64
Surface Carrez du 1er lot float64
Type local             object
Surface reelle bati    int64
Nombre pieces principales int64
Surface terrain        float64
Code departement       object
Code commune           int64
Date mutation          datetime64[ns]
Valeur fonciere        float64
dtype: object

```

#### a.Colonne 'Code postal'

> RETYPAGE MAL AJUSTE en int16 => code postaux négatifs !!!

```
[ ] | 18 cellules masquées
```

RETYPAGE MAL AJUSTE en int16 => code postaux négatifs !!!

> RETYPAGE du 'Code postal' en int32

```
[ ] | 11 cellules masquées
```

#### b. Colonne 'No Voie' : passage de nan à 0, et RETYPAGE en int16

> ANALYSONS :

```
[ ] | 9 cellules masquées
```

> TENTATIVES INFRUCTUEUSES

```
[ ] | 17 cellules masquées
```

TENTATIVES INFRUCTUEUSES

> APRES DE NOMBREUSES TENTATIVES INFRUCTUEUSES : remplissons tout simplement les NA des Voie par 0 :

[ ] ↳ 6 cellules masquées

> **ENFIN RETYPONS la col 'No voie' en int16 :**

[ ] ↳ 3 cellules masquées

> **TENTATIVES INFRUCTUEUSES de remettre des nan dans une colonne de int**

[ ] ↳ 5 cellules masquées

**Il semble que la valeur np.nan est incompatible avec une colonne d'entier!!!**

> REVENONS A NOTRE FICHER avec un No Voie en entier et les valeurs inconnues remplacées par 0 :

[ ] ↳ 2 cellules masquées

> **c. Colonne 'Type de voie' et 'B/T/Q' : remplaçons les valeurs vides par "**

[ ] ↳ 3 cellules masquées

> **d. Colonne 'Surface terrain' : remplaçons les valeurs videsnan par 0 et passons là en int16**

▶ ↳ 5 cellules masquées

✓ **3. Retypage de colonnes**

FICHER BILAN APRES NETTOYAGE ET APRES RETYPAGE DES COLONNES

```
valeurs_foncieres_NETTOYE_RETYPE_FINAL = valeurs_foncieres_nettoye_1_retype_6.copy(deep=True)
```

```
Description_Fichier_Tableau_avc_ValBlanches(valeurs_foncieres_NETTOYE_RETYPE_FINAL)
```



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
Code service CH	float64	34146	0	34146	1	0
Reference document	float64	34146	0	34146	1	0
1 Articles CGI	float64	34146	0	34146	1	0
2 Articles CGI	float64	34146	0	34146	1	0
3 Articles CGI	float64	34146	0	34146	1	0
4 Articles CGI	float64	34146	0	34146	1	0
5 Articles CGI	float64	34146	0	34146	1	0
No disposition	int64	34146	34146	0	3	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Nature mutation	object	34146	34146	0	1	0
Valeur fonciere	float64	34146	34146	0	9677	0
No voie	int16	34146	34146	0	1468	0
B/T/Q	object	34146	34146	0	25	31973
Code type de voie	int64	34146	34146	0	80	0
Type de voie	object	34146	34146	0	80	940
Code voie	object	34146	34146	0	5765	0
Voie	object	34146	34146	0	14130	0
Code ID commune	int64	34146	34146	0	3215	0
Code postal	int32	34146	34146	0	2449	0
Commune	object	34146	34146	0	3110	0
Code departement	object	34146	34146	0	96	0
Code commune	int64	34146	34146	0	666	0
Préfixe de section	float64	34146	1143	33003	159	0
Section	object	34146	34145	1	464	0
No plan	int64	34146	34146	0	1861	0
No Volume	float64	34146	0	34146	1	0
1er lot	object	34146	34146	0	1555	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
2eme lot	float64	34146	0	34146	1	0
Surface Carrez du 2eme lot	float64	34146	0	34146	1	0
3eme lot	float64	34146	0	34146	1	0
Surface Carrez du 3eme lot	float64	34146	0	34146	1	0
4eme lot	float64	34146	0	34146	1	0
Surface Carrez du 4eme lot	float64	34146	0	34146	1	0
5eme lot	float64	34146	0	34146	1	0
Surface Carrez du 5eme lot	float64	34146	0	34146	1	0
Nombre de lots	int64	34146	34146	0	14	0
Code type local	int64	34146	34146	0	2	0
Type local	object	34146	34146	0	2	0
Identifiant local	float64	34146	0	34146	1	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Nature culture	object	34146	253	33893	8	0
Nature culture speciale	object	34146	8	34138	5	0
Surface terrain	int16	34146	34146	0	201	0

Description\_Fichier\_Tableau\_avc\_ValBlanches(valeurs\_foncieries\_NETTOYE\_RETYPE\_FINAL[liste\_col\_utiles\_VF])

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
No voie	int16	34146	34146	0	1468	0
Type de voie	object	34146	34146	0	80	940
Voie	object	34146	34146	0	14130	0
B/T/Q	object	34146	34146	0	25	31973
Code postal	int32	34146	34146	0	2449	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
Type local	object	34146	34146	0	2	0
Surface réelle bati	int64	34146	34146	0	254	0
Nombre pièces principales	int64	34146	34146	0	12	0
Surface terrain	int16	34146	34146	0	201	0
Code département	object	34146	34146	0	96	0
Code commune	int64	34146	34146	0	666	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Valeur foncière	float64	34146	34146	0	9677	0

Commencez à coder ou à [générer](#) avec l'IA.

## V. CREATION DES TABLES

### A. premier test : Table Region

A partir du fichier référentiel géographique et sans s'être préoccupé de nettoyage pour le moment :

Mon objectif selon mon MLD est d'avoir une table Region avec :

- Id\_region : (clé étrangère) : 3 caractères exactement (reg\_id(3carac)=R+0si nec+code reg(1 à 2 car))
- Nom\_region : Libellé de la région (alphabétique)
- Code\_region : Code région (1 à 2 caractères numériques (de 0 à 94) )

```
referentiel_geographique.columns
```

```
Index(['reggrp_nom', 'reg_nom', 'reg_nom_old', 'aca_nom', 'dep_nom',
      'com_code', 'com_code1', 'com_code2', 'com_id', 'com_nom_maj_court',
      'com_nom_maj', 'com_nom', 'uu_code', 'uu_id', 'uucr_id', 'uucr_nom',
      'ze_id', 'dep_code', 'dep_id', 'dep_nom_num', 'dep_num_nom', 'aca_code',
      'aca_id', 'reg_code', 'reg_id', 'reg_code_old', 'reg_id_old', 'fd_id',
      'fr_id', 'fe_id', 'uu_id_99', 'au_code', 'au_id', 'auc_id', 'auc_nom',
      'uu_id_10', 'geolocalisation'],
      dtype='object')
```

```
Region_1 = referentiel_geographique[['reg_id','reg_nom','reg_code']]
Region_1.head()
```

```

reg_id      reg_nom      reg_code
0    R84  Auvergne-Rhône-Alpes      84
1    R84  Auvergne-Rhône-Alpes      84
2    R84  Auvergne-Rhône-Alpes      84
3    R84  Auvergne-Rhône-Alpes      84
4    R84  Auvergne-Rhône-Alpes      84
```

```
Region_2 = Region_1.rename(columns = {'reg_id':'Id_region','reg_nom':'Nom_region','reg_code':'Code_region' })
Region_2.head()
```

	<b>Id_region</b>	<b>Nom_region</b>	<b>Code_region</b>
0	R84	Auvergne-Rhône-Alpes	84
1	R84	Auvergne-Rhône-Alpes	84
2	R84	Auvergne-Rhône-Alpes	84
3	R84	Auvergne-Rhône-Alpes	84
4	R84	Auvergne-Rhône-Alpes	84

Description\_Fichier\_Tableau(Region\_2)

	<b>Type</b>	<b>Nb lignes</b>	<b>Valeurs non-vides</b>	<b>Valeurs vides</b>	<b>Valeurs distinctes</b>
<b>Id_region</b>	object	38916	38916	0	19
<b>Nom_region</b>	object	38916	38916	0	19
<b>Code_region</b>	int64	38916	38916	0	19

Region\_2['Id\_region'].unique()

```
array(['R84', 'R32', 'R93', 'R44', 'R76', 'R28', 'R75', 'R24', 'R27',
      'R53', 'R52', 'R11', 'R01', 'R02', 'R03', 'R04', 'R00', 'R06',
      'R94'], dtype=object)
```

Region\_2['Nom\_region'].unique()

```
array(['Auvergne-Rhône-Alpes', 'Hauts-de-France',
      'Provence-Alpes-Côte d'Azur', 'Grand Est', 'Occitanie',
      'Normandie', 'Nouvelle-Aquitaine', 'Centre-Val de Loire',
      'Bourgogne-Franche-Comté', 'Bretagne', 'Pays de la Loire',
      'Ile-de-France', 'Guadeloupe', 'Martinique', 'Guyane',
      'La Réunion', 'Collectivités d'outre-mer', 'Mayotte', 'Corse'],
      dtype=object)
```

Region\_2['Code\_region'].unique()

```
array([84, 32, 93, 44, 76, 28, 75, 24, 27, 53, 52, 11,  1,  2,  3,  4,  0,
       6, 94], dtype=int64)
```

ATTENTION j'ai 38916 lignes.

Je pourrais tenter l'ajout comme ça et voir si ma base de données refuse d'ajouter les nouvelles lignes mais je vais plutôt tenter de voir qu'il n'y a pas d'anomalies : pour une valeur de ID\_Region, j'ai bien toujours les mêmes valeurs après.

Je peux le tenter à la main...

Commencez à coder ou à [générer](#) avec l'IA.

OU...

Region\_2.duplicated()

```
0      False
1       True
2       True
3       True
4       True
...
38911    True
38912    True
38913    True
38914    True
38915    True
Length: 38916, dtype: bool
```

Region\_2.duplicated().value\_counts()

```
True      38897
False      19
Name: count, dtype: int64
```

Cela signifie que j'ai 19 lignes différentes et que toutes les autres sont des lignes qui sont des répétitions.

Je peux appliquer le résultat de `Region_2.duplicated()` comme un masque conditionnel pour les lignes, et conserver toutes les colonnes. mais il faut qu'il soit à l'inverse.

### Inverser un vecteur avec l'opérateur : ~

```
~Region_2.duplicated()
```

```
0      True
1     False
2     False
3     False
4     False
...
38911  False
38912  False
38913  False
38914  False
38915  False
Length: 38916, dtype: bool
```

```
(~Region_2.duplicated() ).value_counts()
```

```
False    38897
True       19
Name: count, dtype: int64
```

### ✓ SUPPRESSION des doublons avec un masque

Je me sers donc de `~Region_2.duplicated()` comme masque sur les lignes et conserve toutes les colonnes

```
Region_3 = Region_2.loc[~Region_2.duplicated() , :]
display(Region_3)
```

```

   Id_region  Nom_region  Code_region
0         R84  Auvergne-Rhône-Alpes      84
459        R32    Hauts-de-France      32
1614       R93  Provence-Alpes-Côte d'Azur     93
2555       R44           Grand Est       44
3058       R76           Occitanie       76
4728       R28           Normandie       28
5761       R75  Nouvelle-Aquitaine       75
6672       R24    Centre-Val de Loire       24
7251       R27  Bourgogne-Franche-Comté     27
7968       R53           Bretagne       53
17259      R52    Pays de la Loire       52
31754      R11    Ile-de-France       11
38306      R01           Guadeloupe        1
38340      R02           Martinique        2
38374      R03           Guyane          3
38400      R04           La Réunion         4
38424      R00  Collectivités d'outre-mer      0
38426      R06           Mayotte          6
38550      R94           Corse          94
```

```
Region_4 = (Region_3.sort_values(by='Id_region', ascending=True)).reset_index().drop(columns='index')
display(Region_4)
```





	<b>Id_region</b>	<b>Nom_region</b>	<b>Code_region</b>
<b>0</b>	R00	Collectivités d'outre-mer	0
<b>1</b>	R01	Guadeloupe	1
<b>2</b>	R02	Martinique	2
<b>3</b>	R03	Guyane	3
<b>4</b>	R04	La Réunion	4
<b>5</b>	R06	Mayotte	6
<b>6</b>	R11	Ile-de-France	11
<b>7</b>	R24	Centre-Val de Loire	24
<b>8</b>	R27	Bourgogne-Franche-Comté	27
<b>9</b>	R28	Normandie	28
<b>10</b>	R32	Hauts-de-France	32
<b>11</b>	R44	Grand Est	44
<b>12</b>	R52	Pays de la Loire	52
<b>13</b>	R53	Bretagne	53
<b>14</b>	R75	Nouvelle-Aquitaine	75
<b>15</b>	R76	Occitanie	76
<b>16</b>	R84	Auvergne-Rhône-Alpes	84
<b>17</b>	R93	Provence-Alpes-Côte d'Azur	93
<b>18</b>	R94	Corse	94

#### ✓ SUPPRESSION des doublons avec la fonction `drop_duplicates`

```
Region_3bis =Region_2.drop_duplicates(keep='first')  
display(Region_3bis)
```



	<b>Id_region</b>	<b>Nom_region</b>	<b>Code_region</b>
<b>0</b>	R84	Auvergne-Rhône-Alpes	84
<b>459</b>	R32	Hauts-de-France	32
<b>1614</b>	R93	Provence-Alpes-Côte d'Azur	93
<b>2555</b>	R44	Grand Est	44
<b>3058</b>	R76	Occitanie	76
<b>4728</b>	R28	Normandie	28
<b>5761</b>	R75	Nouvelle-Aquitaine	75
<b>6672</b>	R24	Centre-Val de Loire	24
<b>7251</b>	R27	Bourgogne-Franche-Comté	27
<b>7968</b>	R53	Bretagne	53
<b>17259</b>	R52	Pays de la Loire	52
<b>31754</b>	R11	Ile-de-France	11
<b>38306</b>	R01	Guadeloupe	1
<b>38340</b>	R02	Martinique	2
<b>38374</b>	R03	Guyane	3
<b>38400</b>	R04	La Réunion	4
<b>38424</b>	R00	Collectivités d'outre-mer	0
<b>38426</b>	R06	Mayotte	6
<b>38550</b>	R94	Corse	94

```
Region_4bis = (Region_3bis.sort_values(by='Id_region', ascending=True)).reset_index().drop(columns='index')  
display(Region_4bis)
```

	<b>Id_region</b>	<b>Nom_region</b>	<b>Code_region</b>
0	R00	Collectivités d'outre-mer	0
1	R01	Guadeloupe	1
2	R02	Martinique	2
3	R03	Guyane	3
4	R04	La Réunion	4
5	R06	Mayotte	6
6	R11	Ile-de-France	11
7	R24	Centre-Val de Loire	24
8	R27	Bourgogne-Franche-Comté	27
9	R28	Normandie	28
10	R32	Hauts-de-France	32
11	R44	Grand Est	44
12	R52	Pays de la Loire	52
13	R53	Bretagne	53
14	R75	Nouvelle-Aquitaine	75
15	R76	Occitanie	76
16	R84	Auvergne-Rhône-Alpes	84
17	R93	Provence-Alpes-Côte d'Azur	93
18	R94	Corse	94

Description\_Fichier\_Tableau(Region\_4bis)

	<b>Type</b>	<b>Nb lignes</b>	<b>Valeurs non-vides</b>	<b>Valeurs vides</b>	<b>Valeurs distinctes</b>
<b>Id_region</b>	object	19	19	0	19
<b>Nom_region</b>	object	19	19	0	19
<b>Code_region</b>	int64	19	19	0	19

Autant de lignes que de valeurs distinctes, ça veut dire que chaque ligne est différente et aucune valeur n'est répétée.

Region\_4bis.duplicated().value\_counts()

```
False    19
Name: count, dtype: int64
```

Aucun true => pas de valeur répétées

```
print('il y a ', Region_4bis.duplicated().sum(), 'répétitions de lignes')
```

```
il y a 0 répétitions de lignes
```

```
Region_final = Region_4bis.copy(deep=True)
```

```
Region_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19 entries, 0 to 18
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Id_region    19 non-null    object
1   Nom_region   19 non-null    object
2   Code_region  19 non-null    int64
dtypes: int64(1), object(2)
memory usage: 588.0+ bytes
```

## ✚ EXPORTONS MAINTENANT LE FICHIER travaillé

```
Region_final.to_csv('Region.csv', index=False)
```

```
Region_final.to_excel('Region.xlsx',index=False)
```

## ✓ B. premier test : Table Departement

A partir du fichier referentiel géographique et sans s'être préoccupé de nettoyage pour le moment :

Mon objectif selon mon MLD est d'avoir une table Departement avec :

- Id\_dep : l'identifiant du département : 4 caractères exactement (dep\_id (4carac)=D+0 ou 00 si nec+code dep(1 à 3 carac))
- Nom\_departement : Libellé du département (alphabétique)
- Code\_departement : Code département (1 a 3 caract alphanum 1 , 60, 2A , 989 possible)
- Id\_region : (clé étrangère) : 3 caractères exactement (reg\_id(3carac)=R+0si nec+code reg(1 à 2 car))

```
referentiel_geographique.columns
```

```
Index(['reggrp_nom', 'reg_nom', 'reg_nom_old', 'aca_nom', 'dep_nom',
      'com_code', 'com_code1', 'com_code2', 'com_id', 'com_nom_maj_court',
      'com_nom_maj', 'com_nom', 'uu_code', 'uu_id', 'uucr_id', 'uucr_nom',
      'ze_id', 'dep_code', 'dep_id', 'dep_nom_num', 'dep_num_nom', 'aca_code',
      'aca_id', 'reg_code', 'reg_id', 'reg_code_old', 'reg_id_old', 'fd_id',
      'fr_id', 'fe_id', 'uu_id_99', 'au_code', 'au_id', 'auc_id', 'auc_nom',
      'uu_id_10', 'geolocalisation'],
      dtype='object')
```

#Restreignons aux colonnes qui nous seront utiles :

```
Departement_1 = referentiel_geographique[['dep_id','dep_nom','dep_code','reg_id']]
Departement_1.head()
```

```

dep_id  dep_nom  dep_code  reg_id
0    D001    Ain         1    R84
1    D001    Ain         1    R84
2    D001    Ain         1    R84
3    D001    Ain         1    R84
4    D001    Ain         1    R84
```

```
Departement_2 = Departement_1.rename(columns = {'dep_id':'Id_departement',
      'dep_nom':'Nom_departement',
      'dep_code':'Code_departement',
      'reg_id':'Id_region'})
```

```
Departement_2.head()
```

```

Id_departement  Nom_departement  Code_departement  Id_region
0             D001             Ain                1         R84
1             D001             Ain                1         R84
2             D001             Ain                1         R84
3             D001             Ain                1         R84
4             D001             Ain                1         R84
```

```
Description_Fichier_Tableau(Departement_2)
```

```

Type  Nb lignes  Valeurs non-vides  Valeurs vides  Valeurs distinctes
Id_departement  object    38916             38916             0             109
Nom_departement  object    38916             38916             0             109
Code_departement  object    38916             38916             0             109
Id_region         object    38916             38916             0              19
```

J'ai 38916 lignes, mais a priori 109 valeurs distinctes pour tout ce qui est département. Ce qui est plutôt rassurant et 19 regions ce qui est cohérent avec notre fichier région. Il faudra vérifier la cohérence des données. Et également, nous n'avons besoin que de garder les lignes qui sont distinctes, en espérant qu'il y aura bien 109 lignes distinctes et pas de croisement de noms et d'Id par exemple !!!

Il faut aussi que les valeurs pour Id\_région soient cohérentes avec nos valeurs dans la table Region. Cependant comme j'ai créé les valeurs Id\_region à partir de la même colonne sans transformation nécessaire, cela devrait être ok.

```
Departement_2.Id_departement.unique()
```

```
array(['D001', 'D002', 'D003', 'D004', 'D005', 'D006', 'D007', 'D008',  
      'D009', 'D010', 'D011', 'D012', 'D013', 'D014', 'D015', 'D016',  
      'D017', 'D018', 'D019', 'D021', 'D022', 'D023', 'D024', 'D025',  
      'D026', 'D027', 'D028', 'D029', 'D030', 'D031', 'D032', 'D033',  
      'D034', 'D035', 'D036', 'D037', 'D038', 'D039', 'D040', 'D041',  
      'D042', 'D043', 'D044', 'D045', 'D046', 'D047', 'D048', 'D049',  
      'D050', 'D051', 'D052', 'D053', 'D054', 'D055', 'D056', 'D057',  
      'D058', 'D059', 'D060', 'D061', 'D062', 'D063', 'D064', 'D065',  
      'D066', 'D067', 'D068', 'D069', 'D070', 'D071', 'D072', 'D073',  
      'D074', 'D075', 'D076', 'D077', 'D078', 'D079', 'D080', 'D081',  
      'D082', 'D083', 'D084', 'D085', 'D086', 'D087', 'D088', 'D089',  
      'D090', 'D091', 'D092', 'D093', 'D094', 'D095', 'D097', 'D0972',  
      'D973', 'D974', 'D975', 'D976', 'D977', 'D978', 'D984', 'D986',  
      'D987', 'D988', 'D989', 'D02A', 'D02B'], dtype=object)
```

Je vais observer combien j'ai de lignes complètement identiques :

```
Departement_2.duplicated()
```

```
0      False  
1       True  
2       True  
3       True  
4       True  
...  
38911   True  
38912   True  
38913   True  
38914   True  
38915   True  
Length: 38916, dtype: bool
```

```
Departement_2.duplicated().value_counts()
```

```
True      38807  
False      109  
Name: count, dtype: int64
```

```
type(Departement_2.duplicated().value_counts())
```

```
pandas.core.series.Series
```

```
Departement_2.duplicated().value_counts()[False]
```

```
109
```

```
print("Nous avons donc dans ce DataFrame exactement : ",  
      Departement_2.duplicated().value_counts()[False],  
      "valeurs distinctes")
```

```
Nous avons donc dans ce DataFrame exactement : 109 valeurs distinctes
```

## ✓ SUPPRESSION des doublons avec la fonction `drop_duplicates`

```
Departement_3bis = Departement_2.drop_duplicates(keep='first')  
display(Departement_3bis)
```

	Id_departement	Nom_departement	Code_departement	Id_region
0	D001	Ain	1	R84
459	D002	Aisne	2	R32
1293	D003	Allier	3	R84
1614	D004	Alpes-de-Haute-Provence	4	R93
1859	D005	Hautes-Alpes	5	R93
...	...	...	...	...
38467	D987	Polynésie Française	987	R00
38516	D988	Nouvelle-Calédonie	988	R00
38549	D989	Ile de Clipperton	989	R00
38550	D02A	Corse-du-Sud	2A	R94
38677	D02B	Haute-Corse	2B	R94

109 rows × 4 columns

```

Departement_4bis = (Departement_3bis.sort_values(by='Id_departement', ascending=True)
                    ).reset_index().drop(columns='index')
display(Departement_4bis)

```

	Id_departement	Nom_departement	Code_departement	Id_region
0	D001	Ain	1	R84
1	D002	Aisne	2	R32
2	D003	Allier	3	R84
3	D004	Alpes-de-Haute-Provence	4	R93
4	D005	Hautes-Alpes	5	R93
...	...	...	...	...
104	D984	Terres australes et antarctiques françaises	984	R00
105	D986	Wallis et Futuna	986	R00
106	D987	Polynésie Française	987	R00
107	D988	Nouvelle-Calédonie	988	R00
108	D989	Ile de Clipperton	989	R00

109 rows × 4 columns

```

Description_Fichier_Tableau(Departement_4bis)

```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Id_departement	object	109	109	0	109
Nom_departement	object	109	109	0	109
Code_departement	object	109	109	0	109
Id_region	object	109	109	0	19

Autant de lignes que de valeurs distinctes, ça veut dire que chaque ligne est différente et aucune valeur n'est répétée.

```

Departement_4bis.duplicated().value_counts()

```

```

False    109
Name: count, dtype: int64

```

Aucun true => pas de valeur répétées

```

print('il y a ', Departement_4bis.duplicated().sum(), 'répétitions de lignes')

```

```

il y a 0 répétitions de lignes

```

```

Departement_final = Departement_4bis.copy(deep=True)

```

```

Departement_final.info()

```

```

↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 109 entries, 0 to 108
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Id_departement        109 non-null    object
1   Nom_departement       109 non-null    object
2   Code_departement      109 non-null    object
3   Id_region             109 non-null    object
dtypes: object(4)
memory usage: 3.5+ KB

```

## ✧ EXPORTONS MAINTENANT LE FICHER travaillé

```
Departement_final.to_csv('Departement.csv',index=False)
```

```
Departement_final.to_excel('Departement.xlsx',index=False)
```

## ✧ C. Table Commune

Mon objectif selon mon MLD est d'avoir une table Commune avec :

- `Id_codedep_codecommune` : (com\_id ds ref geo)/ ('C' + 2 premiers chiffres(CODDEP) + CODCOM ds donnees\_communes) /('C'+ 2 premiers chiffres(Code departement) + Code commune ds valeurs\_foncieres) : l'identifiant de la commune : 6 caractères exactement  
com\_id (6carac)=C+0si nec+com\_code1(4à5carac) de la forme CXXXXXX C2AXXX possib C974XX possib  
ex :

```

C01275 = C+01+275 (dep01/com275)
C34172 = C+34+172 (dep34/com172)
C2A362 = C+2A+363 (dep2A/com363)
C97424 = C+97+424 (dep974/com424) exception dep à 3 chiffres

```

- `Nom_commune` : (COM ds donnees\_communes) : Nom de commune
- `Code_commune` : (CODCOM ds donnees\_communes) : Code Commune : numérique sur 3 chiffres exactement
- `Population_totale` : (PTOT ds donnees\_communes) : Population totale : entiers ente 0 et 500 000
- `Id_dep CLE ETRANGERE` : l'identifiant du département : 4 caractères exactement (dep\_id (4carac)=D+0 ou 00 si nec+code dep(1 à 3 carac))

Au vu de la répartition des données :

1. Récupérons dans `donnees_communes` :

- le code departement
- le code commune
- le nom de la commune
- la population totale
- qui nous permettront de construire également l'`Id_codedep_codecommune`
- et l'`Id_departement`

2. ANNULE : Je n'ai pas besoin du code postal pour cette étude. De plus il y a un problème d'homogénéité, car il y a plusieurs code postal pour une même `Id_codedep_codecommune` . Soit il faudrait avori une liste de code postal possible pour un `Id_codedep_codecommune`, soit plus simple il faudrait mettre le Code postal dans Bien, soit dans commune il faut faire un autre choix d'identifiant !

```
donnees_communes.columns
```

```

↳ Index(['CODREG', 'CODDEP', 'CODARR', 'CODCAN', 'CODCOM', 'COM', 'PMUN', 'PCAP',
        'PTOT'],
        dtype='object')

```

```
donnees_communes.duplicated().value_counts()
```

```

↳ False    34991
   Name: count, dtype: int64

```

```

Commune_1 = donnees_communes[['COM', 'CODCOM', 'PTOT', 'CODDEP']]
Commune_1.head()

```



	COM	CODCOM	PTOT	CODDEP
0	L'Abergement-Clémenciat	1	798	01
1	L'Abergement-de-Varey	2	257	01
2	Ambérieu-en-Bugey	4	14514	01
3	Ambérieux-en-Dombes	5	1776	01
4	Ambléon	6	118	01

Description\_Fichier\_Tableau(Commune\_1)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
COM	object	34991	34991	0	32732
CODCOM	int64	34991	34991	0	908
PTOT	int64	34991	34991	0	5931
CODDEP	object	34991	34991	0	100

Commencez à coder ou à [générer](#) avec l'IA.

vérifions les formats des codes communes et des codedep

## ✓ PASSONS LA COLONNE CODDEP en STRING

Commune\_1.CODDEP.unique()



```
array(['01', '02', '03', '04', '05', '06', '07', '08', '09', 10, 11, 12,
      13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29,
      '2A', '2B', 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
      44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
      61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
      78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
      95, 971, 972, 973, 974], dtype=object)
```

Remarque : il y a des chiffres et des caractères

Commune\_2 = Commune\_1.astype({'CODDEP': 'string'})

Commune\_2.CODDEP.unique()



```
<StringArray>
[ '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11',
  '12', '13', '14', '15', '16', '17', '18', '19', '21', '22', '23',
  '24', '25', '26', '27', '28', '29', '2A', '2B', '30', '31', '32',
  '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43',
  '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54',
  '55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65',
  '66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76',
  '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87',
  '88', '89', '90', '91', '92', '93', '94', '95', '971', '972', '973',
  '974']
Length: 100, dtype: string
```

## ✓ PASSONS LA COLONNE CODCOM en STRING et reformatons là sur 3 caractères

Commune\_1.CODCOM.unique()



```
array([ 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
      15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29,
      30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
      43, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 56, 57,
      58, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 71, 72,
      73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
      87, 88, 89, 90, 92, 93, 94, 95, 96, 98, 99, 100, 101,
      102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
      115, 116, 117, 118, 121, 123, 124, 125, 127, 128, 129, 130, 133,
      134, 135, 136, 138, 139, 140, 141, 142, 143, 145, 146, 147, 148,
      149, 150, 151, 152, 153, 155, 156, 157, 158, 159, 160, 162, 163,
      165, 166, 167, 169, 170, 171, 173, 174, 175, 177, 179, 180, 181,
      183, 184, 185, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196,
      197, 198, 199, 200, 202, 203, 204, 206, 207, 208, 209, 210, 211,
      212, 213, 214, 215, 216, 219, 224, 225, 227, 228, 229, 230, 231,
```

232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244,  
 245, 246, 247, 248, 249, 250, 252, 254, 255, 257, 258, 259, 260,  
 261, 262, 263, 264, 265, 266, 267, 268, 269, 272, 273, 274, 275,  
 276, 277, 279, 280, 281, 282, 283, 284, 285, 286, 288, 289, 290,  
 291, 293, 294, 295, 296, 297, 298, 299, 301, 302, 303, 304, 305,  
 306, 307, 308, 309, 310, 311, 313, 314, 317, 318, 319, 320, 321,  
 322, 323, 325, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,  
 338, 339, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352,  
 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365,  
 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 378, 379,  
 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392,  
 393, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407,  
 408, 410, 411, 412, 415, 416, 418, 419, 420, 421, 422, 423, 424,  
 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437,  
 439, 441, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453,  
 454, 456, 457, 3, 18, 20, 48, 55, 59, 70, 86, 91, 97,  
 119, 120, 122, 126, 131, 132, 137, 144, 154, 164, 168, 172, 176,  
 178, 182, 186, 201, 205, 217, 218, 220, 221, 222, 223, 226, 251,  
 253, 256, 270, 271, 278, 287, 292, 312, 315, 316, 324, 326, 327,  
 340, 341, 377, 395, 409, 413, 414, 417, 438, 440, 442, 455, 458,  
 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471,  
 472, 473, 474, 476, 477, 478, 480, 481, 482, 483, 484, 485, 486,  
 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499,  
 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512,  
 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525,  
 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538,  
 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551,  
 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564,  
 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577,  
 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590,  
 591, 592, 593, 594, 595, 596, 598, 599, 600, 601, 602, 604, 605,  
 606, 607, 608, 609, 610, 612, 613, 614, 615, 616, 617, 618, 619,  
 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 631, 632, 633,  
 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 647,  
 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660,  
 661, 662, 663, 664, 665, 666, 667, 668, 670, 671, 672, 673, 674,  
 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687,  
 688, 689, 690, 691, 693, 694, 695, 696, 697, 698, 699, 701, 702,  
 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715,  
 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728,  
 729, 730, 731, 732, 734, 735, 736, 737, 738, 739, 740, 741, 742,  
 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755,  
 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768.

là je vais avoir un problème car je veux que les codes communes soient sur 3 chiffres automatiquement !  
 De plus c'est étonnant car dans le fichier initial les chiffres étaient obligatoirement sur 3 chiffres significatifs.

```
Commune_3 = Commune_2[:]
```

```
Commune_3 = Commune_3.astype({'CODCOM':'string'})
```

```
Commune_3.CODCOM.unique()
```

```
<StringArray>
[ '1', '2', '4', '5', '6', '7', '8', '9', '10', '11',
...
'900', '901', '902', '903', '904', '905', '906', '907', '908', '909']
Length: 908, dtype: string
```

```
Commune_3['CODCOM'] = ('00'+Commune_3['CODCOM'])
```

```
Commune_3.CODCOM.unique()
```

```
<StringArray>
[ '001', '002', '004', '005', '006', '007', '008', '009',
'0010', '0011',
...
'00900', '00901', '00902', '00903', '00904', '00905', '00906', '00907',
'00908', '00909']
Length: 908, dtype: string
```

```
a='001'
b='00901'
print(a[-3:])
print(b[-3:])
```

```
001
901
```

#le slice ne s'applique pas terme à terme sur la Series.  
 #Ca fait un slice de la Series, et ça raccourci la dimension de la Series au lieu de raccourcir les termes de la Series !



```
#Commune_3['CODCOM'] = Commune_3['CODCOM'][-3:]
```

```
#Commune_3.CODCOM.unique()
```

Je vais donc le faire terme à terme :

```
#Je pense qu'il y a eu un gros problèmes logique :
#l'opération a été longue, et o la fin j'ai 1807 lignes au lieu de 908, et j'ai des 0010;
#en fait je pense que en modifiant les éléments sur lesquesl j'itérais la boucle est revenu sur des élément suq'elle avait déjà vu.
#il faut que je crée une nouvelle colonne poru ne pas faire ça
#for elt in Commune_3['CODCOM'] :
#     Commune_3.loc[elt,'CODCOM'] = elt[-3:]
```

```
Commune_3.CODCOM.unique()
```

```
<StringArray>
[ '001', '002', '004', '005', '006', '007', '008', '009',
  '0010', '0011',
  ...
  '00900', '00901', '00902', '00903', '00904', '00905', '00906', '00907',
  '00908', '00909']
Length: 908, dtype: string
```

```
Commune_3.loc[Commune_3['CODCOM']=='001',:]
```

	COM	CODCOM	PTOT	CODDEP
0	L'Abergement-Clémenciat	001	798	01
393	Abbécourt	001	513	02
1193	Abrest	001	2964	03
1510	Aiglun	001	1460	04
1708	Abriès-Ristolas	001	389	05
...	...	...	...	...
33346	Les Ableuvenettes	001	58	88
34276	Andelnans	001	1192	90
34377	Abbeville-la-Rivière	001	332	91
34607	Aubervilliers	001	89139	93
34647	Ablon-sur-Seine	001	5879	94

86 rows x 4 columns

```
for elt in Commune_3['CODCOM'].unique() :
    Commune_3.loc[Commune_3['CODCOM']==elt,'New_CODCOM'] = elt[-3:]
```

```
Commune_3['New_CODCOM'].unique()
```

```
array(['001', '002', '004', '005', '006', '007', '008', '009', '010',
       '011', '012', '013', '014', '015', '016', '017', '019', '021',
       '022', '023', '024', '025', '026', '027', '028', '029', '030',
       '031', '032', '033', '034', '035', '036', '037', '038', '039',
       '040', '041', '042', '043', '044', '045', '046', '047', '049',
       '050', '051', '052', '053', '054', '056', '057', '058', '060',
       '061', '062', '063', '064', '065', '066', '067', '068', '069',
       '071', '072', '073', '074', '075', '076', '077', '078', '079',
       '080', '081', '082', '083', '084', '085', '087', '088', '089',
       '090', '092', '093', '094', '095', '096', '098', '099', '100',
       '101', '102', '103', '104', '105', '106', '107', '108', '109',
       '110', '111', '112', '113', '114', '115', '116', '117', '118',
       '121', '123', '124', '125', '127', '128', '129', '130', '133',
       '134', '135', '136', '138', '139', '140', '141', '142', '143',
       '145', '146', '147', '148', '149', '150', '151', '152', '153',
       '155', '156', '157', '158', '159', '160', '162', '163', '165',
       '166', '167', '169', '170', '171', '173', '174', '175', '177',
       '179', '180', '181', '183', '184', '185', '187', '188', '189',
       '190', '191', '192', '193', '194', '195', '196', '197', '198',
       '199', '200', '202', '203', '204', '206', '207', '208', '209',
       '210', '211', '212', '213', '214', '215', '216', '219', '224',
       '225', '227', '228', '229', '230', '231', '232', '233', '234',
       '235', '236', '237', '238', '239', '240', '241', '242', '243',
       '244', '245', '246', '247', '248', '249', '250', '252', '254',
       '255', '257', '258', '259', '260', '261', '262', '263', '264',
       '265', '266', '267', '268', '269', '272', '273', '274', '275',
       '276', '277', '279', '280', '281', '282', '283', '284', '285',
```

'286', '288', '289', '290', '291', '293', '294', '295', '296',  
 '297', '298', '299', '301', '302', '303', '304', '305', '306',  
 '307', '308', '309', '310', '311', '313', '314', '317', '318',  
 '319', '320', '321', '322', '323', '325', '328', '329', '330',  
 '331', '332', '333', '334', '335', '336', '337', '338', '339',  
 '342', '343', '344', '345', '346', '347', '348', '349', '350',  
 '351', '352', '353', '354', '355', '356', '357', '358', '359',  
 '360', '361', '362', '363', '364', '365', '366', '367', '368',  
 '369', '370', '371', '372', '373', '374', '375', '376', '378',  
 '379', '380', '381', '382', '383', '384', '385', '386', '387',  
 '388', '389', '390', '391', '392', '393', '396', '397', '398',  
 '399', '400', '401', '402', '403', '404', '405', '406', '407',  
 '408', '410', '411', '412', '415', '416', '418', '419', '420',  
 '421', '422', '423', '424', '425', '426', '427', '428', '429',  
 '430', '431', '432', '433', '434', '435', '436', '437', '439',  
 '441', '443', '444', '445', '446', '447', '448', '449', '450',  
 '451', '452', '453', '454', '456', '457', '003', '018', '020',  
 '048', '055', '059', '070', '086', '091', '097', '119', '120',  
 '122', '126', '131', '132', '137', '144', '154', '164', '168',  
 '172', '176', '178', '182', '186', '201', '205', '217', '218',  
 '220', '221', '222', '223', '226', '251', '253', '256', '270',  
 '271', '278', '287', '292', '312', '315', '316', '324', '326',  
 '327', '340', '341', '377', '395', '409', '413', '414', '417',  
 '438', '440', '442', '455', '458', '459', '460', '461', '462',  
 '463', '464', '465', '466', '467', '468', '469', '470', '471',  
 '472', '473', '474', '476', '477', '478', '480', '481', '482',  
 '483', '484', '485', '486', '487', '488', '489', '490', '491',  
 '492', '493', '494', '495', '496', '497', '498', '499', '500',  
 '501', '502', '503', '504', '505', '506', '507', '508', '509',  
 '510', '511', '512', '513', '514', '515', '516', '517', '518',  
 '519', '520', '521', '522', '523', '524', '525', '526', '527',

Description\_Fichier\_Tableau(Commune\_3)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
<b>COM</b>	object	34991	34991	0	32732
<b>CODCOM</b>	string[python]	34991	34991	0	908
<b>PTOT</b>	int64	34991	34991	0	5931
<b>CODDEP</b>	string[python]	34991	34991	0	100
<b>New_CODCOM</b>	object	34991	34991	0	908

**DEBARRASSONS NOUS de la colonne CODCOM, et Obtenons la colonne Id\_codedep\_codecommune**

Commune\_4 = Commune\_3[:]

del Commune\_4['CODCOM']

Description\_Fichier\_Tableau(Commune\_4)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
<b>COM</b>	object	34991	34991	0	32732
<b>PTOT</b>	int64	34991	34991	0	5931
<b>CODDEP</b>	string[python]	34991	34991	0	100
<b>New_CODCOM</b>	object	34991	34991	0	908

**Obtenons la colonne Id\_codedep\_codecommune**

Je vais construire la colonne élément par élément

car je ne peux pas malheureusement pas faire un slice directement sur la colonne Commune\_4['CODDEP'] pour ne garder que les premiers termes à chaque fois.

Si je fais un slice, il va slicer les lignes (et ne garder que les 2 dernières lignes)

au lieux de slicer chaque terme de la colonne.

Du coup je ne peux faire des opérations sur les colonnes comme c'est le plus pratique à faire !

for elt in Commune\_4.index :

```
Commune_4.loc[elt, 'Id_codedep_codecommune'] = ('C'+
  (Commune_4.loc[elt, 'CODDEP'])[0:2]+
  Commune_4.loc[elt, 'New_CODCOM']
)
```

Description\_Fichier\_Tableau(Commune\_4)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
COM	object	34991	34991	0	32732
PTOT	int64	34991	34991	0	5931
CODDEP	string[python]	34991	34991	0	100
New_CODCOM	object	34991	34991	0	908
Id_codedep_codecommune	object	34991	34991	0	34991

## CONSTRUISONS LA COLONNE Id\_département :

Commune\_4.CODDEP.unique()



```
<StringArray>
[ '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11',
  '12', '13', '14', '15', '16', '17', '18', '19', '21', '22', '23',
  '24', '25', '26', '27', '28', '29', '2A', '2B', '30', '31', '32',
  '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43',
  '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54',
  '55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65',
  '66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76',
  '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87',
  '88', '89', '90', '91', '92', '93', '94', '95', '971', '972', '973',
  '974']
Length: 100, dtype: string
```

POur rappel l'Id\_département est sur 4 caractères c'est D+0si nec+code dep

Commune\_5 = Commune\_4[:]

for elt in Commune\_5.index :

```
Commune_5.loc[elt, 'Id_département'] = ('D'+
                                         ('00'+Commune_5.loc[elt, 'CODDEP'])[-3:])
)
```



C:\Users\matth\AppData\Local\Temp\ipykernel\_36580\2480043137.py:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
Commune\_5.loc[elt, 'Id\_département'] = ('D'+

Description\_Fichier\_Tableau(Commune\_5)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
COM	object	34991	34991	0	32732
PTOT	int64	34991	34991	0	5931
CODDEP	string[python]	34991	34991	0	100
New_CODCOM	object	34991	34991	0	908
Id_codedep_codecommune	object	34991	34991	0	34991
Id_département	object	34991	34991	0	100

Commune\_5['Id\_département'].unique()



```
array(['D001', 'D002', 'D003', 'D004', 'D005', 'D006', 'D007', 'D008',
      'D009', 'D010', 'D011', 'D012', 'D013', 'D014', 'D015', 'D016',
      'D017', 'D018', 'D019', 'D021', 'D022', 'D023', 'D024', 'D025',
      'D026', 'D027', 'D028', 'D029', 'D02A', 'D02B', 'D030', 'D031',
      'D032', 'D033', 'D034', 'D035', 'D036', 'D037', 'D038', 'D039',
      'D040', 'D041', 'D042', 'D043', 'D044', 'D045', 'D046', 'D047',
      'D048', 'D049', 'D050', 'D051', 'D052', 'D053', 'D054', 'D055',
      'D056', 'D057', 'D058', 'D059', 'D060', 'D061', 'D062', 'D063',
      'D064', 'D065', 'D066', 'D067', 'D068', 'D069', 'D070', 'D071',
      'D072', 'D073', 'D074', 'D075', 'D076', 'D077', 'D078', 'D079',
      'D080', 'D081', 'D082', 'D083', 'D084', 'D085', 'D086', 'D087',
      'D088', 'D089', 'D090', 'D091', 'D092', 'D093', 'D094', 'D095',
      'D971', 'D972', 'D973', 'D974'], dtype=object)
```

Il ne reste plus que le code postal à récupérer dans le fichier valeur

## TRIONS ET RENOMMONS DEJA NOS COLONNES :

```
Commune_5.head()
```



	COM	PTOT	CODDEP	New_CODCOM	Id_codedep_codecommune	Id_departement
0	L'Abergement-Clémenciat	798	01	001	C01001	D001
1	L'Abergement-de-Varey	257	01	002	C01002	D001
2	Ambérieu-en-Bugey	14514	01	004	C01004	D001
3	Ambérieux-en-Dombes	1776	01	005	C01005	D001
4	Ambléon	118	01	006	C01006	D001

```
Commune_6 = Commune_5.rename(columns ={'COM' : 'Nom_commune',
                                         'New_CODCOM' : 'Code_commune',
                                         'PTOT':'Population_totale',
                                         })
```

```
Commune_6.head()
```



	Nom_commune	Population_totale	CODDEP	Code_commune	Id_codedep_codecommune	Id_departement
0	L'Abergement-Clémenciat	798	01	001	C01001	D001
1	L'Abergement-de-Varey	257	01	002	C01002	D001
2	Ambérieu-en-Bugey	14514	01	004	C01004	D001
3	Ambérieux-en-Dombes	1776	01	005	C01005	D001
4	Ambléon	118	01	006	C01006	D001

```
del Commune_6['CODDEP']
Commune_6.head()
```



	Nom_commune	Population_totale	Code_commune	Id_codedep_codecommune	Id_departement
0	L'Abergement-Clémenciat	798	001	C01001	D001
1	L'Abergement-de-Varey	257	002	C01002	D001
2	Ambérieu-en-Bugey	14514	004	C01004	D001
3	Ambérieux-en-Dombes	1776	005	C01005	D001
4	Ambléon	118	006	C01006	D001

```
#NE MARCHE PAS
#Commune_6.reindex(['Id_codedep_codecommune', 'Nom_commune',
#                   'Code_commune', 'Population_totale','Id_departement'],
#                   axis=1
#                   )
#Commune_6.head()
```

```
Commune_6 = Commune_6[['Id_codedep_codecommune', 'Nom_commune',
                       'Code_commune', 'Population_totale','Id_departement']]
```

```
Commune_6.head()
```



	Id_codedep_codecommune	Nom_commune	Code_commune	Population_totale	Id_departement
0	C01001	L'Abergement-Clémenciat	001	798	D001
1	C01002	L'Abergement-de-Varey	002	257	D001
2	C01004	Ambérieu-en-Bugey	004	14514	D001
3	C01005	Ambérieux-en-Dombes	005	1776	D001
4	C01006	Ambléon	006	118	D001

Il ne reste plus que le code postal à récupérer dans le fichier valeur

FINALEMENT NON : Après Analyse du fichier, l'ajout du code postal est problématique car plusieurs code postaux sont possibles pour une même Id\_codedep\_codecommune .

De plus le code postal n'est pas nécessaire pour résoudre les requêtes demandées.  
Je n'ajoute donc pas le Code postal dans mon schéma.

Classons les données par Id croissant

```
Commune_7 = Commune_6.sort_values(by='Id_codedep_codecommune',ascending=True)
Commune_7.head()
```

	Id_codedep_codecommune	Nom_commune	Code_commune	Population_totale	Id_departement
0	C01001	L'Abergement-Clémenciat	001	798	D001
1	C01002	L'Abergement-de-Varey	002	257	D001
2	C01004	Ambérieu-en-Bugey	004	14514	D001
3	C01005	Ambérieux-en-Dombes	005	1776	D001
4	C01006	Ambléon	006	118	D001

```
Commune_final = Commune_7.copy(deep=True)
```

```
Commune_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34991 entries, 0 to 34990
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Id_codedep_codecommune 34991 non-null  object
1   Nom_commune            34991 non-null  object
2   Code_commune           34991 non-null  object
3   Population_totale      34991 non-null  int64
4   Id_departement         34991 non-null  object
dtypes: int64(1), object(4)
memory usage: 1.3+ MB
```

## ✓ EXPORTONS MAINTENANT LE FICHER travaillé

```
Commune_final.to_csv('Commune.csv',index=False)
```

```
Commune_final.to_excel('Commune.xlsx',index=False)
```

## ✓ D. Table Bien

Mon objectif selon mon MLD est d'avoir une table Bien avec :

- Id\_bien : A CREER : (probablement à la fin avec les lignes d'index tenant compte des doublons)
- No\_voie : ( 'No voie' ds valeurs\_foncieres ) : Numéro des rues
- Type\_de\_voie : ( 'Type de voie' ds valeurs\_foncieres ) : Plusieurs valeurs (rue, avenue, chemin, etc.)
- Voie : ( 'Voie' ds valeurs\_foncieres ) : Nom de la rue
- BTQ : ( 'B/T/Q' ds valeurs\_foncieres ) : Indice de répétition
- Code\_postal : ( 'Code postal' ds valeurs\_foncieres ) : Code postal
- Surface\_carrez : ( 'Surface Carrez du 1er lot' ds valeurs\_foncieres ) : Surface Carrez
- Type\_local : ( 'Type local' ds valeurs\_foncieres ) : maison appartement
- Surface\_local : ( 'Surface réelle bati' ds valeurs\_foncieres ) : surface mesurée au sol
- Total\_piece : ( 'Nombre pièces principales' ds valeurs\_foncieres ) : Nbre pièces
- Surface\_terrain : ( 'Surface terrain' ds valeurs\_foncieres ) : Surface du terrain
- Id\_codedep\_codecommune CLE ETRANGERE : (A CREER à partir de 'Code departement' et 'Code commune' ds valeurs\_foncieres) /( 'C'+ 2 premiers chiffres(Code departement) + Code commune ds valeurs\_foncieres) : l'identifiant de la commune : 6 caractères exactement  
com\_id (6carac)=C+0si nec+com\_code1(4à5carac) de la forme CXXXXXX C2AXXX possib C974XX possib  
ex :

C01275 = C+01+275 (dep01/com275)

C34172 = C+34+172 (dep34/com172)

C2A362 = C+2A+363 (dep2A/com363)

C97424 = C+97+424 (dep974/com424) exception dep à 3 chiffres

ATTENTION :  
CONSERVER PONCTUELLEMENT pour construire Id\_codedep\_codecommune et pouvoir faire le lien avec table Commune :

- 'Code departement' ds valeurs\_foncieres
- 'Code commune' ds valeurs\_foncieres

ATTENTION :  
CONSERVER les éléments de la table VENTE pour faire les liens table BIEN et VENTE :

- Id\_vente : A CREER : (probablement à la fin avec les lignes d'index )
- Date\_vente : ( 'Date mutation' ds valeurs\_foncieres ) : Date de signature de l'acte
- Valeur : ( 'Valeur fonciere' ds valeurs\_foncieres ) : prix net vendeur (ou évaluation)

ATTENTION : Je dois tirer les Id\_bien et Id\_vente de ce fichier. Il faut donc que je crée les Id\_bien et Id\_Vente sur les mêmes données de base pour avoir leur relation.

REMARQUE : Si on considère qu'un bien ne peut avoir été vendu qu'une fois dans l'année, il n'y aura pas d'intérêt à séparer les Tables Bien et Vente, car à ce moment là 1 Vente <=> 1 Bien; et une table et 1 ID suffit. Dans la réalité 1 Bien peut être vendu 2 fois la même année. Même si ce n'est pas courant, il ne faut pas empêcher la possibilité.  
Cependant pour les statistiques que nous avons à faire, considérer que 2 ventes du même biens, sont 2 ventes de biens différents ne devraient pas fausser les statistiques pour ce que nous cherchons.

En tout cas ce n'est donc qu'à la fin que je dois extraire à part les Biens et Vente.  
En tout cas je dois pouvoir tracer le lien.

**STRATEGIE :**

1. Conserver les éléments suivants : 'No voie', 'Type de voie', 'Voie', 'B/T/Q', 'Code postal', 'Surface Carrez du 1er lot','Type local', 'Surface reelle bati', 'Nombre pieces principales', 'Surface terrain', 'Code departement', 'Code commune', 'Date mutation', 'Valeur fonciere'
2. CREER le 'Id\_codedep\_codecommune' à partir de 'Code departement' et 'Code commune' ds valeurs\_foncieres
3. Supprimer les attributs 'Code departement' et 'Code commune'
4. CREER un Id\_Vente et un Id\_Bien à partir des mêmes lignes d'index
5. CREER une Table Bien (avec 'Id\_codedep\_codecommune' en clé étrangère) et une Table Vente qui référence les Id\_bien en clé étrangère
6. VERIFIER qu'il n'y a pas de doublon dans Table Bien (hors Id\_unique créé).
7. Si doublon, ne garder qu'un Id\_bien et créer une table de correspondance d'Id remplacé.
8. Puis mettre à jour la Table Vente avec la tablde correspondance d'Id\_Bien doublon

1. Conservons les éléments utiles

#le fichier d'origine, nettoyé des valeurs aberrantes, avec toutes les colonnes utiles ou non :  
valeurs\_foncieres\_NETTOYE\_RETYPE\_FINAL.head()

	Code service CH	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Date mutation	Nature mutation	...	Surface Carrez du 5eme lot	Nombre de lots	Code type local
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	2	2
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	2	2
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	1	2
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	1	2
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2020-01-02	Vente	...	NaN	1	2

#La liste des colonnes utiles :  
liste\_col\_utiles\_VF

[ 'No voie',  
 'Type de voie',  
 'Voie',  
 'B/T/Q',  
 'Code postal',  
 'Surface Carrez du 1er lot',  
 'Type local',  
 'Surface reelle bati',  
 'Nombre pieces principales',

```
'Surface terrain',
'Code departement',
'Code commune',
'Date mutation',
'Valeur fonciere']
```

COMMENTONS par construire notre fichier en conservant les éléments utiles :

```
valeurs_foncieres_1 = valeurs_foncieres_NETTOYE_RETYPE_FINAL[liste_col_utiles_VF]
Description_Fichier_Tableau_avc_ValBlanches(valeurs_foncieres_1)
```



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
<b>No voie</b>	int16	34146	34146	0	1468	0
<b>Type de voie</b>	object	34146	34146	0	80	940
<b>Voie</b>	object	34146	34146	0	14130	0
<b>B/T/Q</b>	object	34146	34146	0	25	31973
<b>Code postal</b>	int32	34146	34146	0	2449	0
<b>Surface Carrez du 1er lot</b>	float64	34146	34146	0	10394	0
<b>Type local</b>	object	34146	34146	0	2	0
<b>Surface reelle bati</b>	int64	34146	34146	0	254	0
<b>Nombre pieces principales</b>	int64	34146	34146	0	12	0
<b>Surface terrain</b>	int16	34146	34146	0	201	0
<b>Code departement</b>	object	34146	34146	0	96	0
<b>Code commune</b>	int64	34146	34146	0	666	0
<b>Date mutation</b>	datetime64[ns]	34146	34146	0	158	0
<b>Valeur fonciere</b>	float64	34146	34146	0	9677	0

## ✓ 2. CREER le 'Id\_codedep\_codecommune' à partir de 'Code departement' et 'Code commune' ds valeurs\_foncieres

ANALYSONS les données dans 'Code departement' et 'Code commune'

```
valeurs_foncieres_1['Code departement'].unique()
```



```
array(['01', '06', '13', '14', '17', '25', '29', '31', '33', '34', '37',
       '38', '40', '44', '51', '54', '56', '58', '59', '60', '69', '74',
       '75', '76', '77', '78', '83', '84', '85', '86', '87', '91', '92',
       '93', '94', '95', '02', '03', '04', '10', '22', '24', '28', '30',
       '32', '42', '45', '50', '62', '63', '70', '82', '88', '90', '80',
       '11', '18', '19', '2A', '35', '39', '43', '49', '64', '72', '73',
       '07', '26', '27', '53', '66', '81', '972', '41', '71', '79', '89',
       '23', '16', '971', '47', '05', '08', '65', '48', '973', '974',
       '09', '46', '52', '36', '61', '2B', '55', '21', '12'], dtype=object)
```

```
liste_val_dep = valeurs_foncieres_1['Code departement'].unique()
```

```
liste_1=[]
for elt in liste_val_dep :
    liste_1.append(len(elt))
```

```
df_1= pd.DataFrame(liste_1)
df_1.value_counts()
```



```
0
2    92
3     4
Name: count, dtype: int64
```

On n'a donc que des departement de longueur 2 et 3, et sous forme de string\_object

```
valeurs_foncieres_1['Code commune'].unique()
```



```
array([103, 4, 88, 123, 5, 28, 208, 212, 338, 366, 300, 56, 232,
       260, 555, 63, 97, 449, 145, 150, 199, 156, 261, 185, 266, 328,
       143, 184, 230, 395, 161, 194, 512, 57, 155, 159, 524, 149, 384,
       119, 236, 238, 278, 105, 110, 111, 112, 115, 118, 217, 451, 655,
       350, 423, 545, 551, 586, 640, 62, 69, 7, 47, 75, 85, 249,
       689, 25, 26, 36, 44, 32, 51, 71, 60, 80, 252, 388, 607,
```

```

262, 408, 190, 73, 166, 209, 27, 323, 201, 204, 206, 333, 715,
754, 414, 113, 168, 322, 527, 3, 133, 189, 13, 550, 23, 169,
253, 46, 304, 256, 109, 234, 286, 2, 454, 368, 378, 491, 382,
41, 264, 389, 14, 102, 104, 106, 107, 114, 117, 120, 351, 447,
58, 59, 192, 259, 288, 379, 372, 383, 621, 646, 96, 9, 34,
70, 35, 127, 10, 225, 326, 377, 425, 570, 589, 12, 22, 48,
50, 1, 6, 16, 33, 42, 52, 68, 76, 79, 203, 218, 500,
572, 21, 223, 160, 29, 83, 325, 343, 362, 170, 101, 205, 207,
210, 220, 31, 39, 514, 522, 533, 157, 301, 191, 421, 526, 281,
284, 312, 317, 187, 332, 132, 198, 99, 147, 108, 263, 342, 513,
893, 89, 176, 381, 181, 257, 311, 116, 61, 285, 307, 419, 450,
487, 146, 165, 358, 481, 228, 121, 137, 164, 86, 174, 24, 40,
72, 45, 18, 37, 43, 74, 78, 219, 306, 258, 354, 92, 19,
202, 213, 265, 299, 134, 66, 81, 186, 229, 470, 162, 347, 410,
604, 385, 235, 53, 401, 130, 277, 428, 30, 94, 387, 144, 215,
279, 167, 122, 15, 573, 329, 271, 367, 509, 684, 243, 373, 438,
445, 468, 172, 466, 490, 686, 272, 363, 549, 631, 691, 20, 49,
8, 226, 173, 139, 334, 93, 195, 485, 296, 197, 178, 250, 175,
612, 163, 341, 152, 251, 124, 158, 297, 380, 396, 650, 126, 153,
477, 600, 657, 685, 64, 65, 424, 452, 244, 722, 84, 100, 415,
136, 318, 376, 529, 324, 327, 179, 563, 269, 95, 55, 135, 129,
502, 463, 601, 142, 283, 310, 11, 224, 276, 711, 316, 430, 418,
440, 624, 287, 339, 345, 432, 471, 77, 54, 488, 231, 293, 582,
478, 482, 826, 305, 216, 291, 188, 17, 282, 67, 82, 183, 498,
517, 90, 98, 464, 659, 268, 280, 313, 443, 331, 247, 128, 255,
337, 474, 548, 242, 483, 91, 241, 349, 579, 38, 539, 585, 441,
200, 462, 353, 516, 248, 653, 578, 289, 386, 222, 246, 397, 346,
577, 138, 399, 214, 699, 374, 321, 141, 510, 87, 274, 303, 360,
275, 543, 131, 407, 356, 177, 553, 409, 211, 547, 758, 404, 290,
596, 335, 361, 402, 642, 674, 687, 537, 494, 503, 508, 270, 294,
392, 151, 154, 431, 644, 390, 643, 521, 692, 391, 182, 140, 599,
446, 314, 442, 455, 616, 531, 458, 520, 557, 434, 196, 584, 764,
171, 469, 476, 302, 413, 416, 701, 348, 125, 267, 364, 606, 637,
702, 340, 587, 649, 355, 480, 574, 598, 227, 765, 292, 479, 489,
369, 394, 233, 273, 493, 193, 254, 561, 422, 400, 648, 667, 670,
240, 562, 672, 465, 525, 411, 357, 566, 899, 688, 580, 148, 810,
308, 330, 506, 559, 501, 426, 560, 613, 647, 484, 405, 309, 448,
245, 535, 645, 180, 457, 427, 344, 352, 536, 552, 295, 473, 433,
439, 528, 575, 680, 436, 519, 665, 398, 571, 795, 511, 495, 237,
660, 393, 661, 681, 298, 403, 461, 554, 635, 315, 641, 435, 652,
540, 475, 591, 675, 636, 617, 370, 569, 638, 544, 567, 507, 420,
437, 429, 634, 678, 239, 359, 679, 371, 417, 662, 630, 565, 695,
486, 683, 497, 492, 628, 623, 523, 627, 546, 534, 320, 581, 319,
639, 456, 705, 221, 603, 744, 666, 755, 609, 538, 895, 671, 651,
663, 459, 785, 752, 738, 467, 406, 620, 721, 615, 453, 542, 518,
412, 594, 743], dtype=int64)

```

Dans 'Code commune' les valeurs sont des entiers.

CREONS UNE colonne Id\_codedep\_codecommune à partir de 'Code departement' et 'Code commune'

```
valeurs_foncieres_2=valeurs_foncieres_1[:]
```

```

for elt in valeurs_foncieres_2.index :
    valeurs_foncieres_2.loc[elt,'Id_codedep_codecommune']=('C'+
                                                            (valeurs_foncieres_2.loc[elt,'Code departement'][:2]+
                                                             ('00'+str(valeurs_foncieres_2.loc[elt,'Code commune']))[-3:]
                                                            )

```

➡ C:\Users\matth\AppData\Local\Temp\ipykernel\_36580\632734150.py:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
valeurs\_foncieres\_2.loc[elt,'Id\_codedep\_codecommune']=('C'+

```
valeurs_foncieres_2.columns
```

➡ Index(['No voie', 'Type de voie', 'Voie', 'B/T/Q', 'Code postal',  
'Surface Carrez du 1er lot', 'Type local', 'Surface réelle bati',  
'Nombre pieces principales', 'Surface terrain', 'Code departement',  
'Code commune', 'Date mutation', 'Valeur fonciere',  
'Id\_codedep\_codecommune'],  
dtype='object')

```
valeurs_foncieres_2.head()
```



	No voie	Type de voie	Voie	B/T/Q	Code postal	Surface Carrez du 1er lot	Type local	Surface reelle bati	Nombre pieces principales	Surface terrain	Code departement	Code commune	Date mutation	Valeur fonciere
0	347	RUE	DU CHATEAU		1170	48.22	Appartement	48	3	0	01	103	2020-01-02	165000.0
1	4	BD	EDOUARD BAUDOUIN		6160	39.11	Appartement	40	1	0	06	4	2020-01-02	355680.0
2	20	RUE	MARCEAU	B	6000	80.25	Appartement	82	3	0	06	88	2020-01-02	229500.0

```
(valeurs_foncieres_2[['Code departement', 'Code commune', 'Id_codedep_codecommune']]).head()
```

	Code departement	Code commune	Id_codedep_codecommune
0	01	103	C01103
1	06	4	C06004
2	06	88	C06088
3	06	123	C06123
4	13	5	C13005

```
(valeurs_foncieres_2[['Code departement', 'Code commune', 'Id_codedep_codecommune']]).tail()
```

	Code departement	Code commune	Id_codedep_codecommune
34164	95	585	C95585
34165	95	598	C95598
34166	95	607	C95607
34167	972	213	C97213
34168	972	229	C97229

```
Description_Fichier_Tableau_avc_ValBlanches(valeurs_foncieres_2)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
No voie	int16	34146	34146	0	1468	0
Type de voie	object	34146	34146	0	80	940
Voie	object	34146	34146	0	14130	0
B/T/Q	object	34146	34146	0	25	31973
Code postal	int32	34146	34146	0	2449	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
Type local	object	34146	34146	0	2	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Surface terrain	int16	34146	34146	0	201	0
Code departement	object	34146	34146	0	96	0
Code commune	int64	34146	34146	0	666	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Valeur fonciere	float64	34146	34146	0	9677	0
Id_codedep_codecommune	object	34146	34146	0	3125	0

### 3. Supprimer les attributs 'Code departement' et 'Code commune'

```
valeurs_foncieres_3_avc_CodedepCodecom = valeurs_foncieres_2[:]
```

```
valeurs_foncieres_3_avc_CodedepCodecom.drop(columns=['Code departement', 'Code commune'], inplace=True)
```

```
C:\Users\matth\AppData\Local\Temp\ipykernel_36580\1773831940.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-values\\_foncières\\_3\\_avc\\_CodedepCodecom.drop\(columns=\['Code departement','Code commune'\], inplace=True\)](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-values_foncières_3_avc_CodedepCodecom.drop(columns=['Code departement','Code commune'], inplace=True))

Description\_Fichier\_Tableau\_avc\_ValBlanches(valeurs\_foncières\_3\_avc\_CodedepCodecom)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
No voie	int16	34146	34146	0	1468	0
Type de voie	object	34146	34146	0	80	940
Voie	object	34146	34146	0	14130	0
B/T/Q	object	34146	34146	0	25	31973
Code postal	int32	34146	34146	0	2449	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
Type local	object	34146	34146	0	2	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Surface terrain	int16	34146	34146	0	201	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Valeur fonciere	float64	34146	34146	0	9677	0
Id_codedep_codecommune	object	34146	34146	0	3125	0

#### ✓ VERIFIONS QUE NOUS N'AVONS PAS DE DOUBLONS DE TRANSACTIONS :

valeurs\_foncières\_3\_avc\_CodedepCodecom.duplicated()



```
0      False
1      False
2      False
3      False
4      False
...
34164   False
34165   False
34166   False
34167   False
34168   False
Length: 34146, dtype: bool
```

valeurs\_foncières\_3\_avc\_CodedepCodecom.duplicated().value\_counts()



```
False      34146
Name: count, dtype: int64
```

```
print("cela signifie que nous avons : ",
      valeurs_foncières_3_avc_CodedepCodecom.duplicated().sum(),
      "double(s) de transaction")
```



```
cela signifie que nous avons : 0 double(s) de transaction
```

Commencez à coder ou à [générer](#) avec l'IA.

#### ✓ 4. CREER un Id\_Vente et un Id\_Bien à partir des mêmes lignes d'index

valeurs\_foncières\_4 = valeurs\_foncières\_3\_avc\_CodedepCodecom[:]

valeurs\_foncières\_4.head()



	No voie	Type de voie	Voie	B/T/Q	Code postal	Surface Carrez du 1er lot	Type local	Surface reelle bati	Nombre pieces principales	Surface terrain	Date mutation	Valeur fonciere	Id_codedep_codecommu
0	347	RUE	DU CHATEAU		1170	48.22	Appartement	48	3	0	2020-01-02	165000.0	C011
1	4	BD	EDOUARD BAUDOIN		6160	39.11	Appartement	40	1	0	2020-01-02	355680.0	C06C

```
valeurs_foncieries_4.reset_index(inplace=True)
```

ATTENTION ici on n'a plus autant de lignes qu'initialement d'ou ce décalage. Cependant je peux garder ce décalage afin de garder une trace du fichier d'origine.

```
valeurs_foncieries_4.dtypes
```

```
➡ index                                int64
  No voie                             int16
  Type de voie                         object
  Voie                                 object
  B/T/Q                               object
  Code postal                          int32
  Surface Carrez du 1er lot            float64
  Type local                           object
  Surface reelle bati                  int64
  Nombre pieces principales            int64
  Surface terrain                      int16
  Date mutation                       datetime64[ns]
  Valeur fonciere                      float64
  Id_codedep_codecommune              object
dtype: object
```

```
valeurs_foncieries_4 = valeurs_foncieries_4.astype({'index':'string'})
```

```
valeurs_foncieries_4.dtypes
```

```
➡ index                                string[python]
  No voie                             int16
  Type de voie                         object
  Voie                                 object
  B/T/Q                               object
  Code postal                          int32
  Surface Carrez du 1er lot            float64
  Type local                           object
  Surface reelle bati                  int64
  Nombre pieces principales            int64
  Surface terrain                      int16
  Date mutation                       datetime64[ns]
  Valeur fonciere                      float64
  Id_codedep_codecommune              object
dtype: object
```

```
valeurs_foncieries_4['Id_vente'] = 'V'+valeurs_foncieries_4['index']
valeurs_foncieries_4['Id_bien'] = 'B'+valeurs_foncieries_4['index']
```

```
valeurs_foncieries_4.dtypes
```

```
➡ index                                string[python]
  No voie                             int16
  Type de voie                         object
  Voie                                 object
  B/T/Q                               object
  Code postal                          int32
  Surface Carrez du 1er lot            float64
  Type local                           object
  Surface reelle bati                  int64
  Nombre pieces principales            int64
  Surface terrain                      int16
  Date mutation                       datetime64[ns]
  Valeur fonciere                      float64
  Id_codedep_codecommune              object
  Id_vente                           string[python]
  Id_bien                             string[python]
dtype: object
```

```
display(valeurs_foncieries_4)
```

	index	No voie	Type de voie	Voie	B/T/Q	Code postal	Surface Carrez du 1er lot	Type local	Surface reelle bati	Nombre pieces principales	Surface terrain	Date mutation	Valeur fonciere	Id_code
0	0	347	RUE	DU CHATEAU		1170	48.22	Appartement	48	3	0	2020-01-02	165000.0	
1	1	4	BD	EDOUARD BAUDOIN		6160	39.11	Appartement	40	1	0	2020-01-02	355680.0	
2	2	20	RUE	MARCEAU	B	6000	80.25	Appartement	82	3	0	2020-01-02	229500.0	
3	3	550	RTE	DES VESPINS RN7		6700	27.51	Appartement	27	1	0	2020-01-02	125000.0	
4	4	9300	RES	LES ARPEGES BD DES ABA		13400	47.33	Appartement	47	2	0	2020-01-02	90000.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
34141	34164	5	PL	JEAN CHARCOT		95200	59.50	Appartement	63	3	0	2020-06-30	115000.0	

valeurs\_foncieres\_4.columns

```
Index(['index', 'No voie', 'Type de voie', 'Voie', 'B/T/Q', 'Code postal',
      'Surface Carrez du 1er lot', 'Type local', 'Surface reelle bati',
      'Nombre pieces principales', 'Surface terrain', 'Date mutation',
      'Valeur fonciere', 'Id_codedep_codecommune', 'Id_vente', 'Id_bien'],
      dtype='object')
```

display(valeurs\_foncieres\_4[['index', 'Valeur fonciere', 'Id\_codedep\_codecommune', 'Id\_vente', 'Id\_bien']])

	index	Valeur fonciere	Id_codedep_codecommune	Id_vente	Id_bien
0	0	165000.0	C01103	V0	B0
1	1	355680.0	C06004	V1	B1
2	2	229500.0	C06088	V2	B2
3	3	125000.0	C06123	V3	B3
4	4	90000.0	C13005	V4	B4
...	...	...	...	...	...
34141	34164	115000.0	C95585	V34164	B34164
34142	34165	129000.0	C95598	V34165	B34165
34143	34166	212000.0	C95607	V34166	B34166
34144	34167	220000.0	C97213	V34167	B34167
34145	34168	121000.0	C97229	V34168	B34168

34146 rows × 5 columns

valeurs\_foncieres\_4.drop(columns=['index'], inplace=True)

Description\_Fichier\_Tableau\_avc\_ValBlanches(valeurs\_foncieres\_4)



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
No voie	int16	34146	34146	0	1468	0
Type de voie	object	34146	34146	0	80	940
Voie	object	34146	34146	0	14130	0
B/T/Q	object	34146	34146	0	25	31973
Code postal	int32	34146	34146	0	2449	0
Surface Carrez du 1er lot	float64	34146	34146	0	10394	0
Type local	object	34146	34146	0	2	0
Surface reelle bati	int64	34146	34146	0	254	0
Nombre pieces principales	int64	34146	34146	0	12	0
Surface terrain	int16	34146	34146	0	201	0
Date mutation	datetime64[ns]	34146	34146	0	158	0
Valeur fonciere	float64	34146	34146	0	9677	0
Id_codedep_codecommune	object	34146	34146	0	3125	0
Id_vente	string[python]	34146	34146	0	34146	0
Id_bien	string[python]	34146	34146	0	34146	0

Commencez à coder ou à [générer](#) avec l'IA.

Commencez à coder ou à [générer](#) avec l'IA.

5. CREER une Table Bien (avec 'Id\_codedep\_codecommune' en clé étrangère) et une Table Vente qui référence les Id\_bien en clé étrangère

a. CREONS une Table Bien (avec 'Id\_codedep\_codecommune' en clé étrangère)

valeurs\_foncieries\_4.columns



```
Index(['No voie', 'Type de voie', 'Voie', 'B/T/Q', 'Code postal',  
      'Surface Carrez du 1er lot', 'Type local', 'Surface reelle bati',  
      'Nombre pieces principales', 'Surface terrain', 'Date mutation',  
      'Valeur fonciere', 'Id_codedep_codecommune', 'Id_vente', 'Id_bien'],  
      dtype='object')
```

```
liste_col_Bien = ['Id_bien', 'No voie', 'B/T/Q',  
                  'Type de voie', 'Voie', 'Code postal',  
                  'Nombre pieces principales', 'Surface Carrez du 1er lot', 'Surface reelle bati',  
                  'Type local', 'Surface terrain', 'Id_codedep_codecommune']
```

```
Bien_1 = valeurs_foncieries_4[ liste_col_Bien ]
```

```
display(Bien_1)
```



	Id_bien	No voie	B/T/Q	Type de voie	Voie	Code postal	Nombre pieces principales	Surface Carrez du 1er lot	Surface reelle bati	Type local	Surface terrain	Id_codedep_codecommune
0	B0	347		RUE	DU CHATEAU	1170	3	48.22	48	Appartement	0	C01103
1	B1	4		BD	EDOUARD BAUDOIN	6160	1	39.11	40	Appartement	0	C06004
2	B2	20	B	RUE	MARCEAU	6000	3	80.25	82	Appartement	0	C06088
3	B3	550		RTE	DES VESPINS RN7	6700	1	27.51	27	Appartement	0	C06123
4	B4	9300		RES	LES ARPEGES BD DES ABA	13400	2	47.33	47	Appartement	0	C13005
...	...	...	...	...	...	...	...	...	...	...	...	...
34141	B34164	5		PL	JEAN CHARCOT	95200	3	59.50	63	Appartement	0	C95585

```
Bien_1.rename(columns ={'Id_bien' : 'Id_bien',
                        'No voie' : 'No_voie',
                        'B/T/Q': 'BTQ',
                        'Type de voie': 'Type_de_voie',
                        'Voie': 'Voie',
                        'Code postal': 'Code_postal',
                        'Nombre pieces principales': 'Total_piece',
                        'Surface Carrez du 1er lot': 'Surface_carrez',
                        'Surface reelle bati': 'Surface_reelle',
                        'Type local': 'Type_local',
                        'Surface terrain': 'Surface_terrain',
                        'Id_codedep_codecommune': 'Id_codedep_codecommune'
                        },
              inplace=True)
```

➡ C:\Users\matth\AppData\Local\Temp\ipykernel\_36580\425386594.py:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
 Bien\_1.rename(columns ={'Id\_bien' : 'Id\_bien',

Description\_Fichier\_Tableau\_avc\_ValBlanches(Bien\_1)

➡

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
<b>Id_bien</b>	string[python]	34146	34146	0	34146	0
<b>No_voie</b>	int16	34146	34146	0	1468	0
<b>BTQ</b>	object	34146	34146	0	25	31973
<b>Type_de_voie</b>	object	34146	34146	0	80	940
<b>Voie</b>	object	34146	34146	0	14130	0
<b>Code_postal</b>	int32	34146	34146	0	2449	0
<b>Total_piece</b>	int64	34146	34146	0	12	0
<b>Surface_carrez</b>	float64	34146	34146	0	10394	0
<b>Surface_reelle</b>	int64	34146	34146	0	254	0
<b>Type_local</b>	object	34146	34146	0	2	0
<b>Surface_terrain</b>	int16	34146	34146	0	201	0
<b>Id_codedep_codecommune</b>	object	34146	34146	0	3125	0

## ▼ b. une Table Vente qui référence les Id\_bien en clé étrangère

```
liste_col_Vente = ['Id_vente', 'Date mutation', 'Valeur fonciere', 'Id_bien']
```

```
Vente_1 = valeurs_fonciere_4[ liste_col_Vente ]
```

```
display(Vente_1)
```

➡

	Id_vente	Date mutation	Valeur fonciere	Id_bien
<b>0</b>	V0	2020-01-02	165000.0	B0
<b>1</b>	V1	2020-01-02	355680.0	B1
<b>2</b>	V2	2020-01-02	229500.0	B2
<b>3</b>	V3	2020-01-02	125000.0	B3
<b>4</b>	V4	2020-01-02	90000.0	B4
...	...	...	...	...
<b>34141</b>	V34164	2020-06-30	115000.0	B34164
<b>34142</b>	V34165	2020-06-30	129000.0	B34165
<b>34143</b>	V34166	2020-06-30	212000.0	B34166
<b>34144</b>	V34167	2020-06-30	220000.0	B34167
<b>34145</b>	V34168	2020-06-30	121000.0	B34168

34146 rows × 4 columns

```
Vente_1.rename(columns ={'Id_vente' : 'Id_vente',
                        'Date mutation' : 'Date_vente',
                        'Valeur fonciere' : 'Valeur',
                        'Id_bien' : 'Id_bien'
                        },
               inplace=True
               )
```

➡ C:\Users\matth\AppData\Local\Temp\ipykernel\_36580\3419524341.py:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
Vente\_1.rename(columns ={'Id\_vente' : 'Id\_vente',

Description\_Fichier\_Tableau\_avc\_ValBlanches(Vente\_1)

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
<b>Id_vente</b>	string[python]	34146	34146	0	34146	0
<b>Date_vente</b>	datetime64[ns]	34146	34146	0	158	0
<b>Valeur</b>	float64	34146	34146	0	9677	0
<b>Id_bien</b>	string[python]	34146	34146	0	34146	0

## ✓ 6. VERIFIER qu'il n'y a pas de doublon dans Table Bien (hors Id\_unique créé).

Description\_Fichier\_Tableau\_avc\_ValBlanches(Bien\_1)

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes	Valeurs blanches
<b>Id_bien</b>	string[python]	34146	34146	0	34146	0
<b>No_voie</b>	int16	34146	34146	0	1468	0
<b>BTQ</b>	object	34146	34146	0	25	31973
<b>Type_de_voie</b>	object	34146	34146	0	80	940
<b>Voie</b>	object	34146	34146	0	14130	0
<b>Code_postal</b>	int32	34146	34146	0	2449	0
<b>Total_piece</b>	int64	34146	34146	0	12	0
<b>Surface_carrez</b>	float64	34146	34146	0	10394	0
<b>Surface_reelle</b>	int64	34146	34146	0	254	0
<b>Type_local</b>	object	34146	34146	0	2	0
<b>Surface_terrain</b>	int16	34146	34146	0	201	0
<b>Id_codedep_codecommune</b>	object	34146	34146	0	3125	0

**ATTENTION AUX SHALLOW COPY et DEEP COPY : Une shallow copy crée un nouvel objet mais qui pointe vers les mêmes données que le premier; une modification sur l'une modifie l'autre. Il faut faire un DEEP COPY si on veut que les données soient indépendantes également**

source : <https://medium.com/@mathieuvdp/attention-au-deep-copy-de-pandas-8ebbdd5b0b92>

```
Bien_2 = Bien_1.copy(deep=False) => shallow copy
```

```
Bien_2 = Bien_1.copy(deep=True) => deep copy
```

#ERREUR quand je faisais : Bien\_2 = Bien\_1 ou Bien\_2 = Bien\_1[:] , ça faisait des shallow copy !!! au lieu de DEEP COPY !!!

```
#Creation d'une DEEP COPY
Bien_2 = Bien_1.copy(deep=True)
```

Etude des doublons (et suppression si nécessaire) sans prendre compte Id\_bien forcément unique par construction

Bien\_2.columns

```
➡ Index(['Id_bien', 'No_voie', 'BTQ', 'Type_de_voie', 'Voie', 'Code_postal',
        'Total_piece', 'Surface_carrez', 'Surface_reelle', 'Type_local',
        'Surface_terrain', 'Id_codedep_codecommune'],
        dtype='object')
```

```
liste_tte_col_Bien_sauf_Id = ['No_voie', 'BTQ', 'Type_de_voie', 'Voie', 'Code_postal',
                             'Total_piece', 'Surface_carrez', 'Surface_reelle', 'Type_local',
```

```
'Surface_terrain', 'Id_codedep_codecommune']
```

```
Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id)
```

```
0      False
1      False
2      False
3      False
4      False
...
34141  False
34142  False
34143  False
34144  False
34145  False
Length: 34146, dtype: bool
```

```
(Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id) ).value_counts()
```

```
False    34131
True       15
Name: count, dtype: int64
```

```
print("Cela signifie que j'ai ",
      (Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id) ).sum() ,
      "doublons de bien")
```

```
Cela signifie que j'ai 15 doublons de bien
```

Pour rappel, nous avons :

- 0 doublon de transaction
- 15 doublons de bien => Cela signifie que quelques biens ont participés à plusieurs transactions

ETUDIONS LES en utilisant `(Bien_2[listete_col_Bien_sauf_Id]).duplicated()` comme masque :

```
Bien_2.loc[Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id) , :]
```

	Id_bien	No_voie	BTQ	Type_de_voie	Voie	Code_postal	Total_piece	Surface_carrez	Surface_reelle	Type_local	Surfa
10820	B10826	1998		RTE	DE LESPECIER	40170	3	46.09	64	Maison	
16074	B16082	4		RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
18664	B18674	7		RUE	DE PORT LA BLANCHE	44300	3	62.60	62	Appartement	
20391	B20402	25		RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
25615	B25628	25		RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
25858	B25871	97		AV	DES LOMBARDS	10000	4	82.00	82	Appartement	
26424	B26437	8		RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
27701	B27715	1		PL	DE HANOVRE	76100	1	16.22	16	Appartement	
28200	B28215	17		RUE	DES ERABLES	78150	4	101.40	101	Appartement	
28579	B28594	23		RUE	D	77144	1	22.02	22	Appartement	

```
#help(pd.DataFrame.duplicated)
```

✓ utilisons l'option `keep=False` , pour avoir tous les doublons y compris le 1er

```
(Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id,keep=False) ).value_counts()
```

```
False    34117
True       29
Name: count, dtype: int64
```



```
print("Cela signifie que j'ai ",
      (Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id,keep=False) ).sum() ,
      "doublons de bien")
```

Cela signifie que j'ai 29 doublons de bien

=> Cela signifie que nous avons 15 doublons provenant de 14 valeurs initiales

```
Bien_2.loc[Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id,keep=False) , :]
```

	Id_bien	No_voie	BTQ	Type_de_voie	Voie	Code_postal	Total_piece	Surface_carrez	Surface_reelle	Type_local	Surf
	2851	B2852	4	RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
	3169	B3170	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	6114	B6119	8	RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
	6823	B6828	97	AV	DES LOMBARDS	10000	4	82.00	82	Appartement	
	8975	B8981	2752	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	9686	B9692	1998	RTE	DE LESPECIER	40170	3	46.09	64	Maison	
	10820	B10826	1998	RTE	DE LESPECIER	40170	3	46.09	64	Maison	
	11335	B11341	62	RUE	JEAN BART	59290	4	85.61	90	Appartement	
	13484	B13492	7	RUE	DE PORT LA BLANCHE	44300	3	62.60	62	Appartement	
	15483	B15491	17	RUE	DES ERABLES	78150	4	101.40	101	Appartement	
	16074	B16082	4	RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
	18664	B18674	7	RUE	DE PORT LA BLANCHE	44300	3	62.60	62	Appartement	
	20391	B20402	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	25614	B25627	23	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	25615	B25628	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	25858	B25871	97	AV	DES LOMBARDS	10000	4	82.00	82	Appartement	
	26424	B26437	8	RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
	27699	B27713	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	
	27701	B27715	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	
	27702	B27716	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	

NOUS DEVONS ELIMINER LES DOUBLONS, mais nous devons d'abord conserver une table avec les ID, afin de pouvoir remplacer les Id\_bien dans la table Id\_Vente

CREONS LA TABLE DE REFERENCEMENT des Id du même BIEN

```
Table_doublon_bien = pd.DataFrame({'Id_bien_reference':[],'Id_bien_doublon':[]})
```

```
Table_doublon_bien.head()
```

Id_bien_reference	Id_bien_doublon
-------------------	-----------------

```
(Bien_2.loc[Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id,keep=False) , :]).sort_values(by=['Surface_carrez','No_voie','Id_bien
```

	Id_bien	No_voie	BTQ	Type_de_voie	Voie	Code_postal	Total_piece	Surface_carrez	Surface_reelle	Type_local	Surfa
	27702	B27716	1	PL	DE HANOVRE	76100	1	16.03	16	Appartement	
	33055	B33077	1	PL	DE HANOVRE	76100	1	16.03	16	Appartement	
	27699	B27713	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	
	27701	B27715	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	
	16074	B16082	4	RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
	2851	B2852	4	RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
	25614	B25627	23	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	28579	B28594	23	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	20391	B20402	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	25615	B25628	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	3169	B3170	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	29685	B29702	2752	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	8975	B8981	2752	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	28467	B28482	2753	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	30507	B30526	2753	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	10820	B10826	1998	RTE	DE LESPECIER	40170	3	46.09	64	Maison	
	9686	B9692	1998	RTE	DE LESPECIER	40170	3	46.09	64	Maison	
	26424	B26437	8	RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
	6114	B6119	8	RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
	32583	B32605	4	RUE	DU HAUT VINAGE	59200	3	61.00	66	Appartement	
	33403	B33425	4	RUE	DU HAUT VINAGE	59200	3	61.00	66	Appartement	

```
Table_doublon_bien_2 = (Bien_2.loc[Bien_2.duplicated(subset = liste_tte_col_Bien_sauf_Id,keep=False) , :]).sort_values(by=['Surface_carrez'])
```

```
Table_doublon_bien_2['Id_bien_reference']=Table_doublon_bien_2['Id_bien']
```

```
Table_doublon_bien_2[['Id_bien','Id_bien_reference','Surface_carrez','No_voie']]
```



	Id_bien	Id_bien_reference	Surface_carrez	No_voie
<b>27702</b>	B27716	B27716	16.03	1
<b>33055</b>	B33077	B33077	16.03	1
<b>27699</b>	B27713	B27713	16.22	1
<b>27701</b>	B27715	B27715	16.22	1
<b>16074</b>	B16082	B16082	19.20	4
<b>2851</b>	B2852	B2852	19.20	4
<b>25614</b>	B25627	B25627	22.02	23
<b>28579</b>	B28594	B28594	22.02	23
<b>20391</b>	B20402	B20402	22.02	25
<b>25615</b>	B25628	B25628	22.02	25
<b>3169</b>	B3170	B3170	22.02	25
<b>29685</b>	B29702	B29702	25.17	2752
<b>8975</b>	B8981	B8981	25.17	2752
<b>28467</b>	B28482	B28482	25.17	2753
<b>30507</b>	B30526	B30526	25.17	2753
<b>10820</b>	B10826	B10826	46.09	1998
<b>9686</b>	B9692	B9692	46.09	1998
<b>26424</b>	B26437	B26437	57.46	8
<b>6114</b>	B6119	B6119	57.46	8
<b>32583</b>	B32605	B32605	61.00	4
<b>33403</b>	B33425	B33425	61.00	4
<b>13484</b>	B13492	B13492	62.60	7
<b>18664</b>	B18674	B18674	62.60	7
<b>25858</b>	B25871	B25871	82.00	97
<b>6823</b>	B6828	B6828	82.00	97
<b>11335</b>	B11341	B11341	85.61	62
<b>33404</b>	B33426	B33426	85.61	62
<b>15483</b>	B15491	B15491	101.40	17
<b>28200</b>	B28215	B28215	101.40	17

```
Table_doublon_bien_2['Id_bien_reference']=['B27716',  
'B27716',  
'B27713',  
'B27713',  
'B27713',  
'B2852',  
'B2852',  
'B25627',  
'B25627',  
'B3170',  
'B3170',  
'B3170',  
'B3170',  
'B8981',  
'B8981',  
'B28482',  
'B28482',  
'B9692',  
'B9692',  
'B6119',  
'B6119',  
'B32605',  
'B32605',  
'B13492',  
'B13492',  
'B6828',  
'B6828',  
'B11341',  
'B11341',  
'B15491',  
'B15491']
```

Table\_doublon\_bien\_2[['Id\_bien','Id\_bien\_reference','Surface\_carrez','No\_voie']]



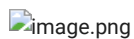
	Id_bien	Id_bien_reference	Surface_carrez	No_voie
<b>27702</b>	B27716	B27716	16.03	1
<b>33055</b>	B33077	B27716	16.03	1
<b>27699</b>	B27713	B27713	16.22	1
<b>27701</b>	B27715	B27713	16.22	1
<b>16074</b>	B16082	B2852	19.20	4
<b>2851</b>	B2852	B2852	19.20	4
<b>25614</b>	B25627	B25627	22.02	23
<b>28579</b>	B28594	B25627	22.02	23
<b>20391</b>	B20402	B3170	22.02	25
<b>25615</b>	B25628	B3170	22.02	25
<b>3169</b>	B3170	B3170	22.02	25
<b>29685</b>	B29702	B8981	25.17	2752
<b>8975</b>	B8981	B8981	25.17	2752
<b>28467</b>	B28482	B28482	25.17	2753
<b>30507</b>	B30526	B28482	25.17	2753
<b>10820</b>	B10826	B9692	46.09	1998
<b>9686</b>	B9692	B9692	46.09	1998
<b>26424</b>	B26437	B6119	57.46	8
<b>6114</b>	B6119	B6119	57.46	8
<b>32583</b>	B32605	B32605	61.00	4
<b>33403</b>	B33425	B32605	61.00	4
<b>13484</b>	B13492	B13492	62.60	7
<b>18664</b>	B18674	B13492	62.60	7
<b>25858</b>	B25871	B6828	82.00	97
<b>6823</b>	B6828	B6828	82.00	97
<b>11335</b>	B11341	B11341	85.61	62
<b>33404</b>	B33426	B11341	85.61	62
<b>15483</b>	B15491	B15491	101.40	17
<b>28200</b>	B28215	B15491	101.40	17

Table\_doublon\_bien\_correspondance = Table\_doublon\_bien\_2[['Id\_bien','Id\_bien\_reference']]  
Table\_doublon\_bien\_correspondance



	Id_bien	Id_bien_reference
27702	B27716	B27716
33055	B33077	B27716
27699	B27713	B27713
27701	B27715	B27713
16074	B16082	B2852
2851	B2852	B2852
25614	B25627	B25627
28579	B28594	B25627
20391	B20402	B3170
25615	B25628	B3170
3169	B3170	B3170
29685	B29702	B8981
8975	B8981	B8981
28467	B28482	B28482
30507	B30526	B28482
10820	B10826	B9692
9686	B9692	B9692
26424	B26437	B6119
6114	B6119	B6119
32583	B32605	B32605
33403	B33425	B32605
13484	B13492	B13492
18664	B18674	B13492
25858	B25871	B6828
6823	B6828	B6828
11335	B11341	B11341
33404	B33426	B11341
15483	B15491	B15491
28200	B28215	B15491

**REFLEXION sur ces doublons :**



Table\_doublon\_bien\_2

	Id_bien	No_voie	BTQ	Type_de_voie	Voie	Code_postal	Total_piece	Surface_carrez	Surface_reelle	Type_local	Surf:
	27702	B27716	1	PL	DE HANOVRE	76100	1	16.03	16	Appartement	
	33055	B33077	1	PL	DE HANOVRE	76100	1	16.03	16	Appartement	
	27699	B27713	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	
	27701	B27715	1	PL	DE HANOVRE	76100	1	16.22	16	Appartement	
	16074	B16082	4	RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
	2851	B2852	4	RUE	DE LA ROCHE BENOTTE	91580	1	19.20	19	Appartement	
	25614	B25627	23	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	28579	B28594	23	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	20391	B20402	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	25615	B25628	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	3169	B3170	25	RUE	D AMSTERDAM	77144	1	22.02	22	Appartement	
	29685	B29702	2752	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	8975	B8981	2752	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	28467	B28482	2753	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	30507	B30526	2753	AV	DE L OCEAN	40550	3	25.17	25	Appartement	
	10820	B10826	1998	RTE	DE LESPECIER	40170	3	46.09	64	Maison	
	9686	B9692	1998	RTE	DE LESPECIER	40170	3	46.09	64	Maison	
	26424	B26437	8	RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
	6114	B6119	8	RUE	CESAR FRANCK	78330	3	57.46	56	Appartement	
	32583	B32605	4	RUE	DU HAUT VINAGE	59200	3	61.00	66	Appartement	
	33403	B33425	4	RUE	DU HAUT VINAGE	59200	3	61.00	66	Appartement	

#### REFLEXION :

- EN FAVEUR de la suppression des doublons :

Pour ce qui est de la maison de 46,09 m2, il paraît peu probable, qu'il existe 2 maisons différentes de même superficie, à la même adresse.

- EN DEFAVEUR de la suppression des doublons :

Pour la maison : il pourrait s'agir d'un lotissement avec des maisons identiques et une entrée commune. Pour les appartements : il est tout à fait possible d'avoir 2 appartements différents à la même adresse, et de superficie identique, car les étages ont été conçus à l'identique. Pour l'appartement de 22,02m2 au 25 rue d'amsterdam, il paraît peu probable que ce soit le même appartement qui ait été vendu 3 fois dans la même année. Au vu des délais des ventes qui prennent au minimum 3 mois, cela paraît très peu probable.

=> JE CHOISIS DONC DE NE PAS RETIRER DE DOUBLON car je ne peux pas être sûr qu'il s'agit bien de la maison.

Je garde cela en tête. Mais au vu des statistiques que je dois faire, Cela ne devrait pas avoir d'impact.

Je retiens également que si je choisis d'éliminer les doublons je suivrai la table de correspondance suivante :

Table\_doublon\_bien\_correspondance



	Id_bien	Id_bien_reference
27702	B27716	B27716
33055	B33077	B27716
27699	B27713	B27713
27701	B27715	B27713
16074	B16082	B2852
2851	B2852	B2852
25614	B25627	B25627
28579	B28594	B25627
20391	B20402	B3170
25615	B25628	B3170
3169	B3170	B3170
29685	B29702	B8981
8975	B8981	B8981
28467	B28482	B28482
30507	B30526	B28482
10820	B10826	B9692
9686	B9692	B9692
26424	B26437	B6119
6114	B6119	B6119
32583	B32605	B32605
33403	B33425	B32605
13484	B13492	B13492
18664	B18674	B13492
25858	B25871	B6828
6823	B6828	B6828
11335	B11341	B11341
33404	B33426	B11341
15483	B15491	B15491
28200	B28215	B15491

Commencez à coder ou à [générer](#) avec l'IA.

Et j'effectuerai les opérations suivantes :

```
Bien_2_sans_doublon = Bien_2.copy(deep=True)
```

```
Bien_2_sans_doublon.head()
```



	Id_bien	No_voie	BTQ	Type_de_voie	Voie	Code_postal	Total_piece	Surface_carrez	Surface_reelle	Type_local	Surface_terri
0	B0	347		RUE	DU CHATEAU	1170	3	48.22	48	Appartement	
1	B1	4		BD	EDOUARD BAUDOIN	6160	1	39.11	40	Appartement	
2	B2	20	B	RUE	MARCEAU	6000	3	80.25	82	Appartement	
					DES						

```
for elt in Table_doublon_bien_correspondance['Id_bien'] :  
    print(elt)
```



B27716  
B33077  
B27713  
B27715  
B16082  
B2852