analyse_exploratoire

May 29, 2024

1 Projet 3 - Anticipez les besoins en consommation de bâtiments

1.1 Analyse exploratoire et création d'un dataset clean

Le but de ce notebook est d'analyser le jeu de données initial et de le traiter afin de produire un dataset "clean", exporté en csv, qui sera la base du travail de machine learning consécutif.

[ ]:
```
import pandas as pd
from MLUtils import DataAnalysis, DataEngineering
from sklearn.preprocessing import OneHotEncoder
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
df = pd.read_csv('data/2016_Building_Energy_Benchmarking_20240529.csv')
```

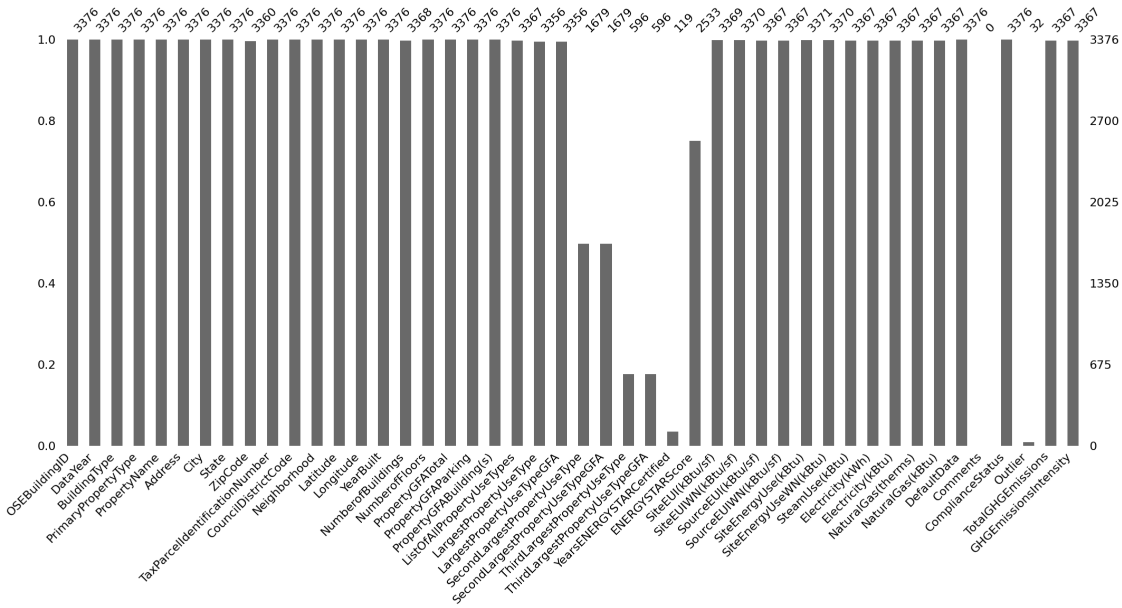
[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3376 entries, 0 to 3375
Data columns (total 46 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   OSEBuildingID                    3376 non-null   int64
 1   DataYear                         3376 non-null   int64
 2   BuildingType                     3376 non-null   object
 3   PrimaryPropertyType              3376 non-null   object
 4   PropertyName                     3376 non-null   object
 5   Address                          3376 non-null   object
 6   City                             3376 non-null   object
 7   State                            3376 non-null   object
 8   ZipCode                          3360 non-null   float64
 9   TaxParcelIdentificationNumber    3376 non-null   object
 10  CouncilDistrictCode              3376 non-null   int64
 11  Neighborhood                     3376 non-null   object
 12  Latitude                         3376 non-null   float64
```

```
 13  Longitude                         3376 non-null   float64
 14  YearBuilt                         3376 non-null   int64
 15  NumberofBuildings                 3368 non-null   float64
 16  NumberofFloors                    3376 non-null   int64
 17  PropertyGFATotal                  3376 non-null   int64
 18  PropertyGFAParking                3376 non-null   int64
 19  PropertyGFABuilding(s)            3376 non-null   int64
 20  ListOfAllPropertyUseTypes         3367 non-null   object
 21  LargestPropertyUseType            3356 non-null   object
 22  LargestPropertyUseTypeGFA         3356 non-null   float64
 23  SecondLargestPropertyUseType      1679 non-null   object
 24  SecondLargestPropertyUseTypeGFA   1679 non-null   float64
 25  ThirdLargestPropertyUseType       596 non-null    object
 26  ThirdLargestPropertyUseTypeGFA    596 non-null    float64
 27  YearsENERGYSTARCertified          119 non-null    object
 28  ENERGYSTARScore                   2533 non-null   float64
 29  SiteEUI(kBtu/sf)                  3369 non-null   float64
 30  SiteEUIWN(kBtu/sf)                3370 non-null   float64
 31  SourceEUI(kBtu/sf)                3367 non-null   float64
 32  SourceEUIWN(kBtu/sf)              3367 non-null   float64
 33  SiteEnergyUse(kBtu)               3371 non-null   float64
 34  SiteEnergyUseWN(kBtu)             3370 non-null   float64
 35  SteamUse(kBtu)                    3367 non-null   float64
 36  Electricity(kWh)                  3367 non-null   float64
 37  Electricity(kBtu)                 3367 non-null   float64
 38  NaturalGas(therms)                3367 non-null   float64
 39  NaturalGas(kBtu)                  3367 non-null   float64
 40  DefaultData                       3376 non-null   bool
 41  Comments                          0 non-null      float64
 42  ComplianceStatus                  3376 non-null   object
 43  Outlier                           32 non-null     object
 44  TotalGHGEmissions                 3367 non-null   float64
 45  GHGEmissionsIntensity             3367 non-null   float64
dtypes: bool(1), float64(22), int64(8), object(15)
memory usage: 1.2+ MB
```

Le jeu initial de données contient 3376 observations réparties en 46 colonnes/variables.

[ ]:
```python
DataAnalysis.show_columns_population(df, type='bar')
```



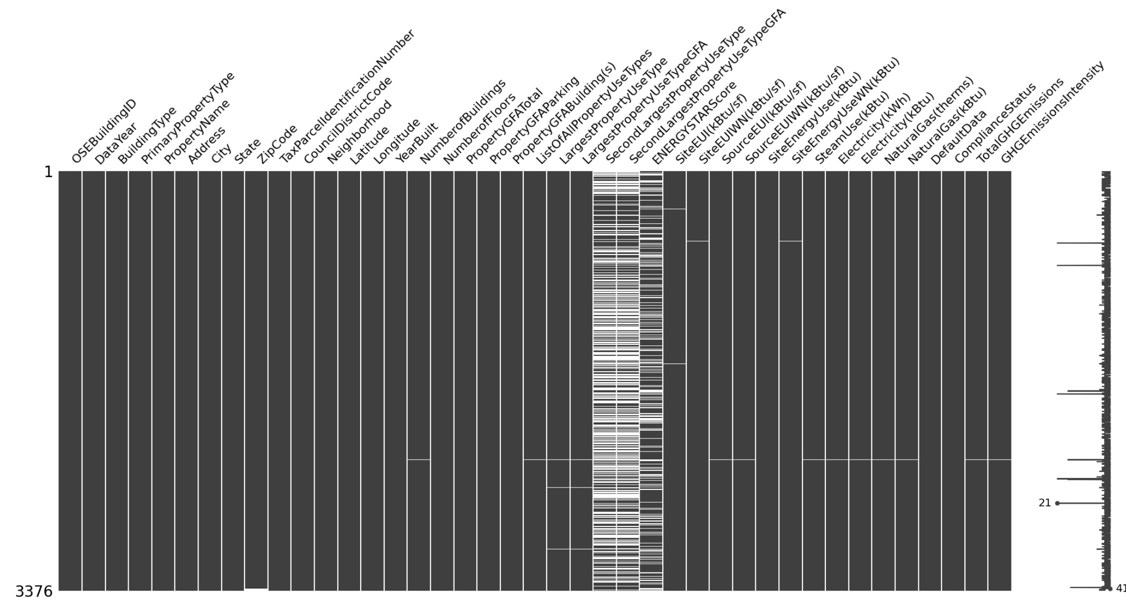
On constate que plusieurs colonnes contiennent trop peu de données pour être exploitées. Nous enlevons donc les colonnes qui ont moins de 30% de données.

[ ]:
```
(df, logs, n_columns_removed) = DataEngineering.
 ↪remove_columns_by_percentage(df, 0.3)

print("Nombre de colonnes supprimées : ", n_columns_removed)

logs
```

```
Nombre de colonnes supprimées :  5
```

[ ]:
```
['La colonne ThirdLargestPropertyUseType a été supprimée car elle ne contient
que 17.65% de valeurs renseignées.',
 'La colonne ThirdLargestPropertyUseTypeGFA a été supprimée car elle ne contient
que 17.65% de valeurs renseignées.',
 'La colonne YearsENERGYSTARCertified a été supprimée car elle ne contient que
3.52% de valeurs renseignées.',
 'La colonne Comments a été supprimée car elle ne contient que 0.0% de valeurs
renseignées.',
 'La colonne Outlier a été supprimée car elle ne contient que 0.95% de valeurs
renseignées.']
```

[ ]:
```
DataAnalysis.show_columns_population(df, type='matrix')
```



[ ]:
```
df.describe()
```

[ ]:
```
       OSEBuildingID  DataYear       ZipCode  CouncilDistrictCode  \
count    3376.000000    3376.0   3360.000000          3376.000000
mean    21208.991114    2016.0  98116.949107             4.439277
std     12223.757015       0.0     18.615205             2.120625
min         1.000000    2016.0  98006.000000             1.000000
25%     19990.750000    2016.0  98105.000000             3.000000
50%     23112.000000    2016.0  98115.000000             4.000000
75%     25994.250000    2016.0  98122.000000             7.000000
max     50226.000000    2016.0  98272.000000             7.000000

          Latitude    Longitude    YearBuilt  NumberofBuildings  \
count  3376.000000  3376.000000  3376.000000        3368.000000
mean     47.624033  -122.334795  1968.573164           1.106888
std       0.047758     0.027203    33.088156           2.108402
min      47.499170  -122.414250  1900.000000           0.000000
25%      47.599860  -122.350662  1948.000000           1.000000
50%      47.618675  -122.332495  1975.000000           1.000000
75%      47.657115  -122.319407  1997.000000           1.000000
max      47.733870  -122.220966  2015.000000         111.000000

       NumberofFloors  PropertyGFATotal  …  SourceEUIWN(kBtu/sf)  \
count     3376.000000      3.376000e+03  …           3367.000000
mean         4.709123      9.483354e+04  …            137.783932
std          5.494465      2.188376e+05  …            139.109807
min          0.000000      1.128500e+04  …             -2.100000
```

```
25%          2.000000     2.848700e+04  ...            78.400002
50%          4.000000     4.417500e+04  ...           101.099998
75%          5.000000     9.099200e+04  ...           148.349998
max         99.000000     9.320156e+06  ...          2620.000000

       SiteEnergyUse(kBtu)  SiteEnergyUseWN(kBtu)  SteamUse(kBtu)  \
count         3.371000e+03           3.370000e+03    3.367000e+03
mean          5.403667e+06           5.276726e+06    2.745959e+05
std           2.161063e+07           1.593879e+07    3.912173e+06
min           0.000000e+00           0.000000e+00    0.000000e+00
25%           9.251286e+05           9.701822e+05    0.000000e+00
50%           1.803753e+06           1.904452e+06    0.000000e+00
75%           4.222455e+06           4.381429e+06    0.000000e+00
max           8.739237e+08           4.716139e+08    1.349435e+08

       Electricity(kWh)  Electricity(kBtu)  NaturalGas(therms)  \
count      3.367000e+03       3.367000e+03        3.367000e+03
mean       1.086639e+06       3.707612e+06        1.368505e+04
std        4.352478e+06       1.485066e+07        6.709781e+04
min       -3.382680e+04      -1.154170e+05        0.000000e+00
25%        1.874229e+05       6.394870e+05        0.000000e+00
50%        3.451299e+05       1.177583e+06        3.237538e+03
75%        8.293178e+05       2.829632e+06        1.189033e+04
max        1.925775e+08       6.570744e+08        2.979090e+06

       NaturalGas(kBtu)  TotalGHGEmissions  GHGEmissionsIntensity
count      3.367000e+03        3367.000000            3367.000000
mean       1.368505e+06         119.723971               1.175916
std        6.709781e+06         538.832227               1.821452
min        0.000000e+00          -0.800000              -0.020000
25%        0.000000e+00           9.495000               0.210000
50%        3.237540e+05          33.920000               0.610000
75%        1.189034e+06          93.940000               1.370000
max        2.979090e+08       16870.980000              34.090000

[8 rows x 28 columns]
```

Grâce à cette analyse, nous pouvons voir que : - La colonne DataYear semble contenir toujours la même valeur - Les colonne OSEBuildingID, PropertyName, Address, City, State, TaxParcelIdentificationNumber, CouncilDistrictCode ne seront pas utile pour nos algorithmes, car bien trop spécifiques

Nous les enlevons donc du dataset.

```
[ ]: df = DataEngineering.remove_columns_by_name(df, ['OSEBuildingID', 'DataYear',
     'PropertyName', 'Address', 'City', 'State', 'TaxParcelIdentificationNumber',
     'CouncilDistrictCode'])
```

1.2 Analyse des colonnes de type number et valeurs aberrantes

[ ]:
```python
# On liste les colonnes qui ont des valeurs de type number
numericColumns = df.select_dtypes(include=['number']).columns
```

[ ]:
```python
import matplotlib.pyplot as plt
import math

data_to_plot = [df[col].dropna() for col in numericColumns]

# Calcule du nombre de lignes nécessaires
num_rows = math.ceil(len(numericColumns) / 2)

fig, axs = plt.subplots(num_rows, 2, figsize=(12*2, 4*num_rows))
axs = axs.ravel()

for idx, col in enumerate(numericColumns):
    axs[idx].boxplot(data_to_plot[idx], vert=True, patch_artist=True)
    axs[idx].set_title(f'Diagramme à moustache de {col}')
    axs[idx].set_ylabel('Valeurs')
    axs[idx].set_xticks([])

# Supprimer les axes non utilisés s'il y en a
for idx in range(len(numericColumns), num_rows*2):
    axs[idx].axis('off')

plt.tight_layout()
plt.show()
```

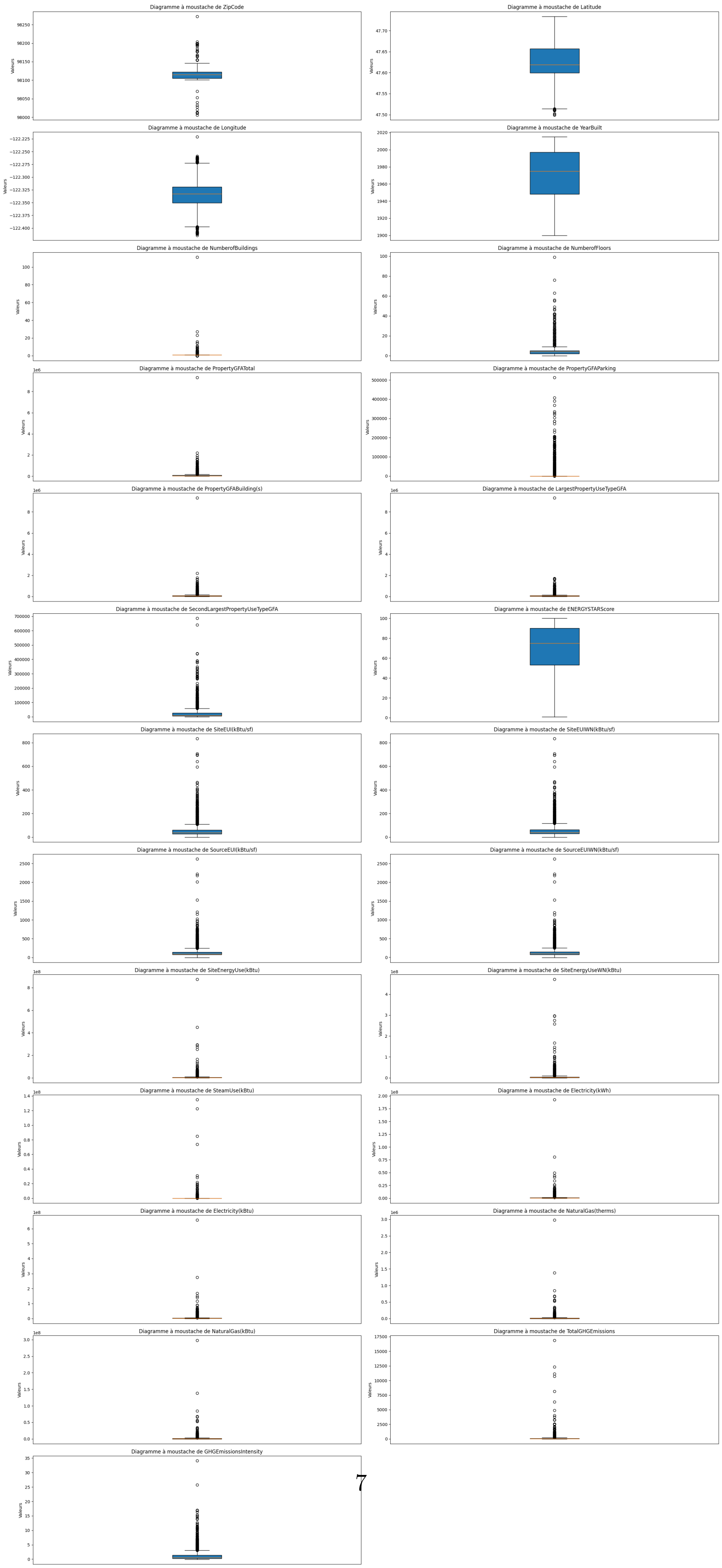


Diagramme à moustache de ZipCode
Diagramme à moustache de Latitude
Diagramme à moustache de Longitude
Diagramme à moustache de YearBuilt
Diagramme à moustache de NumberofBuildings
Diagramme à moustache de NumberofFloors
Diagramme à moustache de PropertyGFATotal
Diagramme à moustache de PropertyGFAParking
Diagramme à moustache de PropertyGFABuilding(s)
Diagramme à moustache de LargestPropertyUseTypeGFA
Diagramme à moustache de SecondLargestPropertyUseTypeGFA
Diagramme à moustache de ENERGYSTARScore
Diagramme à moustache de SiteEUI(kBtu/sf)
Diagramme à moustache de SiteEUIWN(kBtu/sf)
Diagramme à moustache de SourceEUI(kBtu/sf)
Diagramme à moustache de SourceEUIWN(kBtu/sf)
Diagramme à moustache de SiteEnergyUse(kBtu)
Diagramme à moustache de SiteEnergyUseWN(kBtu)
Diagramme à moustache de SteamUse(kBtu)
Diagramme à moustache de Electricity(kWh)
Diagramme à moustache de Electricity(kBtu)
Diagramme à moustache de NaturalGas(therms)
Diagramme à moustache de NaturalGas(kBtu)
Diagramme à moustache de TotalGHGEmissions
Diagramme à moustache de GHGEmissionsIntensity
Valeurs

[ ]:
```python
# Créer un dataframe ne contenant que les colonnes de type number
df_num = df.select_dtypes(include=['number'])
```

[ ]:
```python
correlation_matrix = df_num.corr()

# On sauvegarde la matrice de corrélation
correlation_matrix.to_csv('data/correlation_matrix.csv')

print(correlation_matrix)
```

```
                                ZipCode  Latitude  Longitude  YearBuilt  \
ZipCode                        1.000000  0.030536  -0.120893   0.094818
Latitude                       0.030536  1.000000   0.005250   0.117239
Longitude                     -0.120893  0.005250   1.000000  -0.051111
YearBuilt                      0.094818  0.117239  -0.051111   1.000000
NumberofBuildings             -0.009582  0.020646   0.017858  -0.023712
NumberofFloors                -0.117719 -0.023980  -0.026054   0.146214
PropertyGFATotal              -0.052669 -0.018162   0.025383   0.100417
PropertyGFAParking            -0.076657 -0.001167  -0.003374   0.183176
PropertyGFABuilding(s)        -0.043509 -0.018932   0.027237   0.077203
LargestPropertyUseTypeGFA     -0.036931 -0.015277   0.029323   0.070187
SecondLargestPropertyUseTypeGFA -0.059226 -0.052773   0.018545   0.197447
ENERGYSTARScore                0.002822  0.079948  -0.026404   0.028813
SiteEUI(kBtu/sf)              -0.070757 -0.012730   0.027695  -0.019642
SiteEUIWN(kBtu/sf)            -0.076659 -0.016918   0.027467  -0.030900
SourceEUI(kBtu/sf)            -0.050815 -0.001273   0.019153   0.043394
SourceEUIWN(kBtu/sf)          -0.055920 -0.002326   0.018590   0.039066
SiteEnergyUse(kBtu)           -0.041811 -0.021314   0.033803   0.027251
SiteEnergyUseWN(kBtu)         -0.050046 -0.041985   0.032156   0.069277
SteamUse(kBtu)                -0.038624 -0.015448   0.018502  -0.018234
Electricity(kWh)              -0.036909 -0.018924   0.026537   0.039849
Electricity(kBtu)             -0.036909 -0.018924   0.026537   0.039849
NaturalGas(therms)            -0.028650 -0.020860   0.033180   0.023275
NaturalGas(kBtu)              -0.028650 -0.020860   0.033180   0.023275
TotalGHGEmissions             -0.047686 -0.026089   0.037411   0.012831
GHGEmissionsIntensity         -0.083394 -0.040727   0.039365  -0.146212

                               NumberofBuildings  NumberofFloors  \
ZipCode                                -0.009582       -0.117719
Latitude                                0.020646       -0.023980
Longitude                               0.017858       -0.026054
YearBuilt                              -0.023712        0.146214
NumberofBuildings                       1.000000       -0.026386
NumberofFloors                         -0.026386        1.000000
PropertyGFATotal                        0.693412        0.400488
```

PropertyGFAParking	-0.004774	0.420489
PropertyGFABuilding(s)	0.730487	0.356107
LargestPropertyUseTypeGFA	0.758749	0.339212
SecondLargestPropertyUseTypeGFA	0.112821	0.469908
ENERGYSTARScore	-0.004900	0.023540
SiteEUI(kBtu/sf)	0.033003	0.009351
SiteEUIWN(kBtu/sf)	0.007034	-0.000857
SourceEUI(kBtu/sf)	0.031599	0.037679
SourceEUIWN(kBtu/sf)	0.003896	0.031666
SiteEnergyUse(kBtu)	0.690712	0.205864
SiteEnergyUseWN(kBtu)	0.090486	0.293096
SteamUse(kBtu)	0.397588	0.079497
Electricity(kWh)	0.735028	0.251514
Electricity(kBtu)	0.735028	0.251514
NaturalGas(therms)	0.062324	0.065226
NaturalGas(kBtu)	0.062324	0.065226
TotalGHGEmissions	0.405261	0.136014
GHGEmissionsIntensity	0.027564	-0.042445

	PropertyGFATotal	PropertyGFAParking \
ZipCode	-0.052669	-0.076657
Latitude	-0.018162	-0.001167
Longitude	0.025383	-0.003374
YearBuilt	0.100417	0.183176
NumberofBuildings	0.693412	-0.004774
NumberofFloors	0.400488	0.420489
PropertyGFATotal	1.000000	0.402580
PropertyGFAParking	0.402580	1.000000
PropertyGFABuilding(s)	0.989823	0.268217
LargestPropertyUseTypeGFA	0.974113	0.300578
SecondLargestPropertyUseTypeGFA	0.807411	0.477959
ENERGYSTARScore	0.067342	0.049559
SiteEUI(kBtu/sf)	0.071020	0.097110
SiteEUIWN(kBtu/sf)	0.040080	0.089160
SourceEUI(kBtu/sf)	0.083315	0.134553
SourceEUIWN(kBtu/sf)	0.054662	0.128065
SiteEnergyUse(kBtu)	0.796781	0.171544
SiteEnergyUseWN(kBtu)	0.400813	0.238464
SteamUse(kBtu)	0.440568	0.013501
Electricity(kWh)	0.849576	0.220356
Electricity(kBtu)	0.849576	0.220356
NaturalGas(therms)	0.183408	0.058547
NaturalGas(kBtu)	0.183408	0.058547
TotalGHGEmissions	0.531436	0.088625
GHGEmissionsIntensity	0.020105	-0.043160

	PropertyGFABuilding(s) \
ZipCode	-0.043509

```
Latitude                                    -0.018932
Longitude                                    0.027237
YearBuilt                                    0.077203
NumberofBuildings                            0.730487
NumberofFloors                               0.356107
PropertyGFATotal                             0.989823
PropertyGFAParking                           0.268217
PropertyGFABuilding(s)                       1.000000
LargestPropertyUseTypeGFA                    0.978422
SecondLargestPropertyUseTypeGFA              0.791727
ENERGYSTARScore                              0.064530
SiteEUI(kBtu/sf)                             0.059602
SiteEUIWN(kBtu/sf)                           0.028332
SourceEUI(kBtu/sf)                           0.066762
SourceEUIWN(kBtu/sf)                         0.037617
SiteEnergyUse(kBtu)                          0.811866
SiteEnergyUseWN(kBtu)                        0.384778
SteamUse(kBtu)                               0.461554
Electricity(kWh)                             0.859833
Electricity(kBtu)                            0.859833
NaturalGas(therms)                           0.183916
NaturalGas(kBtu)                             0.183916
TotalGHGEmissions                            0.545503
GHGEmissionsIntensity                        0.027868

                                 LargestPropertyUseTypeGFA  …  \
ZipCode                                          -0.036931  …
Latitude                                         -0.015277  …
Longitude                                         0.029323  …
YearBuilt                                         0.070187  …
NumberofBuildings                                 0.758749  …
NumberofFloors                                    0.339212  …
PropertyGFATotal                                  0.974113  …
PropertyGFAParking                                0.300578  …
PropertyGFABuilding(s)                            0.978422  …
LargestPropertyUseTypeGFA                         1.000000  …
SecondLargestPropertyUseTypeGFA                   0.769156  …
ENERGYSTARScore                                   0.058088  …
SiteEUI(kBtu/sf)                                  0.057341  …
SiteEUIWN(kBtu/sf)                                0.026611  …
SourceEUI(kBtu/sf)                                0.062135  …
SourceEUIWN(kBtu/sf)                              0.032467  …
SiteEnergyUse(kBtu)                               0.836185  …
SiteEnergyUseWN(kBtu)                             0.393574  …
SteamUse(kBtu)                                    0.497636  …
Electricity(kWh)                                  0.875059  …
Electricity(kBtu)                                 0.875059  …
NaturalGas(therms)                                0.198753  …
```

NaturalGas(kBtu)	0.198753	…
TotalGHGEmissions	0.578487	…
GHGEmissionsIntensity	0.053555	…

	SourceEUIWN(kBtu/sf)	SiteEnergyUse(kBtu)	\
ZipCode	-0.055920	-0.041811	
Latitude	-0.002326	-0.021314	
Longitude	0.018590	0.033803	
YearBuilt	0.039066	0.027251	
NumberofBuildings	0.003896	0.690712	
NumberofFloors	0.031666	0.205864	
PropertyGFATotal	0.054662	0.796781	
PropertyGFAParking	0.128065	0.171544	
PropertyGFABuilding(s)	0.037617	0.811866	
LargestPropertyUseTypeGFA	0.032467	0.836185	
SecondLargestPropertyUseTypeGFA	0.100272	0.630121	
ENERGYSTARScore	-0.311054	-0.090196	
SiteEUI(kBtu/sf)	0.940204	0.300966	
SiteEUIWN(kBtu/sf)	0.938051	0.272799	
SourceEUI(kBtu/sf)	0.994317	0.296804	
SourceEUIWN(kBtu/sf)	1.000000	0.268986	
SiteEnergyUse(kBtu)	0.268986	1.000000	
SiteEnergyUseWN(kBtu)	0.387075	0.715149	
SteamUse(kBtu)	0.076800	0.604323	
Electricity(kWh)	0.293289	0.956556	
Electricity(kBtu)	0.293289	0.956556	
NaturalGas(therms)	0.176670	0.514408	
NaturalGas(kBtu)	0.176670	0.514408	
TotalGHGEmissions	0.216232	0.862668	
GHGEmissionsIntensity	0.529583	0.310729	

	SiteEnergyUseWN(kBtu)	SteamUse(kBtu)	\
ZipCode	-0.050046	-0.038624	
Latitude	-0.041985	-0.015448	
Longitude	0.032156	0.018502	
YearBuilt	0.069277	-0.018234	
NumberofBuildings	0.090486	0.397588	
NumberofFloors	0.293096	0.079497	
PropertyGFATotal	0.400813	0.440568	
PropertyGFAParking	0.238464	0.013501	
PropertyGFABuilding(s)	0.384778	0.461554	
LargestPropertyUseTypeGFA	0.393574	0.497636	
SecondLargestPropertyUseTypeGFA	0.626631	0.263866	
ENERGYSTARScore	-0.090163	-0.040441	
SiteEUI(kBtu/sf)	0.397474	0.106617	
SiteEUIWN(kBtu/sf)	0.394437	0.093233	
SourceEUI(kBtu/sf)	0.388278	0.090563	
SourceEUIWN(kBtu/sf)	0.387075	0.076800	

SiteEnergyUse(kBtu)	0.715149	0.604323
SiteEnergyUseWN(kBtu)	1.000000	0.472701
SteamUse(kBtu)	0.472701	1.000000
Electricity(kWh)	0.587712	0.546965
Electricity(kBtu)	0.587712	0.546965
NaturalGas(therms)	0.727617	0.026827
NaturalGas(kBtu)	0.727617	0.026827
TotalGHGEmissions	0.859042	0.683254
GHGEmissionsIntensity	0.434785	0.194053

	Electricity(kWh)	Electricity(kBtu) \
ZipCode	-0.036909	-0.036909
Latitude	-0.018924	-0.018924
Longitude	0.026537	0.026537
YearBuilt	0.039849	0.039849
NumberofBuildings	0.735028	0.735028
NumberofFloors	0.251514	0.251514
PropertyGFATotal	0.849576	0.849576
PropertyGFAParking	0.220356	0.220356
PropertyGFABuilding(s)	0.859833	0.859833
LargestPropertyUseTypeGFA	0.875059	0.875059
SecondLargestPropertyUseTypeGFA	0.634493	0.634493
ENERGYSTARScore	-0.057299	-0.057299
SiteEUI(kBtu/sf)	0.285053	0.285053
SiteEUIWN(kBtu/sf)	0.253013	0.253013
SourceEUI(kBtu/sf)	0.323180	0.323180
SourceEUIWN(kBtu/sf)	0.293289	0.293289
SiteEnergyUse(kBtu)	0.956556	0.956556
SiteEnergyUseWN(kBtu)	0.587712	0.587712
SteamUse(kBtu)	0.546965	0.546965
Electricity(kWh)	1.000000	1.000000
Electricity(kBtu)	1.000000	1.000000
NaturalGas(therms)	0.290987	0.290987
NaturalGas(kBtu)	0.290987	0.290987
TotalGHGEmissions	0.691111	0.691111
GHGEmissionsIntensity	0.177903	0.177903

	NaturalGas(therms)	NaturalGas(kBtu) \
ZipCode	-0.028650	-0.028650
Latitude	-0.020860	-0.020860
Longitude	0.033180	0.033180
YearBuilt	0.023275	0.023275
NumberofBuildings	0.062324	0.062324
NumberofFloors	0.065226	0.065226
PropertyGFATotal	0.183408	0.183408
PropertyGFAParking	0.058547	0.058547
PropertyGFABuilding(s)	0.183916	0.183916
LargestPropertyUseTypeGFA	0.198753	0.198753

SecondLargestPropertyUseTypeGFA	0.387937	0.387937
ENERGYSTARScore	-0.102422	-0.102422
SiteEUI(kBtu/sf)	0.260207	0.260207
SiteEUIWN(kBtu/sf)	0.262725	0.262725
SourceEUI(kBtu/sf)	0.177507	0.177507
SourceEUIWN(kBtu/sf)	0.176670	0.176670
SiteEnergyUse(kBtu)	0.514408	0.514408
SiteEnergyUseWN(kBtu)	0.727617	0.727617
SteamUse(kBtu)	0.026827	0.026827
Electricity(kWh)	0.290987	0.290987
Electricity(kBtu)	0.290987	0.290987
NaturalGas(therms)	1.000000	1.000000
NaturalGas(kBtu)	1.000000	1.000000
TotalGHGEmissions	0.732294	0.732294
GHGEmissionsIntensity	0.494864	0.494864

	TotalGHGEmissions	GHGEmissionsIntensity
ZipCode	-0.047686	-0.083394
Latitude	-0.026089	-0.040727
Longitude	0.037411	0.039365
YearBuilt	0.012831	-0.146212
NumberofBuildings	0.405261	0.027564
NumberofFloors	0.136014	-0.042445
PropertyGFATotal	0.531436	0.020105
PropertyGFAParking	0.088625	-0.043160
PropertyGFABuilding(s)	0.545503	0.027868
LargestPropertyUseTypeGFA	0.578487	0.053555
SecondLargestPropertyUseTypeGFA	0.506537	0.105724
ENERGYSTARScore	-0.101633	-0.269263
SiteEUI(kBtu/sf)	0.286608	0.730897
SiteEUIWN(kBtu/sf)	0.274616	0.745573
SourceEUI(kBtu/sf)	0.230243	0.524232
SourceEUIWN(kBtu/sf)	0.216232	0.529583
SiteEnergyUse(kBtu)	0.862668	0.310729
SiteEnergyUseWN(kBtu)	0.859042	0.434785
SteamUse(kBtu)	0.683254	0.194053
Electricity(kWh)	0.691111	0.177903
Electricity(kBtu)	0.691111	0.177903
NaturalGas(therms)	0.732294	0.494864
NaturalGas(kBtu)	0.732294	0.494864
TotalGHGEmissions	1.000000	0.470212
GHGEmissionsIntensity	0.470212	1.000000

[25 rows x 25 columns]

1.3 Visualisation de la matrice de correlation

[ ]:
```
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

1.4 Analyse des colonnes contenant des valeurs autres que des numbers

[ ]:
```
# create a dataframe with columns which are not number type
df_not_num = df.select_dtypes(exclude=['number'])
```

[ ]:
```
df_not_num.info()
```

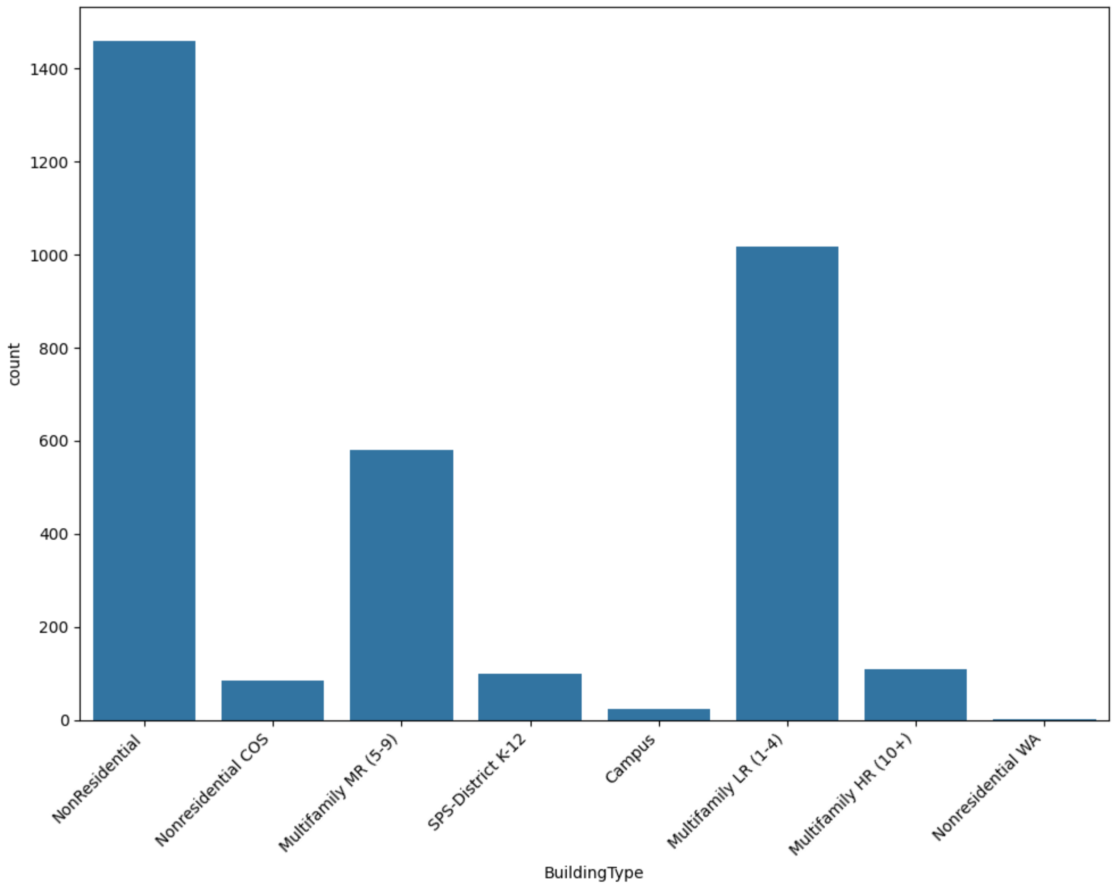
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3376 entries, 0 to 3375
Data columns (total 8 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   BuildingType                  3376 non-null   object
```

```
 1   PrimaryPropertyType           3376 non-null   object
 2   Neighborhood                  3376 non-null   object
 3   ListOfAllPropertyUseTypes     3367 non-null   object
 4   LargestPropertyUseType        3356 non-null   object
 5   SecondLargestPropertyUseType  1679 non-null   object
 6   DefaultData                   3376 non-null   bool
 7   ComplianceStatus              3376 non-null   object
dtypes: bool(1), object(7)
memory usage: 188.1+ KB
```

[ ]:
```
df_not_num.sample(5)
```

[ ]:
```
               BuildingType   PrimaryPropertyType          Neighborhood  \
2980  Multifamily LR (1-4)  Low-Rise Multifamily  MAGNOLIA / QUEEN ANNE
893   Multifamily MR (5-9)  Mid-Rise Multifamily               DOWNTOWN
317         NonResidential          Large Office            LAKE UNION
539         NonResidential   Distribution Center      GREATER DUWAMISH
3217  Multifamily MR (5-9)  Mid-Rise Multifamily               DELRIDGE

               ListOfAllPropertyUseTypes LargestPropertyUseType  \
2980                 Multifamily Housing    Multifamily Housing
893   Multifamily Housing, Office, Other    Multifamily Housing
317                               Office                 Office
539                  Distribution Center    Distribution Center
3217                 Multifamily Housing    Multifamily Housing

     SecondLargestPropertyUseType  DefaultData ComplianceStatus
2980                          NaN        False        Compliant
893                         Other        False        Compliant
317                           NaN        False        Compliant
539                           NaN        False        Compliant
3217                          NaN        False        Compliant
```

[ ]:
```
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='BuildingType', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')
plt.tight_layout()
```



1.5 Nous allons simplifier la colonne 'BuildingType' en classant les valeurs en 2 catégories :

- 'Multifamily' : valeur 0
- 'Autres' : valeur 1

[ ]:
```
# Nous allons simplifier la colonne 'BuildingType' en classant les valeurs en 2␣
↪catégories
df['BuildingType'].unique()
```

[ ]:
```
array(['NonResidential', 'Nonresidential COS', 'Multifamily MR (5-9)',
       'SPS-District K-12', 'Campus', 'Multifamily LR (1-4)',
       'Multifamily HR (10+)', 'Nonresidential WA'], dtype=object)
```

[ ]:
```
multifamily_values = ['Multifamily LR (1-4)', 'Multifamily MR (5-9)',␣
↪'Multifamily HR (10+)']
non_multifamily_values = ['NonResidential', 'Nonresidential COS',␣
↪'Nonresidential WA', 'SPS-District K-12', 'Campus']
```

```
# replace values in column 'BuildingType'
df['BuildingType'] = df['BuildingType'].replace(multifamily_values,␣
↪'Multifamily')

# test wether the value in column 'BuildingType' is 0. If not, replace by 1
df['BuildingType'] = df['BuildingType'].apply(lambda x: 'Other' if x !=␣
↪'Multifamily' else x)
```

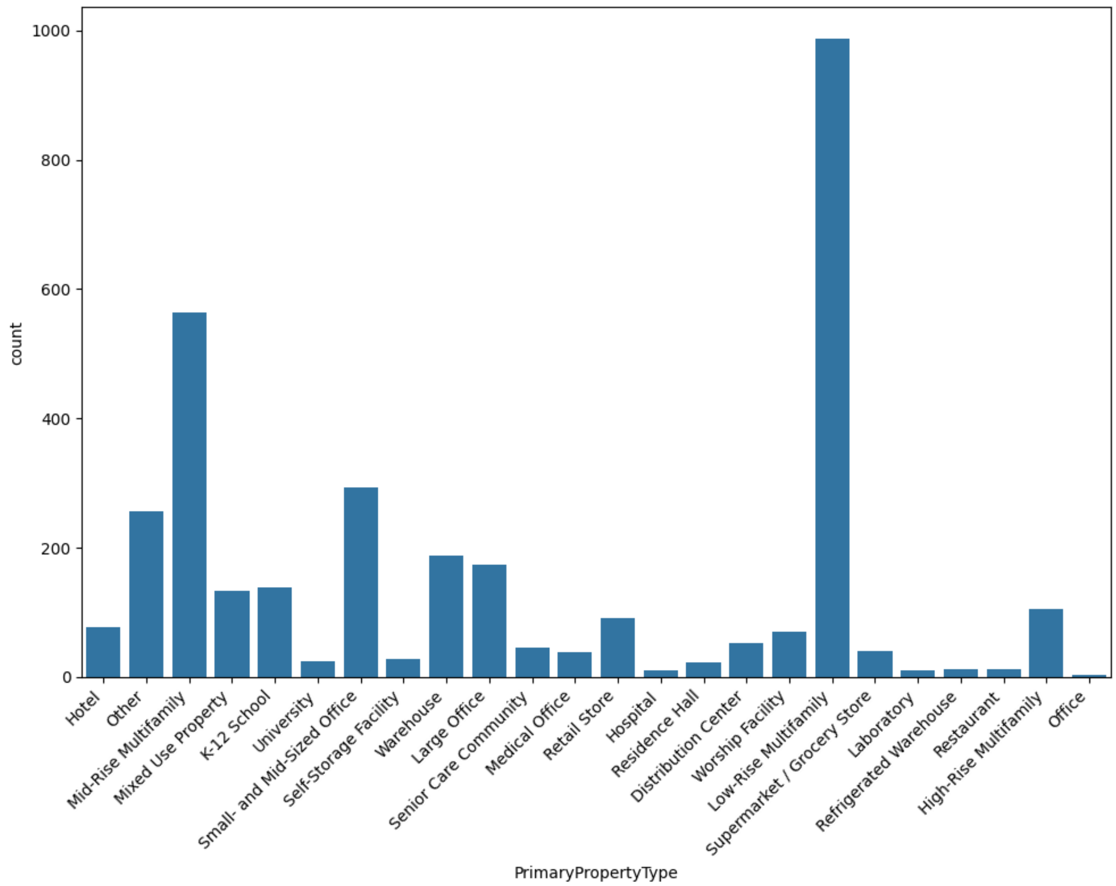
[ ]:
```
# Vérifions que les valeurs sont bien uniquement des 0 et des 1
df['BuildingType'].unique()
```

[ ]:
```
array(['Other', 'Multifamily'], dtype=object)
```

[ ]:
```
encoder = OneHotEncoder()

encoded_data = encoder.fit_transform(df[['BuildingType']])

#encoded_df = pd.DataFrame(encoded_data, columns=encoder.
↪get_feature_names_out(['BuildingType']))
encoded_df = pd.DataFrame(encoded_data)

df = pd.concat([df, encoded_df], axis=1)
```

Les valeurs de BuildingType sont maintenant remplacées.

[ ]:
```
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='PrimaryPropertyType', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
↪horizontalalignment='right')
plt.tight_layout()
```



[ ]:
```
# list possible values of column 'PrimaryPropertyType'
df['PrimaryPropertyType'].unique()
```

[ ]:
```
array(['Hotel', 'Other', 'Mid-Rise Multifamily', 'Mixed Use Property',
       'K-12 School', 'University', 'Small- and Mid-Sized Office',
       'Self-Storage Facility', 'Warehouse', 'Large Office',
       'Senior Care Community', 'Medical Office', 'Retail Store',
       'Hospital', 'Residence Hall', 'Distribution Center',
       'Worship Facility', 'Low-Rise Multifamily',
       'Supermarket / Grocery Store', 'Laboratory',
       'Refrigerated Warehouse', 'Restaurant', 'High-Rise Multifamily',
       'Office'], dtype=object)
```

[ ]:
```
# Lists for each category
residential_buildings = ["Low-Rise Multifamily", "Mid-Rise Multifamily",␣
 ↪"High-Rise Multifamily", "Senior Care Community", "Residence Hall"]
commercial_office_buildings = ["Hotel", "Small- and Mid-Sized Office", "Large␣
 ↪Office", "Retail Store", "Medical Office", "Restaurant", "Laboratory"]
educational_healthcare_facilities = ["K-12 School", "University", "Hospital"]
```

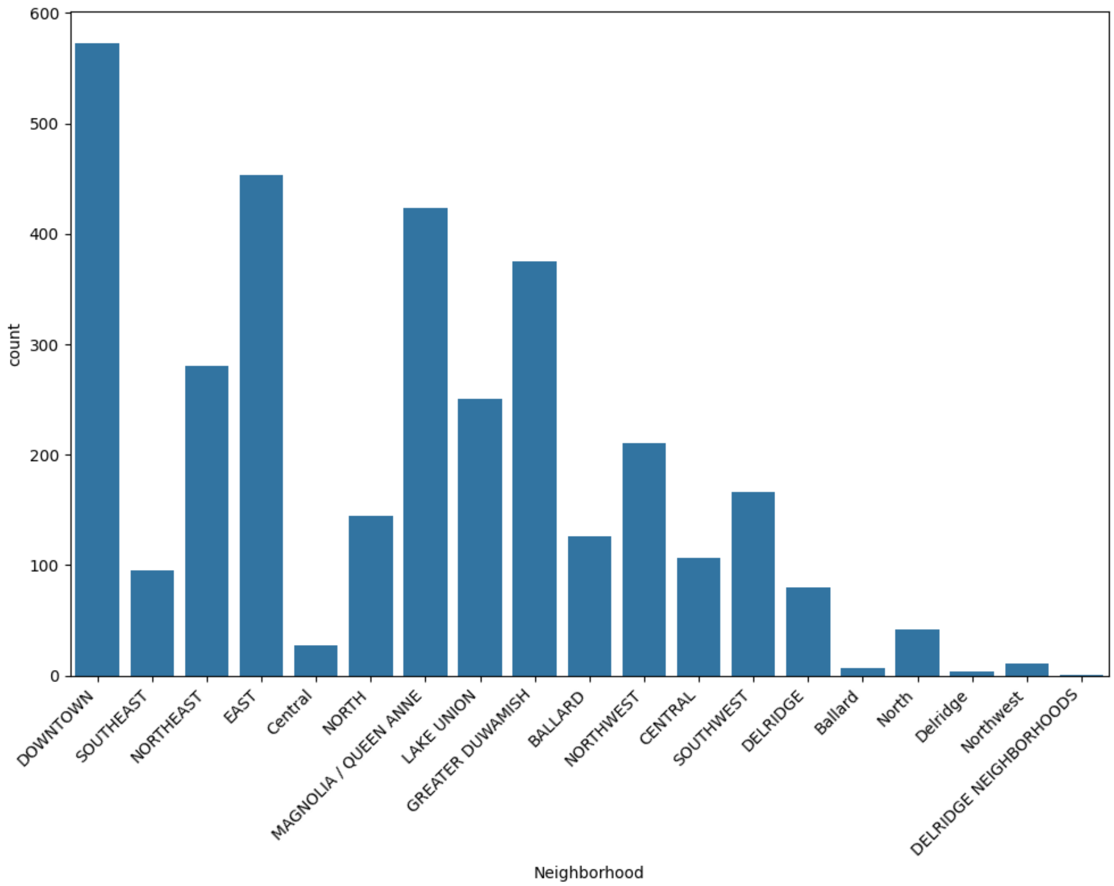
```
industrial_special_purpose = ["Warehouse", "Distribution Center", "Refrigerated␣
↪Warehouse", "Self-Storage Facility", "Mixed Use Property", "Supermarket /␣
↪Grocery Store", "Worship Facility", "Office", "Other"]

# Function to map property type to a category number
def property_type_to_number(property_type):
    if property_type in residential_buildings:
        return 'Residential'
    elif property_type in commercial_office_buildings:
        return 'Commercial'
    elif property_type in educational_healthcare_facilities:
        return 'EducationalHealthcare'
    elif property_type in industrial_special_purpose:
        return 'IndustrialOther'
    else:
        return 'IndustrialOther'  # For any property type that doesn't fit into␣
↪these categories

df['PrimaryPropertyType'] = df['PrimaryPropertyType'].apply(lambda x:␣
↪property_type_to_number(x))
```

[ ]:
```
# list possible values of column 'PrimaryPropertyType'
df['PrimaryPropertyType'].unique()
```

[ ]:
```
array(['Commercial', 'IndustrialOther', 'Residential',
       'EducationalHealthcare'], dtype=object)
```

[ ]:
```
encoder = OneHotEncoder()

encoded_data = encoder.fit_transform(df[['PrimaryPropertyType']])

encoded_df = pd.DataFrame(encoded_data)

df = pd.concat([df, encoded_df], axis=1)
```

[ ]:
```
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='Neighborhood', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
↪horizontalalignment='right')
plt.tight_layout()
```



[ ]:
```
# list possible values of column 'Neighborhood'
df['Neighborhood'].unique()
```

[ ]:
```
array(['DOWNTOWN', 'SOUTHEAST', 'NORTHEAST', 'EAST', 'Central', 'NORTH',
       'MAGNOLIA / QUEEN ANNE', 'LAKE UNION', 'GREATER DUWAMISH',
       'BALLARD', 'NORTHWEST', 'CENTRAL', 'SOUTHWEST', 'DELRIDGE',
       'Ballard', 'North', 'Delridge', 'Northwest',
       'DELRIDGE NEIGHBORHOODS'], dtype=object)
```

[ ]:
```
# Function to normalize and regroup neighborhood names
def normalize_neighborhood(neighborhood):
    # Normalize the case (convert all to uppercase)
    normalized_neighborhood = neighborhood.upper()

    # Special handling for 'DELRIDGE' to include 'DELRIDGE NEIGHBORHOODS'
    if 'DELRIDGE' in normalized_neighborhood:
        return 'DELRIDGE'

    return normalized_neighborhood
```

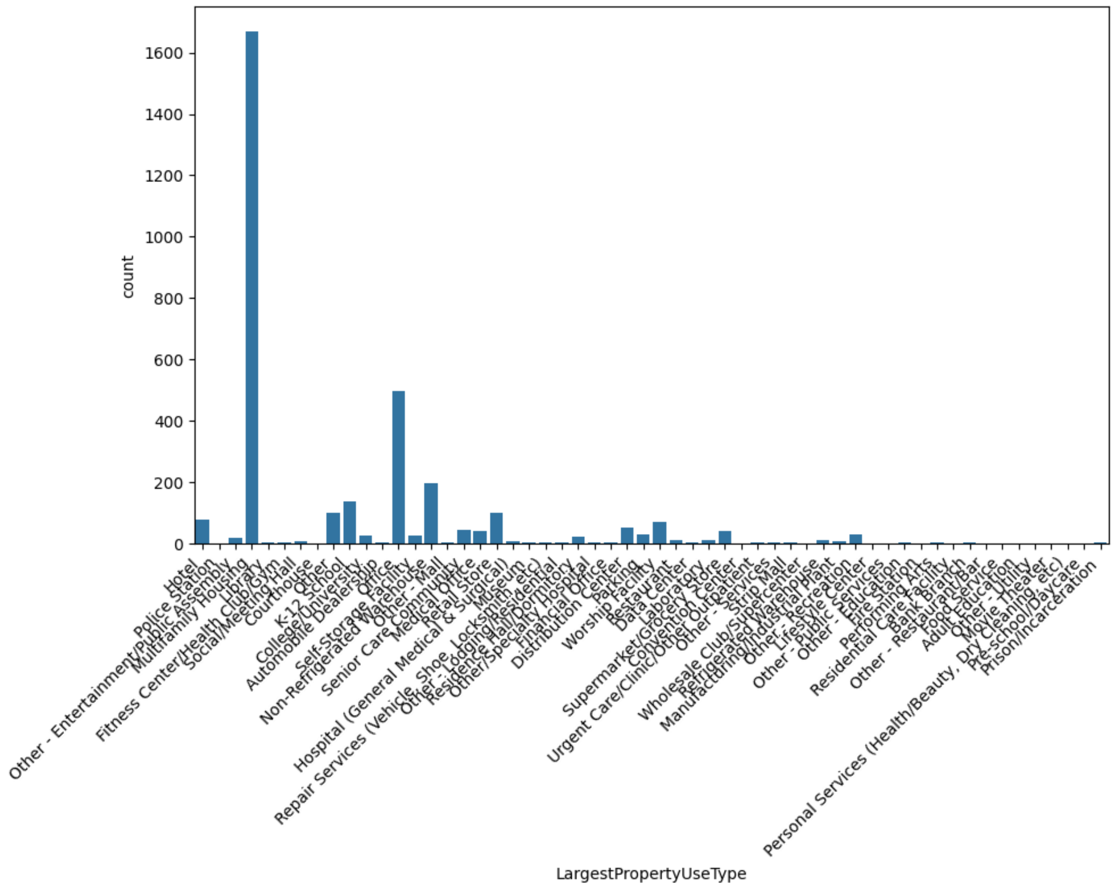
```
# Assuming you have a DataFrame 'df' with a column 'Neighborhood'
# Apply the function to the DataFrame
df['Neighborhood'] = df['Neighborhood'].apply(lambda x:␣
↪normalize_neighborhood(x))
```

[ ]:
```
# list possible values of column 'Neighborhood'
df['Neighborhood'].unique()
```

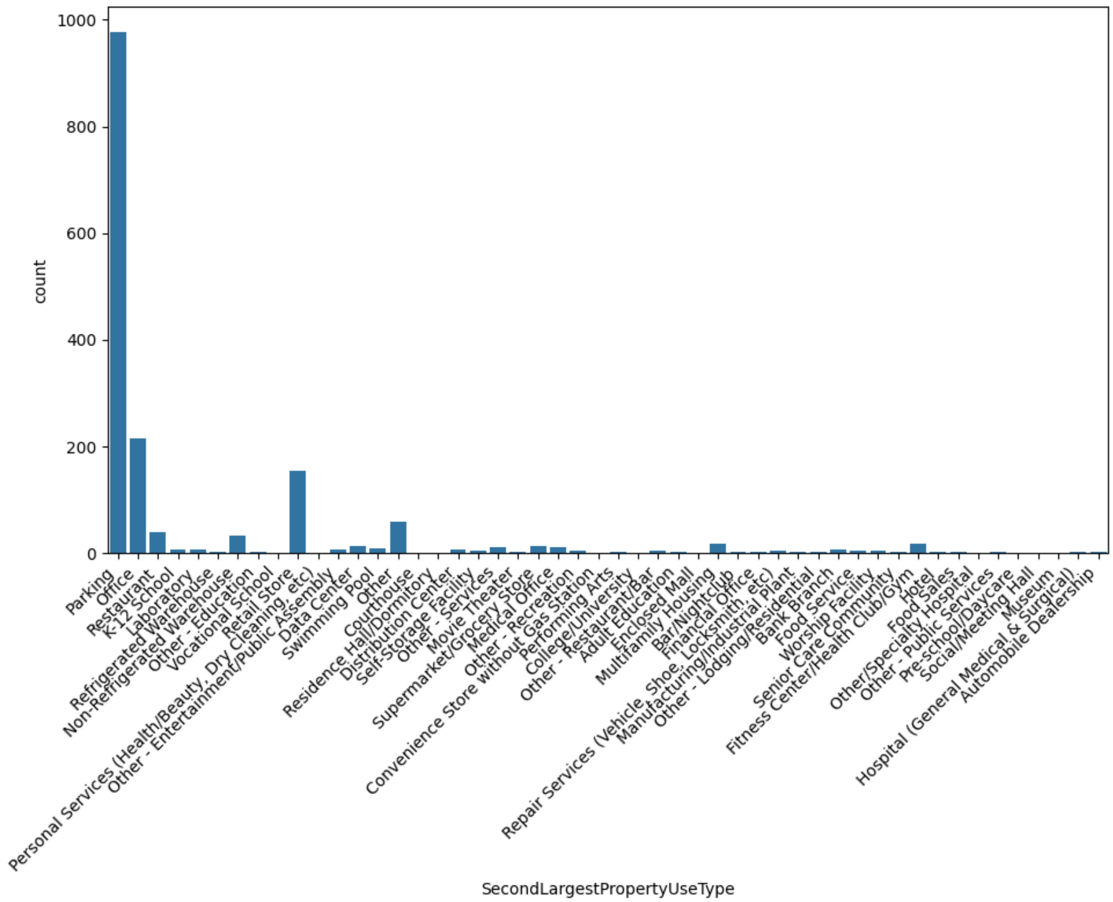
[ ]:
```
array(['DOWNTOWN', 'SOUTHEAST', 'NORTHEAST', 'EAST', 'CENTRAL', 'NORTH',
       'MAGNOLIA / QUEEN ANNE', 'LAKE UNION', 'GREATER DUWAMISH',
       'BALLARD', 'NORTHWEST', 'SOUTHWEST', 'DELRIDGE'], dtype=object)
```

[ ]:
```
encoder = OneHotEncoder()

encoded_data = encoder.fit_transform(df[['Neighborhood']])

encoded_df = pd.DataFrame(encoded_data)

df = pd.concat([df, encoded_df], axis=1)

# Now df has the original data along with the one-hot encoded neighborhood␣
↪columns
```

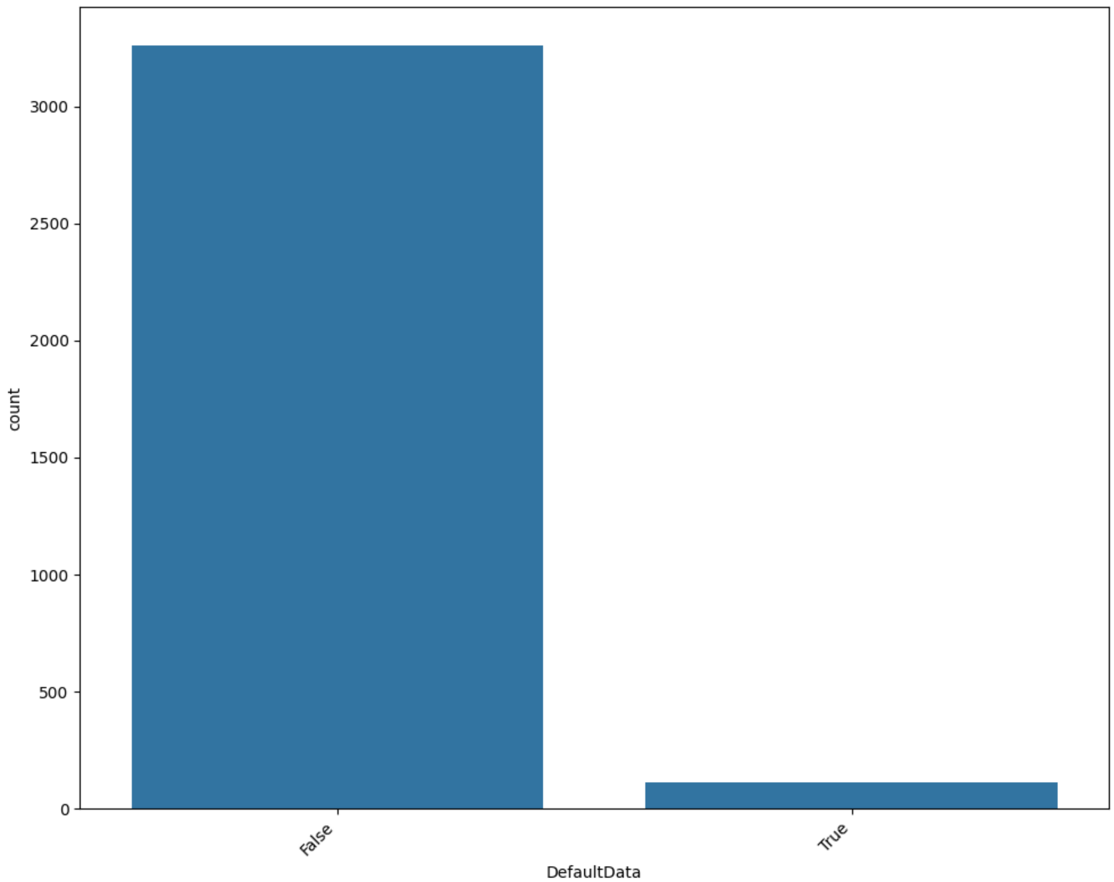
[ ]:
```
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='LargestPropertyUseType', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
↪horizontalalignment='right')
plt.tight_layout()
```



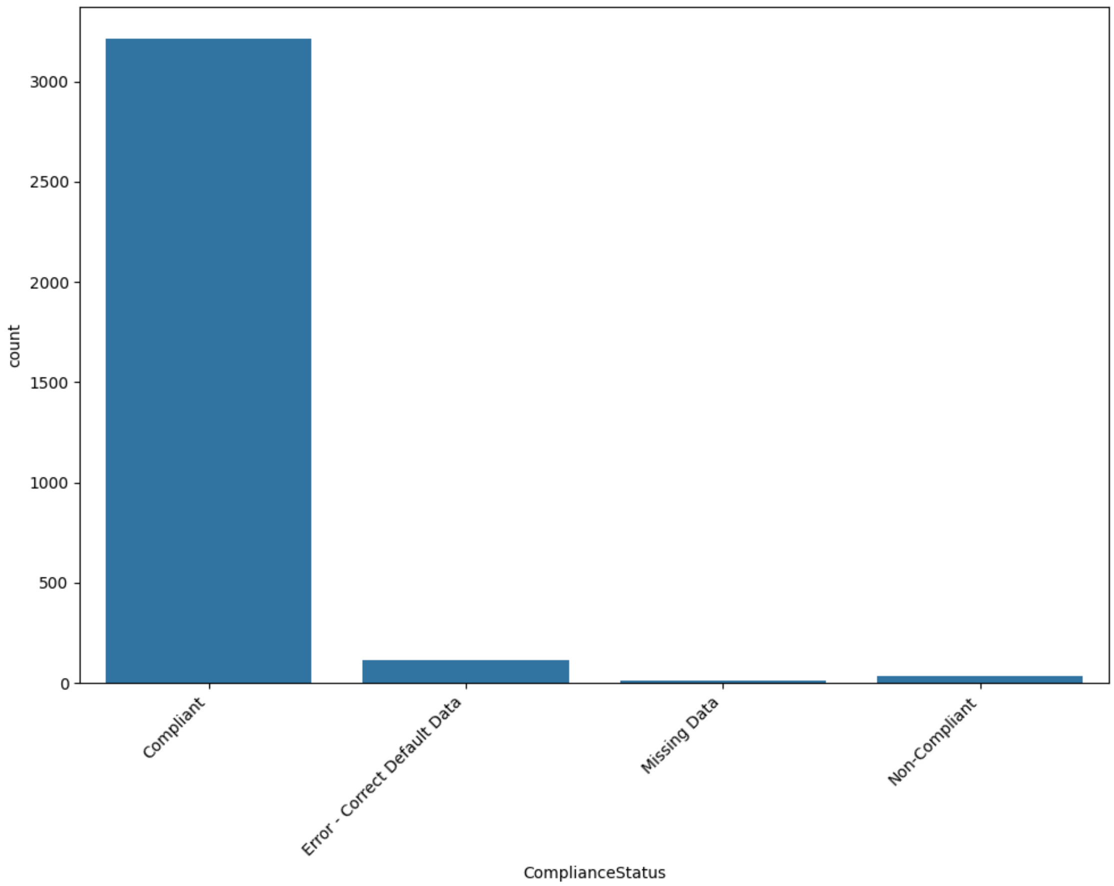
[ ]:
```
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='SecondLargestPropertyUseType', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
  ↪horizontalalignment='right')
plt.tight_layout()
```



[ ]:
```python
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='DefaultData', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
    ↪horizontalalignment='right')
plt.tight_layout()
```



[ ]:
```
plt.figure(figsize=(10, 8))
count_plot = sns.countplot(x='ComplianceStatus', data=df_not_num)
count_plot.set_xticklabels(count_plot.get_xticklabels(), rotation=45,␣
↪horizontalalignment='right')
plt.tight_layout()
```



1.6 Grâce aux analyses des champs non numériques, nous pouvons exclure les champs suivants :

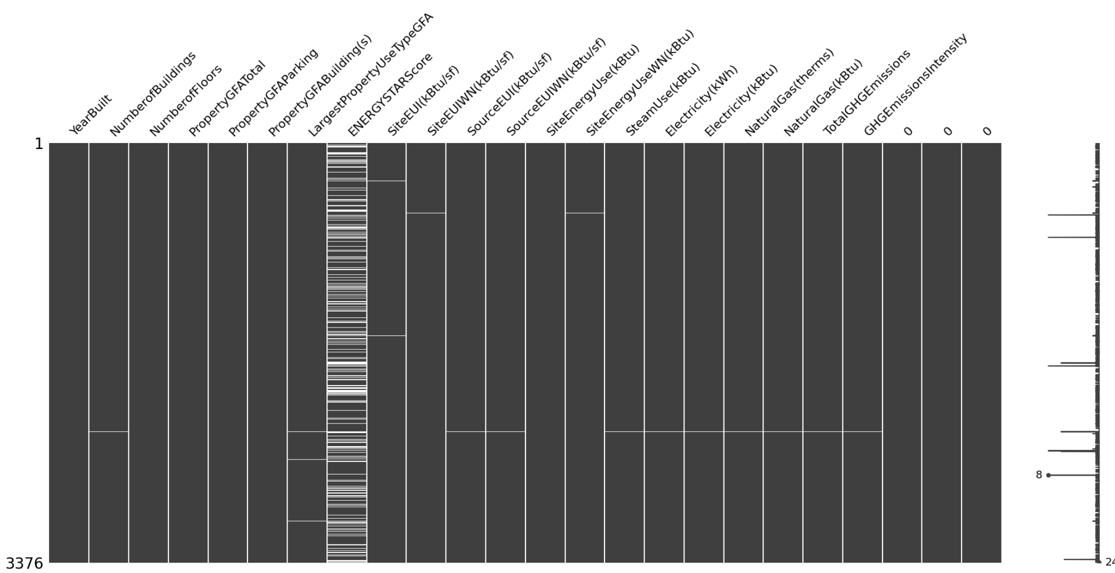
- ComplianceStatus
- DefaultData
- LargestPropertyType, SecondLargestPropertyType et SecondLargestPropertyUseTypeGFA : ces colonnes sont inutiles étant donné que nous avons pris en compte PrimaryPropertyType.
- ListOfAllPropertyUseTypes est inutile dans le contexte
- ZipCode, Latitude et Longitude : il a été décidé de garder le quartier (Neighborhood) et donc les coordonnées exactes des batiments devient obsolète.

Ces champs sont trop peu diversifié pour pouvoir être utiles dans nos modèles. Les autres champs présentent pour certains de nombreuses valeurs possibles. Nous allons plus tard essayer de les combiner afin d'avoir un champ unique plus exploitable.

[ ]:
```
# array of columns to remove
columns_to_remove = [
    'ComplianceStatus',
    'DefaultData',
    'LargestPropertyUseType',
```

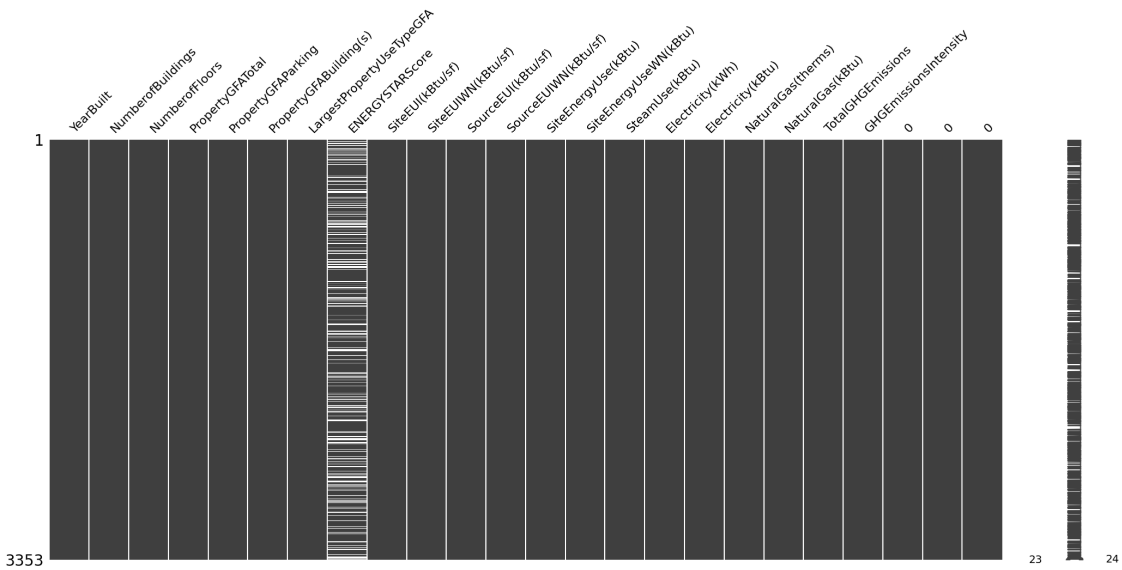
```
    'SecondLargestPropertyUseType',
    'SecondLargestPropertyUseTypeGFA',
    'Neighborhood',
    'ListOfAllPropertyUseTypes',
    'Latitude',
    'Longitude',
    'ZipCode',
    'BuildingType',
    'PrimaryPropertyType'
]

df = DataEngineering.remove_columns_by_name(df, columns_to_remove)
```

[ ]:
```
# Analyse après nettoyage et engineering des données
DataAnalysis.show_columns_population(df, type='matrix')
```



[ ]:
```
# create a variable containing columns names without ENERGYSTARScore
columns_without_energystarscore = df.columns.drop('ENERGYSTARScore')
# eliminate rows with missing values for columns_without_energystarscore
df = df.dropna(subset=columns_without_energystarscore)
```

[ ]:
```
DataAnalysis.show_columns_population(df, type='matrix')
```

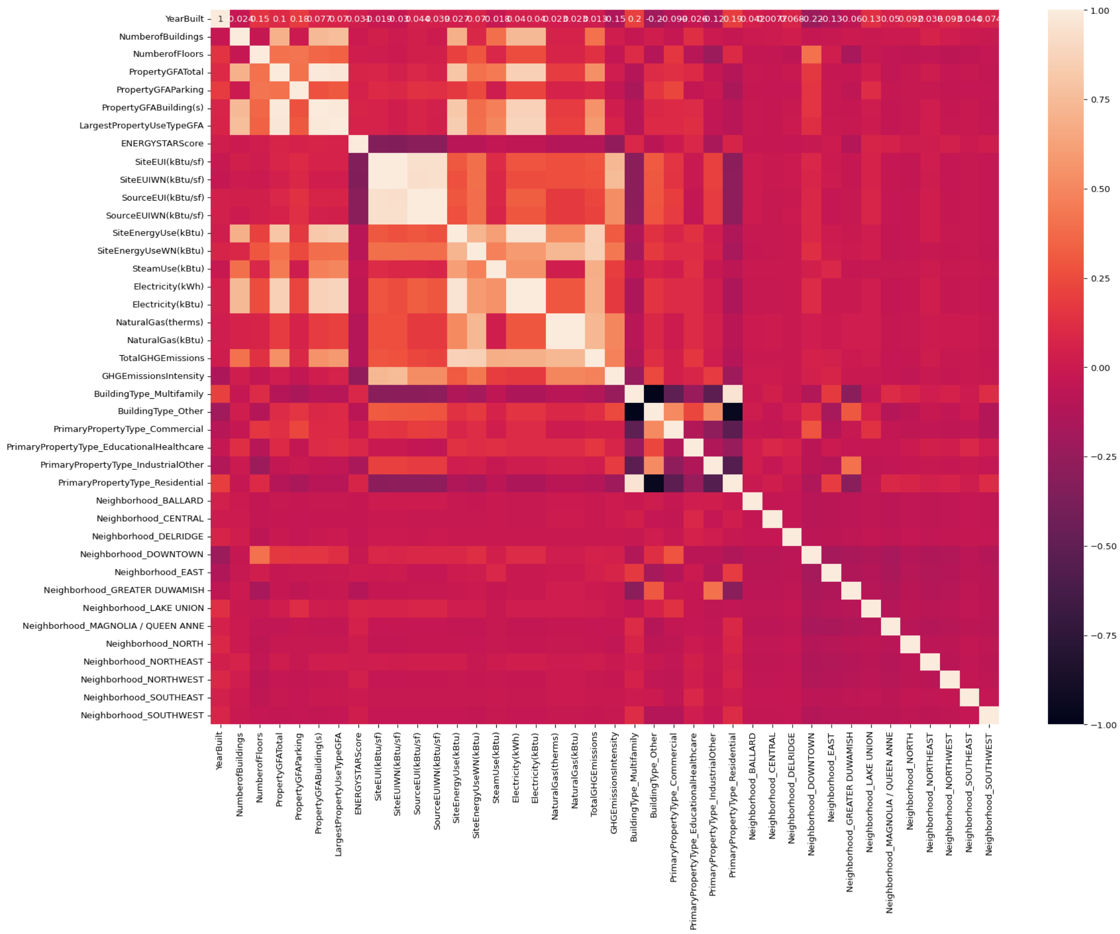


[ ]:
```python
correlation_matrix_clean = df.corr()

# write correlation matrix to file
correlation_matrix_clean.to_csv('data/correlation_matrix_clean.csv')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
TypeError: float() argument must be a string or a real number, not 'csr_matrix'

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
Cell In[43], line 1
----> 1 correlation_matrix_clean = df.corr()
      3 # write correlation matrix to file
      4 correlation_matrix_clean.to_csv('data/correlation_matrix_clean.csv')

File ~/anaconda3/envs/OC-P3/lib/python3.11/site-packages/pandas/core/frame.py:
↪11049, in DataFrame.corr(self, method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File ~/anaconda3/envs/OC-P3/lib/python3.11/site-packages/pandas/core/frame.py:
↪1993, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1991 if dtype is not None:
```

```
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1994 if result.dtype is not dtype:
   1995     result = np.asarray(result, dtype=dtype)

File ~/anaconda3/envs/OC-P3/lib/python3.11/site-packages/pandas/core/internals/
 ↪managers.py:1694, in BlockManager.as_array(self, dtype, copy, na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File ~/anaconda3/envs/OC-P3/lib/python3.11/site-packages/pandas/core/internals/
 ↪managers.py:1753, in BlockManager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

ValueError: setting an array element with a sequence.
```

[ ]:
```
# Set the size of the plot
# Note: The size is set in inches and the dpi (dots per inch) determines the␣
 ↪resolution.
# For a 1920x1080 resolution with a typical screen dpi of 96, use the following␣
 ↪dimensions:
plt.figure(figsize=(1920/96, 1400/96), dpi=96)

# Create the heatmap
sns.heatmap(correlation_matrix_clean, annot=True)

# Show the plot
plt.show()
```



[ ]:
```python
# Transformer la colonne NumberofFloors en logarithme afin de réduire l'effet des outliers
# Définir une fonction pour appliquer le logarithme en toute sécurité
def safe_log(x, min_val=0.0001):
    return np.log(x + min_val)

# Appliquer la fonction logarithmique sécurisée
df['NumberofFloors'] = df['NumberofFloors'].apply(safe_log)
```

[ ]:
```python
# Extraire les corrélations avec 'SiteEnergyUse(kBtu)'
# correlations = correlation_matrix_clean['SiteEnergyUse(kBtu)']

# # Définir un seuil de corrélation, par exemple 0.75
# threshold = 0.75

# # Identifier les variables fortement corrélées (à l'exclusion de la variable elle-même)
```

```
# strongly_correlated = correlations[abs(correlations) > threshold].
  ↪drop(['SiteEnergyUse(kBtu)', 'TotalGHGEmissions'])
# #strongly_correlated.drop('TotalGHGEmissions')

# # Afficher les variables fortement corrélées
# print(strongly_correlated)

# # Supprimer ces variables du dataset
# df = df.drop(columns=strongly_correlated.index)
```

1.6.1 Création d'une nouvelle colonne "Age" qui est une transformation de la colonne YearBuilt (2023 - YearBuilt)

[ ]:
```
df['Age'] = 2017 - df['YearBuilt']

df.drop(columns='YearBuilt', inplace=True)
```

1.6.2 Utilisation du logarithme pour la colonne "Age" qui est une transformation qui permet de mieux exploiter cette valeur par la suite

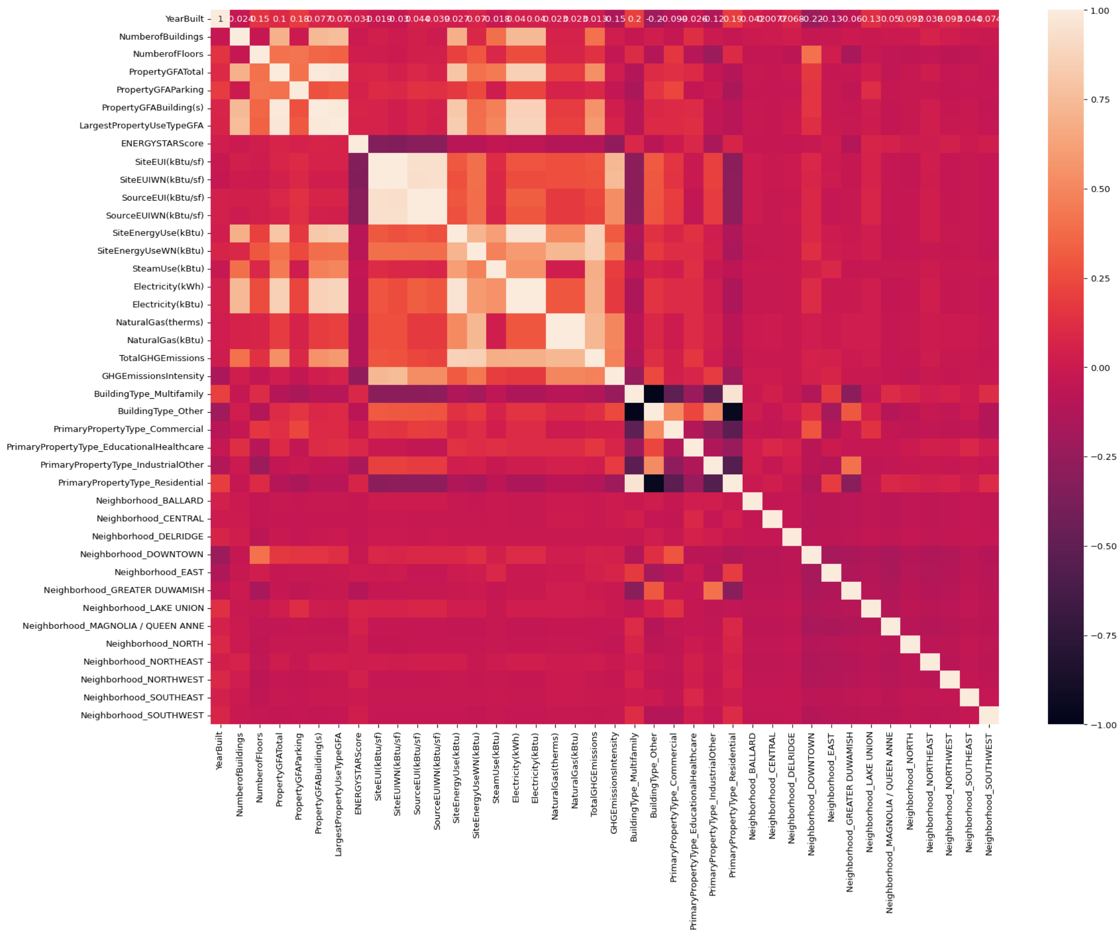
[ ]:
```
# Make Age a logarithmic feature
df['Age'] = df['Age'].apply(safe_log)
```

Construction d'une nouvelle variable, qui sera ratio d'utilisation d'énergie par âge

[ ]:
```
# remove rows where 'SiteEUI(kBtu/sf)' is 0
df = df[df['SiteEUI(kBtu/sf)'] != 0]

df['EnergyUse_Age_Ratio'] = df['SiteEUI(kBtu/sf)'] / df['Age']
```

1.7 On réaffiche les correlations pour voir si les nouvelles colonnes ont un impact

[ ]:
```
plt.figure(figsize=(1920/96, 1400/96), dpi=96)

# Create the heatmap
sns.heatmap(correlation_matrix_clean, annot=True)

# Show the plot
plt.show()
```



1.8 Génération du fichier csv clean pour les modèles de machine learning.

[ ]:
```
df.sample(5)
```

[ ]:
```
      NumberofBuildings  NumberofFloors  PropertyGFATotal  PropertyGFAParking  \
40                  1.0        0.693197             52554                   0
397                 1.0        0.000100            101101                   0
1458                1.0        0.000100             22047                   0
679                 1.0        2.079454            148474                   0
337                 1.0        1.609458            127800                   0

      PropertyGFABuilding(s)  LargestPropertyUseTypeGFA  ENERGYSTARScore  \
40                     52554                    51029.0             83.0
397                   101101                   101101.0              1.0
1458                   22047                    22898.0            100.0
679                   148474                    64445.0             64.0
337                   127800                   127800.0             56.0
```

```
      SiteEUI(kBtu/sf)  SiteEUIWN(kBtu/sf)  SourceEUI(kBtu/sf)  …  \
40           50.099998           53.500000          108.699997  …
397         215.600006          211.100006          676.599976  …
1458         11.600000           11.900000           36.400002  …
679          61.099998           64.000000          124.800003  …
337          21.700001           22.000000           64.400002  …

      Neighborhood_GREATER DUWAMISH  Neighborhood_LAKE UNION  \
40                              1.0                      0.0
397                             1.0                      0.0
1458                            0.0                      0.0
679                             0.0                      0.0
337                             0.0                      1.0

      Neighborhood_MAGNOLIA / QUEEN ANNE  Neighborhood_NORTH  \
40                                   0.0                 0.0
397                                  0.0                 0.0
1458                                 0.0                 0.0
679                                  0.0                 0.0
337                                  0.0                 0.0

      Neighborhood_NORTHEAST  Neighborhood_NORTHWEST  Neighborhood_SOUTHEAST  \
40                       0.0                     0.0                     0.0
397                      0.0                     0.0                     0.0
1458                     0.0                     1.0                     0.0
679                      0.0                     0.0                     0.0
337                      0.0                     0.0                     0.0

      Neighborhood_SOUTHWEST       Age  EnergyUse_Age_Ratio
40                       0.0  4.762175            10.520403
397                      0.0  4.219509            51.095992
1458                     0.0  4.007335             2.894692
679                      0.0  3.258100            18.753258
337                      0.0  4.634730             4.682042

[5 rows x 41 columns]
```

[ ]:

```python
# write the resulting dataframe to a csv file
df.to_csv('data/clean.csv', index=False)
```