De doelstelling van het project is een kaart te creëren die weergeeft hoeveel gebouwen er liggen binnen de zones die volgens het gewestplan bestemd zijn als landbouwgebied. De kaart toont het resultaat voor het Vlaams Gewest en toont het aantal gebouwen per km^2 landbouwgebied. De kaart maakt abstractie van het feit dat het gewestplan op verschillende plaatsen is vervangen door andere plannen.

Het duurt vermoedelijk 3 à 4 minuten om de volledige code uit te voeren.

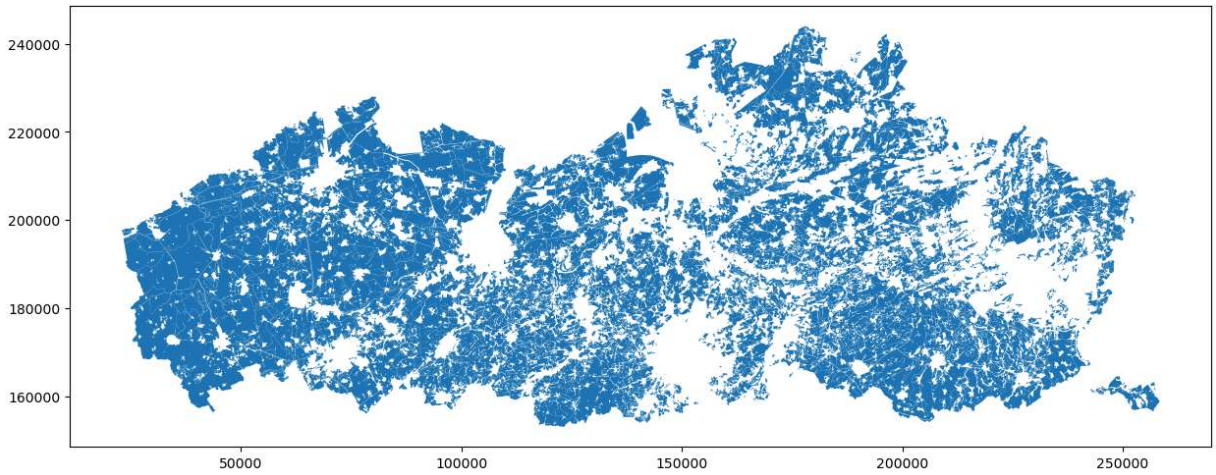
1. De eerste stap is het inladen van de geodata van al die gebieden die volgens het gewestplan een bestemming hebben die in te delen valt als een landbouwbestemming. Deze geodata is beschikbaar via een wfs van Digitaal Vlaanderen. De verzamel-url mercator bevat een laag "lu_gwp_gv" die alle grondvlakken omvat van de eenheden op het gewestplan. Uit deze laag worden enkel de gebieden geselecteerd die een bestemming hebben in de categorie landbouw. Het dataframe "ldb_gwp" weerhoudt van elke karteeringseenheid enkel het id-nummer en de coördinaten.

In [ ]:
```
import geopandas as gpd
import pandas as pd

import requests

# WFS voor de landbouwgebieden in Vlaanderen volgens het gewestplan
url_gwp = "https://www.mercator.vlaanderen.be/raadpleegdienstenmercatorpubliek/wfs"

params = {
    "SERVICE": "WFS",
    "REQUEST": "GetFeature",
    "VERSION": "2.0.0",
    "TYPENAMES": "lu_gwp_gv",
    #via deze filter worden enkel de grondvlakken opgevraagd met een bestemming die
    "cql_filter": "categorie = 'LDB'",
    "outputFormat": "JSON"
}

responseLDB = requests.Request('GET', url_gwp, params=params).prepare()

ldb_gwp = gpd.read_file(responseLDB.url)

# Enkel de kolommen id en de geometry zijn relevant en worden overgehouden uit de d
ldb_gwp = ldb_gwp[['id', 'geometry']]

ldb_gwp.head()
```

Out[ ]:

	id	geometry
0	0209b349-2ea2-4ba5-bc77-56cc32f69fdf	MULTIPOLYGON (((163725.812 219765.308, 163743....
1	22feae60-b9ac-4cb8-aa09-23fc300bc0a1	MULTIPOLYGON (((57741.632 215700.621, 57677.77...
2	0084cd95-900b-497f-b7a7-c3fbdf068810	MULTIPOLYGON (((47844.742 210666.199, 47627.85...
3	153d730f-047b-4bef-afaa-5e57e1dc33cb	MULTIPOLYGON (((163085.953 219492.152, 163180....
4	2a5c0188-1537-4c66-ac5f-a74c0a640662	MULTIPOLYGON (((163375.609 219177.511, 163352....

In [ ]:
```
import matplotlib.pyplot as plt

# Kaart als test
ldb_gwp.plot()

plt.rcParams['figure.figsize'] = [15, 15]

plt.show()
```



In [ ]:
```
# De WFS levert 7573 landbouwgebieden aan
ldb_gwp.shape
```

Out[ ]:
```
(7573, 2)
```

2. De tweede stap bestaat uit het inladen van de geodata van de statistische sectoren die in Vlaanderen liggen. De WFS-service van Digitaal Vlaanderen omvat de NISCODE van elke statistische sector. Op basis van de NISCODE kunnende Vlaamse statistische sectoren geselecteerd worden, uit een dataset die heel het Belgische grondgebied beslaat. De statistische sectoren zijn nuttig voor het project om 2 redenen:

- De bevraging van de WFS-service van het GRB voor gebouwpolygonen kan in kleinere delen worden opgesplitst. Sommige karteringseenheden van het gewestplan bevatten immers nog te veel polygonen om in 1 keer op te vragen (max 10 000 resultaten)

- De statistische sectoren zijn een logischere eenheid voor de weergave van het aantal gebouwen in landbouwgebied. De karteringseenheden van het gewestplan zijn immers onderling zeer verschillend in grootte. Het gebruik van statistische sectoren laat bovendien toe om ook een analyse te maken van het aantal gebouwen in landbouwgebied per gemeente.

Aan het dataframe wordt de oppervlakte van elke statistische sector toegevoegd om later het aantal gebouwen per km² te kunnen berekenen.

In [ ]:
```python
import geopandas as gpd
import requests

# WFS voor de statistische sectoren in Vlaanderen
stat_url = "https://geo.api.vlaanderen.be/StatistischeSectoren/wfs"

params = {
    "service": "WFS",
    "request": "GetFeature",
    "typeName": "StatistischeSectoren:StatSec",
    "version": "2.0.0",
    #filter waarbij enkel de Vlaamse statistische sectoren worden gevraagd op basis
    "cql_filter": "(NISCODE > 23000 AND NISCODE < 25000) OR (NISCODE < 20000) OR (N
    "outputFormat": "json"
}

# GET-verzoek naar de URL
response_stat = requests.get(stat_url, params=params)

stat = gpd.read_file(response_stat.url)

# enkel de ID, NISCODE, de naam van de sector en de geometrie zijn noodzakelijk
stat = stat[['id', 'NISCODE', 'SECNAAM', 'geometry']]

# de kolom 'id' wordt hernoemd naar ID_STAT om een unieke kolomnaam te hebben over
stat.rename(columns={'id': 'ID_STAT'}, inplace=True)

# De oppervlakte van elke statistische sector wordt berekend
stat['opp_stat'] = stat.geometry.area

stat.head()
```

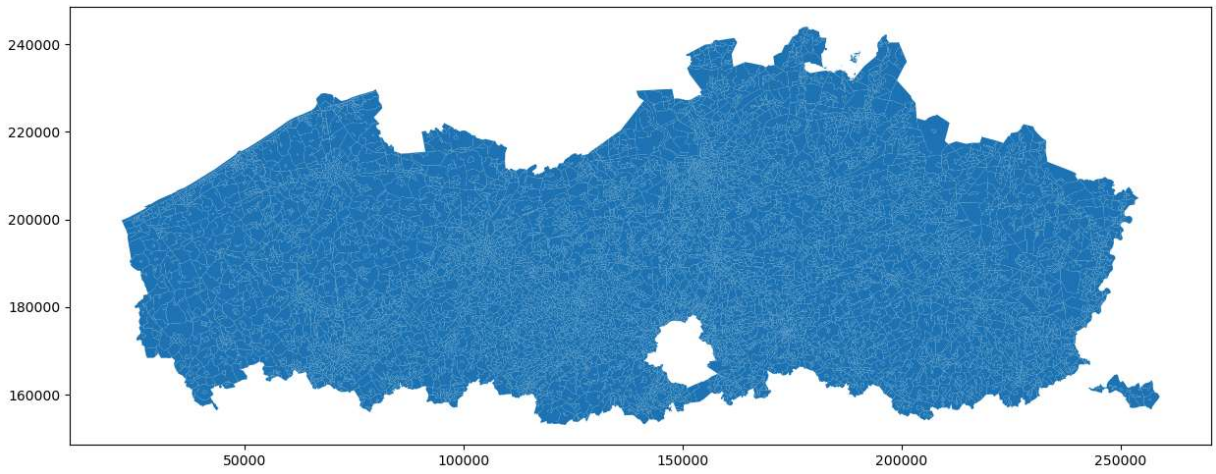
Out[ ]:

	ID_STAT	NISCODE	SECNAAM	geometry	opp_stat
0	StatSec.1	11001	AARTSELAAR-CENTRUM	MULTIPOLYGON (((151281.422 203009.293, 151249....	531589.575160
1	StatSec.2	11001	DE LEEUWERIK	MULTIPOLYGON (((151325.594 203677.293, 151318....	671157.586199
2	StatSec.3	11001	BUERSTEDE	MULTIPOLYGON (((150721.484 203547.496, 150729....	280848.161640
3	StatSec.4	11001	KLEINE GRIPPE	MULTIPOLYGON (((151882.891 203814.199, 151889....	255623.650854
4	StatSec.5	11001	LEUG	MULTIPOLYGON (((150790.516 202810.371, 150745....	337269.057140

In [ ]:
```
# De WFS stuurt 9194 statisctische sectoren terug
stat.shape
```

Out[ ]:
```
(9194, 5)
```

In [ ]:
```
# kaart als test of volledig Vlaanderen wordt gedekt
stat.plot()

plt.show()
```



3. In deze stap worden de polygonen van het gewestplpan geclipt op basis van de grenzen van de statistische sectoren. Het resultaat is een nieuw dataframe met 24411 geclipte polygonen van het gewestplan aangevuld met de NISCODE en het identificatienummer van de statistische sector waartoe de polygoon behoort.

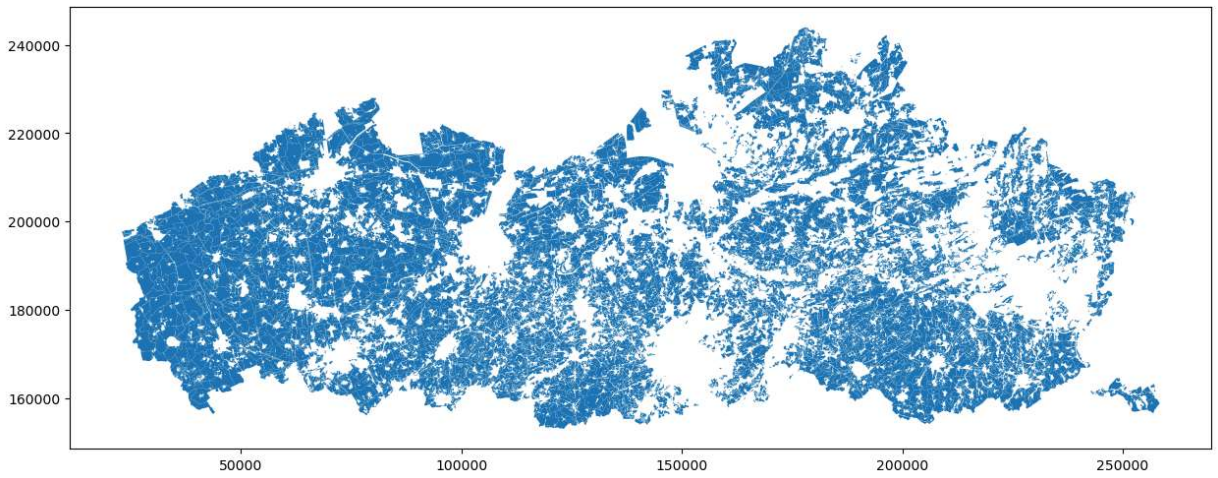
In [ ]:
```
# Maak een lege lijst om de geclipte resultaten in op te slaan
clip_ldb_gwp_list = []

# In volgende operatie wordt elk landbouwgebied geclipt op basis van de geometrieën

for index, row in stat.iterrows():
```

```
    # Clip 'ldb_gwp' op de huidige multipolygon
    clipped_ldb_gwp = ldb_gwp.clip(row['geometry'])
    # Voeg de NISCODE en ID_STAT toe aan het geclippte GeoDataFrame
    clipped_ldb_gwp['NISCODE'] = row['NISCODE']
    clipped_ldb_gwp['ID_STAT'] = row['ID_STAT']
    # Voeg het geclippte GeoDataFrame toe aan de lijst
    clip_ldb_gwp_list.append(clipped_ldb_gwp)

# Voeg de geclippte GeoDataFrames samen in één GeoDataFrame
stat_ldb_gwp = pd.concat(clip_ldb_gwp_list, ignore_index=True)

stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1
1	d5de8179-4b94-4890-b579-ff8d111b5d09	MULTIPOLYGON (((150020.431 203331.154, 150021....	11001	StatSec.3
2	3a5ba118-d979-4466-8738-9c8b84470763	MULTIPOLYGON (((150432.364 203583.112, 150432....	11001	StatSec.3
3	3e2caaee-b6bf-4612-b401-68ff0229dd19	MULTIPOLYGON (((151755.965 203183.373, 151752....	11001	StatSec.4
4	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5

In [ ]:
```
# kaart als test
stat_ldb_gwp.plot()

plt.show()
```



In [ ]:
```
# Het clippen van de landbouwgebieden levert 24411 polygonen op. De landbouwgebiede
stat_ldb_gwp.shape
```

Out[ ]: (24411, 4)

4. Om de resultaten op een kaart te kunnen groeperen per gemeente zijn de polygonen van de grenzen van de Vlaamse gemeenten noodzakelijk. Deze zijn beschikbaar via een een WFS van Digitaal Vlaanderen. Aan het dataframe wordt de oppervlakte van elke gemeente toegevoegd om later het aantal gebouwen per km² te kunnen berekenen.

In [ ]:
```
# WFS voor de gemeentegrenzen in Vlaanderen
url_Refgem="https://geo.api.vlaanderen.be/VRBG/wfs"

params = {
    "SERVICE": "WFS",
    "REQUEST": "GetFeature",
    "VERSION": "2.0.0",
    "TYPENAMES": "VRBG:Refgem",
    "outputFormat": "JSON"
}

responseRefgem = requests.Request('GET', url_Refgem, params=params).prepare()

gemeentegrenzen = gpd.read_file(responseRefgem.url)

# enkel de NISCODE, de naam en de geometrie zijn noodzakelijk
gemeentegrenzen = gemeentegrenzen[['NISCODE', 'NAAM', 'geometry']]

# de oppervlakte van elke gemeente wordt berekend op basis van de geometrie
gemeentegrenzen['oppervlakte'] = gemeentegrenzen.geometry.area

gemeentegrenzen.head()
```

Out[ ]:

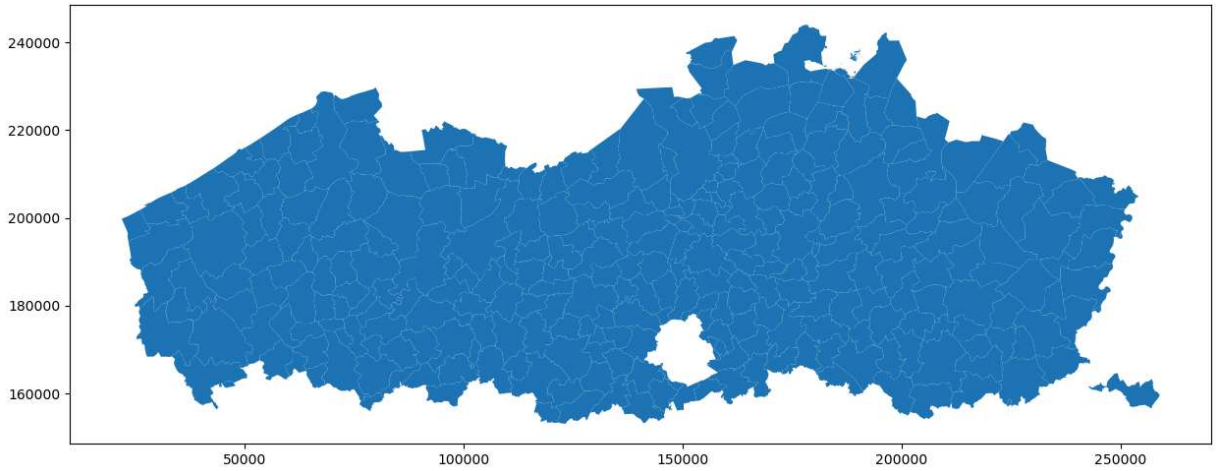
	NISCODE	NAAM	geometry	oppervlakte
0	72043	Pelt	MULTIPOLYGON (((224591.910 218998.880, 224547....	8.378474e+07
1	72042	Oudsbergen	MULTIPOLYGON (((231467.154 204604.461, 231454....	1.163707e+08
2	12041	Puurs-Sint-Amands	MULTIPOLYGON (((148689.993 197941.126, 148665....	4.913837e+07
3	44085	Lievegem	MULTIPOLYGON (((96423.253 207308.996, 96416.05...	8.078116e+07
4	44084	Aalter	MULTIPOLYGON (((82999.396 205969.226, 82979.68...	1.198508e+08

In [ ]:
```
# De WFS levert 300 polygonen aan
gemeentegrenzen.shape
```

Out[ ]:
```
(300, 4)
```

In [ ]:
```
# Kaart ter controle of alle gemeenten in de dataframe zitten
gemeentegrenzen.plot()
```

```
plt.show()
```



5. In een volgende stap wordt van elke geclipte polygoon de bounding box berekend. Op basis van deze bounding boxes kan de laag GBG van het GRB bevraagd worden zonder het maximum van 10000 resultaten niet te overschrijden. Naast de bouding box wordt ook de oppervlakte toegevoegd en de naam van de gemeente. Ten slotte worden de geclipte polygonen die kleiner zijn dan 1 hectare gewist aangezien ze te klein zijn voor statistische analyse en voor te veel uitschieters zouden zorgen.

In [ ]:
```
# Voor elke geclipt landbouwgebied wordt de bouding box berekend
bounding_boxes = stat_ldb_gwp['geometry'].bounds

# De boundig box kolommen worden toegevoegd aan de dataframe
stat_ldb_gwp['minx'] = bounding_boxes['minx']
stat_ldb_gwp['miny'] = bounding_boxes['miny']
stat_ldb_gwp['maxx'] = bounding_boxes['maxx']
stat_ldb_gwp['maxy'] = bounding_boxes['maxy']

# De oppervlakte van elke geclipt landbouwgebied wordt berekend
stat_ldb_gwp['opp_ldb'] = stat_ldb_gwp.geometry.area

# Elk geclipt landbouwgebied wordt voorzien van een uniek ID
stat_ldb_gwp['ID_STAT_LDB'] = range(1, len(stat_ldb_gwp) + 1)

# De geclipte landbouwgebieden waarvan de oppervlakte kleiner is dan 1 hectare word
stat_ldb_gwp = stat_ldb_gwp[stat_ldb_gwp['opp_ldb'] >= 10000]

# Voor elke geclipt landbouwgebied wordt de naam van de gemeente toegevoegd op basi
stat_ldb_gwp = pd.merge(stat_ldb_gwp, gemeentegrenzen[['NISCODE', 'NAAM']], on='NIS

stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT	minx	miny	
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1	151489.587910	201819.512243	151611
1	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5	150701.078125	201824.066791	151143
2	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151750.471 202174.322, 151819.765 20...	11001	StatSec.6	151750.471248	202166.824250	152033
3	b9304c99-5dfe-4e3d-a134-50ec24e4526a	POLYGON ((145710.257 226947.227, 145713.937 22...	11002	StatSec.101	145710.257427	226920.590000	146577
4	7e9c2f06-28bd-4ab6-9936-62c825e9e866	POLYGON ((145698.703 227999.637, 145726.437 22...	11002	StatSec.101	145519.593300	227847.355300	145733

In [ ]:
```
stat_ldb_gwp.shape
```

Out[ ]:
```
(16563, 11)
```

6. In deze stap worden de polygonen van de laag "GBG" uit de WFS van het GRB geladen op basis van de bouding boxes. Deze operatie duurt een achttal uur gezien het hoge aantal data. Onderstaande code leest enkel het resultaat in van het uiteindelijke excelbestand dat lokaal is opgeslaan. De aanpassing van het pad naar de correcte bestandslocatie is dus noodzakelijk. De polygonen die wel in de bouding box liggen, maar niet in het eigenlijke landbouwgebied zijn al gefilterd met het intersect commando en niet opgenomen in het excelbestand.

   De code voor de bevraging van het GRB is apart opgeleverd.

In [ ]:
```
import pandas as pd

gbg_results = pd.read_excel(r"C:\Users\dries\Documents\LAM_2\Project_Geo-ICT\PROJEC
```

```
gbg_results.head()
```

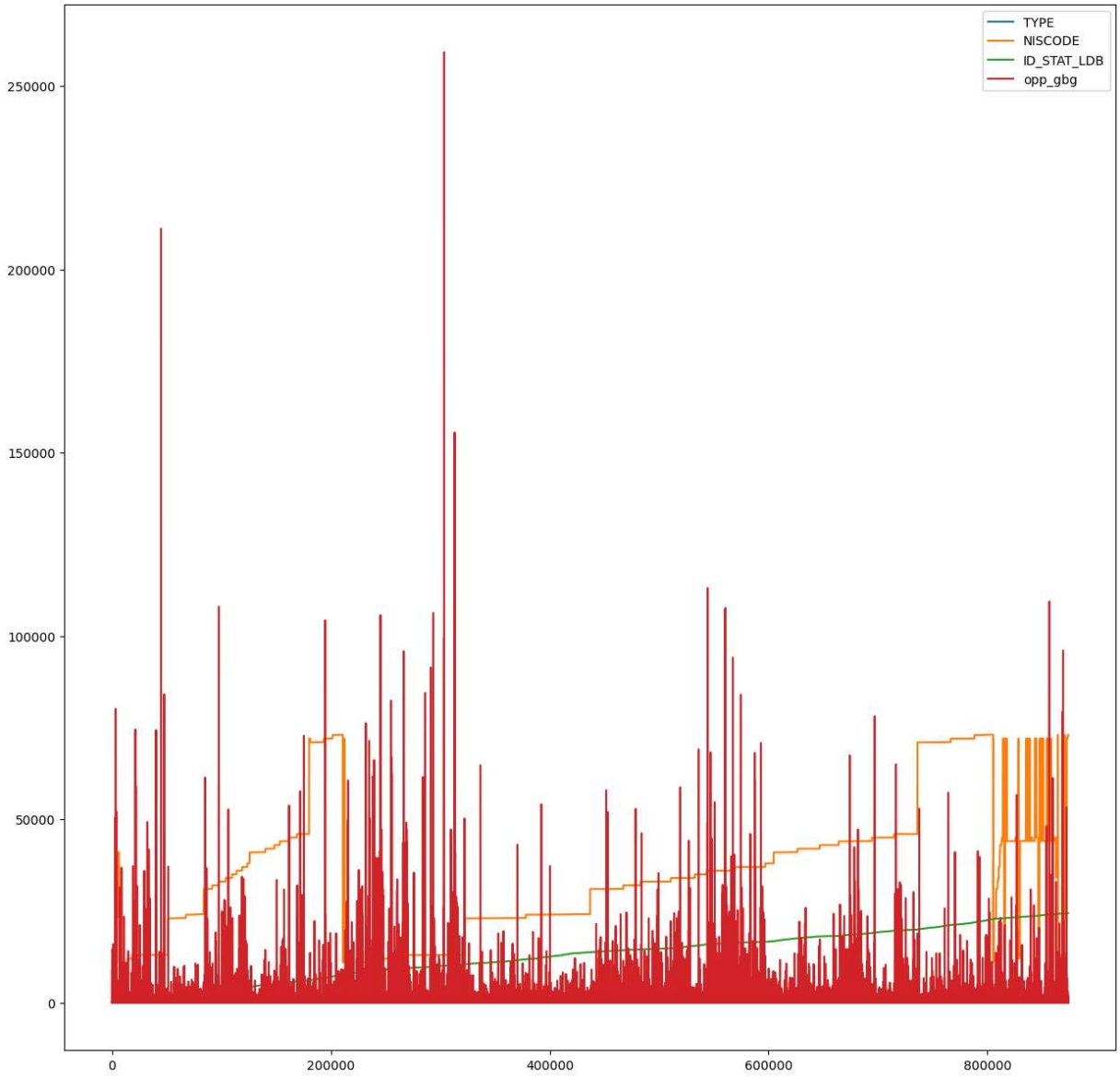
Out[ ]:

	id	TYPE	LBLTYPE	geometry	NISCODE	ID_STAT	ID_LDB	ID_
0	GBG.6159262	2	bijgebouw	POLYGON ((151518.3120123 201843.11201412, 1515...	11001	StatSec.1	97e45686-e632-4930-a4a8-cf0e80833b5f	
1	GBG.3019916	2	bijgebouw	POLYGON ((151526.06932431 201869.91623814, 151...	11001	StatSec.1	97e45686-e632-4930-a4a8-cf0e80833b5f	
2	GBG.750390	2	bijgebouw	POLYGON ((151578.98132434 202105.3879823, 1515...	11001	StatSec.1	97e45686-e632-4930-a4a8-cf0e80833b5f	
3	GBG.7155788	2	bijgebouw	POLYGON ((151568.78497234 202087.71438229, 151...	11001	StatSec.1	97e45686-e632-4930-a4a8-cf0e80833b5f	
4	GBG.6475217	2	bijgebouw	POLYGON ((151494.87098829 201820.35105411, 151...	11001	StatSec.1	97e45686-e632-4930-a4a8-cf0e80833b5f	

In [ ]:
```
# De WFS leverde 873818 polygonen aan binnen landbouwgebied
gbg_results.shape
```

Out[ ]:
```
(873818, 9)
```

In [ ]:
```
gbg_results.plot()

plt.show()
```



7. In deze stap wordt het dataframe van de geclipte landbouwgebieden uitgebreid met data uit het GRB in volgende kolommen:

    - De som van de oppervlaktes van alle gebouwen in het landbouwgebied
    - Het aantal gebouwen in het landbouwgebied
    - Het aantal gebouwen per km² landbouwgebied

In [ ]:
```
# De resultaten van de gebouwen worden gesorteerd per geclipt landbouwgebied en de
gbg_per_stat_ldb = gbg_results.groupby('ID_STAT_LDB')['opp_gbg'].sum().reset_index(

# Controleer of de kolom 'opp_gbg' al bestaat in de dataframe van de geclipte landb
if 'opp_gbg' in stat_ldb_gwp.columns:
    # Verwijder de kolom 'opp_gbg' als deze al bestaat
    stat_ldb_gwp.drop(columns=['opp_gbg'], inplace=True)

# De lijst met de opgetelde oppervlaktes wordt toegevoegd aan de dataframe van de g
stat_ldb_gwp = pd.merge(stat_ldb_gwp, gbg_per_stat_ldb, on='ID_STAT_LDB', how='left
```

```
stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT	minx	miny	
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1	151489.587910	201819.512243	151611
1	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5	150701.078125	201824.066791	151143
2	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151750.471 202174.322, 151819.765 20...	11001	StatSec.6	151750.471248	202166.824250	152033
3	b9304c99-5dfe-4e3d-a134-50ec24e4526a	POLYGON ((145710.257 226947.227, 145713.937 22...	11002	StatSec.101	145710.257427	226920.590000	146577
4	7e9c2f06-28bd-4ab6-9936-62c825e9e866	POLYGON ((145698.703 227999.637, 145726.437 22...	11002	StatSec.101	145519.593300	227847.355300	145733

In [ ]:

```
# De resultaten van de gebouwen worden gesorteerd per geclipt landbouwgebied en het
aantal_gbg_per_stat_ldb = gbg_results.groupby('ID_STAT_LDB').size().reset_index()

# Het aantal gebouwen wordt toegevoegd aan de lijst onder de kolom 'aantal_gbg'
aantal_gbg_per_stat_ldb.columns = ['ID_STAT_LDB', 'aantal_gbg']

# Controleer of de kolom 'aantal_gbg' al bestaat in de lijst
if 'aantal_gbg' in stat_ldb_gwp.columns:
    # Verwijder de kolom 'aantal_gbg' als deze al bestaat
    stat_ldb_gwp.drop(columns=['aantal_gbg'], inplace=True)

# De lijst met het aantal gebouowen wordt toegevoegd aan de dataframe van de geclip
stat_ldb_gwp = pd.merge(stat_ldb_gwp, aantal_gbg_per_stat_ldb, on='ID_STAT_LDB', ho

stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT	minx	miny	
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1	151489.587910	201819.512243	151611
1	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5	150701.078125	201824.066791	151143
2	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151750.471 202174.322, 151819.765 20...	11001	StatSec.6	151750.471248	202166.824250	152033
3	b9304c99-5dfe-4e3d-a134-50ec24e4526a	POLYGON ((145710.257 226947.227, 145713.937 22...	11002	StatSec.101	145710.257427	226920.590000	146577
4	7e9c2f06-28bd-4ab6-9936-62c825e9e866	POLYGON ((145698.703 227999.637, 145726.437 22...	11002	StatSec.101	145519.593300	227847.355300	145733

In [ ]:
```
# aan de dataframe wordt een kolom toegevoegd met het aantal gebouwen per km² landb
stat_ldb_gwp['gbg_ldb'] = (stat_ldb_gwp['aantal_gbg']/stat_ldb_gwp['opp_ldb'])*10**
```

In [ ]:
```
stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT	minx	miny	
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1	151489.587910	201819.512243	151611
1	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5	150701.078125	201824.066791	151143
2	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151750.471 202174.322, 151819.765 20...	11001	StatSec.6	151750.471248	202166.824250	152033
3	b9304c99-5dfe-4e3d-a134-50ec24e4526a	POLYGON ((145710.257 226947.227, 145713.937 22...	11002	StatSec.101	145710.257427	226920.590000	146577
4	7e9c2f06-28bd-4ab6-9936-62c825e9e866	POLYGON ((145698.703 227999.637, 145726.437 22...	11002	StatSec.101	145519.593300	227847.355300	145733

8. In deze stap wordt het dataframe van de statistische sectoren uitgebreid met data uit het GRB in volgende kolommen:

   - De som van de oppervlaktes van alle gebouwen in landbouwgebied per statistische sector
   - Het aantal gebouwen in landbouwgebied per statistische sector
   - Het aantal gebouwen per km² landbouwgebied per statistische sector

In [ ]:
```
# De dataframe van de geclipte landbouwgebieden wordt gesorteerd per statistische s
gbg_per_stat = stat_ldb_gwp.groupby('ID_STAT')['opp_ldb'].sum().reset_index()

# Controleer of de kolom 'opp_ldb' al bestaat in de dataframe van de statistische s
if 'opp_ldb' in stat.columns:
    # Verwijder de kolom 'opp_ldb' als deze al bestaat
    stat.drop(columns=['opp_ldb'], inplace=True)

# De lijst met de opgetelde oppervlakte van het landbouwgebied wordt toegevoegd aan
stat = pd.merge(stat, gbg_per_stat, on='ID_STAT', how='left')

# Indien er geen waarde in de kolom 'opp_ldb' word de statistische sector verwijder
```

```
stat = stat.dropna(subset=['opp_ldb'])

stat.head()
```

Out[ ]:

	ID_STAT	NISCODE	SECNAAM	geometry	opp_stat	op
0	StatSec.1	11001	AARTSELAAR-CENTRUM	MULTIPOLYGON (((151281.422 203009.293, 151249....	5.315896e+05	18526.9
4	StatSec.5	11001	LEUG	MULTIPOLYGON (((150790.516 202810.371, 150745....	3.372691e+05	13410.6
5	StatSec.6	11001	TEN DORPE	MULTIPOLYGON (((151979.438 202728.934, 151850....	2.182698e+05	11293.4
100	StatSec.101	11002	ZANDVLIET-KERN(ZANDVL.-5 DIST)	MULTIPOLYGON (((145788.641 228199.231, 145732....	1.342175e+06	404044.5
101	StatSec.102	11002	STALSHOEK(ZANDVLIET-5 DISTR.)	MULTIPOLYGON (((147287.938 228082.668, 147279....	6.900929e+05	201277.2

In [ ]:

```
# De dataframe van de geclipte landbouwgebieden wordt gesorteerd per statistische s
gbg_per_stat = stat_ldb_gwp.groupby('ID_STAT')['opp_gbg'].sum().reset_index()

# Controleer of de kolom 'opp_gbg' al bestaat in ovl_final_gemeentegrenzen DataFram
if 'opp_gbg' in stat.columns:
    # Verwijder de kolom 'opp_gbg' als deze al bestaat
    stat.drop(columns=['opp_gbg'], inplace=True)

# De lijst met de opgetelde oppervlakte van de gebouwen wordt toegevoegd aan de dat
stat = pd.merge(stat, gbg_per_stat, on='ID_STAT', how='left')

stat.head()
```

Out[ ]:

	ID_STAT	NISCODE	SECNAAM	geometry	opp_stat	opp_
0	StatSec.1	11001	AARTSELAAR-CENTRUM	MULTIPOLYGON (((151281.422 203009.293, 151249....	5.315896e+05	18526.925
1	StatSec.5	11001	LEUG	MULTIPOLYGON (((150790.516 202810.371, 150745....	3.372691e+05	13410.691
2	StatSec.6	11001	TEN DORPE	MULTIPOLYGON (((151979.438 202728.934, 151850....	2.182698e+05	11293.457
3	StatSec.101	11002	ZANDVLIET-KERN(ZANDVL.-5 DIST)	MULTIPOLYGON (((145788.641 228199.231, 145732....	1.342175e+06	404044.501
4	StatSec.102	11002	STALSHOEK(ZANDVLIET-5 DISTR.)	MULTIPOLYGON (((147287.938 228082.668, 147279....	6.900929e+05	201277.243

In [ ]:

```
# De dataframe van de geclipte landbouwgebieden wordt gesorteerd per statistische s
aantal_gbg_per_stat = stat_ldb_gwp.groupby('ID_STAT')['aantal_gbg'].sum().reset_ind

# Het aantal gebouwen wordt toegevoegd aan de lijst onder de kolom 'aantal_gbg'
aantal_gbg_per_stat.columns = ['ID_STAT', 'aantal_gbg']

# Controleer of de kolom 'aantal_gbg' al bestaat in de lijst
if 'aantal_gbg' in stat.columns:
    # Verwijder de kolom 'aantal_gbg' als deze al bestaat
    stat.drop(columns=['aantal_gbg'], inplace=True)

# De lijst met het aantal gebouwen wordt toegevoegd aan de dataframe van de statist
stat = pd.merge(stat, aantal_gbg_per_stat, on='ID_STAT', how='left')

stat.head()
```

Out[ ]:

	ID_STAT	NISCODE	SECNAAM	geometry	opp_stat	opp_
0	StatSec.1	11001	AARTSELAAR-CENTRUM	MULTIPOLYGON (((151281.422 203009.293, 151249....	5.315896e+05	18526.925
1	StatSec.5	11001	LEUG	MULTIPOLYGON (((150790.516 202810.371, 150745....	3.372691e+05	13410.691
2	StatSec.6	11001	TEN DORPE	MULTIPOLYGON (((151979.438 202728.934, 151850....	2.182698e+05	11293.457
3	StatSec.101	11002	ZANDVLIET-KERN(ZANDVL.-5 DIST)	MULTIPOLYGON (((145788.641 228199.231, 145732....	1.342175e+06	404044.501
4	StatSec.102	11002	STALSHOEK(ZANDVLIET-5 DISTR.)	MULTIPOLYGON (((147287.938 228082.668, 147279....	6.900929e+05	201277.243

In [ ]:

```
# aan de dataframe statistische sectoren wordt een kolom 'gbg_ldb_stat' toegevoegd
stat['gbg_ldb_stat'] = (stat['aantal_gbg']/stat['opp_ldb'])*10**6

stat.head()
```

Out[ ]:

	ID_STAT	NISCODE	SECNAAM	geometry	opp_stat	opp_
0	StatSec.1	11001	AARTSELAAR-CENTRUM	MULTIPOLYGON (((151281.422 203009.293, 151249....	5.315896e+05	18526.925
1	StatSec.5	11001	LEUG	MULTIPOLYGON (((150790.516 202810.371, 150745....	3.372691e+05	13410.691
2	StatSec.6	11001	TEN DORPE	MULTIPOLYGON (((151979.438 202728.934, 151850....	2.182698e+05	11293.457
3	StatSec.101	11002	ZANDVLIET-KERN(ZANDVL.-5 DIST)	MULTIPOLYGON (((145788.641 228199.231, 145732....	1.342175e+06	404044.501
4	StatSec.102	11002	STALSHOEK(ZANDVLIET-5 DISTR.)	MULTIPOLYGON (((147287.938 228082.668, 147279....	6.900929e+05	201277.243

9. In deze stap wordt het dataframe van de gemeenten uitgebreid met data uit het GRB in volgende kolommen:

   - De som van de oppervlaktes van alle gebouwen in landbouwgebied per gemeente
   - Het aantal gebouwen in landbouwgebied per gemeente
   - Het aantal gebouwen per km² landbouwgebied per gemeente

In [ ]:
```
# De dataframe van de geclipte landbouwgebieden wordt gesorteerd per NISCODE en de
ldb_per_NISCODE = stat_ldb_gwp.groupby('NISCODE')['opp_ldb'].sum().reset_index()

# Controleer of de kolom 'opp_ldb' al bestaat in de dataframe van de gemeenten
if 'opp_ldb' in gemeentegrenzen.columns:
    # Verwijder de kolom 'opp_ldb' als deze al bestaat
    gemeentegrenzen.drop(columns=['opp_ldb'], inplace=True)

# De lijst met de opgetelde oppervlakte van het landbouwgebied wordt toegevoegd aan
gemeentegrenzen = pd.merge(gemeentegrenzen, ldb_per_NISCODE, on='NISCODE', how='lef

# Gemeenten zonder landbouwgebied worden verwijderd
gemeentegrenzen = gemeentegrenzen.dropna(subset=['opp_ldb'])

gemeentegrenzen.head()
```

Out[ ]:

	NISCODE	NAAM	geometry	oppervlakte	opp_ldb
0	72043	Pelt	MULTIPOLYGON (((224591.910 218998.880, 224547....	8.378474e+07	3.194366e+07
1	72042	Oudsbergen	MULTIPOLYGON (((231467.154 204604.461, 231454....	1.163707e+08	4.629791e+07
2	12041	Puurs-Sint-Amands	MULTIPOLYGON (((148689.993 197941.126, 148665....	4.913837e+07	2.650007e+07
3	44085	Lievegem	MULTIPOLYGON (((96423.253 207308.996, 96416.05...	8.078116e+07	6.039470e+07
4	44084	Aalter	MULTIPOLYGON (((82999.396 205969.226, 82979.68...	1.198508e+08	8.092602e+07

In [ ]:

```
# De dataframe van de geclipte landbouwgebieden wordt gesorteerd per NISCODE en de
gbg_per_NISCODE = stat_ldb_gwp.groupby('NISCODE')['opp_gbg'].sum().reset_index()

# Controleer of de kolom 'opp_gbg' al bestaat in de dataframe van de gemeenten
if 'opp_gbg' in gemeentegrenzen.columns:
    # Verwijder de kolom 'opp_gbg' als deze al bestaat
    gemeentegrenzen.drop(columns=['opp_gbg'], inplace=True)

# De lijst met de oppervlakte van de gebouwen wordt toegevoegd aan de dataframe van
gemeentegrenzen = pd.merge(gemeentegrenzen, gbg_per_NISCODE, on='NISCODE', how='lef

gemeentegrenzen.head()
```

Out[ ]:

	NISCODE	NAAM	geometry	oppervlakte	opp_ldb	opp_gbg
0	72043	Pelt	MULTIPOLYGON (((224591.910 218998.880, 224547....	8.378474e+07	3.194366e+07	6.394691e+05
1	72042	Oudsbergen	MULTIPOLYGON (((231467.154 204604.461, 231454....	1.163707e+08	4.629791e+07	9.052869e+05
2	12041	Puurs-Sint-Amands	MULTIPOLYGON (((148689.993 197941.126, 148665....	4.913837e+07	2.650007e+07	5.970782e+05
3	44085	Lievegem	MULTIPOLYGON (((96423.253 207308.996, 96416.05...	8.078116e+07	6.039470e+07	1.183029e+06
4	44084	Aalter	MULTIPOLYGON (((82999.396 205969.226, 82979.68...	1.198508e+08	8.092602e+07	1.810233e+06

In [ ]:

```
# De dataframe van de geclipte landbouwgebieden wordt gesorteerd per NISCODE en het
aantal_gbg_per_NISCODE = gbg_results.groupby('NISCODE').size().reset_index()

# De waarden in de kolom NISCODE worden gelezen als een string
aantal_gbg_per_NISCODE['NISCODE'] = aantal_gbg_per_NISCODE['NISCODE'].astype(str)

# Het aantal gebouwen wordt toegevoegd aan de lijst onder de kolom 'aantal_gbg'
aantal_gbg_per_NISCODE.columns = ['NISCODE', 'aantal_gbg']

# Controleer of de kolom 'aantal_gbg' al bestaat in de dataframe
if 'aantal_gbg' in gemeentegrenzen.columns:
    # Verwijder de kolom 'aantal_gbg' als deze al bestaat
    gemeentegrenzen.drop(columns=['aantal_gbg'], inplace=True)

# De lijst met het aantal gebouwen wordt toegevoegd aan de dataframe van de gemeent
gemeentegrenzen = gemeentegrenzen.merge(aantal_gbg_per_NISCODE, on='NISCODE', how='

gemeentegrenzen.head()
```

Out[ ]:

	NISCODE	NAAM	geometry	oppervlakte	opp_ldb	opp_gbg	aant
0	72043	Pelt	MULTIPOLYGON (((224591.910 218998.880, 224547....	8.378474e+07	3.194366e+07	6.394691e+05	
1	72042	Oudsbergen	MULTIPOLYGON (((231467.154 204604.461, 231454....	1.163707e+08	4.629791e+07	9.052869e+05	
2	12041	Puurs-Sint-Amands	MULTIPOLYGON (((148689.993 197941.126, 148665....	4.913837e+07	2.650007e+07	5.970782e+05	
3	44085	Lievegem	MULTIPOLYGON (((96423.253 207308.996, 96416.05...	8.078116e+07	6.039470e+07	1.183029e+06	
4	44084	Aalter	MULTIPOLYGON (((82999.396 205969.226, 82979.68...	1.198508e+08	8.092602e+07	1.810233e+06	

In [ ]:

```
# aan de dataframe gemeenten wordt een kolom 'gbg_ldb_niscode' toegevoegd met het a
gemeentegrenzen['gbg_ldb_niscode'] = (gemeentegrenzen['aantal_gbg']/gemeentegrenzen

gemeentegrenzen.head()
```

Out[ ]:

	NISCODE	NAAM	geometry	oppervlakte	opp_ldb	opp_gbg	aant
0	72043	Pelt	MULTIPOLYGON (((224591.910 218998.880, 224547....	8.378474e+07	3.194366e+07	6.394691e+05	
1	72042	Oudsbergen	MULTIPOLYGON (((231467.154 204604.461, 231454....	1.163707e+08	4.629791e+07	9.052869e+05	
2	12041	Puurs-Sint-Amands	MULTIPOLYGON (((148689.993 197941.126, 148665....	4.913837e+07	2.650007e+07	5.970782e+05	
3	44085	Lievegem	MULTIPOLYGON (((96423.253 207308.996, 96416.05...	8.078116e+07	6.039470e+07	1.183029e+06	
4	44084	Aalter	MULTIPOLYGON (((82999.396 205969.226, 82979.68...	1.198508e+08	8.092602e+07	1.810233e+06	

10. In deze stap wordt het dataframe van de geclipte polygonen nogmaals uitgebreid met de resultaten van de analyse per statistische sector en per gemeente. Dit omdat de geclipte polygonen het meeste interessant zijn om cartografisch te tonen omdat ze de reëele ligging van het landbouwgebied weergeven. De resultaten van het aantal gebouwen per km² landbouwgrond zijn dan weer interessanter op het niveau van de statistische sector of gemeente. De geclipte landbouwpolygonen van het gewestplan hebben immers noooit als doel gehad te dienen tot statistische analyse.

    Concreet worden toegevoegd:

    - de kolom met het aantal gebouwen per km² landbouwgebied uit het dataframe van de statistische sectoren
    - de kolom met het aantal gebouwen per km² landbouwgebied uit het dataframe van de gemeenten

In [ ]:
```
# Voeg de kolom 'gbg_ldb_stat' toe aan stat_ldb_gwp DataFrame op basis van overeenk
stat_ldb_gwp = stat_ldb_gwp.merge(stat[['ID_STAT', 'gbg_ldb_stat']], on='ID_STAT',

stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT	minx	miny	
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1	151489.587910	201819.512243	151611
1	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5	150701.078125	201824.066791	151143
2	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151750.471 202174.322, 151819.765 20...	11001	StatSec.6	151750.471248	202166.824250	152033
3	b9304c99-5dfe-4e3d-a134-50ec24e4526a	POLYGON ((145710.257 226947.227, 145713.937 22...	11002	StatSec.101	145710.257427	226920.590000	146577
4	7e9c2f06-28bd-4ab6-9936-62c825e9e866	POLYGON ((145698.703 227999.637, 145726.437 22...	11002	StatSec.101	145519.593300	227847.355300	145733

In [ ]:

```
# Voeg de kolom 'gbg_ldb_stat' toe aan stat_ldb_gwp DataFrame op basis van overeenk
stat_ldb_gwp = stat_ldb_gwp.merge(gemeentegrenzen[['NISCODE', 'gbg_ldb_niscode']],

stat_ldb_gwp.head()
```

Out[ ]:

	id	geometry	NISCODE	ID_STAT	minx	miny	
0	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151489.588 201819.512, 151512.609 20...	11001	StatSec.1	151489.587910	201819.512243	151611
1	6aa9b437-4916-44b1-bcaa-2a533f6b1543	POLYGON ((150870.515 202200.402, 150962.500 20...	11001	StatSec.5	150701.078125	201824.066791	151143
2	97e45686-e632-4930-a4a8-cf0e80833b5f	POLYGON ((151750.471 202174.322, 151819.765 20...	11001	StatSec.6	151750.471248	202166.824250	152033
3	b9304c99-5dfe-4e3d-a134-50ec24e4526a	POLYGON ((145710.257 226947.227, 145713.937 22...	11002	StatSec.101	145710.257427	226920.590000	146577
4	7e9c2f06-28bd-4ab6-9936-62c825e9e866	POLYGON ((145698.703 227999.637, 145726.437 22...	11002	StatSec.101	145519.593300	227847.355300	145733

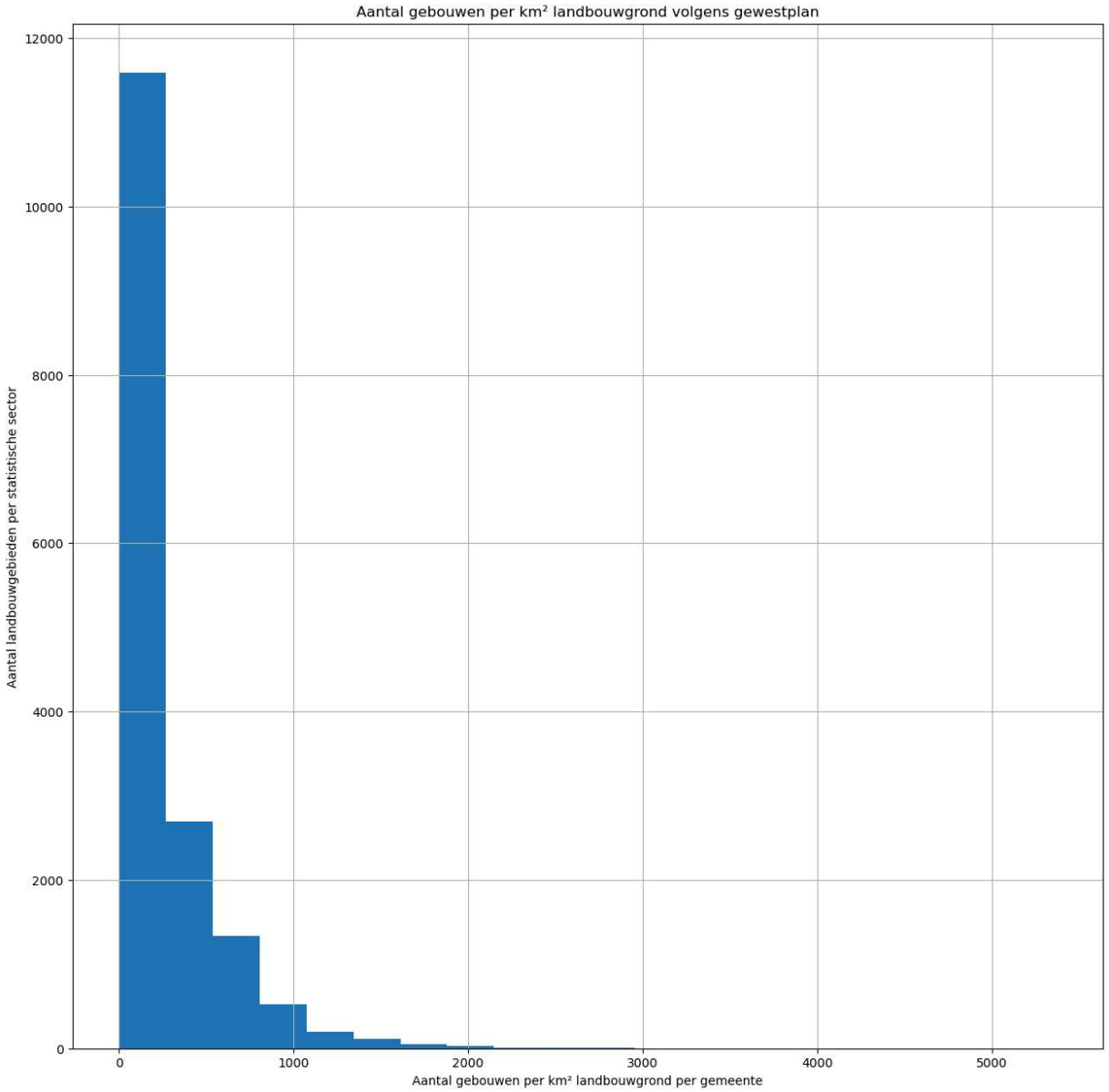
11. De laatste stap is de cartografische voorstelling van de data. Er zijn op basis van de samengestelde dataframes verschillende mogelijkheden om de resultaten te analyseren en te tonen. In dit voorbeeld is gekozen om een choropleetkaart te maken op basis van de polygonen van het gewestplan. Dus er worden enkel zones gekleurd op de kaart die ook landbouwgebied zijn volgens het gewestplan. De lijnen tussen de verschillende gewestplanpolygonen zijn onzichtbaar gemaakt omdat deze niet relevant zijn voor de weer te geven data. De kleuren zijn dan weer gebaseerd op een indeling van klassen van het aantal gebouwen per km² per statistische sector.

    Als ondergrond is gekozen voor een open street map

In [ ]:
```
import matplotlib.pyplot as plt
import mapclassify

plt.rcParams['figure.figsize'] = [15, 15]
```

In [ ]:
```
stat_ldb_gwp['gbg_ldb_stat'].hist(bins=20)
plt.xlabel('Aantal gebouwen per km² landbouwgrond per gemeente')
```

```
plt.ylabel('Aantal landbouwgebieden per statistische sector')
plt.title('Aantal gebouwen per km² landbouwgrond volgens gewestplan')
plt.show()
```



In [ ]:
```
import geopandas as gpd
import contextily as cx
```

In [ ]:
```
stat_ldb_gwp_WM = stat_ldb_gwp.to_crs(epsg=3857)
```

In [ ]:
```
cx.providers.keys()
```

Out[ ]:
```
dict_keys(['OpenStreetMap', 'MapTilesAPI', 'OpenSeaMap', 'OPNVKarte', 'OpenTopoMa
p', 'OpenRailwayMap', 'OpenFireMap', 'SafeCast', 'Stadia', 'Thunderforest', 'CyclO
SM', 'Jawg', 'MapBox', 'MapTiler', 'Stamen', 'TomTom', 'Esri', 'OpenWeatherMap',
'HERE', 'HEREv3', 'FreeMapSK', 'MtbMap', 'CartoDB', 'HikeBike', 'BasemapAT', 'nlma
ps', 'NASAGIBS', 'NLS', 'JusticeMap', 'GeoportailFrance', 'OneMapSG', 'USGS', 'Way
markedTrails', 'OpenAIP', 'OpenSnowMap', 'AzureMaps', 'SwissFederalGeoportal', 'Ga
ode', 'Strava'])
```

In [ ]:

```
ax = stat_ldb_gwp_WM.plot(column='gbg_ldb_stat', scheme='quantiles', k=5, cmap='OrR
cx.add_basemap(ax, source='https://a.tile.openstreetmap.org/{z}/{x}/{y}.png', zoom
ax.set_xlim(275000, 670000)
ax.set_ylim(6560000, 6714000)
ax.set_axis_off()
```