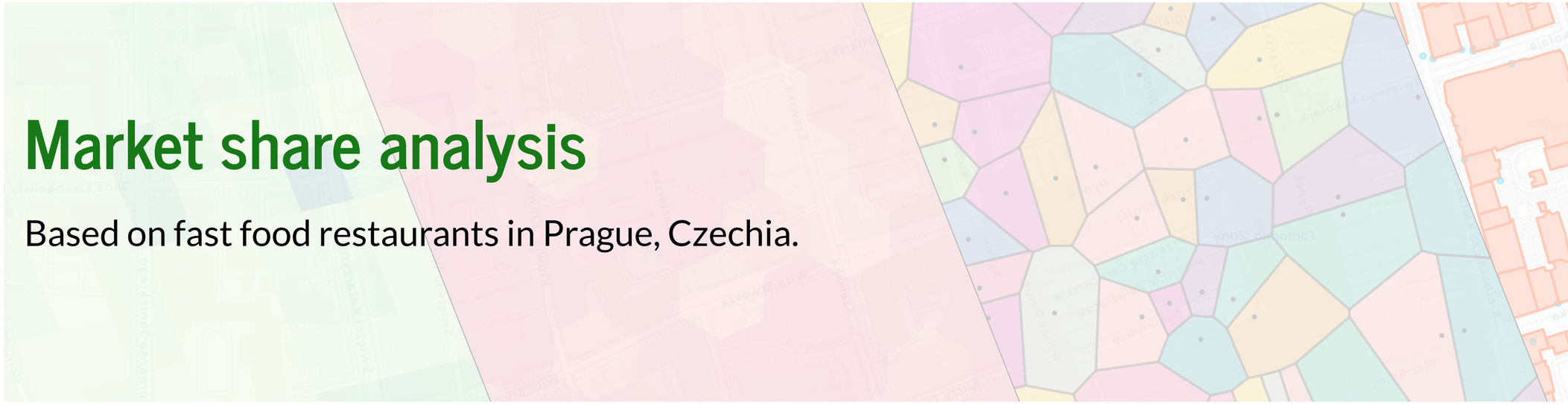
Market share analysis

Based on fast food restaurants in Prague, Czechia.



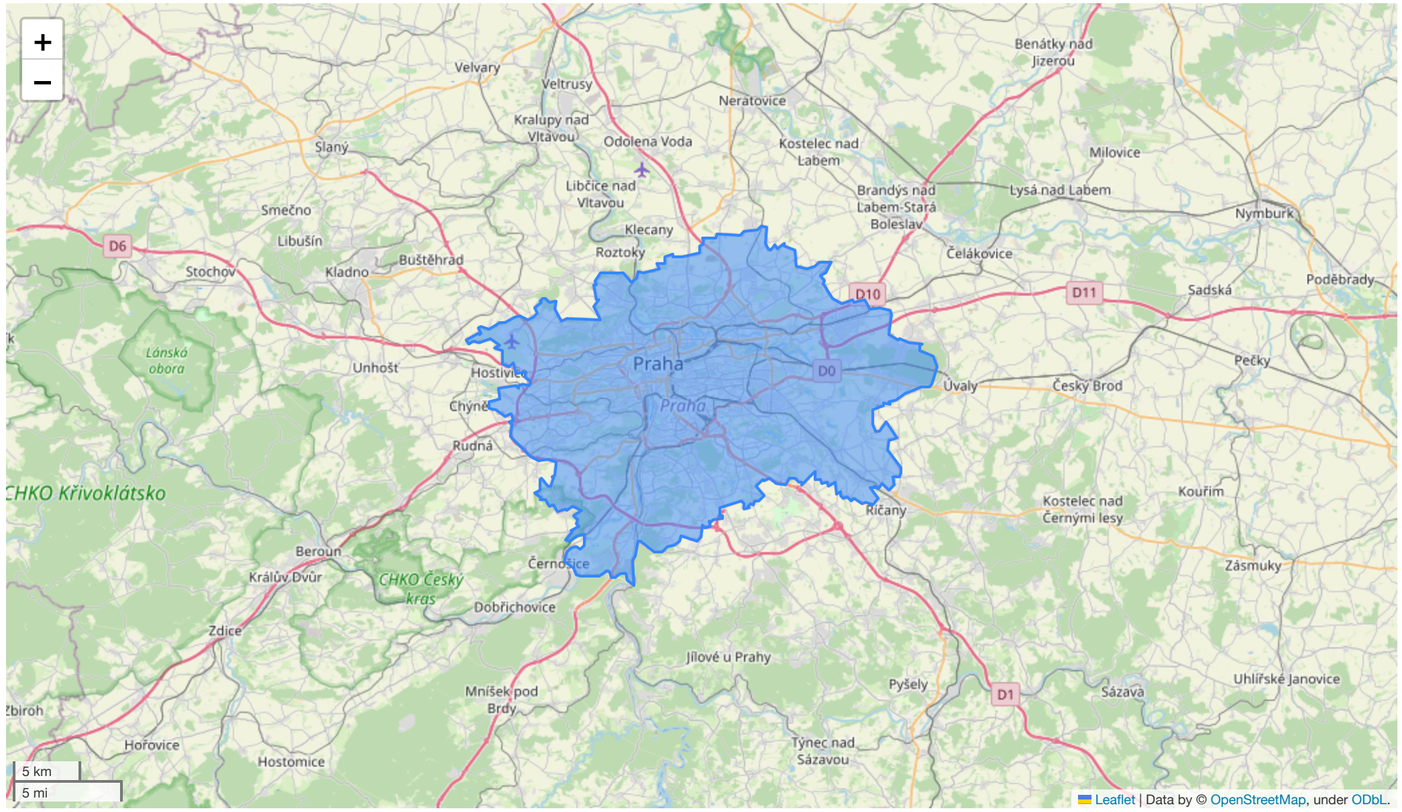
The example will show how to analyse the local market share, defined by the nearest restaurant to a particular Prague resident.

The analysis uses demographic data from the Czech Statistical Office, with residential buildings and fast food restaurant positions downloaded from OpenStreetMap.

In [2]:
```
from srai.regionalizers import geocode_to_region_gdf

prague_area = geocode_to_region_gdf('Praha, CZ')
prague_area.explore(height=600)
```

Out[2]:

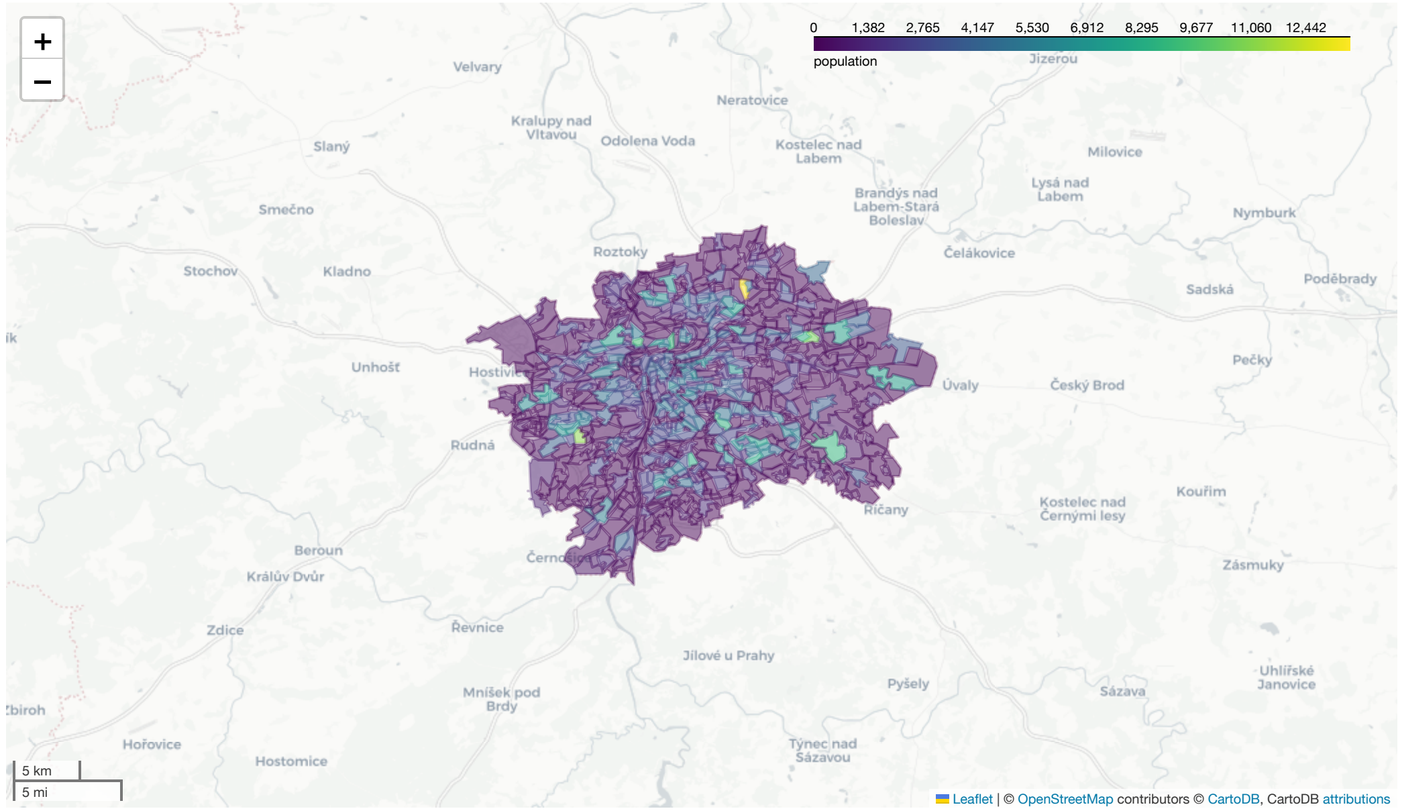


Load demographic data

In [3]:
```
import geopandas as gpd

cadastral_data = gpd.read_file('data/cadastral_data.geojson')
cadastral_data.explore(column='population', tiles="CartoDB positron", style_kwds=dict(opacity
```

Out[3]:



Visit our research group at `kraina.ai`

Load residential buildings from OpenStreetMap

Data that we need is defined by `building=residential` tag in OSM.

Additionally, we want to use `building:flats` information to add weight to each building.

Later we will parse numer of flats per building to a number (OSM tags values are strings).

In [4]:
```
from srai.loaders import OSMOnlineLoader
from utils import map_flats

loader = OSMOnlineLoader()
buildings = loader.load(
    prague_area, {"building": "residential", "building:flats": True}
)
buildings = buildings[
    (buildings["building"] == "residential") & (buildings["building:flats"].notna())
]
```

```
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Downloading building:flats: True : 100%|██████████| 2/2 [00:53<00:00, 26.56s/it]
```

In [5]:
```
buildings["building:flats"] = buildings["building:flats"].apply(map_flats)
buildings.geometry = buildings.geometry.apply(lambda geometry: geometry.centroid)

buildings.head()
```

Out[5]:

feature_id	geometry	building	building:flats
relation/60006	POINT (14.40563 50.08765)	residential	10
relation/60008	POINT (14.40624 50.08758)	residential	9
relation/60009	POINT (14.40635 50.08745)	residential	5
relation/60010	POINT (14.40827 50.08784)	residential	5
relation/60011	POINT (14.40804 50.08783)	residential	3

Population interpolation

Using cadastral information and exact buildings positions, we will interpolate the population over each building using flats number as a weight.

In [6]:
```
from utils import interpolate_spatial_data

interpolate_spatial_data(
    regions=cadastral_data,
    features=buildings,
    weight_column="building:flats",
    result_column="population",
)

buildings.head()
```

```
100%|██████████| 925/925 [00:04<00:00, 196.64it/s]
```

Out[6]:

feature_id	geometry	building	building:flats	population
relation/60006	POINT (14.40563 50.08765)	residential	10	15.624615
relation/60008	POINT (14.40624 50.08758)	residential	9	14.062154
relation/60009	POINT (14.40635 50.08745)	residential	5	7.812308
relation/60010	POINT (14.40827 50.08784)	residential	5	7.812308
relation/60011	POINT (14.40804 50.08783)	residential	3	4.687385

Plotting buildings with population

In [7]:
```
from utils import plot_population

plot_population(buildings)
```

```
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/utils.py:141: UserWarning: Matplotl
ib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI backen
d, so cannot show the figure.
  _ = ax1.axis("off"), ax2.axis("off"), fig.show()
```

Loading data about fast food restaurants

Those features are defined in OSM with `amenity=fast_food` tag.

From those, we will filter out `KFC` and `McDonald's` to simplify the analysis.

In [8]:
```
brands = ["KFC", "McDonald's"]
```

In [9]:
```
pois = loader.load(prague_area, {"amenity": "fast_food", "brand": True})
pois = pois[
    (pois["amenity"] == "fast_food") & (pois["brand"].isin(brands))
]
pois.head()
```

```
Downloading brand: True       : 100%|██████████| 2/2 [00:21<00:00, 10.71s/it]
```

Out[9]:

feature_id	geometry	amenity	brand
node/29902474	POINT (14.44226 50.03388)	fast_food	McDonald's
node/71672422	POINT (14.42329 50.08016)	fast_food	McDonald's
node/280948613	POINT (14.49996 50.11965)	fast_food	McDonald's
node/304155379	POINT (14.39634 50.09984)	fast_food	KFC
node/317006604	POINT (14.43166 50.07531)	fast_food	KFC

In [12]:
```
pois.geometry = pois.geometry.apply(lambda geometry: geometry.centroid)
pois.head()
```

Out[12]:

feature_id	geometry	amenity	brand
node/29902474	POINT (14.44226 50.03388)	fast_food	McDonald's
node/71672422	POINT (14.42329 50.08016)	fast_food	McDonald's
node/280948613	POINT (14.49996 50.11965)	fast_food	McDonald's
node/304155379	POINT (14.39634 50.09984)	fast_food	KFC
node/317006604	POINT (14.43166 50.07531)	fast_food	KFC

In [13]:
```
pois.brand.value_counts()
```

Out[13]:
```
brand
McDonald's    37
KFC           30
Name: count, dtype: int64
```

Segmenting the area

Using `VoronoiRegionalizer` from `srai` library, we can divide the geospatial space into regions using Voronoi diagram.

Here we will be using restaurants as seeds to segment the Prague.

In [14]:
```
from srai.regionalizers import VoronoiRegionalizer

voronoi_regions = VoronoiRegionalizer(seeds=pois).transform(gdf=prague_area)
voronoi_regions.head()
```

```
<frozen importlib._bootstrap>:241: RuntimeWarning: numpy.ndarray size changed, may indicat
e binary incompatibility. Expected 88 from C header, got 96 from PyObject
Generating regions:   0%|
| 0/67 [00:00<?, ?it/s]/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/pyt
hon3.10/site-packages/spherical_geometry/great_circle_arc.py:365: RuntimeWarning: invalid
value encountered in divide
  return P / l
Generating regions: 100%|
| 67/67 [00:03<00:00, 17.79it/s]
```

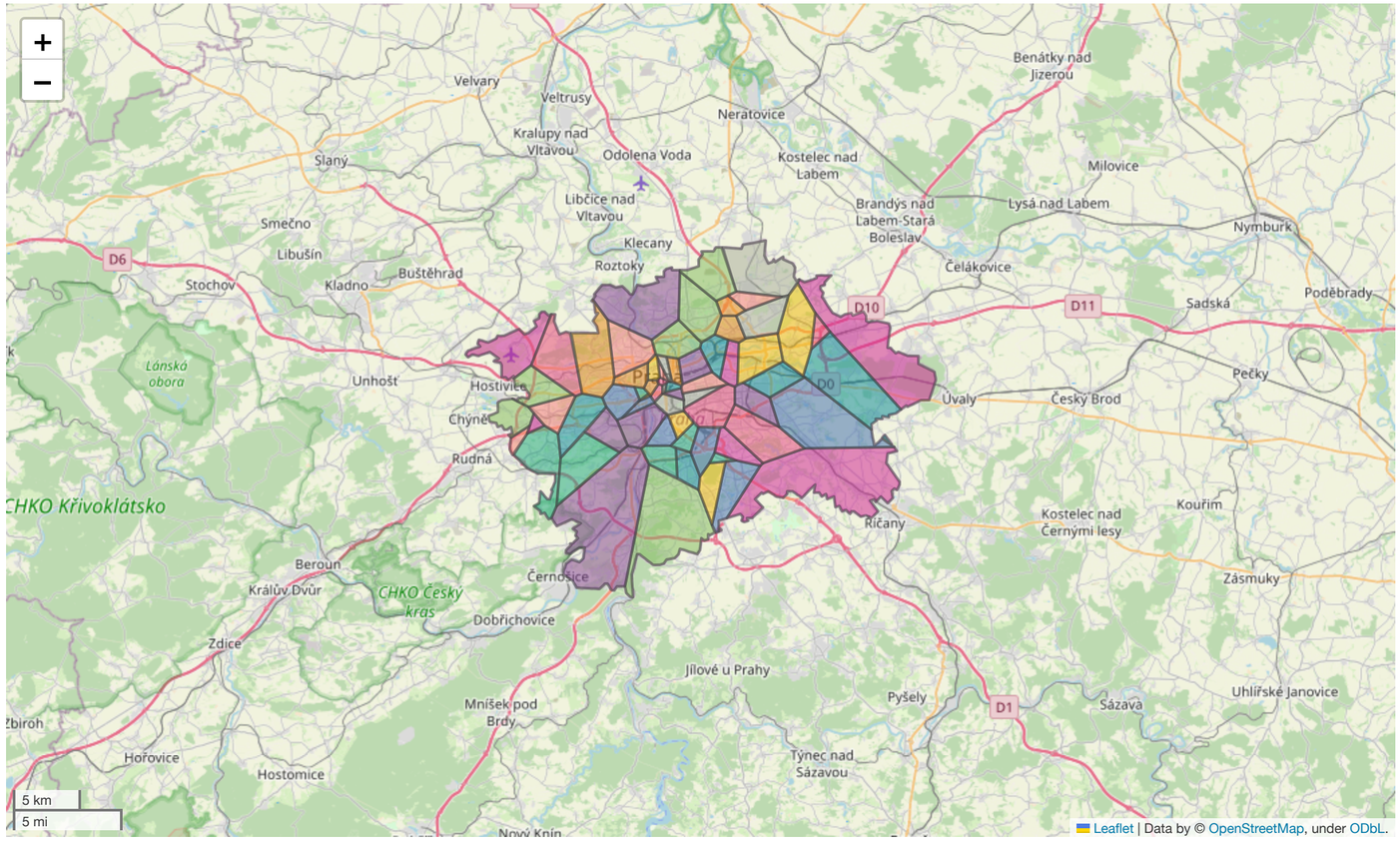
Out[14]:

region_id	geometry
node/3030632237	MULTIPOLYGON (((14.52930 50.03067, 14.57167 50...
node/5141761322	POLYGON ((14.41336 50.02974, 14.44130 50.02125...
way/909280503	POLYGON ((14.38805 49.94965, 14.38797 49.94978...
way/118749708	POLYGON ((14.31886 50.04974, 14.32236 50.03443...
node/4245196374	POLYGON ((14.29060 50.05544, 14.28786 50.04977...

In [15]:
```
from srai.plotting import plot_regions

plot_regions(voronoi_regions)
```

Out[15]:



Now we can join buildings with population into those generated regions. This way, we can assign the closest restaurant to each building.

In [16]:
```
population_in_regions = (
    voronoi_regions.sjoin(buildings).groupby("region_id")["population"].sum()
)
regions_with_population = (
    voronoi_regions.join(pois[["brand"]]).join(population_in_regions).fillna(0)
)
regions_with_population.head()
```

Out[16]:

region_id	geometry	brand	population
node/3030632237	MULTIPOLYGON (((14.52930 50.03067, 14.57167 50...	McDonald's	40833.673424
node/5141761322	POLYGON ((14.41336 50.02974, 14.44130 50.02125...	McDonald's	73422.872771
way/909280503	POLYGON ((14.38805 49.94965, 14.38797 49.94978...	KFC	38665.384888
way/118749708	POLYGON ((14.31886 50.04974, 14.32236 50.03443...	McDonald's	6888.584151
node/4245196374	POLYGON ((14.29060 50.05544, 14.28786 50.04977...	KFC	438.387097

Using simple grouping operation, we can see what is the Prague's market share between those two brands.

In [17]:
```
brand_population = (
    regions_with_population.groupby("brand")
    .agg({"population": "sum", "geometry": "count"})
    .reset_index()
)
brand_population.rename(columns={"geometry": "locations"}, inplace=True)
brand_population["percentage"] = (
    100 * brand_population["population"] / brand_population["population"].sum()
)
brand_population.sort_values(by='percentage', ascending=False)
```

Out[17]:

	brand	population	locations	percentage
1	McDonald's	723334.197739	37	55.759818
0	KFC	573897.802261	30	44.240182

Map plotting

To analyse this market further, we will plot the regions in two distinc gradients based on brand's color.

In [18]:
```
from utils import plot_market_share

plot_market_share(regions_with_population, pois)
```

Out[18]: