



escola
britânica de
artes criativas
& tecnologia

Módulo | Análise de Dados: Análise Exploratória de Dados de Logística

Caderno de **Exercícios**

Professor [André Perez](#)

Tópicos

1. Contexto;
2. Manipulação;
3. Visualização;
4. Storytelling.

1. Contexto

Este projeto é uma iniciativa da EBAC (Escola Britânica de Artes Criativas e Tecnologia) que propõe uma análise exploratória aplicada ao conjunto de dados da Loggi BUD, com o objetivo de investigar o Problema de Roteamento de Veículos com Capacidade (CVRP) na cidade de Brasília. O CVRP é um desafio essencial em logística, pois busca otimizar a entrega de mercadorias considerando restrições como a capacidade dos veículos e a demanda dos clientes, ao mesmo tempo que minimiza os custos totais de transporte.

Para abordar este problema, utilizamos um conjunto de ferramentas e amplamente reconhecidas na análise de dados como o Pandas, Seaborn e Geopandas.

A análise explora múltiplos aspectos do conjunto de dados, incluindo:

- A distribuição espacial das entregas na cidade de Brasília e seus arredores.
- A identificação de regiões com maior concentração de entregas.
- A análise de padrões que possam otimizar a logística de distribuição.

Por meio dessa abordagem, o projeto não apenas fornece insights sobre o conjunto de dados, mas também demonstra como métodos de análise e visualização podem ser

aplicados para resolver problemas reais no contexto da logística urbana.

2. Pacotes e bibliotecas

```
In [1]: import json

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import geopandas
import geopy
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter
```

3. Exploração de dados

Coleta de Dados

3.1 Loggi BUD

O Loggi Benchmark for Urban Deliveries (BUD) é um repositório do GitHub ([link](#)) com dados e códigos para problemas típicos que empresas de logística enfrentam: otimização das rotas de entrega, alocação de entregas nos veículos da frota com capacidade limitada, etc. Os dados são sintetizados de fontes públicas (IBGE, IPEA, etc.) e são representativos dos desafios que a startup enfrenta no dia a dia, especialmente com relação a sua escala.

Atenção: Vou trabalhar com um sub conjunto dos dados originais presentes neste [link](#). Em especial, o professor André Perez consolidou em um único arquivo `JSON` as instâncias de treino de `cvrp` da cidade de Brasília.

O dado bruto é um arquivo do tipo `JSON` com uma lista de instâncias de entregas. Cada instância representa um conjunto de **entregas** que devem ser realizadas pelos **veículos** do **hub** regional.

```
[
  {
    "name": "cvrp-0-df-0",
    "region": "df-0",
    "origin": {"lng": -47.802664728268745, "lat": -15.657013854445248},
    "vehicle_capacity": 180,
    "deliveries": [
      {
        "id": "ed0993f8cc70d998342f38ee827176dc",
        "point": {"lng": -47.7496622016347, "lat": -15.65879313293694},
        "size": 10
      }
    ]
  }
]
```

```

    },
    {
        "id": "c7220154adc7a3def8f0b2b8a42677a9",
        "point": {"lng": -47.75887552060412, "lat":
-15.651440380492554},
        "size": 10
    },
    ...
]
}
]
...

```

Onde:

- **name:** uma `string` com o nome único da instância;
- **region:** uma `string` com o nome único da região do **hub**;
- **origin:** um `dict` com a latitude e longitude da região do **hub**;
- **vehicle_capacity:** um `int` com a soma da capacidade de carga dos **veículos** do **hub**;
- **deliveries:** uma `list` de `dict` com as **entregas** que devem ser realizadas.

Sendo que:

- **id:** uma `string` com o id único da **entrega**;
- **point:** um `dict` com a latitude e longitude da **entrega**;
- **size:** um `int` com o tamanho ou a carga que a **entrega** ocupa no **veículo**.

In [2]: `!wget -q "https://raw.githubusercontent.com/andre-marcos-perez/ebac-course-utils`

In [3]: `# Abrindo o arquivo json
with open('deliveries.json', mode='r', encoding='utf8') as file:
 data = json.load(file) #salvando em um dicionário`

Wrangling da Estrutura

In [4]: `# Transformando a base de dados (json) em um DataFrame (pandas)
deliveries_df = pd.DataFrame(data)
deliveries_df.head()`

Out[4]:

	name	region	origin	vehicle_capacity	deliv
0	cvrp-2-df-33	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...}	180	'313483a19d2f8d65cd5024c8d215c'
1	cvrp-2-df-73	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...}	180	'bf3fc630b1c29601a4caf1bdd474'
2	cvrp-2-df-20	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...}	180	'b30f1145a2ba4e0b9ac0162b68d04'
3	cvrp-1-df-71	df-1	{'lng': -47.89366206897872, 'lat': -15.8051175...}	180	'be3ed547394196c12c7c27c89ac74'
4	cvrp-2-df-87	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...}	180	'a6328fb4dc0654eb28a996a270b0f'

Como podemos perceber a coluna origin contém um dicionário dentro dela, vamos criar novas colunas agregando esses dados. O processo chamado flatten(achatamento) de dados aninhados como este.

In [5]: `#visualização da coluna "origin"`
`deliveries_df["origin"].head()`

Out[5]:

	origin
0	{'lng': -48.05498915846707, 'lat': -15.8381445...}
1	{'lng': -48.05498915846707, 'lat': -15.8381445...}
2	{'lng': -48.05498915846707, 'lat': -15.8381445...}
3	{'lng': -47.89366206897872, 'lat': -15.8051175...}
4	{'lng': -48.05498915846707, 'lat': -15.8381445...}

dtype: object

In [6]: `# normalizando a coluna "origin" e a salvando em um novo df`
`hub_origin_df = pd.json_normalize(deliveries_df['origin'])`
`hub_origin_df.head()`


Out[6]:

	lng	lat
0	-48.054989	-15.838145
1	-48.054989	-15.838145
2	-48.054989	-15.838145
3	-47.893662	-15.805118
4	-48.054989	-15.838145

In [7]: *# Concatenando ambos dataframes com o método merge conservando os índices*
`deliveries_df = pd.merge(left=deliveries_df, right=hub_origin_df, how='inner', le
deliveries_df.head()`

Out[7]:

	name	region	origin	vehicle_capacity	deliv
0	cvrp- 2-df- 33	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...	180	'313483a19d2f8d65cd5024c8d215c
1	cvrp- 2-df- 73	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...	180	'bf3fc630b1c29601a4caf1bdd474
2	cvrp- 2-df- 20	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...	180	'b30f1145a2ba4e0b9ac0162b68d04
3	cvrp- 1-df- 71	df-1	{'lng': -47.89366206897872, 'lat': -15.8051175...	180	'be3ed547394196c12c7c27c89ac74
4	cvrp- 2-df- 87	df-2	{'lng': -48.05498915846707, 'lat': -15.8381445...	180	'a6328fb4dc0654eb28a996a270b0f

◀  ▶

In [8]: *# Apagando a coluna "origin"*
`deliveries_df = deliveries_df.drop("origin", axis=1)`

Reordenando as colunas
`deliveries_df = deliveries_df[["name", "region", "lng", "lat", "vehicle_capacity
deliveries_df.head()`

Out[8]:

	name	region	lng	lat	vehicle_capacity	de
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	'313483a19d2f8d65cd5024c8d2
1	cvrp-2-df-73	df-2	-48.054989	-15.838145	180	'bf3fc630b1c29601a4caf1bdd4
2	cvrp-2-df-20	df-2	-48.054989	-15.838145	180	'b30f1145a2ba4e0b9ac0162b68c
3	cvrp-1-df-71	df-1	-47.893662	-15.805118	180	'be3ed547394196c12c7c27c89ac
4	cvrp-2-df-87	df-2	-48.054989	-15.838145	180	'a6328fb4dc0654eb28a996a270

In [9]: `# Renomeando as colunas "lng" e "lat" para "hub_lat" e "hub_lng"`
`deliveries_df.rename(columns={"lng": "hub_lng", "lat": "hub_lat"}, inplace=True)`
`deliveries_df.head()`

Out[9]:

	name	region	hub_lng	hub_lat	vehicle_capacity	de
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	'313483a19d2f8d65cd5024c8d2
1	cvrp-2-df-73	df-2	-48.054989	-15.838145	180	'bf3fc630b1c29601a4caf1bdd4
2	cvrp-2-df-20	df-2	-48.054989	-15.838145	180	'b30f1145a2ba4e0b9ac0162b68c
3	cvrp-1-df-71	df-1	-47.893662	-15.805118	180	'be3ed547394196c12c7c27c89ac
4	cvrp-2-df-87	df-2	-48.054989	-15.838145	180	'a6328fb4dc0654eb28a996a270

Agora vamos normalizar a coluna deliveries, usando o explode para criar uma nova linha para cada valor que está contido em cada índice que continha em cada um, uma lista de dicionários.

In [10]: `# Transformando cada elemento da lista na coluna "deliveries" em uma linha do df`
`deliveries_exploded_df = deliveries_df[["deliveries"]].explode("deliveries")`
`deliveries_exploded_df.head()`

Out[10]:

deliveries

```
0 {'id': '313483a19d2f8d65cd5024c8d215cfbd', 'po...
0 {'id': '320c94b17aa685c939b3f3244c3099de', 'po...
0 {'id': '3663b42f4b8decb33059febaba46d5c8', 'po...
0 {'id': 'e11ab58363c38d6abc90d5fba87b7d7', 'poi...
0 {'id': '54cb45b7bbbd4e34e7150900f92d7f4b', 'po...
```

In [11]:

```
# Normalizando a coluna e já concatenando os dataframes em um único dataframe
deliveries_normalized_df = pd.concat([
    pd.DataFrame(deliveries_exploded_df["deliveries"].apply(lambda record: record[
    pd.DataFrame(deliveries_exploded_df["deliveries"].apply(lambda record: record[
    pd.DataFrame(deliveries_exploded_df["deliveries"].apply(lambda record: record[
]), axis= 1)
deliveries_normalized_df
```

Out[11]:

	delivery_size	delivery_lng	delivery_lat
0	9	-48.116189	-15.848929
0	2	-48.118195	-15.850772
0	1	-48.112483	-15.847871
0	2	-48.118023	-15.846471
0	7	-48.114898	-15.858055
...
198	8	-48.064269	-15.997694
198	4	-48.065176	-16.003597
198	9	-48.065841	-16.003808
198	1	-48.062327	-16.001568
198	9	-48.059420	-16.009234

636149 rows × 3 columns

In [12]:


```
# Apagando a coluna "deliveries" do df principal
deliveries_df = deliveries_df.drop("deliveries", axis=1)

# Concatenando o df principal com o df das entregas explodido e conservando os índices
deliveries_df = pd.merge(left=deliveries_df, right=deliveries_normalized_df, how='outer')
deliveries_df.reset_index(inplace=True, drop=True)

deliveries_df.head()
```

Out[12]:

	name	region	hub_lng	hub_lat	vehicle_capacity	delivery_size	delivery_lng	d
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	9	-48.116189	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	2	-48.118195	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	1	-48.112483	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	2	-48.118023	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	7	-48.114898	



Estrutura

In [13]: *# Descobrindo o número de linhas e colunas*
`deliveries_df.shape`

Out[13]: (636149, 8)

In [14]: *# Descobrindo o nome das colunas*
`deliveries_df.columns`

Out[14]: Index(['name', 'region', 'hub_lng', 'hub_lat', 'vehicle_capacity',
'delivery_size', 'delivery_lng', 'delivery_lat'],
dtype='object')

In [15]: *# Descobrindo o número de índices e o passo (de quanto em quanto)*
`deliveries_df.index`

Out[15]: RangeIndex(start=0, stop=636149, step=1)

In [16]: `deliveries_df.info()`


```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636149 entries, 0 to 636148
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   name                   636149 non-null object
1   region                 636149 non-null object
2   hub_lng                636149 non-null float64
3   hub_lat                636149 non-null float64
4   vehicle_capacity       636149 non-null int64
5   delivery_size          636149 non-null int64
6   delivery_lng           636149 non-null float64
7   delivery_lat           636149 non-null float64
dtypes: float64(4), int64(2), object(2)
memory usage: 38.8+ MB


```

Schema

In [17]: `deliveries_df.head()`

Out[17]:

	name	region	hub_lng	hub_lat	vehicle_capacity	delivery_size	delivery_lng	d
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	9	-48.116189	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	2	-48.118195	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	1	-48.112483	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	2	-48.118023	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	7	-48.114898	



In [18]: `# Tipo de dado de cada coluna`
`deliveries_df.dtypes`

Out[18]: 0

name	object
region	object
hub_lng	float64
hub_lat	float64
vehicle_capacity	int64
delivery_size	int64
delivery_lng	float64
delivery_lat	float64

dtype: object

```
In [19]: # Atributos categóricos
deliveries_df.select_dtypes("object").describe().transpose()
```

	count	unique	top	freq
name	636149	199	cvrp-1-df-87	5636
region	636149	3	df-1	304708

```
In [20]: # Atributos numéricos
deliveries_df.drop(["name", "region"], axis=1).select_dtypes('int64').describe()
```

	count	mean	std	min	25%	50%	75%	max
vehicle_capacity	636149.0	180.000000	0.000000	180.0	180.0	180.0	180.0	180.0
delivery_size	636149.0	5.512111	2.874557	1.0	3.0	6.0	8.0	10.0

```
In [21]: # Verificando dados faltantes no df
deliveries_df.isna().any()
```

Out[21]: 0

name	False
region	False
hub_lng	False
hub_lat	False
vehicle_capacity	False
delivery_size	False
delivery_lng	False
delivery_lat	False

dtype: bool

In [22]: *# Por fim o .head() do nosso df*
deliveries_df.head()

Out[22]:

	name	region	hub_lng	hub_lat	vehicle_capacity	delivery_size	delivery_lng	d
--	------	--------	---------	---------	------------------	---------------	--------------	---

0	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	9	-48.116189	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	2	-48.118195	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	1	-48.112483	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	2	-48.118023	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	180	7	-48.114898	



4. Manipulação

Enriquecimento dos dados

Usaremos aqui os dados de latitude e longitude existente na nossa base de dados para acessar e agregar mais informações aos nossos dados. A geocodificação reversa transforma essas coordenadas geográficas de um local em suas respectivas descrições textuais.

```
In [23]: # Isolando num df com as colunas 'region', 'hub_lng' e 'hub_lat'
hub_df = deliveries_df[["region", "hub_lng", "hub_lat"]]

# Apagando os valores duplicados e ordenando o df pela coluna 'region'
hub_df = hub_df.drop_duplicates().sort_values(by="region").reset_index(drop=True)
hub_df.head()
```

```
Out[23]:
```

	region	hub_lng	hub_lat
0	df-0	-47.802665	-15.657014
1	df-1	-47.893662	-15.805118
2	df-2	-48.054989	-15.838145

Aqui poderemos enriquecer nossos dados com essas informações. Segue um exemplo do uso do Nominatim, a biblioteca que usaremos para processar nossa base e fazer geolocalização reversa. Para usar essa ferramenta teremos que estabelecer um timer e respeitar seu uso gratuito que somente aceita uma consulta por segundo.

```
In [24]: # Identificando o usuário
geolocator = Nominatim(user_agent="ebac_geocoder")

# Passando as coordenadas para a geocodificação reversa
location = geolocator.reverse("-15.657013854445248, -47.802664728268745")

#Exibindo as informações
print(json.dumps(location.raw, indent=2, ensure_ascii=False))
```

```
{
  "place_id": 14416233,
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0. http://osm.org/copyrig
ht",
  "osm_type": "way",
  "osm_id": 240210480,
  "lat": "-15.656916027876347",
  "lon": "-47.80264463632131",
  "class": "highway",
  "type": "secondary",
  "place_rank": 26,
  "importance": 0.053411383993285995,
  "addresstype": "road",
  "name": "Rua 7",
  "display_name": "Rua 7, Quadra 2, Vila DNOCS, Sobradinho, Região Geográfica Imediata do Distrito Federal, Região Integrada de Desenvolvimento do Distrito Federal e Entorno, Região Geográfica Intermediária do Distrito Federal, Distrito Federal, Região Centro-Oeste, 73015-202, Brasil",
  "address": {
    "road": "Rua 7",
    "residential": "Quadra 2",
    "suburb": "Vila DNOCS",
    "town": "Sobradinho",
    "municipality": "Região Geográfica Imediata do Distrito Federal",
    "county": "Região Integrada de Desenvolvimento do Distrito Federal e Entorno",
    "state_district": "Região Geográfica Intermediária do Distrito Federal",
    "state": "Distrito Federal",
    "ISO3166-2-lvl4": "BR-DF",
    "region": "Região Centro-Oeste",
    "postcode": "73015-202",
    "country": "Brasil",
    "country_code": "br"
  },
  "boundingbox": [
    "-15.6572841",
    "-15.6565043",
    "-47.8047361",
    "-47.8007862"
  ]
}
```

```
In [25]: # Definindo o delay de 1 consulta/segundo
geocoder = RateLimiter(geolocator.reverse, min_delay_seconds=1)
```

Vamos então aplicar a geocodificação nas coordenadas das três regiões e extrair informações de **cidade** e **bairro**.

```
In [26]: # Criando uma coluna chamada 'coordinates' com os valores de 'hub_lat' e 'hub_lng'
hub_df["coordinates"] = hub_df["hub_lat"].astype(str) + ", " + hub_df["hub_lng"]

# Criando uma coluna chamada 'geodata' com as informações da geocodificação
hub_df["geodata"] = hub_df["coordinates"].apply(geocoder)
hub_df.head()
```

Out[26]:	region	hub_lng	hub_lat	coordinates	geodata
0	df-0	-47.802665	-15.657014	-15.657013854445248, -47.802664728268745	(Rua 7, Quadra 2, Vila DNOCS, Sobradinho, Regi...
1	df-1	-47.893662	-15.805118	-15.80511751066334, -47.89366206897872	(SQS 303, Asa Sul, Brasília, Plano Piloto, Reg...
2	df-2	-48.054989	-15.838145	-15.83814451122274, -48.05498915846707	(Armazém do Bolo, lote 4/8, CSB 4/5, Setor B S...

```
In [27]: # Criando um novo df com a coluna 'geodata' normalizada
hub_geodata_df = pd.json_normalize(hub_df["geodata"].apply(lambda data: data.raw
hub_geodata_df.head()
```

Out[27]:	place_id	licence	osm_type	osm_id	lat
0	14416233	Data © OpenStreetMap contributors, ODbL 1.0. h...	way	240210480	-15.656916027876347
1	14619249	Data © OpenStreetMap contributors, ODbL 1.0. h...	way	66353368	-15.805172753950067
2	11654896	Data © OpenStreetMap contributors, ODbL 1.0. h...	node	6249717596	-15.8384371

3 rows × 32 columns

```
In [28]: print(hub_geodata_df.columns)

Index(['place_id', 'licence', 'osm_type', 'osm_id', 'lat', 'lon', 'class',
      'type', 'place_rank', 'importance', 'addresstype', 'name',
      'display_name', 'boundingbox', 'address.road', 'address.residential',
      'address.suburb', 'address.town', 'address.municipality',
      'address.county', 'address.state_district', 'address.state',
      'address.ISO3166-2-lvl4', 'address.region', 'address.postcode',
      'address.country', 'address.country_code', 'address.neighbourhood',
      'address.city', 'address.shop', 'address.house_number',
      'address.quarter'],
      dtype='object')
```

```
In [29]: # Selecionando as colunas do meu interesse
hub_geodata_df = hub_geodata_df[["address.town", "address.suburb", "address.city

# Renomeando as colunas
hub_geodata_df.rename(columns={"address.town": "hub_town", "address.suburb": "hu
```

```
# Filtrando valores nulos
hub_geodata_df["hub_city"] = np.where(hub_geodata_df["hub_city"].notna(), hub_ge
hub_geodata_df["hub_suburb"] = np.where(hub_geodata_df["hub_suburb"].notna(), hu

# Apagando a coluna 'hub_town'
hub_geodata_df = hub_geodata_df.drop("hub_town", axis=1)

hub_geodata_df.head()
```

Out[29]:

	hub_suburb	hub_city
0	Vila DNOCS	Sobradinho
1	Asa Sul	Brasília
2	Taguatinga Centro	Taguatinga

In [30]:

```
# Concatenando os dataframes preservando os índices
hub_df = pd.merge(left=hub_df, right=hub_geodata_df, left_index=True, right_index=True)

# Selecionando as colunas do meu interesse
hub_df = hub_df[["region", "hub_suburb", "hub_city"]]

hub_df.head()
```

Out[30]:

	region	hub_suburb	hub_city
0	df-0	Vila DNOCS	Sobradinho
1	df-1	Asa Sul	Brasília
2	df-2	Taguatinga Centro	Taguatinga

In [31]:

```
# Concatenando os dataframes
deliveries_df = pd.merge(left=deliveries_df, right=hub_df, how="inner", on="region")

# Selecionando as colunas do meu interesse
deliveries_df = deliveries_df[["name", "region", "hub_lng", "hub_lat", "hub_city"]]

deliveries_df.head()
```

Out[31]:

	name	region	hub_lng	hub_lat	hub_city	hub_suburb	vehicle_capacity	deli
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	

Baixando os dados de geocodificação reversa referente a coluna 'deliveries' disponibilizados pelo Professor André

In [32]: `!wget "https://raw.githubusercontent.com/andre-marcos-perez/ebac-course-utils/main/dataset/deliveries-geodata.csv"`

```
--2025-03-22 12:09:09-- https://raw.githubusercontent.com/andre-marcos-perez/ebac-course-utils/main/dataset/deliveries-geodata.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38648908 (37M) [text/plain]
Saving to: 'deliveries-geodata.csv'

deliveries-geodata. 100%[=====>] 36.86M  227MB/s  in 0.2s

2025-03-22 12:09:10 (227 MB/s) - 'deliveries-geodata.csv' saved [38648908/38648908]
```

In [33]: `# Abrindo o arquivo csv com o pandas e criando um df`
`deliveries_geodata_df = pd.read_csv("deliveries-geodata.csv")`
`deliveries_geodata_df.head()`


```
Out[33]:
```

	delivery_lng	delivery_lat	delivery_city	delivery_suburb
0	-48.116189	-15.848929	Ceilândia	P Sul
1	-48.118195	-15.850772	Ceilândia	P Sul
2	-48.112483	-15.847871	Ceilândia	P Sul
3	-48.118023	-15.846471	Ceilândia	P Sul
4	-48.114898	-15.858055	Sol Nascente/Pôr do Sol	Sol Nascente/Pôr do Sol

```
In [34]: # Concatenando os dataframes
deliveries_df = pd.merge(left=deliveries_df, right=deliveries_geodata_df[["deliv
deliveries_df.head()
```

```
Out[34]:
```

	name	region	hub_lng	hub_lat	hub_city	hub_suburb	vehicle_capacity	deli
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	

Qualidade

Aqui vamos verificar a qualidade de nossos dados.

```
In [35]: deliveries_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636149 entries, 0 to 636148
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   name                   636149 non-null object
1   region                 636149 non-null object
2   hub_lng                636149 non-null float64
3   hub_lat                636149 non-null float64
4   hub_city               636149 non-null object
5   hub_suburb             636149 non-null object
6   vehicle_capacity       636149 non-null int64
7   delivery_size          636149 non-null int64
8   delivery_lng           636149 non-null float64
9   delivery_lat           636149 non-null float64
10  delivery_city           634447 non-null object
11  delivery_suburb         476264 non-null object
dtypes: float64(4), int64(2), object(6)
memory usage: 58.2+ MB
```

```
In [36]: # Verificando quais colunas possuem valores nulos
deliveries_df.isna().any()
```

```
Out[36]:
```

	0
name	False
region	False
hub_lng	False
hub_lat	False
hub_city	False
hub_suburb	False
vehicle_capacity	False
delivery_size	False
delivery_lng	False
delivery_lat	False
delivery_city	True
delivery_suburb	True

dtype: bool

Aqui justamente a divisão que ocorre dos bairros entre `delivery_city` e `delivery_suburb` tem valores nulos. O bairro que não tem em uma coluna, tem em outra.

5. Visualização

```
In [37]: !wget -q "https://geoftp.ibge.gov.br/cartas_e_mapas/bases_cartograficas_continua
!unzip -q distrito-federal.zip -d ./maps
```

```
!cp ./maps/LIM_Unidade_Federacao_A.shp ./distrito-federal.shp
!cp ./maps/LIM_Unidade_Federacao_A.shx ./distrito-federal.shx
```

```
In [38]: # Lendo o arquivo com o geopandas e armazenando na variável 'mapa'
mapa = geopandas.read_file("distrito-federal.shp")

# Selecionando a primeira linha
mapa = mapa.loc[[0]]

mapa.head()
```

```
Out[38]:
```

	geometry
0	POLYGON Z ((-47.31048 -16.03602 0, -47.31057 -...

```
In [39]: # Criando um df com as colunas de outro df e removendo os valores duplicados
hub_df = deliveries_df[["region", "hub_lng", "hub_lat"]].drop_duplicates().reset_index()

# Criando um df a partir de outro já existente
geo_hub_df = geopandas.GeoDataFrame(hub_df, geometry=geopandas.points_from_xy(hub_df["hub_lng"], hub_df["hub_lat"]))

geo_hub_df.head()
```

```
Out[39]:
```

	region	hub_lng	hub_lat	geometry
0	df-2	-48.054989	-15.838145	POINT (-48.05499 -15.83814)
1	df-1	-47.893662	-15.805118	POINT (-47.89366 -15.80512)
2	df-0	-47.802665	-15.657014	POINT (-47.80266 -15.65701)

```
In [40]: # Criando um df a partir de outro já existente
geo_deliveries_df = geopandas.GeoDataFrame(deliveries_df, geometry=geopandas.points_from_xy(deliveries_df["hub_lng"], deliveries_df["hub_lat"]))

geo_deliveries_df.head()
```

Out[40]:

	name	region	hub_lng	hub_lat	hub_city	hub_suburb	vehicle_capacity	deli
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	

In [41]: `geo_deliveries_df.info()`

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 636149 entries, 0 to 636148
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   name                   636149 non-null object
1   region                 636149 non-null object
2   hub_lng                636149 non-null float64
3   hub_lat                636149 non-null float64
4   hub_city               636149 non-null object
5   hub_suburb             636149 non-null object
6   vehicle_capacity       636149 non-null int64
7   delivery_size          636149 non-null int64
8   delivery_lng           636149 non-null float64
9   delivery_lat           636149 non-null float64
10  delivery_city           634447 non-null object
11  delivery_suburb         476264 non-null object
12  geometry                636149 non-null geometry
dtypes: float64(4), geometry(1), int64(2), object(6)
memory usage: 63.1+ MB
```

Mapa de Entregas por Região

In [42]:

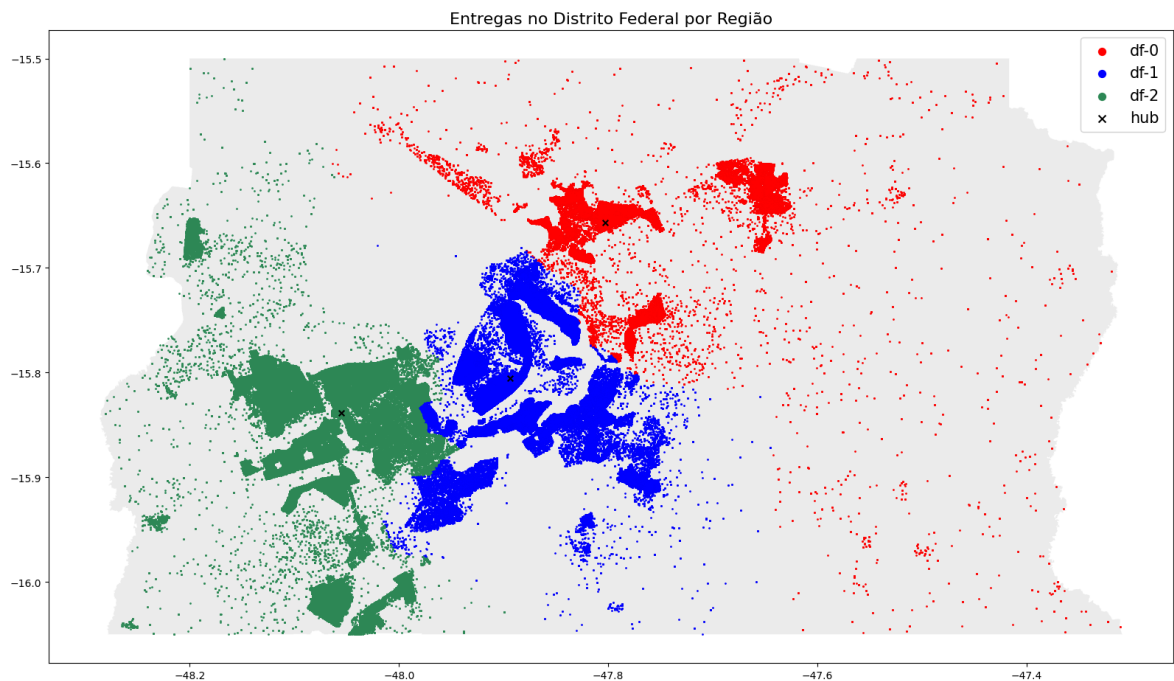
```
# cria o plot vazio
fig, ax = plt.subplots(figsize = (50/2.54, 50/2.54))

# plot do mapa do distrito federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

# plot das entregas
geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="seafoamgreen")
```

```
# plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="hub")

# plot da legenda
plt.title("Entregas no Distrito Federal por Região", fontdict={"fontsize": 16})
lgnd = plt.legend(prop={"size": 15})
for handle in lgnd.legend_handles:
    handle.set_sizes([50])
```



```
In [43]: # Criar figura e subplots em layout 2x2
fig, axes = plt.subplots(2, 2, figsize=(20, 12))

# Reduzindo a quantidade de pontos plotados (amostragem aleatória se necessário)
sample_size = 5000 # Número máximo de pontos a exibir por região

df_0 = geo_deliveries_df[geo_deliveries_df["region"] == "df-0"].sample(min(sample_size, len(df_0)))
df_1 = geo_deliveries_df[geo_deliveries_df["region"] == "df-1"].sample(min(sample_size, len(df_1)))
df_2 = geo_deliveries_df[geo_deliveries_df["region"] == "df-2"].sample(min(sample_size, len(df_2)))

# Configuração de plots
plot_params = [
    (axes[0, 0], "delivery_city", "Brasília", "yellow", "Entregas no DF (Brasília)", "df-0"),
    (axes[0, 1], "delivery_suburb", "Brasília", "yellow", "Entregas no DF (subúrbio)", "df-0"),
    (axes[1, 0], "delivery_city", "Asa Norte", "purple", "Entregas no DF (Asa Norte)", "df-1"),
    (axes[1, 1], "delivery_suburb", "Asa Norte", "purple", "Entregas no DF (subúrbio)", "df-1")
]

for ax, column, value, color, title in plot_params:
    # Plot do mapa base do DF
    mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

    # Usando scatter para melhorar performance
    ax.scatter(df_0.geometry.x, df_0.geometry.y, s=1, color="red", label="df-0",
              df_1.geometry.x, df_1.geometry.y, s=1, color="blue", label="df-1",
              df_2.geometry.x, df_2.geometry.y, s=1, color="seagreen", label="df-2")

    # Plot das entregas em destaque (se existirem)
    df_filtered = geo_deliveries_df[geo_deliveries_df[column] == value]
```

```

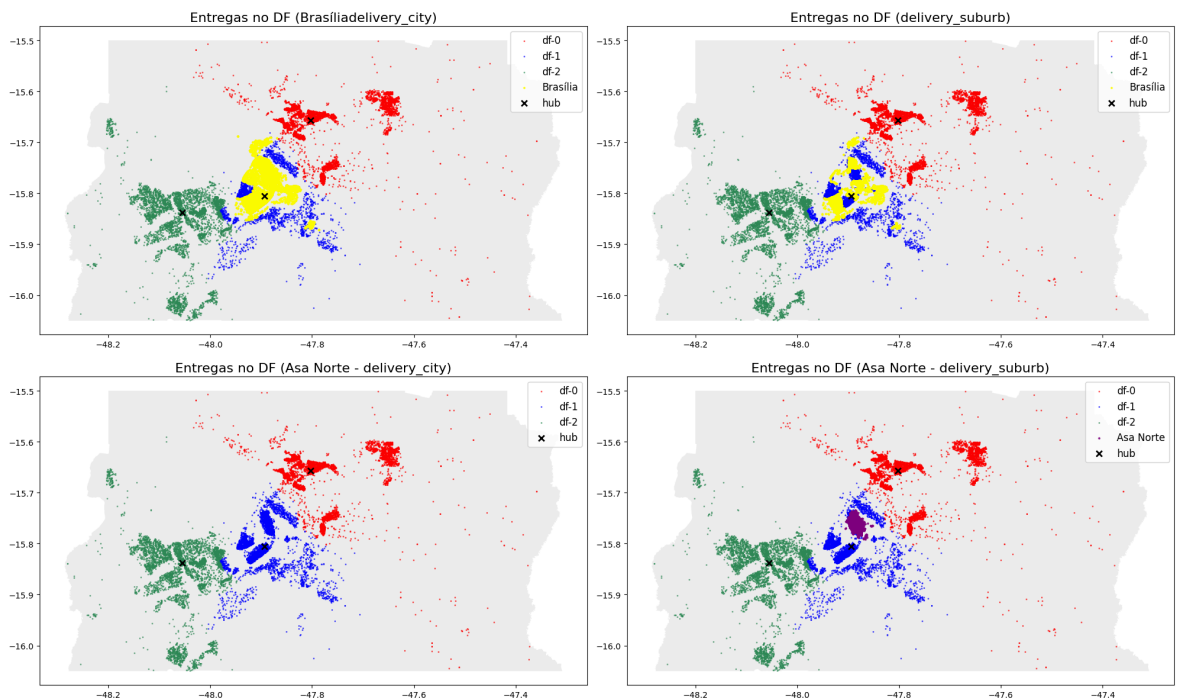
if not df_filtered.empty:
    ax.scatter(df_filtered.geometry.x, df_filtered.geometry.y, s=3, color=cc

# Plot dos hubs (destaque maior)
ax.scatter(geo_hub_df.geometry.x, geo_hub_df.geometry.y, s=50, marker="x", c

# Configuração do gráfico
ax.set_title(title, fontsize=16)
ax.legend(prop={"size": 12})

# Ajustar layout
plt.tight_layout()
plt.show()

```



Aqui antes de tratar os dados, plotei um gráfico mostrando em cada df onde estão localizados esses valores 'Brasília' e 'Asa Norte' em deliveries_suburb e deliveries_city, que são os maiores valores de número de entregas. Posteriormente irei abordar outros valores e compará-los. Note como Asa Norte não tem valores em deliveries_city então não é possível mesclar essas duas séries e ter um valor consistente e confiável para esse valor 'Asa Norte'.

Gráfico de Entregas por Região

```

In [44]: # Criando um df com colunas de outro df
data = pd.DataFrame(deliveries_df[['region', 'vehicle_capacity']].value_counts(n

# Renomeando a primeira coluna
data.rename(columns={0: "region_percent"}, inplace=True)

data.head()

```

```
Out[44]:
```

	region	vehicle_capacity	proportion
0	df-1	180	0.478988
1	df-2	180	0.410783
2	df-0	180	0.110229

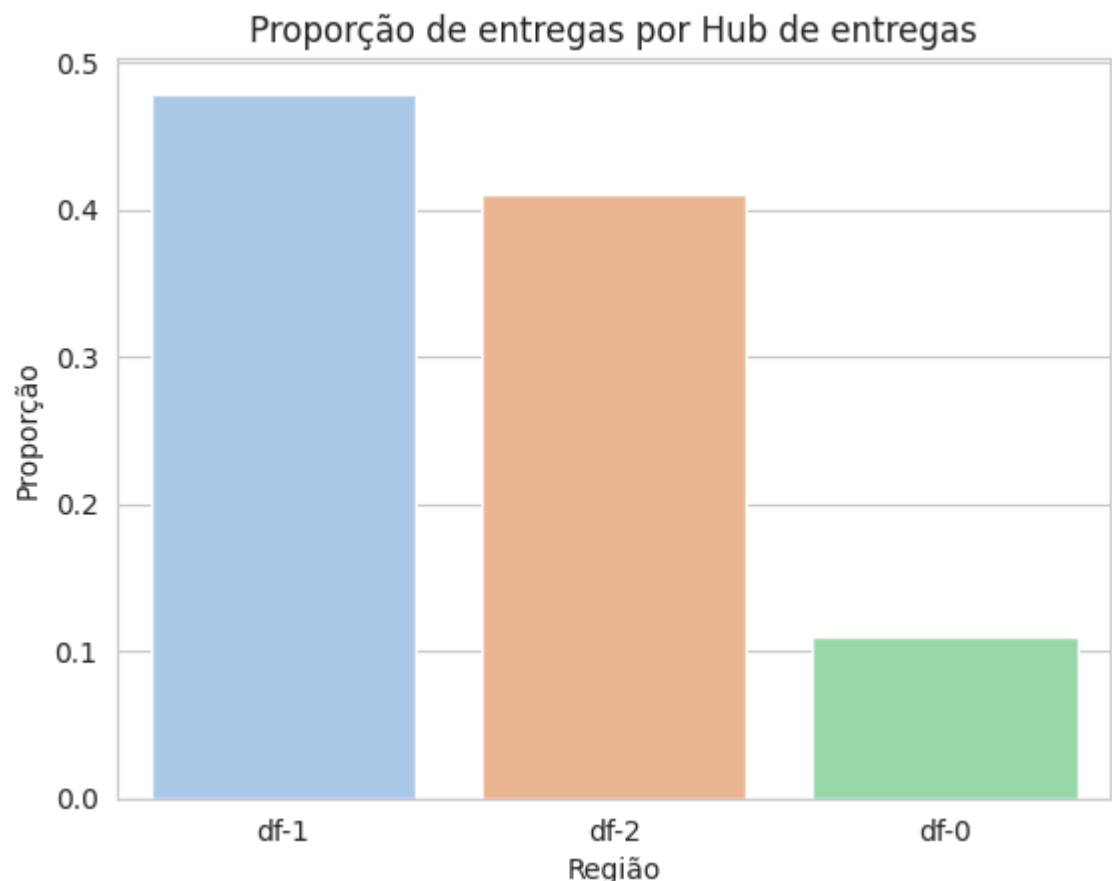
```
In [45]: # Renomeando a coluna "proportion" por "region_percent"
data = data.rename(columns={"proportion": "region_percent"})

# Gerando um gráfico de barras
with sns.axes_style('whitegrid'):
    grafico = sns.barplot(data=data, x="region", y="region_percent", errorbar=None)
    grafico.set(title='Proporção de entregas por Hub de entregas', xlabel='Região')
```

<ipython-input-45-f74256be5a74>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
grafico = sns.barplot(data=data, x="region", y="region_percent", errorbar=None,
palette="pastel")
```



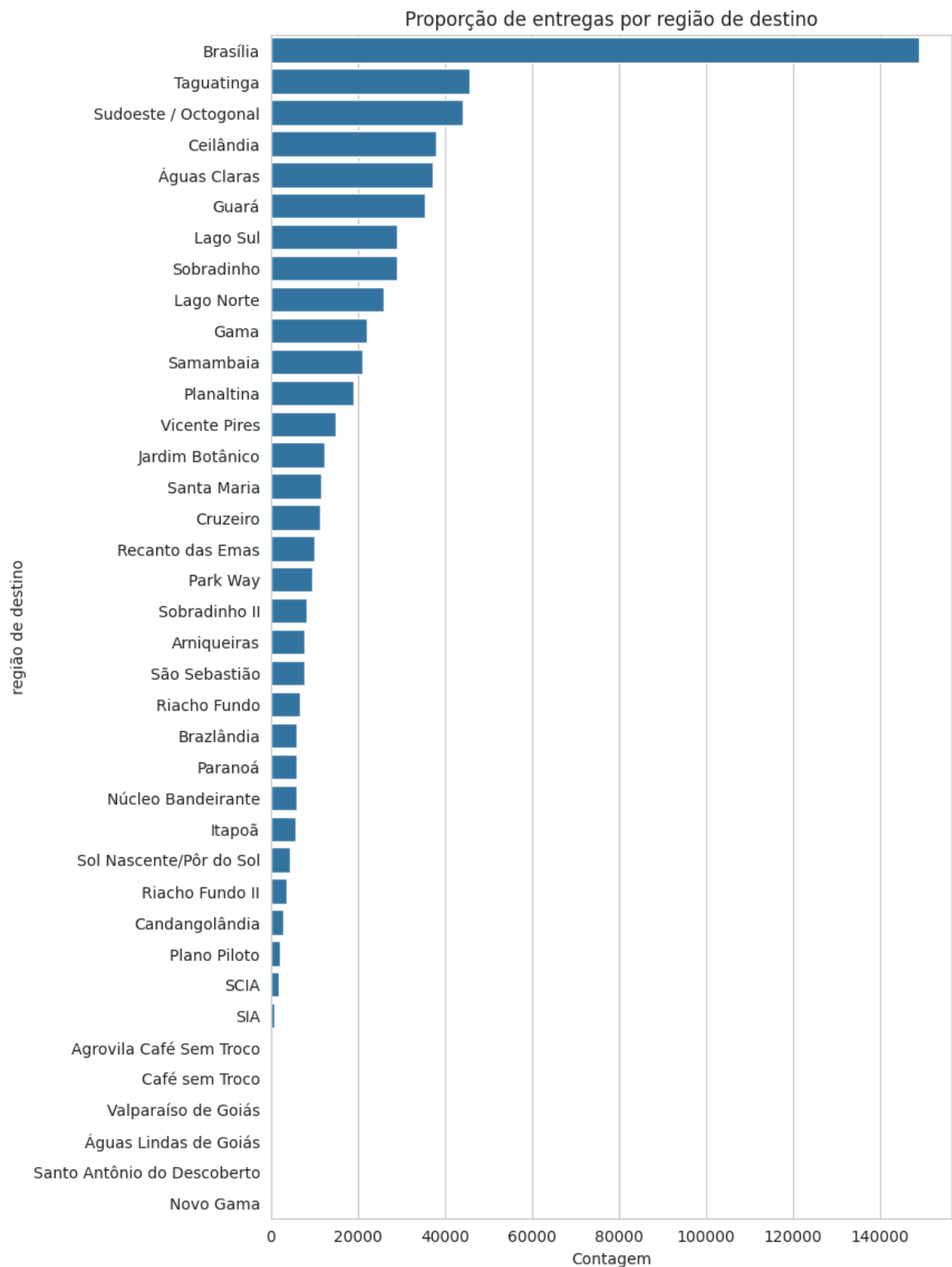
Vamos estender a análise e verificar os pontos no mapa de cada região e os locais com maior e menor volume de entregas.

```
In [46]: data = pd.DataFrame(geo_deliveries_df[['delivery_city', 'delivery_suburb']].values)
data = data.sort_values(by='proportion', ascending=False)
data.head(30)
```

Out[46]:

	delivery_city	delivery_suburb	proportion
0	Brasília	Brasília	0.112952
1	Brasília	Asa Norte	0.102368
2	Taguatinga	Taguatinga	0.084634
3	Brasília	Asa Sul	0.079158
4	Águas Claras	Águas Claras	0.075599
5	Guará	Guará	0.074291
6	Samambaia	Samambaia	0.035415
7	Sobradinho	Sobradinho	0.020967
8	Recanto das Emas	Recanto das Emas	0.020075
9	Ceilândia	P Sul	0.017106
10	Gama	Setor Sul	0.013921
11	Sobradinho II	Setor de Mansões	0.013474
12	Sobradinho	Grande Colorado	0.013136
13	Riacho Fundo	Colônia Agrícola Sucupira	0.012991
14	Gama	Setor Central	0.011227
15	Ceilândia	Ceilândia Sul	0.010927
16	Taguatinga	Setor M Norte	0.010889
17	Vicente Pires	Colônia Agrícola Samambaia	0.010089
18	Ceilândia	Ceilândia Centro	0.009360
19	Sol Nascente/Pôr do Sol	Sol Nascente/Pôr do Sol	0.009255
20	Ceilândia	Setor O	0.009121
21	Arniquireas	Vila Areal	0.009006
22	Samambaia	Setor de Mansões de Samambaia - SMSE - Setor d...	0.008848
23	Santa Maria	Residencial Santos Dummont	0.008611
24	Núcleo Bandeirante	Vila Cauhy	0.008607
25	Ceilândia	Ceilândia Norte	0.008193
26	Ceilândia	P Norte	0.007909
27	Santa Maria	Residencial Ribeirão	0.007819
28	Santa Maria	Santa Maria	0.007483
29	Arniquireas	Arniquireas	0.007420


```
In [47]: with sns.axes_style('whitegrid'):
grafico = sns.countplot(data=deliveries_df, y='delivery_city', order = deliver
grafico.set(title='Proporção de entregas por região de destino', ylabel='região
grafico.figure.set_size_inches(w=20/2.54, h=35/2.54)
```



```
In [48]: # Contando os valores únicos na coluna 'delivery_city'
city_counts = deliveries_df['delivery_city'].value_counts()

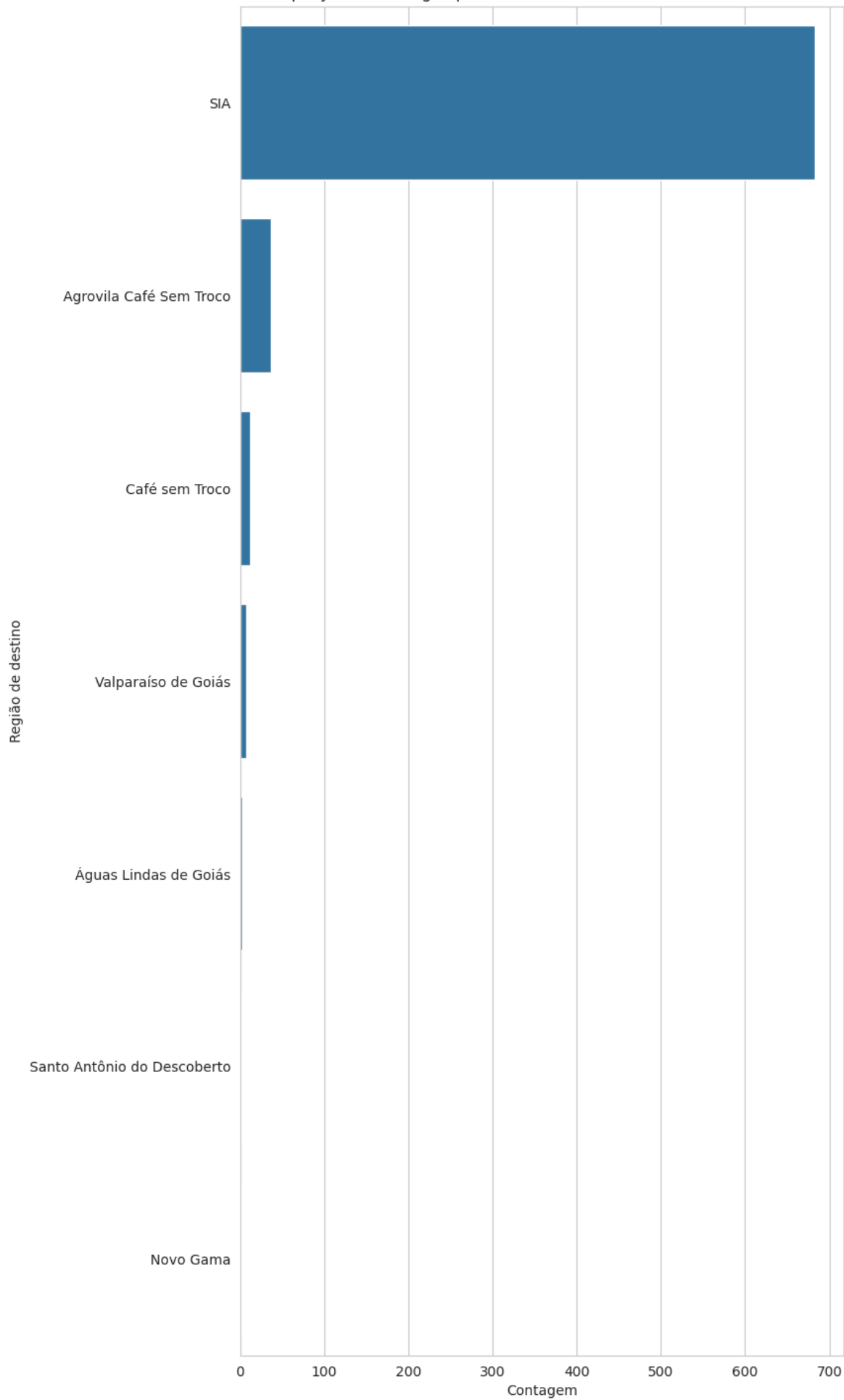
# Filtrando bairros com menos de 1000 ocorrências
city_below_1000 = city_counts[city_counts < 1000]

# Transformando os dados em um DataFrame para o Seaborn
city_below_df = city_below_1000.reset_index()
```

```
city_below_df.columns = ['Bairro', 'Ocorrências']

with sns.axes_style('whitegrid'):
    grafico = sns.barplot(
        data=city_below_df,
        y='Bairro',
        x='Ocorrências',
        order=city_below_df.sort_values('Ocorrências', ascending=False)['Bairro']
    )
    grafico.set(title='Proporção de entregas por bairros com menos de 1000 ocorrên',
                ylabel='Região de destino',
                xlabel='Contagem')
    grafico.figure.set_size_inches(w=20/2.54, h=45/2.54)
```

Proporção de entregas por bairros com menos de 1000 ocorrências



```
In [49]: deliveries_df['delivery_suburb'].head()
```

```
Out[49]:
```

	delivery_suburb
0	P Sul
1	P Sul
2	P Sul
3	P Sul
4	Sol Nascente/Pôr do Sol

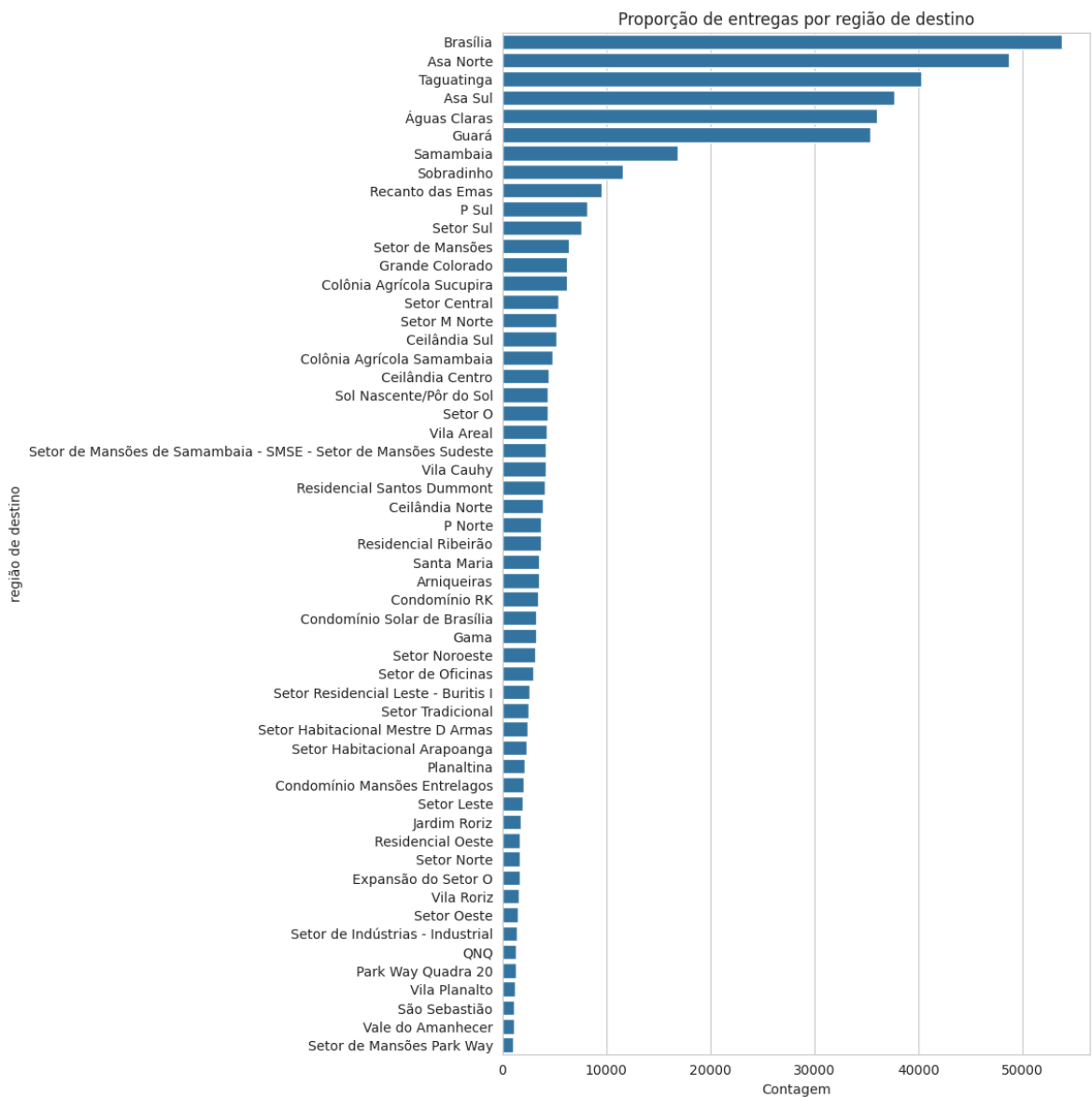
dtype: object

```
In [50]: # Contando os valores únicos na coluna 'delivery_suburb'
suburb_counts = deliveries_df['delivery_suburb'].value_counts()

# Filtrando bairros com mais de 1000 ocorrências
suburbs_above_50 = suburb_counts[suburb_counts > 1000]

# Filtrando as linhas originais onde o bairro aparece mais de 1000 vezes
filtered_deliveries = deliveries_df[deliveries_df['delivery_suburb'].isin(suburb_counts[suburb_counts > 1000].index)]

with sns.axes_style('whitegrid'):
    grafico = sns.countplot(data=filtered_deliveries, y='delivery_suburb', order=f
    grafico.set(title='Proporção de entregas por região de destino', ylabel='região
    grafico.figure.set_size_inches(w=20/2.54, h=35/2.54)
```

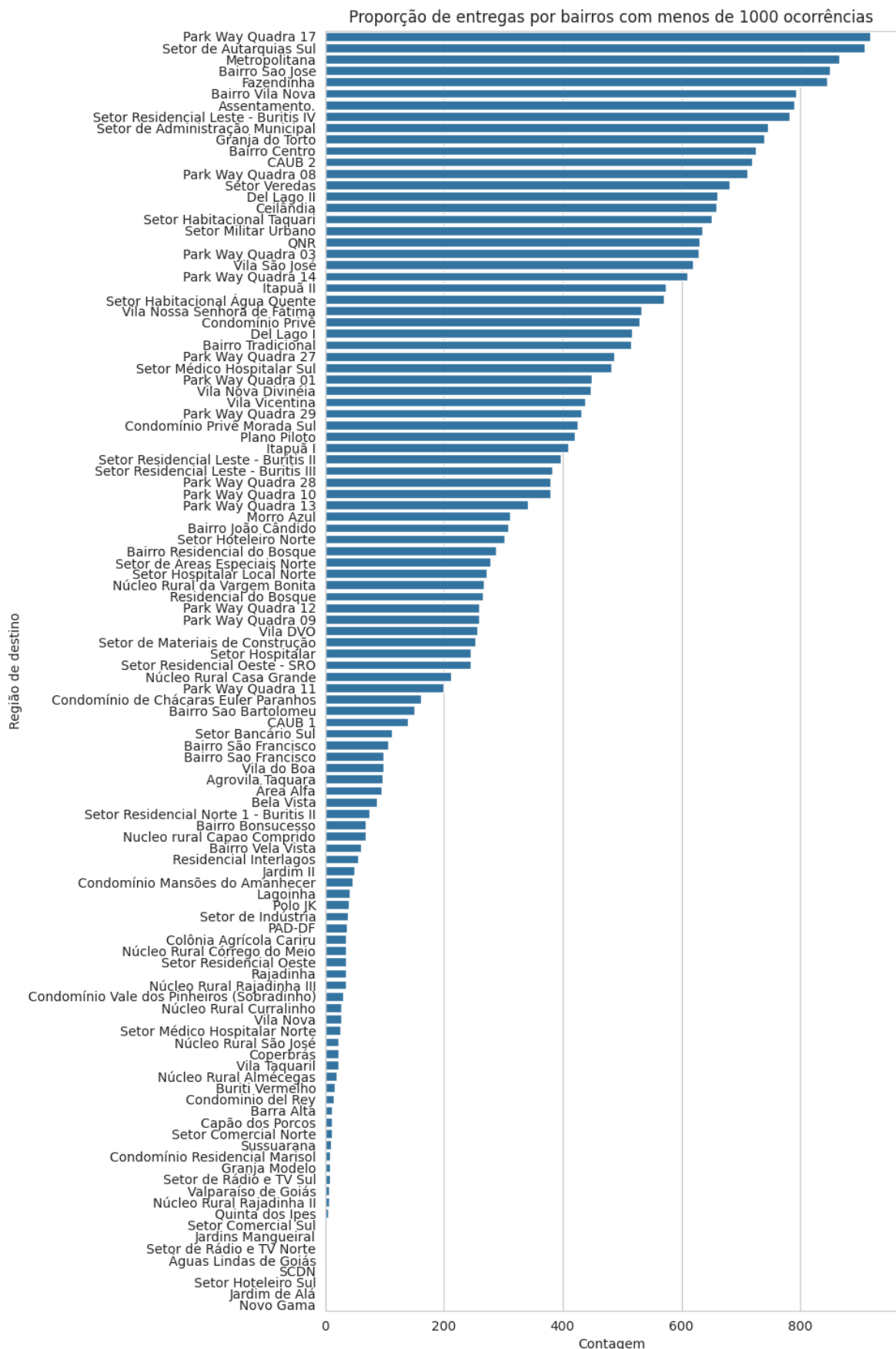


```
In [51]: # Contando os valores únicos na coluna 'delivery_suburb'
suburb_counts = deliveries_df['delivery_suburb'].value_counts()

# Filtrando bairros com menos de 1000 ocorrências
suburbs_below_1000 = suburb_counts[suburb_counts < 1000]

# Transformando os dados em um DataFrame para o Seaborn
suburbs_below_df = suburbs_below_1000.reset_index()
suburbs_below_df.columns = ['Bairro', 'Ocorrências']

with sns.axes_style('whitegrid'):
    grafico = sns.barplot(
        data=suburbs_below_df,
        y='Bairro',
        x='Ocorrências',
        order=suburbs_below_df.sort_values('Ocorrências', ascending=False)['Bairro']
    )
    grafico.set(title='Proporção de entregas por bairros com menos de 1000 ocorrên',
                ylabel='Região de destino',
                xlabel='Contagem')
    grafico.figure.set_size_inches(w=20/2.54, h=45/2.54)
```



```
In [52]: # Filtrar casos onde "Brasília" está em uma coluna e a outra coluna tem um valor
conflitos = deliveries_df[
    ((deliveries_df['delivery_suburb'] == "Brasília") & (deliveries_df['delivery_city'] == "Brasília") & (deliveries_df['delivery_s']
)]

# Exibindo os casos conflitantes
```

```
print("Casos conflitantes:")
print(conflitos[['delivery_suburb', 'delivery_city']])
```

Casos conflitantes:

	delivery_suburb	delivery_city
11874	Asa Norte	Brasília
11875	Asa Norte	Brasília
11876	Asa Norte	Brasília
11877	Asa Norte	Brasília
11878	Asa Norte	Brasília
...
627684	Asa Norte	Brasília
627685	Asa Norte	Brasília
627686	Asa Norte	Brasília
627687	Asa Norte	Brasília
627688	Asa Norte	Brasília

[95188 rows x 2 columns]

```
In [53]: print("Ocorrências restantes de 'Brasília' em delivery_suburb:")
print((deliveries_df['delivery_suburb'] == "Brasília").sum())

print("Ocorrências restantes de 'Brasília' em delivery_city:")
print((deliveries_df['delivery_city'] == "Brasília").sum())

deliveries_df.loc[
    (deliveries_df['delivery_suburb'] == "Brasília") & (deliveries_df['delivery_
    ['delivery_suburb', 'delivery_city']
] = None # Substituir por nulo ou outro valor padrão

# Corrigindo delivery_suburb com prioridade para delivery_city
deliveries_df.loc[
    (deliveries_df['delivery_suburb'] == "Brasília") &
    (deliveries_df['delivery_city'].notna()) &
    (deliveries_df['delivery_city'] != "Brasília"),
    'delivery_suburb'
] = deliveries_df['delivery_city']

# Corrigindo delivery_city com prioridade para delivery_suburb
deliveries_df.loc[
    (deliveries_df['delivery_city'] == "Brasília") &
    (deliveries_df['delivery_suburb'].notna()) &
    (deliveries_df['delivery_suburb'] != "Brasília"),
    'delivery_city'
] = deliveries_df['delivery_suburb']

print("Ocorrências restantes de 'Brasília' em delivery_suburb:")
print((deliveries_df['delivery_suburb'] == "Brasília").sum())

print("Ocorrências restantes de 'Brasília' em delivery_city:")
print((deliveries_df['delivery_city'] == "Brasília").sum())
```

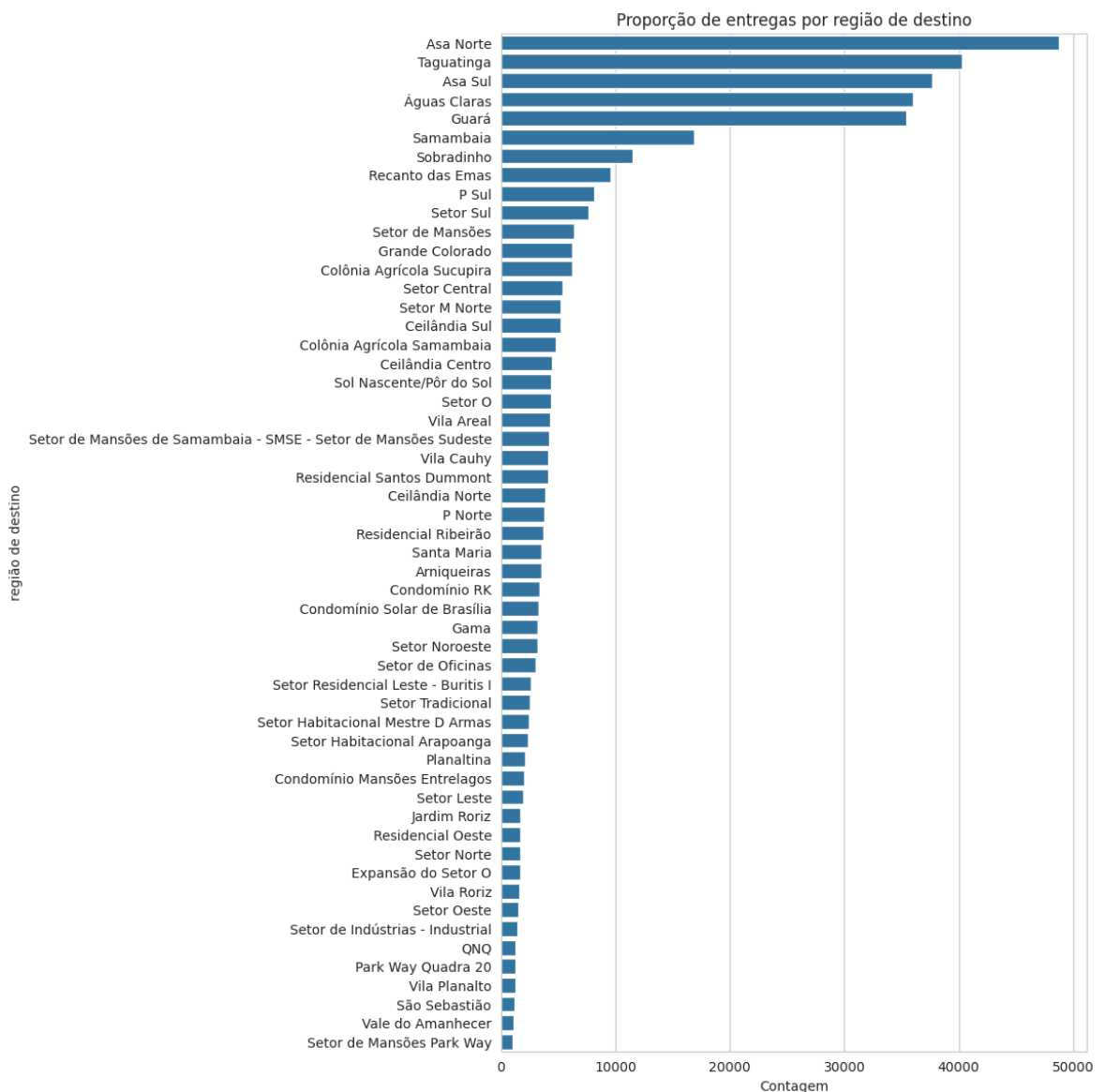
```
Ocorrências restantes de 'Brasília' em delivery_suburb:
53795
Ocorrências restantes de 'Brasília' em delivery_city:
148983
Ocorrências restantes de 'Brasília' em delivery_suburb:
0
Ocorrências restantes de 'Brasília' em delivery_city:
0
```

```
In [54]: # Contando os valores únicos na coluna 'delivery_suburb'
suburb_counts = deliveries_df['delivery_suburb'].value_counts()

# Filtrando bairros com mais de 1000 ocorrências
suburbs_above_50 = suburb_counts[suburb_counts > 1000]

# Filtrando as linhas originais onde o bairro aparece mais de 1000 vezes
filtered_deliveries = deliveries_df[deliveries_df['delivery_suburb'].isin(suburb_counts[suburb_counts > 1000].index)]

with sns.axes_style('whitegrid'):
    grafico = sns.countplot(data=filtered_deliveries, y='delivery_suburb', order=sorted(suburb_counts[suburb_counts > 1000].index, reverse=True))
    grafico.set(title='Proporção de entregas por região de destino', ylabel='região de destino')
    grafico.figure.set_size_inches(w=20/2.54, h=35/2.54)
```




```
In [55]: geo_deliveries_df.head()
```

```
Out[55]:
```

	name	region	hub_lng	hub_lat	hub_city	hub_suburb	vehicle_capacity	deli
0	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
1	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
2	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
3	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	
4	cvrp-2-df-33	df-2	-48.054989	-15.838145	Taguatinga	Taguatinga Centro	180	

```
In [56]: # cria o plot vazio
fig, ax = plt.subplots(figsize = (50/2.54, 50/2.54))

# plot do mapa do distrito federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

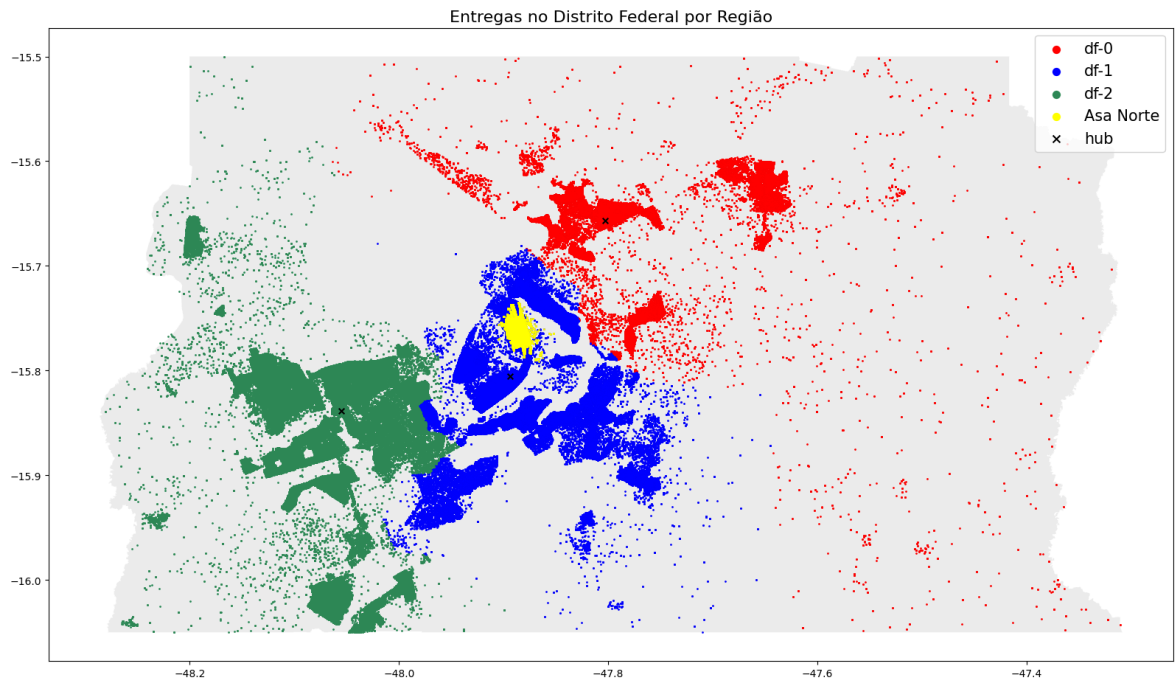
geo_deliveries_df.query("region == 'df-0'").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1'").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2'").plot(ax=ax, markersize=1, color="seafoamgreen")

# plot das entregas
# Verificar se há "Asa Norte" antes de iterar
# Verificar se "Asa Norte" existe em delivery_suburb
if 'Asa Norte' in geo_deliveries_df['delivery_suburb'].values:
    # Iterar pelas regiões
    for region, group in geo_deliveries_df.groupby('region'):
        # Só plotar grupos que têm "Asa Norte"
        if group['delivery_suburb'].str.contains('Asa Norte').any():
            group.query("delivery_suburb == 'Asa Norte'").plot(
                ax=ax,
                markersize=1,
                color="yellow",
                label='Asa Norte'
            )

# plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="hub")

# plot da legenda
plt.title("Entregas no Distrito Federal por Região", fontdict={"fontsize": 16})
lgnd = plt.legend(loc="upper right", prop={"size": 15})
```

```
for handle in lgnd.legend_handles:
    handle.set_sizes([50])
```



```
In [57]: # cria o plot vazio
fig, ax = plt.subplots(figsize = (50/2.54, 50/2.54))

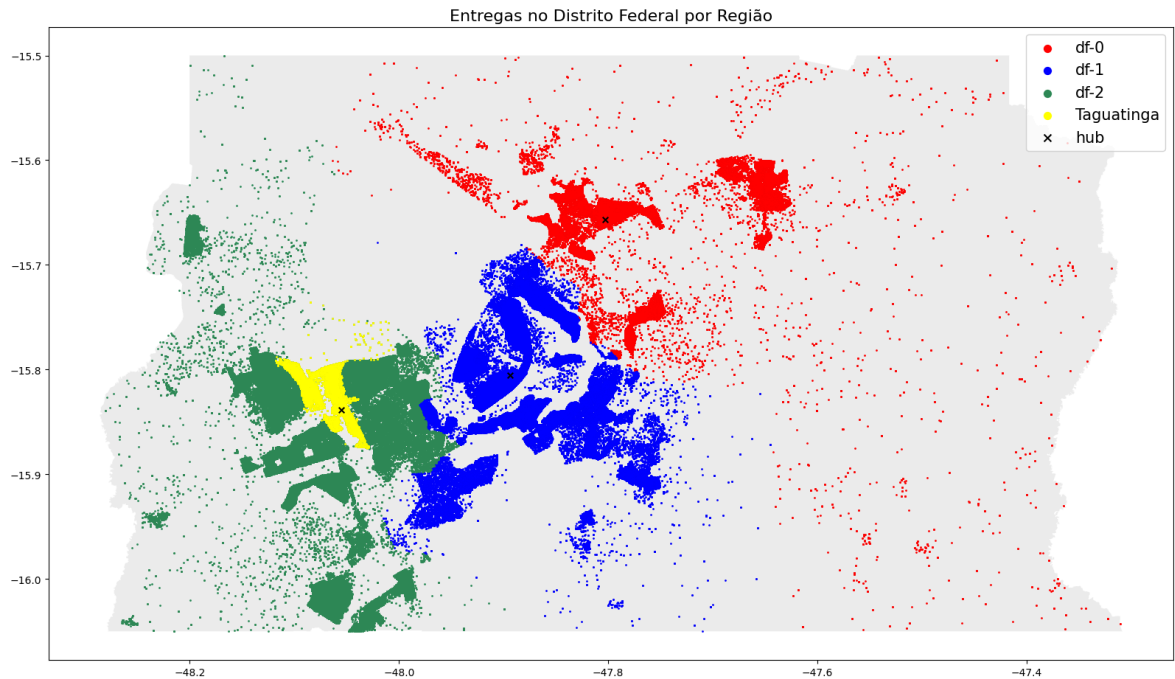
# plot do mapa do distrito federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="green")

# plot das entregas
# Verificar se há "Asa Norte" antes de iterar
# Verificar se "Asa Norte" existe em delivery_suburb
if 'Taguatinga' in geo_deliveries_df['delivery_city'].values:
    # Iterar pelas regiões
    for region, group in geo_deliveries_df.groupby('region'):
        # Só plotar grupos que têm "Asa Norte"
        if group['delivery_city'].str.contains('Taguatinga').any():
            group.query("delivery_city == 'Taguatinga']").plot(
                ax=ax,
                markersize=1,
                color="yellow",
                label='Taguatinga'
            )

# plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="hub")

# plot da legenda
plt.title("Entregas no Distrito Federal por Região", fontdict={"fontsize": 16})
lgnd = plt.legend(loc="upper right", prop={"size": 15})
for handle in lgnd.legend_handles:
    handle.set_sizes([50])
```



```
In [58]: # cria o plot vazio
fig, ax = plt.subplots(figsize = (50/2.54, 50/2.54))

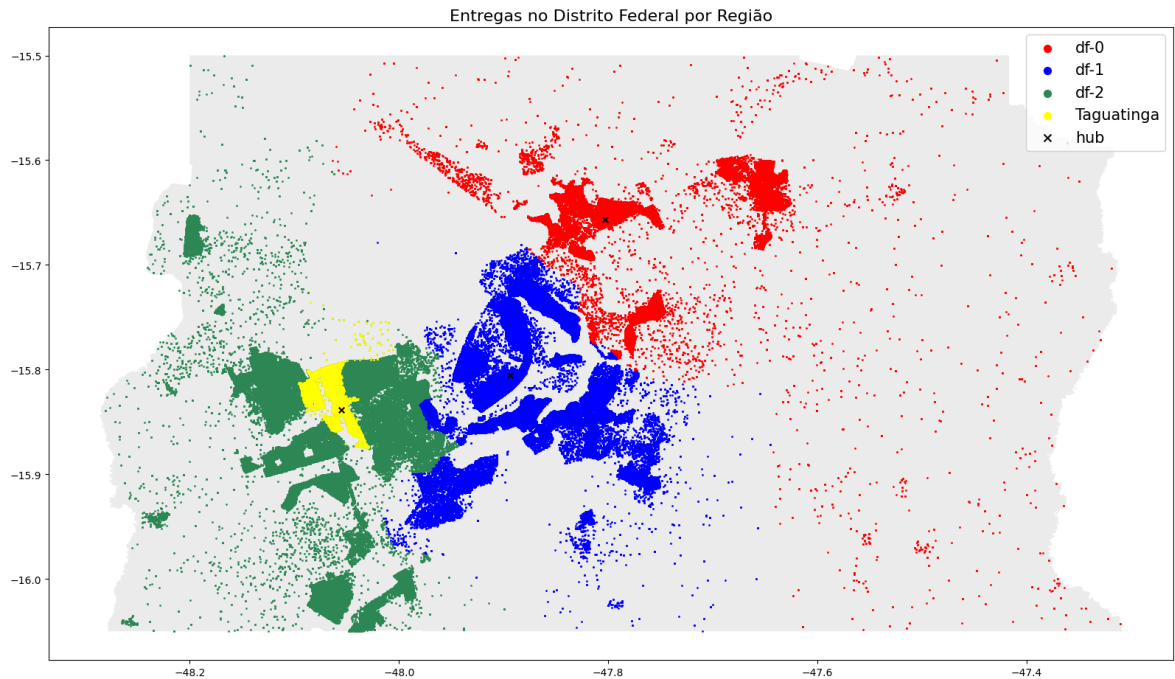
# plot do mapa do distrito federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="sea")

# plot das entregas
# Verificar se "Asa Norte" antes de iterar
# Verificar se "Asa Norte" existe em delivery_suburb
if 'Taguatinga' in geo_deliveries_df['delivery_suburb'].values:
    # Iterar pelas regiões
    for region, group in geo_deliveries_df.groupby('region'):
        # Só plotar grupos que têm "Asa Norte"
        if group['delivery_suburb'].str.contains('Taguatinga').any():
            group.query("delivery_suburb == 'Taguatinga']").plot(
                ax=ax,
                markersize=1,
                color="yellow",
                label='Taguatinga'
            )

# plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="hub")

# plot da legenda
plt.title("Entregas no Distrito Federal por Região", fontdict={"fontsize": 16})
lgnd = plt.legend(loc="upper right", prop={"size": 15})
for handle in lgnd.legend_handles:
    handle.set_sizes([50])
```



```
In [59]: # Criar a figura com duas colunas para os plots lado a lado
fig, axes = plt.subplots(1, 2, figsize=(100/2.54, 50/2.54)) # Converte tamanho

# Primeiro gráfico (usando delivery_city)
ax = axes[0]

# Plot do mapa do Distrito Federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="seafoamgreen")

# Plot das entregas (delivery_city)
if 'Taguatinga' in geo_deliveries_df['delivery_city'].values:
    for region, group in geo_deliveries_df.groupby('region'):
        if group['delivery_city'].str.contains('Taguatinga').any():
            group.query("delivery_city == 'Taguatinga']").plot(
                ax=ax,
                markersize=1,
                color="yellow",
                label='Taguatinga'
            )

# Plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="hub")

# Configurações do primeiro gráfico
ax.set_title("Entregas no DF por Região (delivery_city)", fontsize=16)
lgnd = ax.legend(loc="upper right", prop={"size": 15})
for handle in lgnd.legend_handles:
    handle.set_sizes([50])

# Segundo gráfico (usando delivery_suburb)
ax = axes[1]

# Plot do mapa do Distrito Federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")
```

```

geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="seafoam")

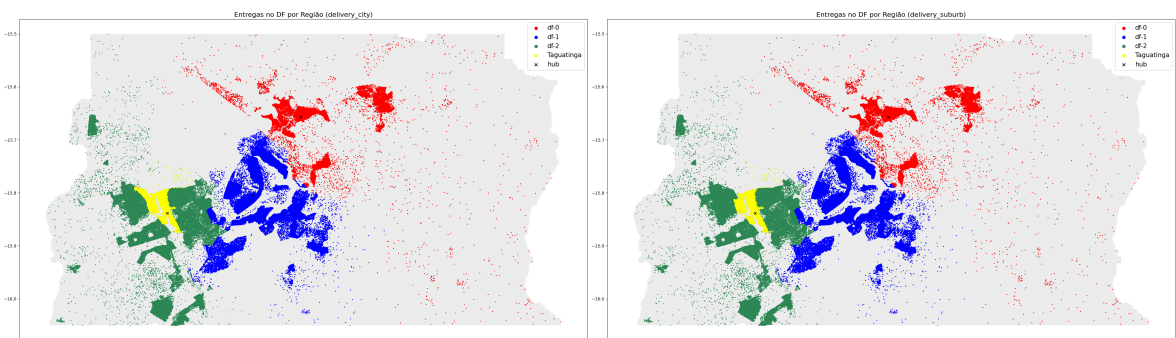
# Plot das entregas (delivery_suburb)
if 'Taguatinga' in geo_deliveries_df['delivery_suburb'].values:
    for region, group in geo_deliveries_df.groupby('region'):
        if group['delivery_suburb'].str.contains('Taguatinga').any():
            group.query("delivery_suburb == 'Taguatinga']").plot(
                ax=ax,
                markersize=1,
                color="yellow",
                label='Taguatinga'
            )

# Plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="hub")

# Configurações do segundo gráfico
ax.set_title("Entregas no DF por Região (delivery_suburb)", fontsize=16)
lgnd = ax.legend(loc="upper right", prop={"size": 15})
for handle in lgnd.legend_handles:
    handle.set_sizes([50])

# Ajustar layout para evitar sobreposição
plt.tight_layout()
plt.show()

```



```

In [60]: # Contando os valores únicos na coluna 'delivery_suburb'
suburb_counts = geo_deliveries_df['delivery_suburb'].value_counts()

# Filtrando bairros com menos de 1000 ocorrências
suburbs_below_1000 = suburb_counts[suburb_counts < 1000].index

# Filtrando as entregas nesses bairros
geo_deliveries_below_1000 = geo_deliveries_df[geo_deliveries_df['delivery_suburb']

# Criando um plot vazio
fig, ax = plt.subplots(figsize=(50/2.54, 50/2.54))

# Plot do mapa do Distrito Federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

# Plot das entregas por região
geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="seafoam")

# Destacando bairros com menos de 1000 entregas (em amarelo)

```

```

geo_deliveries_below_1000.plot(ax=ax, markersize=2, color="yellow", label="Bairr

# Plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="Hub")

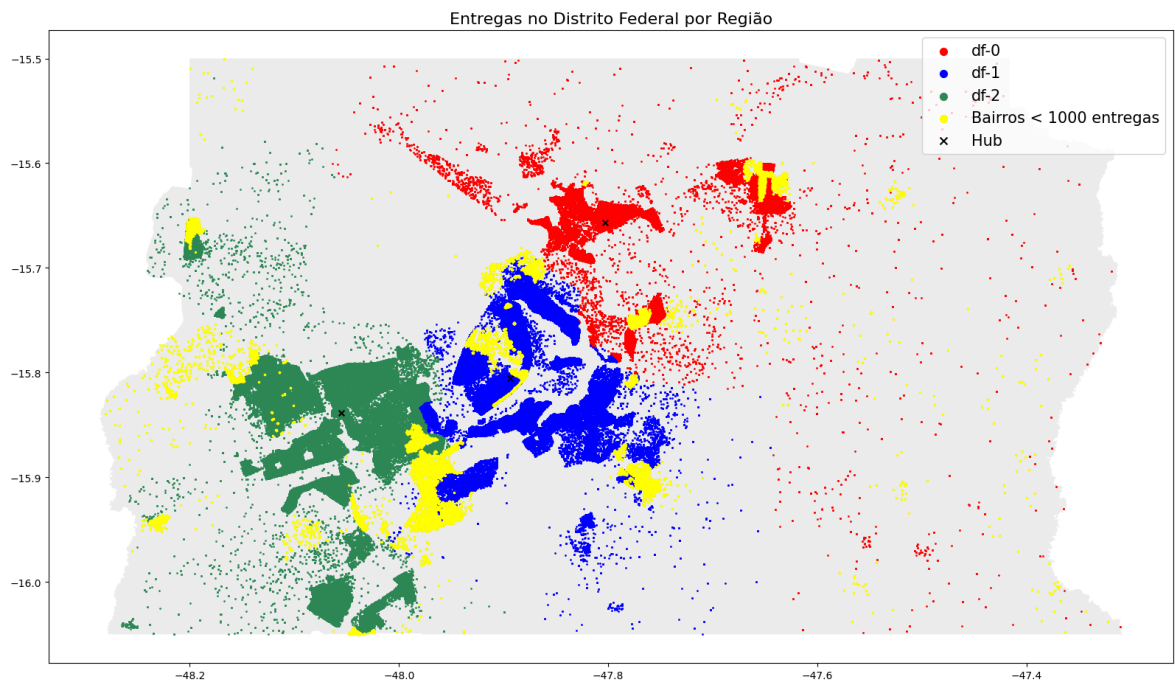
# Definir a posição da legenda para evitar Lentidão
plt.legend(loc="upper right", prop={"size": 15})

# Ajustar tamanho dos marcadores na Legenda
for handle in plt.gca().get_legend().legend_handles:
    handle.set_sizes([50])

# Título
plt.title("Entregas no Distrito Federal por Região", fontdict={"fontsize": 16})

plt.show()

```



```

In [61]: # Contando os valores únicos na coluna 'delivery_suburb'
suburb_counts = geo_deliveries_df['delivery_suburb'].value_counts()

# Filtrando bairros com 1000 ou mais ocorrências
suburbs_above_1000 = suburb_counts[suburb_counts >= 1000].index

# Filtrando as entregas nesses bairros
geo_deliveries_above_1000 = geo_deliveries_df[geo_deliveries_df['delivery_suburb

# Criando um plot vazio
fig, ax = plt.subplots(figsize=(50/2.54, 50/2.54))

# Plot do mapa do Distrito Federal
mapa.plot(ax=ax, alpha=0.4, color="lightgrey")

# Plot das entregas por região
geo_deliveries_df.query("region == 'df-0']").plot(ax=ax, markersize=1, color="red")
geo_deliveries_df.query("region == 'df-1']").plot(ax=ax, markersize=1, color="blue")
geo_deliveries_df.query("region == 'df-2']").plot(ax=ax, markersize=1, color="sea

# Destacando bairros com mais de 1000 entregas (em roxo)
geo_deliveries_above_1000.plot(ax=ax, markersize=2, color="yellow", label="Bairr

```

```

# Plot dos hubs
geo_hub_df.plot(ax=ax, markersize=30, marker="x", color="black", label="Hub")

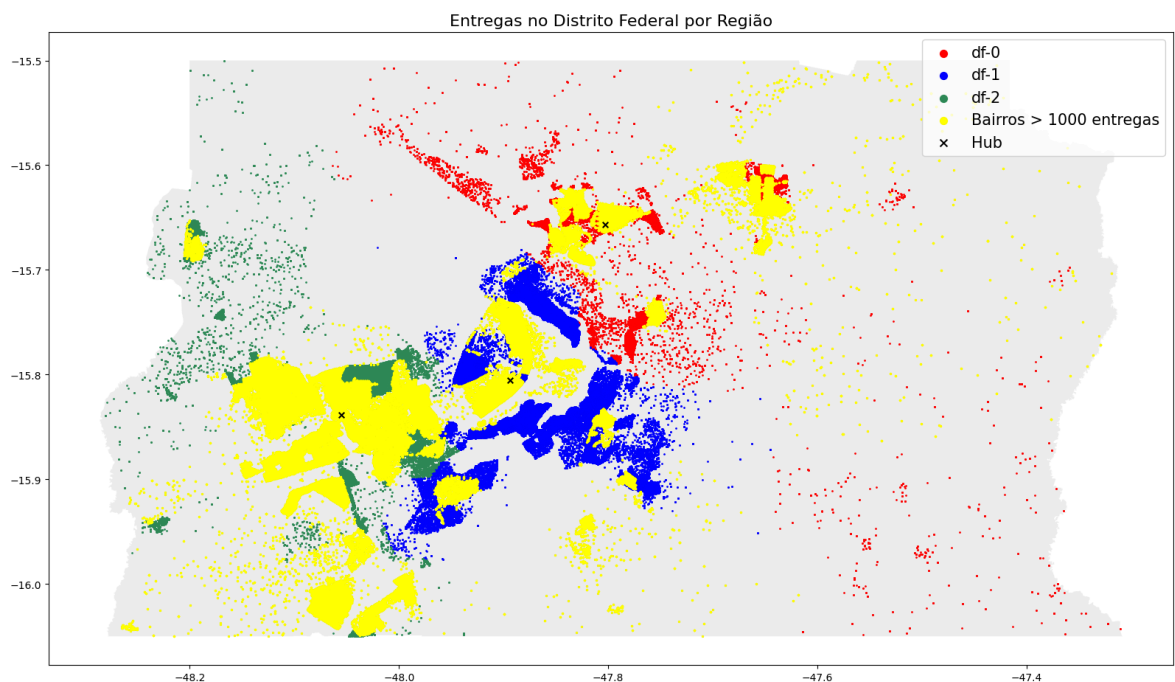
# Definir a posição da legenda para evitar lentidão
plt.legend(loc="upper right", prop={"size": 15})

# Ajustar tamanho dos marcadores na legenda
for handle in plt.gca().get_legend().legend_handles:
    handle.set_sizes([50])

# Título
plt.title("Entregas no Distrito Federal por Região", fontdict={"fontsize": 16})

plt.show()

```



Análise do Volume de Entregas e Inconsistências nos Dados

Ao longo deste projeto, analisamos os volumes de entrega por região no Distrito Federal. Durante o tratamento dos dados e o cruzamento das colunas `delivery_city` e `delivery_suburb`, utilizando uma ferramenta de enriquecimento geoespacial, identificamos algumas inconsistências.

O processo de geocodificação gerou uma dispersão dos valores entre essas colunas, o que resultou em divergências ao analisá-las conjuntamente. Um exemplo claro dessa inconsistência foi a presença do valor "**Brasília**" como bairro, apesar de não ser um bairro oficial. Esse valor representava entregas em várias regiões, como **Asa Norte, Asa Sul e Vila Planalto**, comprometendo a precisão da análise.

Ao remover "**Brasília**" das colunas e refiltrar os dados, notamos que bairros com menor volume de entregas apresentavam pontos de entrega próximos a onde "**Brasília**" estava originalmente categorizada. Isso sugere que o volume real desses

bairros pode estar **subestimado**, o que pode impactar a priorização logística dessas regiões.

Outro caso identificado foi o bairro **Taguatinga**, que apresentou pequenas diferenças de ocorrência entre `delivery_city` e `delivery_suburb`. Além disso, algumas entregas associadas a **Ceilândia Norte** estavam misturadas na coluna `delivery_city`, enquanto o desenho do bairro no mapa se manteve consistente.

Oportunidades de Otimização Logística

A análise geral das regiões de entrega revelou insights sobre a eficiência da distribuição e possíveis melhorias. Observamos que o hub **df-0** cobre uma área extensa e possui um volume de entregas disperso, mas significativo. A localização desse hub é estratégica, porém, ao aplicar filtros para visualizar os maiores e menores volumes de entrega, percebemos que certas regiões atendidas por ele estão bastante espalhadas, o que pode estar gerando trajetos logísticos menos eficientes.

Sugestões para otimização da distribuição:

1. Criação de um novo hub logístico

- Um **quarto hub**, localizado mais ao sul, poderia redistribuir a carga dos hubs **df-0** e **df-1**, reduzindo distâncias percorridas e melhorando o tempo de entrega.

2. Sub-hubs ou pontos de distribuição menores

- Pequenos pontos de apoio podem ser estabelecidos para armazenar volumes menores e facilitar a distribuição final.
- Essa estratégia pode ser aplicada especialmente em regiões com alta dispersão, onde a concentração de entregas não é suficiente para justificar um grande hub.

3. Balanceamento da carga entre regiões

- Avaliar se há **subutilização ou sobrecarga** em certos bairros pode ajudar a equilibrar melhor o número de veículos e otimizar os percursos.
 - Caso alguns veículos estejam percorrendo distâncias maiores do que o necessário, a redistribuição pode ajudar a diminuir custos e tempo de entrega.
-

Análises Futuras e Expansão do Projeto

Embora a análise tenha identificado padrões interessantes, algumas informações adicionais poderiam trazer um nível maior de precisão:

- Mapeamento de clusters de entrega:** Identificar se existem regiões com alta densidade de entregas que poderiam ser beneficiadas por uma logística mais eficiente.

- **Simulação de diferentes cenários:** Testar variações na quantidade de hubs e veículos para comparar impactos em custo e tempo de entrega.
- **Análise de horários de entrega:** Caso existam dados temporais, seria possível verificar se há horários de pico que impactam a eficiência logística.

Com base nesses insights, este estudo pode servir como base para uma reavaliação da distribuição logística, promovendo **eficiência operacional** e **melhor aproveitamento da estrutura existente**.