

TECNOLOGIA EM SISTEMAS PARA INTERNET

**André Luiz Gomes dos Santos
Brenda Lopes Miranda Teixeira
Luiz Henrique de Paiva Ventura**

**RELATÓRIO DE PRÁTICA INTEGRADA
DE
CIÊNCIA DE DADOS E INTELIGÊNCIA ARTIFICIAL**

**Brasília - DF
22/03/2020**

Sumário

1. Objetivos	3
2. Descrição do problema	4
3. Desenvolvimento	5
3.1 Código implementado	5
4. Considerações Finais	6
Referências	7

1. Objetivos

Nesta etapa do projeto será apresentado :

- Quatro estados que possuem maior frequência de relatos
- tipos de OVNI's mais populares
- Gráfico de barras agrupadas
- Gráfico de barras empilhadas
- mapas
- Criar um mapa do país inteiro (EUA) e plotar no mapa as ocorrências para todas as cidades. De forma a construir uma representação visual da quantidade de ocorrências por cidade e estado.
- Criar um mapa do estado da Califórnia.

2. Descrição do problema

De acordo com o relatório anterior que foi apresentado uma lista de avistamento de OVNI's por 20 anos, nesta etapa será sintetizada todas essas listas, categorizando as por objetos voadores não identificados mais populares, será mostrada em gráficos de barras ampliadas e agrupadas, também será mostrado mapas que a visualização seja mais clara.

As maiores dificuldades encontradas foram para agrupar e contar a quantidade dos gráficos e criar mapas.

3. Desenvolvimento

Este trabalho está sendo desenvolvido usando um Script Python por ser uma linguagem orientada a objetos é bastante maleável, o grupo G2 está utilizando a plataforma Google Colaboratory, assim todos podem modificar e acrescentar o código quando necessário. Todos os códigos estão sendo disponibilizados no github.

3.1 Código implementado

Abaixo segue as imagens do código implementado.

Trecho de código das importações

Imagem 1 - Importações


```
[4] 1 #importações
    2 import pandas as pd
    3 import seaborn as sns
    4 #!pip install -U pandasql
    5 import pandasql
    6 import numpy as np
    7 import matplotlib.pyplot as plt
    8 #!pip install folium
    9 import folium
   10 from folium.plugins import HeatMap
   11
```

Fonte: Própria

Este código se refere aos quatro estados com maior frequência de relatos.

Imagem 2 - Relatos de Estados

```
1 #!pip install -U pandasql
2
3 baseOvnis = pd.read_csv("OVNIS.csv")
4
5 #Filtrar os estados
6 estados = baseOvnis['State'].value_counts().head(4)
7
8 #Criar o dataframe
9 df_relatos = pd.DataFrame(estados)
10
11 #Criar coluna com os nome dos estado da index
12 df_relatos['stateName'] = df_relatos.index
13
14 #redefinir index
15 df_relatos.reset_index(drop=True, inplace=True)
16
17 df_relatos
18
```




	State	stateName
0	7917	CA
1	4359	FL
2	3230	WA
3	2883	TX

Fonte: Própria

Abaixo é realizado um filtro mostrando os tipos de OVNIS mais recorrentes.


Imagem 3 - OVNIS Populares



```

1 #Filtrar os Ovnis mais populares
2 tipos = baseOvnis['Shape'].value_counts().head(4)
3
4 #Criar o dataframe
5 df_ovnis = pd.DataFrame(tipos)
6
7 #Criar coluna com os nome dos Ovnis da index
8 df_ovnis['shapesName'] = df_ovnis.index
9
10 #trocar nome
11 df_ovnis.columns = ['Quantidade', 'shapesName']
12
13 #redefinir index
14 df_ovnis.reset_index(drop=True, inplace=True)
15
16 df_ovnis

```



	Quantidade	shapesName
0	15407	Light
1	8303	Circle
2	6462	Triangle
3	6442	Fireball

Fonte: Própria

Aqui fazemos um filtro de aparições por estado usando comandos SQL.

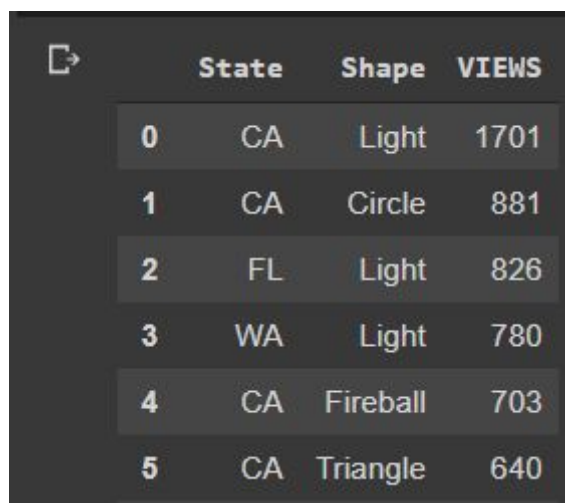
Imagem 4 - Filtro

```
#Aparições por estado
import pandas
#!pip install -U pandasql
import pandasql

baseOvnis = pd.read_csv("OVNIS.csv")

q= """
SELECT State, Shape, COUNT(*) AS VIEWS
FROM baseOvnis
WHERE State IN('CA','FL','WA','TX') AND Shape IN('Light','Circle','Triangle','Fireball')
GROUP BY State, Shape
ORDER BY VIEWS DESC
"""

T1 = pandasql.sqldf(q, locals())
# pd.DataFrame(T1)
# T1
```

A screenshot of a Jupyter Notebook interface. On the left, there is a small icon of a document with an arrow. To the right, a table is displayed with four columns: 'State', 'Shape', and 'VIEWS'. The table contains six rows of data, indexed from 0 to 5. The data is as follows:

	State	Shape	VIEWS
0	CA	Light	1701
1	CA	Circle	881
2	FL	Light	826
3	WA	Light	780
4	CA	Fireball	703
5	CA	Triangle	640

Fonte: Própria

A seguir é mostrada a sequência de códigos para o gráfico 1.

Imagem 5

```
import numpy as np
import matplotlib.pyplot as plt

Light = [1701, 826, 780, 579]
Circle = [881, 551, 330, 290]
Fireball = [703, 541, 296, 183]
Triangle = [640, 344, 301, 245]

# Definindo a largura das barras
barWidth = 0.15

# Aumentando o gráfico
plt.figure(figsize=(7,5))
```

Fonte: Própria

Imagem 6 - Gráfico 1

```
# Definindo a posição das barras
r1 = np.arange(len(Light))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
r4 = [x + barWidth for x in r3]

# Criando as barras
plt.bar(r1, Light, color='green', width=barWidth, label='Light')
plt.bar(r2, Circle, color='yellow', width=barWidth, label='Circle')
plt.bar(r3, Sphere, color='blue', width=barWidth, label='Fireball')
plt.bar(r4, Fireball, color='red', width=barWidth, label='Triangle')
```

Fonte: Própria

Imagem 7 - Gráfico 1

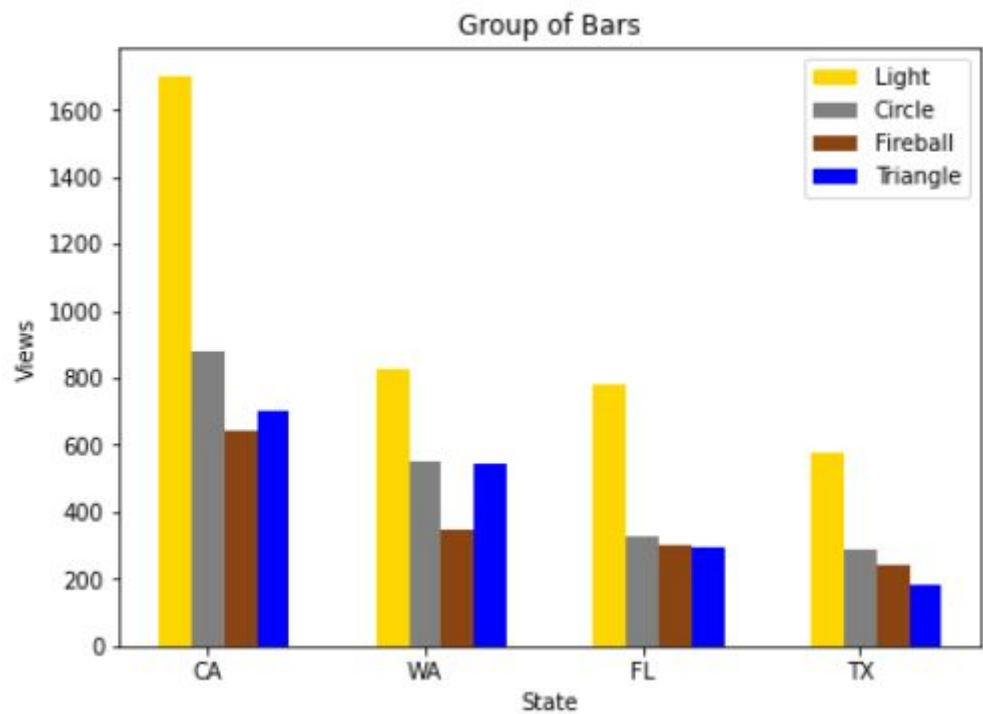
```
# Adiciando legendas as barras
plt.xlabel('State')
plt.xticks([r + barWidth for r in range(len(Light))], ['CA', 'WA', 'FL', 'TX'])
plt.ylabel('Views')
plt.title('Group of Bars')

# Criando a legenda e exibindo o gráfico
plt.legend()
plt.show()
```

Fonte: Própria

Ao rodar o comando o gráfico fica assim:

Gráfico 1: Visualizações por estado



Fonte: Própria

A seguir é mostrada a sequência de códigos para o gráfico 1.

Imagem 8 - Gráfico 2

```
Light = np.array((1701, 826, 780, 579))
Circle = np.array((881, 551, 330, 290))
Fireball = np.array((703, 541, 296, 183))
Triangle = np.array((640, 344, 301, 245))

#shape = ['Light','Circle','Fireball', 'Triangle']

states = ['CA','WA','FL', 'TX']

# Aumentando o gráfico
plt.figure(figsize=(7,5))
```

Fonte: Própria

Imagem 9 - Gráfico 2

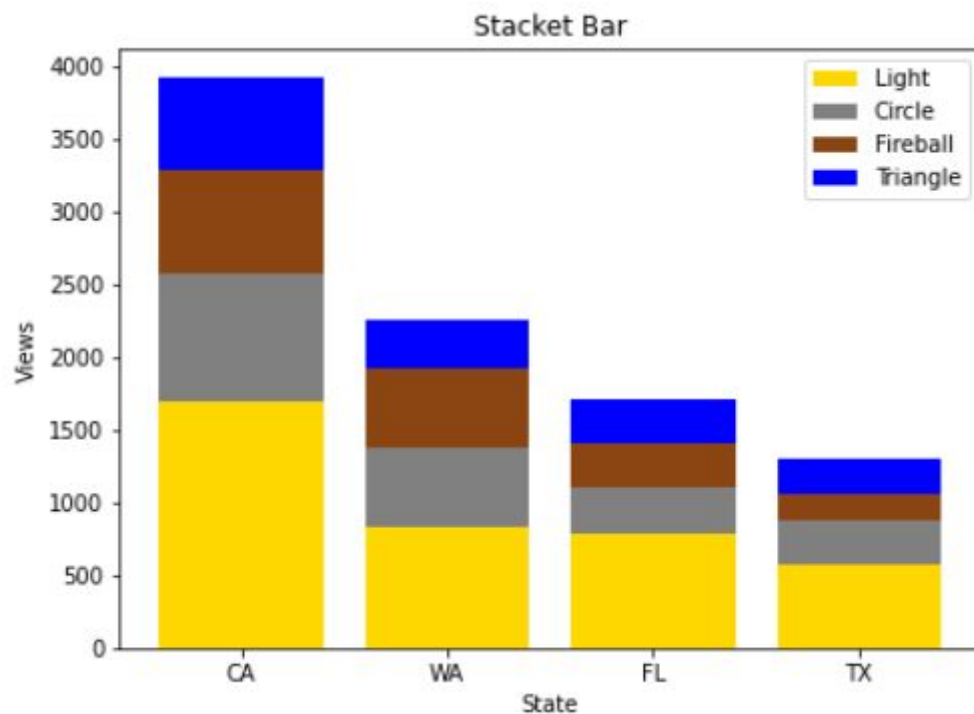

```

14 # Criando as barras
15 plt.bar(states, Light, color = 'gold')
16 plt.bar(states, Circle, color = 'grey', bottom = Light)
17 plt.bar(states, Fireball, color = 'saddlebrown', bottom = Light + Circle)
18 plt.bar(states, Triangle, color = 'blue', bottom = Light + Circle + Fireball)
19
20 # Adicionando legendas as barras
21 plt.xlabel('State')
22 plt.ylabel('Views')
23 plt.title('Stacked Bar')
24 plt.legend(('Light', 'Circle', 'Fireball', 'Triangle'))
25
26 plt.show()

```

Fonte: Própria

Gráfico 2 - Gráfico 2

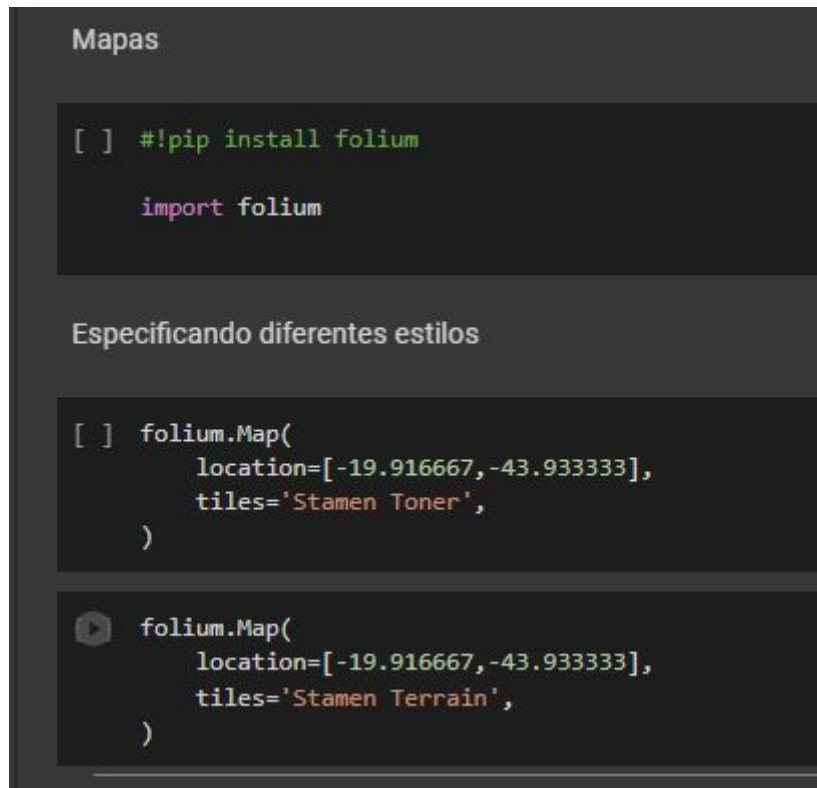


Fonte: Própria

3.2 Mapas

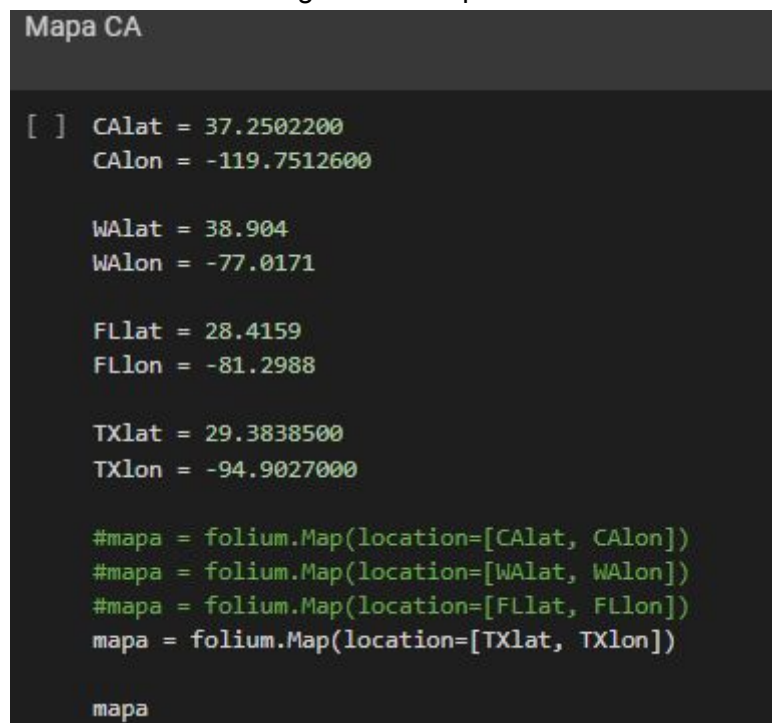
Para a realização dos mapas foi utilizado a biblioteca Folium. Foi feita as importações e especificados os estilos.

Imagem 10 - biblioteca



Fonte: Própria

Imagem 11 - Mapa CA



Fonte: Própria

Também foi instalado e importado a biblioteca ZipCode.

Imagem 12 - Zipcode

```

#instalando a biblioteca zipcode
#!pip install zipcodes
#importando a biblioteca zipcode
import zipcodes
#listando o conteúdo da biblioteca zipcode que está em formato json
zipcodes_json = zipcodes.list_all()
#transformando em um Dataframe a biblioteca zipcode
df_zipcodes = pd.DataFrame(zipcodes_json)
df_zipcodes

#Filtrando as colunas 'Zip_code', 'city','state','lat','long' do DataFrame
df_zipcodes = df_zipcodes[['zip_code','city','state','lat','long']]
df_zipcodes

# Removendo os dados duplicados nas colunas city e state
df_zipcodes.drop_duplicates(subset=['city','state'],inplace=True)
df_zipcodes

```

	zip_code	city	state	lat	long
0	00501	Holtsville	NY	40.8179	-73.0453
2	00601	Adjuntas	PR	18.1967	-66.7367
3	00602	Aguada	PR	18.3529	-67.1775
4	00603	Aguadilla	PR	18.4586	-67.1299
7	00606	Maricao	PR	18.1667	-66.9392
...
42626	99925	Klawock	AK	55.5498	-132.9676
42627	99926	Metlakatla	AK	55.1450	-131.5439
42628	99927	Point Baker	AK	56.1513	-133.3490
42629	99928	Ward Cove	AK	55.4104	-131.7237
42630	99929	Wrangell	AK	56.1800	-132.0304
29791 rows × 5 columns					

Fonte: Própria

Aqui voltamos para o arquivo CSV gerado e ordenamos os dados .

Imagem 13 - Csv

```

#!pip install -U pandasql
# import pandas
import pandasql

#importar o arquivo csv
baseOvnis = pd.read_csv("OVNIS.csv")
# SQL com filtro, dataframe
query = '''
    SELECT State, City,Count(*) as Views FROM baseOvnis group by State,City order by Views desc
'''

df_filtrado = pandasql.sqldf(query, locals())
#df_filtrado.to_csv('filtrado_mapa.csv',index=False)
df_filtrado

```

	State	City	Views
0	AZ	Phoenix	360
1	NV	Las Vegas	334
2	WA	Seattle	322
3	OR	Portland	279
4	CA	San Diego	269
...
20584	YT	Richards Bay (KwaZulu-Natal)(South Africa)	1
20585	YT	Teslin (Canada)	1
20586	YT	Watson Lake (Canada)	1
20587	YT	Yukon City (Canada)	1
20588	YT	Yukon Territory (location undisclosed) (Canada)	1

20589 rows × 3 columns

Fonte: Própria

Para seguir um padrão, resolver colocar a fonte em letras minúsculas. Nas imagens abaixo segue os códigos de como essa padronização foi realizada.

Imagem 14 - Padronização de dados

Fonte: Própria

```

# Padronizar as colunas do DF em letras minúsculas
df_filtrado.columns = map(str.lower, df_filtrado.columns)
df_filtrado

# Padronizar os dados das cidades do DF em letras minúsculas
df_filtrado["city"] = df_filtrado["city"].str.lower()
df_filtrado

# Padronizar as colunas do DFzipcodes em letras minúsculas
df_zipcodes.columns = map(str.lower, df_zipcodes.columns)
df_zipcodes

# Padronizar os dados das cidades do DFzipcodes em letras minúsculas
df_zipcodes["city"] = df_zipcodes["city"].str.lower()
df_zipcodes

```

Imagem 15 - Padronização de dados

```

# Padronizar os dados das cidades do DFzipcodes em letras minúsculas
df_zipcodes["city"] = df_zipcodes["city"].str.lower()
df_zipcodes

# Padronizar os dados dos estados do DF em letras minúsculas
df_filtrado["state"] = df_filtrado["state"].str.lower()
df_filtrado

# Padronizar os dados dos estados do DFzipcodes em letras minúsculas
df_zipcodes["state"] = df_zipcodes["state"].str.lower()
df_zipcodes

```

Fonte: Própria

Depois da padronização dos dados foi realizado o Merge SQL entre o Dataframe do Zipcodes e o Dataframe filtrado.

Imagem 16 -Merge SQL

```

[ ] #Unindo os dataframes DF e dfzipcodes nas colunas estado e cidade
df_merged = df_filtrado.merge(df_zipcodes, on=['state','city'])
df_merged

[ ] #Gerar um arquivo csv
df_merged.to_csv('objetosvoadores.csv')

```


	state	city	views	zip_code	lat	long
0	az	phoenix	360	85001	33.4486	-112.0733
1	nv	las vegas	334	89101	36.1736	-115.1264
2	wa	seattle	322	98101	47.6110	-122.3335
3	or	portland	279	97201	45.5074	-122.6898
4	ca	san diego	269	92101	32.7199	-117.1805
...
10971	wy	recluse	1	82725	44.8203	-105.7762
10972	wy	rozet	1	82727	44.1855	-105.2337
10973	wy	saratoga	1	82331	41.4684	-106.7911
10974	wy	shawnee	1	82229	42.8910	-105.1056
10975	wy	worland	1	82401	43.9071	-108.0607

10976 rows × 6 columns

Fonte: Própria

A seguir o código para determinar as coordenadas e pontos dos Estados Unidos.

Imagem 17 - Estados Unidos

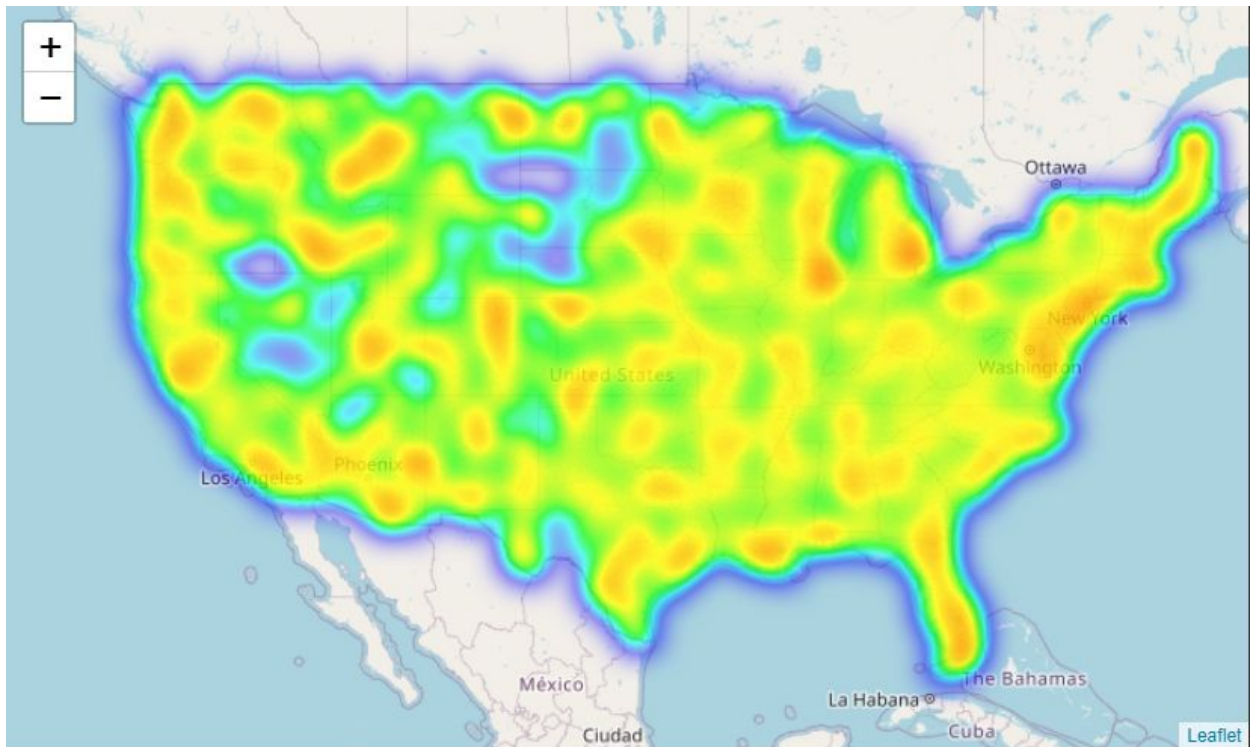
```
[26] 1 #Extrair dados para o mapa
      2 coordenadas = pd.read_csv('objetosvoadores.csv')
      3
      4 #Latitude x Longitude
      5 Lat1 = coordenadas.lat
      6 Long1 = coordenadas.long
      7

[27] 1 #mapa dos EUA
      2 m = folium.Map(
      3     location=[37.8427887, -98.3807258],
      4     zoom_start=4.4
      5 )
      6
      7 HeatMap(list(zip(Lat1, Long1)),radius=13).add_to(m)
      8
      9 m
     10
```

Fonte: Própria

A seguir o mapa dos Estados Unidos com os pontos de avistamento.

Imagem 18 - Mapa Estados Unidos



Fonte: Própria

A seguir o código para determinar as coordenadas e pontos da Califórnia.

Imagem 19 -Califórnia

```

1 #filtrar Califórnia
2
3 coordenadas = pd.read_csv('objetosvoadores.csv')
4
5 q= """
6 SELECT *
7 FROM coordenadas
8 WHERE state = 'ca'
9 """
10
11 T2 = pandasql.sqldf(q, locals())
12 CalFil = pd.DataFrame(T2)
13
14 Lat2 = CalFil.lat
15 Long2 = CalFil.long

```

```

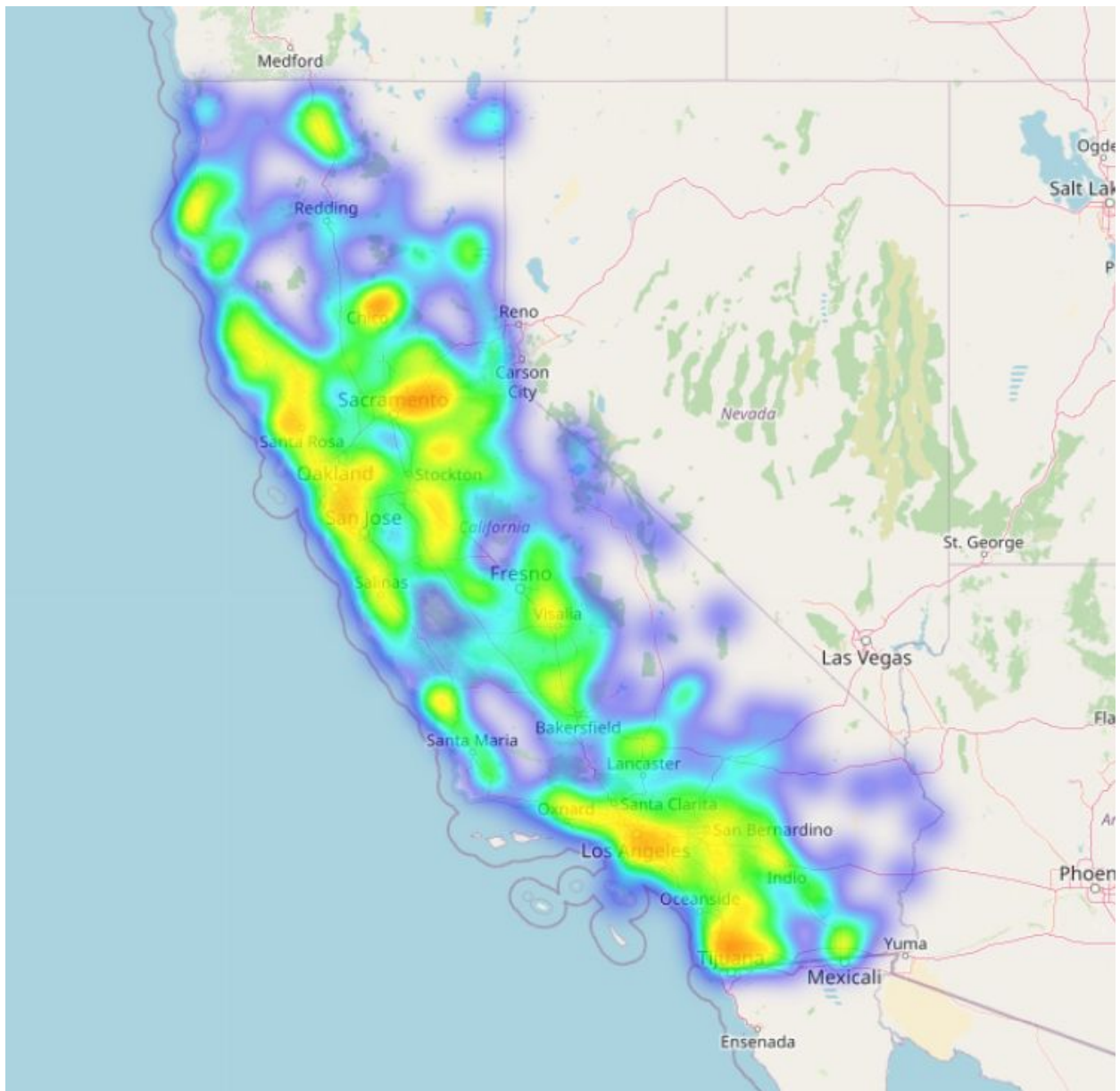
1 #mapa de calor
2
3 m2 = folium.Map(
4     location=[37.2454668, -120.7021918],
5     zoom_start=6
6 )
7
8 HeatMap(list(zip(Lat2, Long2)),radius=13).add_to(m2)
9
10 m2

```

Fonte: Própria

Também temos o mapa de calor da Califórnia:

Imagem 20 -Califórnia



Fonte: Própria

Por fim o link:

Segue o link do GitHub assim será possível visualizar os mapas e gráficos:
<https://github.com/Prof-Fabio-Henrique/pratica-integrada-icd-e-iam-2020-2-grupo-2>.

4. Considerações Finais

Com muito zelo este segundo relatório é finalizado com o apoio de todos e trabalhando juntos para que através de comandos tudo ocorresse conforme o esperado.

Referências

Silveira. Guilherme. **Select count(*), count(1) e count(nome): a batalha dos counts de SQL**. 2017. Disponível em:

<<https://www.alura.com.br/artigos/select-count-count1-e-countnome-a-batalha-dos-counts-de-sql>> Acesso em: 14 de março de 2021.

Teixeira. Douglas. **Manipulação de Listas em Python**. Disponível em:

<<https://algoritmoempython.com.br/cursos/programacao-python/listas/>> Acesso em: 18 de março de 2021.