INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Brasília
Campus Brasília

TECNOLOGIA EM SISTEMAS PARA INTERNET

Jorge Kayodê Lima Trindade
Nathália Teixeira Guimarães

RELATÓRIO DE PRÁTICA INTEGRADA
DE
CIÊNCIA DE DADOS E INTELIGÊNCIA ARTIFICIAL

Brasília - DF

18/08/2021

Sumário

1. Objetivos

Este relatório trata de três partes do projeto de exploração dos dados de avistamento de OVNIS registrados no site NUFORC.

A ***primeira*** parte teve como objetivo realizar as seguintes tarefas:

1. Construir um gráfico baseado nas consultas relativas aos **quatro estados que possuem maior quantidade de relatos de OVNI** e os **formatos mais populares**.
2. Construir um gráfico de **barras agrupadas** com os dados dos estados e formatos de OVNIS mais relatados.
3. Construir um gráfico de **barras empilhadas** com os dados dos estados e formatos de OVNIS mais relatados.
4. Construir um mapa usando a **latitude e longitude** dos **Estados Unidos** e plotar os casos de avistamentos de OVNIS relativos às ocorrências por cidade e estado.
5. Construir um mapa usando a latitude e longitude da **Califórnia** e plotar os casos de avistamentos de OVNIS relativos às ocorrências ocorridas nessa região.
6. Identificar e justificar em que região da Califórnia está concentrada a maior parte das ocorrências de avistamento de OVNIS.

A ***segunda*** parte teve como objetivo realizar as seguintes tarefas:

1. Carregar o arquivo OVNIS.csv em um dataframe;
2. Remover registros que tenham valores vazios (None, Unknown), para City, State e Shape;
3. Manter somente os registros referentes aos 51 estados dos Estados Unidos.
4. Remover variáveis irrelevantes para a análise (Duration, Summary e Posted).
5. Manter somente os registros de Shapes mais populares (com mais de 1000 ocorrências);

6. Salvar o dataframe final em um arquivo CSV com o nome “df_OVNI_limpo”.

A ***terceira*** parte teve como objetivo realizar as seguintes tarefas:

1. Carregar o arquivo ‘df_OVNI_limpo.csv’ em um dataframe.
2. Dividir o conteúdo da coluna Date/Time em duas novas colunas no mesmo dataframe e deletar a coluna Date/Time.
3. Fazer o mesmo procedimento para dias da semana. Descobrir o dia da semana com mais ocorrências de OVNIS.
4. Separar as variáveis mês e dia.
5. Salvar o dataframe resultante em um arquivo .csv com o nome ‘df_OVNI_preparado’.

2. Descrição do problema

O banco de dados dos OVNIS avistados nos Estados Unidos precisa ser limpo e preparado para uma melhor análise de suas informações. Para isso, nessa fase os valores nulos, vazios e desconhecidos serão removidos do DataFrame. Além disso, os dados serão restringidos aos relatos ocorridos nos estados dos Estados Unidos.

É necessário também que algumas informações sejam divididas em mais de uma coluna, para que sua leitura possa ser mais objetiva e melhor aproveitada. Por fim, outras colunas serão criadas a partir das informações concedidas pelas colunas já existentes, assim, alguns dados implícitos, como os dias da semana, poderão ser analisados a partir da coluna ‘Data’, que fornece o momento o qual ocorreu o avistamento de um OVNI.

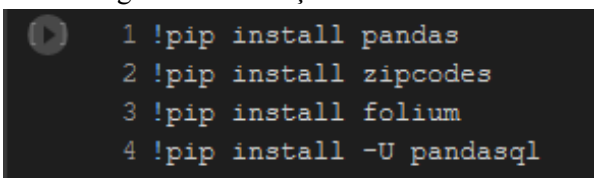
3. Desenvolvimento

Este projeto é desenvolvido em python e nesta etapa foram utilizadas ferramentas como pandas, pandasql, numpy, matplotlib, zipcodes, folium para a extração, limpeza e geração de mapas dos relatos de OVNIS ocorridos nos estados dos Estados Unidos..

3.1 Exploração com Gráficos e Mapas

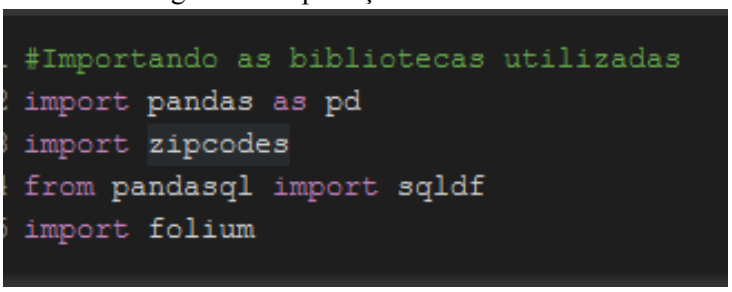
Aqui ficam os códigos que vocês utilizaram no projeto bem como explicações do que cada código faz e referência para o github onde é possível encontrá-lo.

Figura 1 - Instalação de bibliotecas



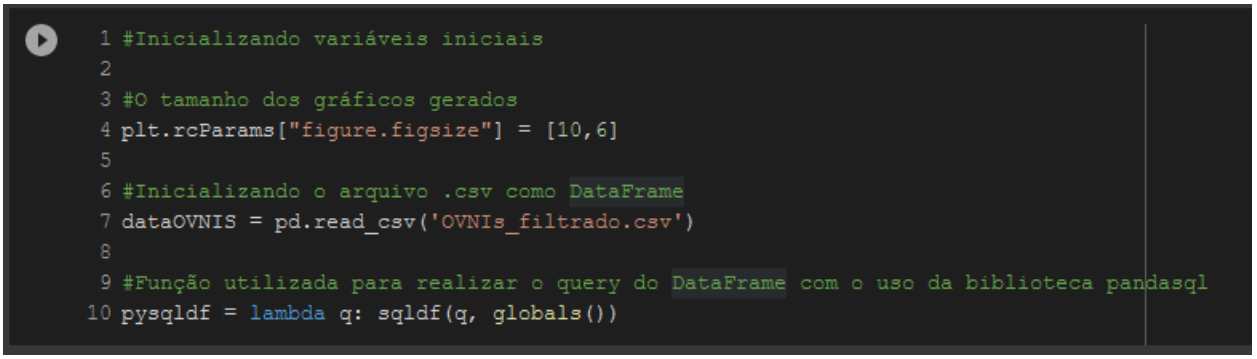
Essas 4 bibliotecas são a base do projeto, daqui instalamos as bibliotecas e temos todas as funções que precisaremos usar.
Fonte: Própria

Figura 2 - Importação das bibliotecas



As bibliotecas pandas e pandasql foram utilizadas para a codificação até o ponto em que geramos os 2 gráficos em barras. Já zipcodes e folium, para a construir os mapas, dos EUA e seus estados, com o mapa de calor baseado no DataFrame de relatos de OVNIS filtrado.
Fonte: Própria

Figura 3 - Inicialização de variáveis



Temos 2 variáveis principais que veremos repetidamente: dataOVNIS, o DataFrame que com os relatos e pysqldf, uma função lambda que nos permite acessar dataOVNIS como uma tabela SQL. Essa declaração logo no começo é essencial para otimização do código.
Fonte: Própria

Figura 4 - Função para query em sqlite3

```
#Essa função é usada para selecionar parte do DataFrame
# de acordo com a Forma
def select_shape(ELEMENTS,FORMA,GROUP):
    q="""
        SELECT """+ELEMENTS+"""
        FROM dataOVNIS
        WHERE Forma = """+FORMA+"""
        AND(Estado = 'CA'
        OR Estado = 'WA'
        OR Estado = 'FL'
        OR Estado = 'TX')
        GROUP BY """+GROUP+"""
        HAVING COUNT(*) > 9
        ORDER BY Estado asc

        LIMIT 4
      """
    return q
```

Esse foi criado reproduzindo a sintaxe do sqlite3, linguagem de escolha do pandasql.
Fonte: Própria

Figura 5 - Construção do DataFrame de formas por estado

```
#Para encontrar a contagem de formas em cada estado
query5 = pysqldf(select_shape('COUNT(*)',"""'Light'""",'Estado'))
query6 = pysqldf(select_shape('COUNT(*)',"""'Circle'""",'Estado'))
query7 = pysqldf(select_shape('COUNT(*)',"""'Fireball'""",'Estado'))
query8 = pysqldf(select_shape('COUNT(*)',"""'Triangle'""",'Estado'))

#Ordem das linhas do dataFrame, ['CA','FL','TX','WA'])
#Ordem das colunas do dataFrame, ['Light','Circle','Fireball','Triangle'])
estadosFORMA = pd.concat([query5,query6,query7,query8], ignore_index=True,axis=1)
estadosFORMA
```

Aqui fazemos o query para encontrar as formas mais populares por Forma, resultando em um DataFrame linhas ou colunas nomeadas.
Fonte: Própria

Figura 6 - Visualização da variável ‘estadosFORMA’

	0	1	2	3
0	2474	1193	1000	987
1	1088	652	630	476
2	861	390	262	433
3	1146	477	465	404

Fonte: Própria

Figura 7- Tratamento inicial de ‘estadosFORMA’

```
#Tratamento para gerar um gráfico em barras
estados = ['CA','FL','TX','WA']
light = estadosFORMA[0].tolist()
circle = estadosFORMA[1].tolist()
fireball = estadosFORMA[2].tolist()
triangle = estadosFORMA[3].tolist()

#Um novo DataFrame é criado dessa vez com os estados e tipos de forma
# explicitados para facilitar a criação dos mapas,
# não seria possível com o shape de índices e labels prévio
plotOVNIS = pd.DataFrame({
    'light': light,
    'circle': circle,
    'fireball': fireball,
    'triangle': triangle}, index=estados)

plotOVNIS
```

Logo antes da criação dos gráficos em barra, realizamos um tratamento no DataFrame 4x4, agora dando nome às linhas e colunas que queremos visualizar
Fonte: Própria

Figura 8 - Resultado da variável ‘plotOVNIS’

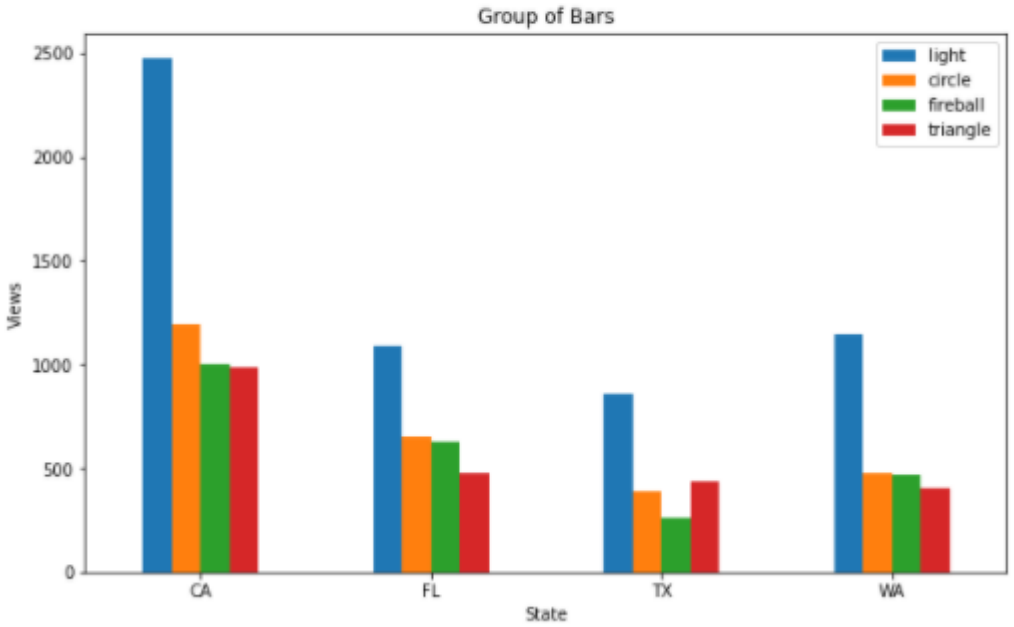
	light	circle	fireball	triangle
CA	2474	1193	1000	987
FL	1088	652	630	476
TX	861	390	262	433
WA	1146	477	465	404

Fonte: Própria

Figura 9- Gráfico de Barras Agrupadas

Plotando os mapas

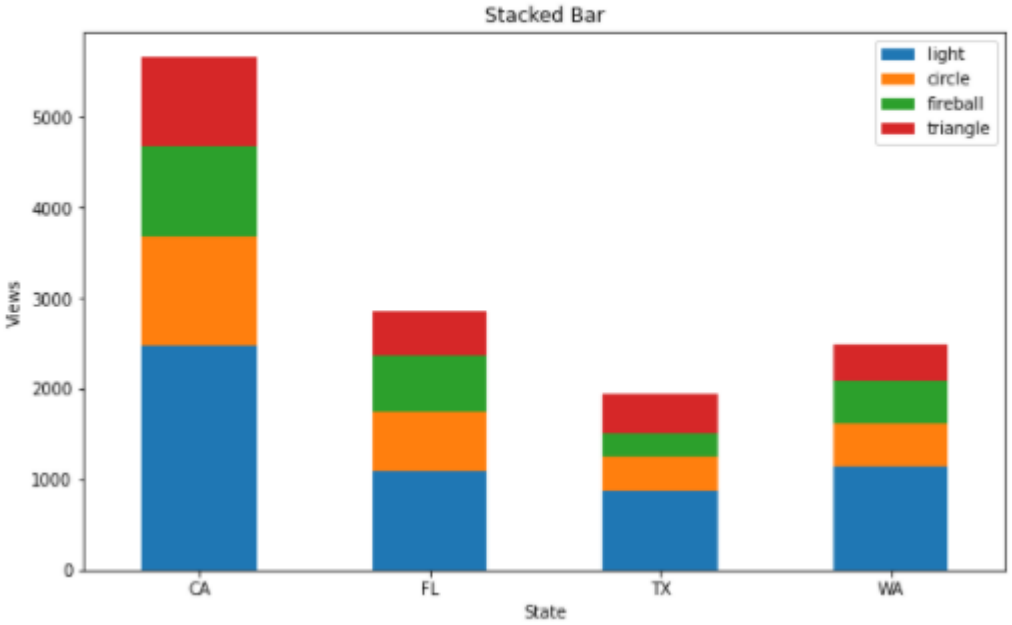
```
[12]  1  ax = plotOVNIS.plot.bar(rot=0, xlabel='State', ylabel='Views', title='Group of Bars'
      2  , legend='Shape')
```



Os dois gráficos foram gerados usando apenas funções da biblioteca pandas
Fonte: Própria

Figura 10- Gráfico de Barras Empilhadas

```
ax1 = plotOVNIS.plot.bar(rot=0, xlabel='State', ylabel='Views', title='Stacked Bar'
, legend='Shape', stacked=True)
```



A diferença entre a criação desses gráficos é apenas a inclusão do parâmetro ‘stacked=True’
Fonte: Própria

Figura 11- Biblioteca Folium

MAPAS

```
# Instalação e impotação da biblioteca do Folium
!pip install folium
import folium
```

Fonte: Própria

Aqui foi instalada e importada a biblioteca folium para a criação dos mapas com a base de dados.

Figura 12- Latitudes e Longitudes dos Estados

```
[29]
# Coordenadas geográficas dos estados "CA, FL, TX, WA"

CAlatitude = 36.778259
CAlongitude = -119.417931

FLlatitude = 25.761681
FLlongitude = -80.191788

TXlatitude = 29.749907
TXlongitude = -95.358421

WAlatitude = 47.751076
WAlongitude = -120.740135

# Mapa de cada um dos Estados "CA, FL, TX, WA"
CAmapa = folium.Map(location=[CAlatitude, CAlongitude])
FLmapa = folium.Map(location=[FLlatitude, FLlongitude])
TXmapa = folium.Map(location=[TXlatitude, TXlongitude])
WAmapa = folium.Map(location=[WAlatitude, WAlongitude])

#Testar mapas
CAmapa
FLmapa
TXmapa
WAmapa
```

Fonte: Própria

Foi feita a pesquisa no Google para saber as coordenadas de cada um dos Estados de maior incidência de avistamento de OVNIS. Depois utilizamos o folium.Map para encontrar a localização desses estados, de modo a testar se as coordenadas estavam certas.

Figura 13- Biblioteca zipcodes

```
# Instalação e importação da biblioteca do zipcodes
!pip install zipcodes
import zipcodes
```

Fonte: Própria

Instalação e importação da biblioteca zipcodes para auxílio na mapeação das latitudes e longitudes para criação dos mapas.

Figura 14- Zipcode

```
# Criação de DataFrame com o conteúdo da biblioteca zipcodes
zipcodes_j = zipcodes.list_all()
df_zipcodes = pd.DataFrame(zipcodes_j)
df_zipcodes.head()
```

Fonte: Própria

Aqui listamos o conteúdo em formato json da zipcodes e criamos um DataFrame com a biblioteca

Figura 15- Dataframe com Biblioteca zipcodes (resultado)

	zip_code	zip_code_type	active	city	acceptable_cities	unacceptable_cities	state	county	timezone	area_codes	world_region	country	lat	long
0	00501	UNIQUE	True	Holtsville	[]	[I R S Service Center]	NY	Suffolk County	America/New_York	[631]	NA	US	40.8179	-73.0453
1	00544	UNIQUE	True	Holtsville	[]	[Irs Service Center]	NY	Suffolk County	America/New_York	[631]	NA	US	40.7888	-73.0394
2	00601	STANDARD	True	Adjuntas	[]	[Colinas Del Gigante, Jard De Adjuntas, Urb Sa...	PR	Adjuntas Municipio	America/Puerto_Rico	[787, 939]	NA	US	18.1967	-66.7367
3	00602	STANDARD	True	Aguada	[]	[Alts De Aguada, Bo Guaniquilla, Comunidad Las...	PR	Aguada Municipio	America/Puerto_Rico	[787, 939]	NA	US	18.3529	-67.1775
4	00603	STANDARD	True	Aguadilla	[Ramey]	[Bda Caban, Bda Esteves, Bo Borinquen, Bo Ceib...	PR	Aguadilla Municipio	America/Puerto_Rico	[787]	NA	US	18.4586	-67.1299

Fonte: Própria

DataFrame criado com a biblioteca zipcodes.

Figura 16- Filtro colunas DF zipcodes

```
# Filtro das colunas "zip_code", "city", "state", "lat", "long" do df_zipcodes
df_zipcodes = df_zipcodes[['zip_code','city','state','lat','long']]

# Redefinição do nome das colunas
df_zipcodes.columns = ['zip_code','Cidade','Estado','Lat','Long']
df_zipcodes
```

	zip_code	Cidade	Estado	Lat	Long
0	00501	Holtsville	NY	40.8179	-73.0453
1	00544	Holtsville	NY	40.7888	-73.0394
2	00601	Adjuntas	PR	18.1967	-66.7367
3	00602	Aguada	PR	18.3529	-67.1775
4	00603	Aguadilla	PR	18.4586	-67.1299
...	...	...	...	...	...
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
42631	99950	Ketchikan	AK	55.8159	-132.9799

42632 rows × 5 columns

Fonte: Própria

Filtro com as colunas do DataFrame que iremos precisar usar na nossa coleta de coordenadas.

Figura 17- Remoção dados duplicados das Colunas ‘Cidade’ e ‘Estado’

```
# Remoção de dados duplicados nas colunas "city" e "state"
df_zipcodes.drop_duplicates(subset=['Cidade','Estado'],inplace=True)
df_zipcodes
```

	zip_code	Cidade	Estado	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

Remoção de dados duplicados das colunas ‘Estado’ e ‘Cidade’.

Figura 18- Cidades com mais relatos de OVNIS

```
# Filtro de Estados com mais visualização de OVNIS no DataFrame 'dataOVNIS'
import pandas as pd
import pandasql

q='''
  SELECT Estado, Cidade, COUNT(*) as Relatos
  FROM dataOVNIS
  GROUP BY Estado, Cidade
  ORDER BY Relatos desc
'''

df_filtro = pandasql.sqldf(q, locals())

# Criando arquivo .csv ['df_filtro'] com o diltro da query
df_filtro.to_csv('df_filtro_mapa.csv', index=False)
df_filtro
```

Fonte: Própria

Pesquisa utilizando ‘pandasql’ para saber quais as cidades com mais relatos de avistamento de OVNIS.

Figura 19- Estados com mais relatos de OVNIS (resultado)

	Estado	Cidade	Relatos
0	AZ	Phoenix	556
1	WA	Seattle	553
2	NV	Las Vegas	470
3	OR	Portland	431
4	CA	San Diego	399
...	...	...	...
21398	WY	Wyoming (I-80, westbound)	1
21399	WY	Wyoming (rural; central)	1
21400	WY	Yellowstone	1
21401	WY	Yellowstone North Entrance	1
21402	WY	na	1

21403 rows × 3 columns

Fonte: Própria

Concluímos que as cinco cidades com mais relatos são: Phoenix, Seattle, Las Vegas, Portland e San Diego, que é a **única na Califórnia**. Além disso, na Sprint 1 identificamos que o Estado da Califórnia é o que possui mais relatos de OVNIS, logo, a cidade em que estão concentradas a maior parte desses relatos é em **San Diego**

.

Figura 20- Coordenadas: fundição dos dataframes ‘df_zipcodes’ e ‘df_filtro’

Cruzamento de DataFrames "df_zicodes" e "df_filtro"

```
df_coordenadas = df_filtro.merge(df_zipcodes, on=['Estado', 'Cidade'])

# Gerar arquivo .csv ['df_coordenadas'] com o cruzamento dos DataFrames
df_coordenadas.to_csv('OVNIS_coordenadas.csv')
df_coordenadas
```

	Estado	Cidade	Relatos	zip_code	Lat	Long
0	AZ	Phoenix	556	85001	33.4486	-112.0733
1	WA	Seattle	553	98101	47.6110	-122.3335
2	NV	Las Vegas	470	89101	36.1736	-115.1264
3	OR	Portland	431	97201	45.5074	-122.6898
4	CA	San Diego	399	92101	32.7199	-117.1805
...	...	...	...	...	...	...
12285	WY	Opal	1	83124	41.7683	-110.2402
12286	WY	Recluse	1	82725	44.8203	-105.7762
12287	WY	Rozet	1	82727	44.1855	-105.2337
12288	WY	Saratoga	1	82331	41.4684	-106.7911
12289	WY	Shawnee	1	82229	42.8910	-105.1056

12290 rows × 6 columns

Fonte: Própria

Criamos um DataFrame que extrai as coordenadas dos Estados a partir da fundição das informações do DataFrame criado com as informações da biblioteca zipcodes (‘df_zipcode’) mais o DataFrame com as informações de relatos de OVNIS (‘df_filtro’).

Figura 21 - Importação de Plugins da biblioteca folium

Extração de dados para criação do MAPA do EUA

```
[159]  1  from folium import plugins
```

Fonte: Própria

Importação da biblioteca plugins.

Figura 22- Definição da Latitude e da Longitude

```
coordenadas = pd.read_csv('OVNIS_coordenadas.csv')

# Definição de Latitude e Longitude
Lat = coordenadas.Lat
Long = coordenadas.Long
```

Fonte: Própria

Criamos uma variável ‘coordenadas’, que lê o arquivo ‘OVNIS_coordenadas.csv’ e utilizamos as informações de latitude e longitude deste arquivo para determinar a latitude e longitude (dos estados e cidades) que utilizaremos na criação do nosso mapa dos Estados Unidos.

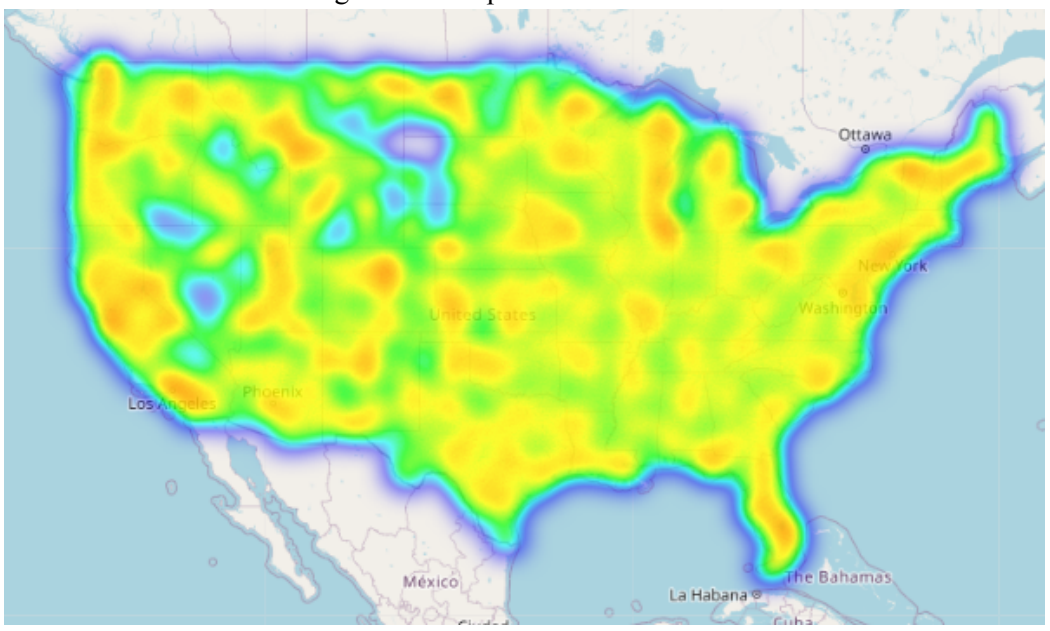
Figura 23- Uso do folium e parâmetros para criação do Mapa dos EUA

```
# Mapeação de calor do EUA

EUA_Mapa = folium.Map(
    location = [37.8427887, -98.3807258],
    zoom_start=4
)

EUA_Mapa.add_child(plugins.HeatMap(list(zip(Lat, Long)), radius=13))

EUA_Mapa
```

Fonte: Própria

Criação do mapa dos Estados Unidos utilizando o folium.map e delimitando a área de cobertura de calor a ser preenchida no mapa a partir do parâmetro ‘location’ e ‘plugins.HeatMap’. Utilizamos as coordenadas de latitude e longitude dos Estados Unidos (encontradas no google) e aquelas retiradas do DataFrame ‘OVNIS_coordenadas.csv’.

Figura 24 - Mapa de Calor dos EUA



Fonte: Própria

Figura 25- Pesquisa para extrair as coordenadas da Califórnia

Extração de dados para criação do MAPA da CALIFÓRNIA

```
coordenadas_CA = pd.read_csv('OVNIS_coordenadas.csv')

q='''
  SELECT *
  FROM coordenadas_CA
  WHERE Estado = 'CA'
'''

filtro_CA = pandasql.sqldf(q, locals())

# DataFrame ['df_california'] com o filtro da query
df_california = pd.DataFrame(filtro_CA)
df_california
```

Fonte: Própria

Pesquisa para selecionar no arquivo 'OVNIS_coordenadas.csv' somente as coordenadas da califórnia e salvar os dados colhidos na pesquisa em um DataFrame.

Figura 26- Dataframe com coordenadas da Califórnia

	Unnamed: 0	Estado	Cidade	Relatos	zip_code	Lat	Long
0	4	CA	San Diego	399	92101	32.7199	-117.1805
1	5	CA	Los Angeles	385	90001	33.9736	-118.2479
2	15	CA	Sacramento	243	94203	38.5819	-121.4935
3	16	CA	San Jose	225	95101	37.3435	-121.8887
4	18	CA	San Francisco	199	94101	37.7700	-122.4100
...	...	...	...	...	...	...	...
802	8095	CA	Wheatland	1	95692	39.0510	-121.3977
803	8096	CA	Wilton	1	95693	38.3988	-121.2547
804	8097	CA	Winterhaven	1	92283	33.0549	-115.0698
805	8098	CA	Wofford Heights	1	93285	35.6173	-118.6098
806	8099	CA	Woodlake	1	93286	36.5378	-119.0342

807 rows × 7 columns

Fonte: Própria

Figura 27- Latitude e Longitude da Califórnia

```
# Definição de Latitude e Longitude do mapa da Califórnia
CALat = df_california.Lat
CALong = df_california.Long
```

Fonte: Própria

Definindo a latitude e longitude da região da Califórnia. Para isso, utilizamos as informações colhidas na pesquisa feita anteriormente e salva no DataFrame 'df_California'.

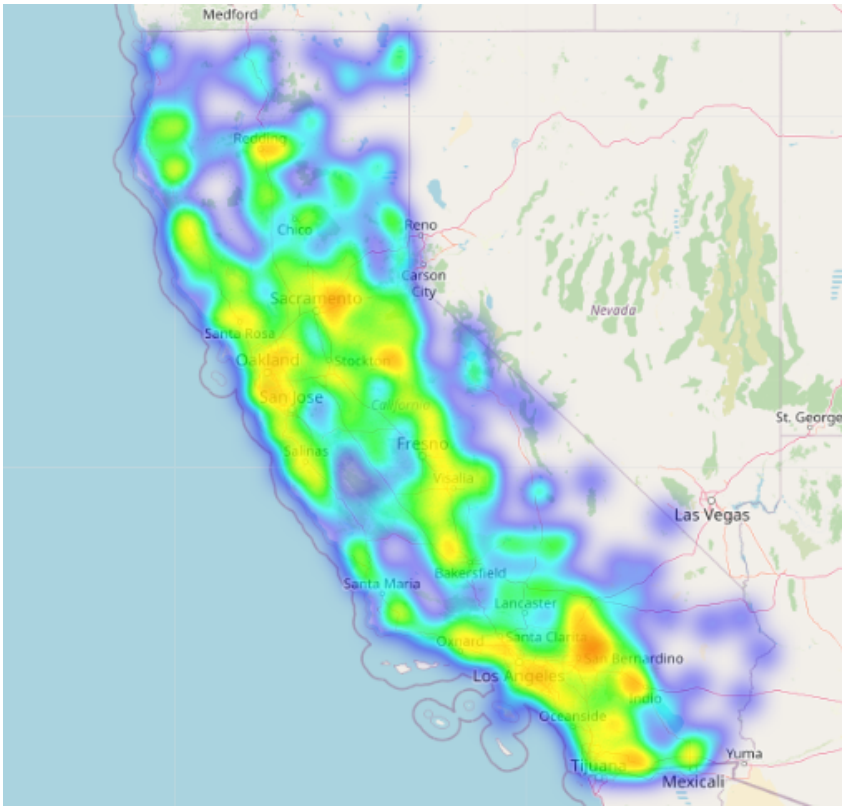
Figura 28- Uso do folium e parâmetros para criação do Mapa da Califórnia

```
# Mapeação de Calor da Califórnia

CA_Mapa = folium.Map(
    location = [36.778259, -119.417931],
    zoom_start=5.5
)

CA_Mapa.add_child(plugins.HeatMap(list(zip(CALat, CALong)), radius=13).add_to(CA_Mapa))

CA_Mapa
```

Fonte: Própria

Criação do mapa da Califórnia utilizando o folium.map e delimitando a área de cobertura de calor a ser preenchida no mapa a partir do parâmetro 'location' e 'plugins.HeatMap'. Utilizamos as coordenadas de latitude e longitude da Califórnia (encontradas no google) e aquelas retiradas do DataFrame 'df_california'.

Figura 29- Mapa de Calor da Califórnia



Fonte: Própria

3.2 Limpeza de Dados

Figura 30- Importação e instalação de bibliotecas pandas

SEQUÊNCIA DE TAREFAS

```
[10]  1  import pandas as pd
      2  !pip install pandasql
      3  import pandasql
```

Fonte: Própria

Importamos e instalamos as biliotecas ‘pandasql’ necessárias para as pesquisas SQL a serem feitas.

Figura 31- Arquivo OVNIS.csv carregado em um DataFrame

```
# 1- Carregar o seu arquivo OVNIS.csv em um dataframe;
dataOVNIS = pd.read_csv('OVNIs_filtrado.csv')

dataOVNIS
```

	Unnamed: 0	Data/Hora	Cidade	Estado	Forma	Duracao	Descricao	Postagem
0	37	11/17/97 03:00	Wasilla	AK	Triangle	3 minutes	A huge trianglar object moving in a true north...	9/2/05
1	37	1/15/98 13:00	Alaska (remote)	AK	Disk	20 seconds	saucer shaped object passed directly above me	3/16/00
2	58	1/8/98 22:38	Fairbanks	AK	Oval	3 Secs	A large phlorescent green oval shaped object m...	2/16/99
3	55	3/15/98 20:30	Anchorage	AK	Oval	15 minutes	Driving home after daughters birthday, noticed...	6/18/98
4	57	3/15/98 20:00	North Pole	AK	Sphere	90 sec	Red orb, blue orb...similar to the Marfa Lights	5/24/05
...	...	...	...	...	...	...	...	...
89271	224	7/17/17 21:00	Cody	WY	Cylinder	30 seconds	Craft heading Northeast at a high rate of spee...	7/23/17
89272	293	7/13/17 04:00	Pinedale	WY	Sphere	NaN	Star-like orb flashing colors in night sky.	7/23/17
89273	34	10/28/17 20:00	Cody and Wapiti (between)	WY	Fireball	10 seconds	There were fireballs coming down. ((anonymous ...	11/3/17
89274	282	10/15/17 20:15	Douglas	WY	Sphere	3 minutes	2 of my friends and I were standing on my balc...	10/19/17
89275	460	10/4/17 02:30	Casper	WY	Unknown	1	7 foot tall, 'Grey', Casper, Wyoming, paid no ...	10/19/17

89276 rows × 8 columns

Fonte: Própria

Carregamos o arquivo ‘OVNIS_filtrado’, resultado final das coletas, extrações e limpezas feitas na Sprint 1.

Figura 32- Remover registros vazios das colunas ‘Cidade’, ‘Estado’ e ‘Forma’

```
# 2- Remover registros que tenham valores vazios (None, Unknown, NaN) para City, State e Shape;
dataOVNIS['Cidade'].dropna()
dataOVNIS.drop(dataOVNIS.index[dataOVNIS['Cidade'] == 'None'], inplace = True)
dataOVNIS.drop(dataOVNIS.index[dataOVNIS['Cidade'] == None], inplace = True)

dataOVNIS['Estado'].dropna()
dataOVNIS.drop(dataOVNIS.index[dataOVNIS['Estado'] == 'None'], inplace = True)
dataOVNIS.drop(dataOVNIS.index[dataOVNIS['Estado'] == None], inplace = True)

dataOVNIS['Forma'].dropna()
dataOVNIS.drop(dataOVNIS.index[dataOVNIS['Forma'] == 'None'], inplace = True)
dataOVNIS.drop(dataOVNIS.index[dataOVNIS['Forma'] == None], inplace = True)

dataOVNIS
```

Fonte: Própria

Aqui nós removemos todos os registros com valores vazios nas colunas ‘Cidade’, ‘Estado’ e ‘Forma’, visto que utilizaremos as informações contidas nessas colunas e é importante para uma pesquisa eficiente que os dados sejam concretos e limpos.

Figura 33- DataFrame ‘dataOVNIS’ após remoção dos valores vazios

	Unnamed: 0	Data/Hora	Cidade	Estado	Forma	Duracao	Descricao	Postagem
0	37	11/17/97 03:00	Wasilla	AK	Triangle	3 minutes	A huge trianglar object moving in a true north...	9/2/05
1	37	1/15/98 13:00	Alaska (remote)	AK	Disk	20 seconds	saucer shaped object passed directly above me	3/16/00
2	58	1/8/98 22:38	Fairbanks	AK	Oval	3 Secs	A large phlorescent green oval shaped object m...	2/16/99
3	55	3/15/98 20:30	Anchorage	AK	Oval	15 minutes	Driving home after daughters birthday, noticed...	6/18/98
4	57	3/15/98 20:00	North Pole	AK	Sphere	90 sec	Red orb, blue orb...similar to the Marfa Lights	5/24/05
...	...	...	...	...	...	...	...	...
89271	224	7/17/17 21:00	Cody	WY	Cylinder	30 seconds	Craft heading Northeast at a high rate of spee...	7/23/17
89272	293	7/13/17 04:00	Pinedale	WY	Sphere	NaN	Star-like orb flashing colors in night sky.	7/23/17
89273	34	10/28/17 20:00	Cody and Wapiti (between)	WY	Fireball	10 seconds	There were fireballs coming down. ((anonymous ...	11/3/17
89274	282	10/15/17 20:15	Douglas	WY	Sphere	3 minutes	2 of my friends and I were standing on my balc...	10/19/17
89275	460	10/4/17 02:30	Casper	WY	Unknown	1	7 foot tall, 'Grey', Casper, Wyoming, paid no ...	10/19/17

89274 rows × 8 columns

Fonte: Própria

Resultado do DataFrame após remoção dos valores vazios nas colunas citadas anteriormente.

Figura 34- Limitar DataFrame aos Relatos feitos nos Estados dos EUA

```
# 3- Manter somente os registros referentes aos 51 estados dos Estados Unidos;

q='''
    SELECT *
    FROM dataOVNIS
    WHERE Estado in ('AK','AL','AR','AZ','CA','CO','CT','DE','FL','GA','HI','IA','ID','IL',
    'IN','KS','KY','LA','MA','MD','ME','MI','MN','MO','MS','MT','NC','ND',
    'NE','NH','NJ','NM','NV','NY','OH','OK','OR','PA','RI','SC','SD','TN',
    'TX','UT','VT','VA','WA','WI','WV','WY')
    ORDER BY Estado
'''
consulta = pandasql.sqldf(q, locals())

EUA_ovnis = pd.DataFrame(consulta)
EUA_ovnis
```

Fonte: Própria

Na Sprint 1 já havíamos restringido o uso dos dados de relatos de OVNIS fornecidos pelo site NUFORC aos relatos ocorridos somente nos Estados dos EUA, mas nessa fase da Sprint 2 foi solicitado que fizéssemos isso novamente, então utilizamos uma pesquisa com ‘pandasql’ para fazê-lo.

Figura 35- Resultado do filtro para os 51 Estados dos EUA

	Unnamed: 0	Data/Hora	Cidade	Estado	Forma	Duracao	Descricao	Postagem
0	37	11/17/97 03:00	Wasilla	AK	Triangle	3 minutes	A huge trianglar object moving in a true north...	9/2/05
1	37	1/15/98 13:00	Alaska (remote)	AK	Disk	20 seconds	saucer shaped object passed directly above me	3/16/00
2	58	1/8/98 22:38	Fairbanks	AK	Oval	3 Secs	A large phlorescent green oval shaped object m...	2/16/99
3	55	3/15/98 20:30	Anchorage	AK	Oval	15 minutes	Driving home after daughters birthday, noticed...	6/18/98
4	57	3/15/98 20:00	North Pole	AK	Sphere	90 sec	Red orb, blue orb...similar to the Marfa Lights	5/24/05
...	...	...	...	...	...	...	...	...
89269	224	7/17/17 21:00	Cody	WY	Cylinder	30 seconds	Craft heading Northeast at a high rate of spee...	7/23/17
89270	293	7/13/17 04:00	Pinedale	WY	Sphere	None	Star-like orb flashing colors in night sky.	7/23/17
89271	34	10/28/17 20:00	Cody and Wapiti (between)	WY	Fireball	10 seconds	There were fireballs coming down. ((anonymous ...	11/3/17
89272	282	10/15/17 20:15	Douglas	WY	Sphere	3 minutes	2 of my friends and I were standing on my balc...	10/19/17
89273	460	10/4/17 02:30	Casper	WY	Unknown	1	7 foot tall, 'Grey', Casper, Wyoming, paid no ...	10/19/17

89274 rows × 8 columns

Fonte: Própria

Devido a isso, é importante observar que o resultado de linhas permanece o mesmo da pesquisa anteriormente feita.

Figura 36- Pesquisa para descobrir os formatos de OVNIS mais populares

```
# 5- Manter somente os registros de Shapes mais populares (com mais de 1000 ocorrências);

# Contagem de ocorrências por formato de ovni
contagem_Formatos = EUA_ovnis['Forma'].value_counts()
formas_pop = contagem_Formatos[contagem_Formatos>1000]
formas_pop
```

```
Light        19073
Circle        9467
Triangle      8428
Fireball      7579
Unknown       6497
Other         6018
Sphere        5986
Disk          4342
Oval          3924
Formation     2889
Changing      2300
Cigar         1923
Flash         1712
Rectangle     1460
Cylinder      1387
Diamond       1334
Chevron       1070
Name: Forma, dtype: int64
```

Fonte: Própria

Para conseguirmos extrair do nosso DataFrame somente os registros de formatos de OVNIS relatados mais de mil vezes, precisamos antes saber o ranking de formatos mais relatados.

Figura 37 - Filtro do DataFrame com os OVNIS mais populares

```
# Manter no DF somente os registros com os formatos mais populares

q='''
  SELECT *
  FROM EUA_ovnis
  WHERE Forma in ('Light', 'Circle','Triangle', 'Fireball', 'Sphere','Disk','Oval',
  'Formation','Changing','Cigar','Flash','Rectangle','Cylinder','Diamond','Chevron')
'''

filtro_Shapes = pandasql.sqldf(q, locals())
filtro_Shapes

```

Fonte: Própria

Após sabermos os formatos mais populares, fizemos uma pesquisa SQL limitando o retorno de dados somente aos formatos vistos relatados mais de mil vezes.

Figura 38 - Filtro do DataFrame com os OVNIS mais populares (resultado)

	Unnamed: 0	Data/Hora	Cidade	Estado	Forma
0	37	11/17/97 03:00	Wasilla	AK	Triangle
1	37	1/15/98 13:00	Alaska (remote)	AK	Disk
2	58	1/8/98 22:38	Fairbanks	AK	Oval
3	55	3/15/98 20:30	Anchorage	AK	Oval
4	57	3/15/98 20:00	North Pole	AK	Sphere
...	...	...	...	...	...
72869	86	7/26/17 21:50	Buffalo	WY	Circle
72870	224	7/17/17 21:00	Cody	WY	Cylinder
72871	293	7/13/17 04:00	Pinedale	WY	Sphere
72872	34	10/28/17 20:00	Cody and Wapiti (between)	WY	Fireball
72873	282	10/15/17 20:15	Douglas	WY	Sphere

72874 rows × 5 columns

Fonte: Própria

Figura 39 - DataFrame final salvo no arquivo ‘df_OVNI_limpo.csv’

```
#6- Salvar o dataframe final em um arquivo CSV com o nome "df_OVNI_limpo".

OVNI_limpo = pd.DataFrame(filtro_Shapes)
OVNI_limpo.to_csv("df_OVNI_limpo.csv", index = False)
OVNI_limpo
```

	Unnamed: 0	Data/Hora	Cidade	Estado	Forma
0	37	11/17/97 03:00	Wasilla	AK	Triangle
1	37	1/15/98 13:00	Alaska (remote)	AK	Disk
2	58	1/8/98 22:38	Fairbanks	AK	Oval
3	55	3/15/98 20:30	Anchorage	AK	Oval
4	57	3/15/98 20:00	North Pole	AK	Sphere
...	...	...	...	...	...
72869	86	7/26/17 21:50	Buffalo	WY	Circle
72870	224	7/17/17 21:00	Cody	WY	Cylinder
72871	293	7/13/17 04:00	Pinedale	WY	Sphere
72872	34	10/28/17 20:00	Cody and Wapiti (between)	WY	Fireball
72873	282	10/15/17 20:15	Douglas	WY	Sphere

72874 rows × 5 columns

Fonte: Própria

Por fim, geramos um DataFrame com esta última pesquisa e salvamos o resultado dela em um arquivo csv chamado ’df_OVNI_limpo’.

3.3 Acréscimo de variáveis

Figura 40- Importação e instalação de bibliotecas Python e Numpy

SEQUÊNCIA DE TAREFAS

```
[2]  1  import pandas as pd
     2  !pip install pandasql
     3  import pandasql
     4  import numpy as np
```

Fonte: Própria

Importação e instalação das bibliotecas pandas, pandasql e numpy a serem usadas nessa fase da Sprint 2.

Figura 41- Arquivo ‘df_OVNI_limpo’ carregado em um DataFrame

```
1  # 1-Carregar o seu arquivo df_OVNI_limpo.csv (arquivo gerado após a limpeza de
2  # dados efetuada na atividade 2.3) em um dataframe;
3
4  dataOVNIS = pd.read_csv('df_OVNI_limpo.csv')
5  #dataOVNIS
```

Fonte: Própria

Carregamos o arquivo gerado na fase anterior da Sprint 2 (‘df_OVNI_limpo.csv’) em um DataFrame chamado ‘dataOVNIS’.

Figura 42- Divisão dos conteúdos da coluna ‘Data/Hora’

```
1  # 2-Dividir o conteúdo da coluna Date / Time em duas novas colunas no mesmo dataframe e deletar a coluna Date / Time .
2
3  # Dividir Data e Hora
4  dataOVNIS['Data'], dataOVNIS['Hora'] = dataOVNIS['Data/Hora'].str.split(' ',1).str
5
6  # Deletar coluna 'DateTime' e 'Unnamed: 0'
7  dataOVNIS.drop(columns=['Data/Hora'], inplace=True)
8  dataOVNIS.drop(columns=['Unnamed: 0'], inplace=True)
9
10 dataOVNIS
```

Fonte: Própria

Aqui fizemos a divisão da coluna ‘Data/Hora’ e duas colunas, uma para data e outra para hora’ utilizando o split e aproveitamos para deletar a coluna ‘Unnamed: 0’ com o drop.

Figura 43- Divisão dos conteúdos da coluna ‘Data/Hora’ (resultado)

	Cidade	Estado	Forma	Data	Hora
0	Wasilla	AK	Triangle	11/17/97	03:00
1	Alaska (remote)	AK	Disk	1/15/98	13:00
2	Fairbanks	AK	Oval	1/8/98	22:38
3	Anchorage	AK	Oval	3/15/98	20:30
4	North Pole	AK	Sphere	3/15/98	20:00
...	...	...	...	...	...
72869	Buffalo	WY	Circle	7/26/17	21:50
72870	Cody	WY	Cylinder	7/17/17	21:00
72871	Pinedale	WY	Sphere	7/13/17	04:00
72872	Cody and Wapiti (between)	WY	Fireball	10/28/17	20:00
72873	Douglas	WY	Sphere	10/15/17	20:15

72874 rows × 5 columns

Fonte: Própria

Na figura 43temos o resultado do nosso DataFrame com as duas novas colunas denominadas ‘Data’ e ‘Hora’.

Figura 44 - Criação de uma coluna com os dias da semana

```
# 3-Fazer o mesmo procedimento para dias da semana. Será que existe um dia da semana com mais
# ocorrências de relatórios para OVNIs? Para descobrir isso, você deve criar uma nova coluna chamada weekdays.

# Extrair os dias da semana dos registro da coluna 'Data'
datas = pd.to_datetime(dataOVNIS['Data'])
datas = datas.dt.dayofweek

# Inserção dos dias da semana
dias_semana = {0: 'Segunda-feira',
               1:'Terça-feira',
               2: 'Quarta-feira',
               3: 'Quinta-feira',
               4: 'Sexta-feira',
               5: 'Sábado',
               6: 'Domingo'}

dataOVNIS['Dias_da_Semana'] = datas.map(dias_semana)

dataOVNIS
```

Fonte: Própria

Aqui nós criamos um dicionário com os dias da semana e fizemos a extração e mapeamento das informações da coluna ‘Data’ para conseguirmos coletar qual dia da semana ocorreram cada um dos relatos de avistamento de OVNIS.

Figura 45 - Criação de uma coluna com os dias da semana (resultado)

	Cidade	Estado	Forma	Data	Hora	Dias_da_Semana
0	Wasilla	AK	Triangle	11/17/97	03:00	Segunda-feira
1	Alaska (remote)	AK	Disk	1/15/98	13:00	Quinta-feira
2	Fairbanks	AK	Oval	1/8/98	22:38	Quinta-feira
3	Anchorage	AK	Oval	3/15/98	20:30	Domingo
4	North Pole	AK	Sphere	3/15/98	20:00	Domingo
...	...	...	...	...	...	...
72869	Buffalo	WY	Circle	7/26/17	21:50	Quarta-feira
72870	Cody	WY	Cylinder	7/17/17	21:00	Segunda-feira
72871	Pinedale	WY	Sphere	7/13/17	04:00	Quinta-feira
72872	Cody and Wapiti (between)	WY	Fireball	10/28/17	20:00	Sábado
72873	Douglas	WY	Sphere	10/15/17	20:15	Domingo

72874 rows × 6 columns

Fonte: Própria

Na figura 45 temos o resultado do DataFrame ‘dataOVNIS’ após a inclusão da coluna ‘Dias_da_Semana’.

Figura 46 - Criação de uma coluna para ‘mês’ e outra para ‘dia’

```
# 4-Separar as variáveis mês (Month) e dia (Day). Desse modo, será possível refinar as pesquisas.

dataOVNIS['Dia'] = dataOVNIS['Data'].str.split('/', expand = True)[1]
dataOVNIS['Mês'] = dataOVNIS['Data'].str.split('/', expand = True)[0]

dataOVNIS
```

Fonte: Própria

Aqui precisamos criar uma coluna para ‘Dia’ e outra para ‘Mês’. Para isso, utilizamos o split e determinamos a posição de índice nas datas trazidas pela coluna ‘Data’. Assim, na coluna ‘Mês’, eu teria todas as informações da coluna ‘Data’ posicionadas no índice 0. O mesmo ocorre para a coluna ‘Dia’, só mudando o índice, que passa a ser 1, conforme as datas norte americanas.

Figura 47 - Criação de uma coluna para ‘mês’ e outra para ‘dia’ (resultado)

	Cidade	Estado	Forma	Data	Hora	Dias_da_Semana	Dia	Mês
0	Wasilla	AK	Triangle	11/17/97	03:00	Segunda-feira	17	11
1	Alaska (remote)	AK	Disk	1/15/98	13:00	Quinta-feira	15	1
2	Fairbanks	AK	Oval	1/8/98	22:38	Quinta-feira	8	1
3	Anchorage	AK	Oval	3/15/98	20:30	Domingo	15	3
4	North Pole	AK	Sphere	3/15/98	20:00	Domingo	15	3
...	...	...	...	...	...	...	...	...
72869	Buffalo	WY	Circle	7/26/17	21:50	Quarta-feira	26	7
72870	Cody	WY	Cylinder	7/17/17	21:00	Segunda-feira	17	7
72871	Pinedale	WY	Sphere	7/13/17	04:00	Quinta-feira	13	7
72872	Cody and Wapiti (between)	WY	Fireball	10/28/17	20:00	Sábado	28	10
72873	Douglas	WY	Sphere	10/15/17	20:15	Domingo	15	10

72874 rows × 8 columns

Na figura ‘’ temos o resultado do DataFrame após a inclusão das colunas ‘Dia’ e ‘Mês’.

4. Considerações Finais

Comparado ao início do desenvolvimento da Sprint 2(2.1), várias porções de código foram cortadas: um query feito por estado, sem considerar as formas; a função que realizava esse query, e um query temporário para encontrar a contagem de relatos dos 4 estados mais frequentes, mas que se tornou relevante por não considerar as formas também.

Além disso, é necessário pontuar que apenas os alunos Jorge e Nathália participaram do desenvolvimento da Sprint 2.

Referências

Minerando dados. **Plotando Mapas Interativos com Python - Visualize dados de Vendas por Região -**. Disponível em: <https://minerandodados.com.br/plotando-mapas-interativos-com-python-visualize-dados-de-vendas-por-regiao/>. Acesso em: 17 de agosto de 2021.