

TECNOLOGIA EM SISTEMAS PARA INTERNET

**Daniel Evangelista Pereira
Ribson Coelho Cardoch Valdés
Douglas Seidi Shibata**

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

03/10/2020

Brasília - DF

2020

Sumário

1. Objetivos	3
2. Descrição do problema	4
3. Desenvolvimento	5
3.1 Código implementado	5
4. Considerações Finais	15
Referências	16

1. Objetivos

Esta etapa do projeto tem como finalidade a exploração dos dados utilizando a linguagem SQL no script do notebook em python, que foram coletados na primeira fase. E a partir desta exploração serão gerados mapas e gráficos para a melhor visualização dos dados coletados.

2. Descrição do problema

Nesta Etapa foi feita a exploração dos dados e gerados gráficos e mapas para a melhor visualização.

3. Desenvolvimento

As tecnologias utilizadas para a elaboração desta segunda fase do projeto, será a linguagem python e algumas bibliotecas para ajudar no desenvolvimento, com a biblioteca matplotlib e seaborn para a geração de gráficos, por meio do notebook e o ambiente de desenvolvimento Google Colab.

3.1 Código implementado

5.5 - Exploração com gráficos e mapas

Realizando a leitura do arquivo com apenas dados dos estados unidos, e atribuindo a variável `df_just_usa` que será o dataframe para realizar a análise.

```
[19] 1 df_just_usa = pd.read_csv('apenas_usa.csv')
```

Realizando consulta utilizando a query do SQL para ter a informação do número de ocorrências, agrupando por estado e formato para saber a quantidade de casos por estados e quais formas foram mais recorrentes naquele estado.

```
1 # Executa o seu comando SQL e retorna um dataframe
2 query = '''
3 | SELECT State, city, Count(City) as 'Numero de Ocorrencias', Shape FROM df_just_usa group by State, shape having count(City)>=10 order by count(City) desc
4 | '''
5 ocorrencia_usa_all_state = pandasql.sqldf(query.lower(), locals())
6 ocorrencia_usa_all_state
```

	State	City	numero de ocorrencias	Shape
0	CA	Davis	1700	Light
1	CA	Chino	880	Circle
2	FL	Davenport	824	Light
3	WA	Coulee Dam	778	Light
4	CA	Corona	703	Fireball
...
803	UT	South Jordan	10	Rectangle

Dividindo pelo estado com o maior número de ocorrência, que neste caso é o estado da Califórnia e mostrando os formatos mais recorrentes. E atribuindo a um novo dataframe chamado `ca_ocorrencias`

```
1 # Executa o seu comando SQL e retorna um dataframe
2 query = '''
3 | SELECT State, Count(State) as 'Numero de Ocorrencias', Shape FROM df_just_usa where State like '%CA%' group by State, Shape having count(State)>=10 order
4 | '''
5 ca_ocorrencias = pandasql.sqldf(query.lower(), locals())
6 ca_ocorrencias
```

	State	numero de ocorrencias	Shape
0	CA	1700	Light
1	CA	880	Circle
2	CA	703	Fireball
3	CA	640	Triangle

Dividindo pelo estado com o maior número de ocorrência, que neste caso é o estado da Flórida e mostrando os formatos mais recorrentes. E atribuindo a um novo dataframe chamado fl_ocorrencias

```
1 # Executa o seu comando SQL e retorna um dataframe
2 query = '''
3 | SELECT State,Count(State) as 'Numero de Ocorrencias', Shape FROM df_just_usa where State like '%FL%' group by State,Shape having count(State)>=10 order b
4 | '''
5 fl_ocorrencias = pandasql.sqldf(query.lower(), locals())
6 fl_ocorrencias
```

	State	numero de ocorrencias	Shape
0	FL	824	Light
1	FL	551	Circle
2	FL	541	Fireball
3	FL	343	Triangle

Dividindo pelo estado com o maior número de ocorrência, que neste caso é o estado de Washington e mostrando os formatos mais recorrentes. E atribuindo a um novo dataframe chamado wa_ocorrencias

```
[ ] 1 # Executa o seu comando SQL e retorna um dataframe
2 query = '''
3 | SELECT State,Count(State) as 'Numero de Ocorrencias', Shape FROM df_just_usa where State like '%WA%' group by State,Shape having count(State)>=10 order b
4 | '''
5 wa_ocorrencias = pandasql.sqldf(query.lower(), locals())
6 wa_ocorrencias =wa_ocorrencias.drop(index=3)
7 wa_ocorrencias
```

	State	numero de ocorrencias	Shape
0	WA	779	Light
1	WA	329	Circle
2	WA	296	Fireball
4	WA	245	Triangle

Dividindo pelo estado com o maior número de ocorrência, que neste caso é o estado do Texas e mostrando os formatos mais recorrentes. E atribuindo a um novo dataframe chamado tx_ocorrencias

```
[ ] 1 # Executa o seu comando SQL e retorna um dataframe
2 query = '''
3 | SELECT State,Count(State) as 'Numero de Ocorrencias', Shape FROM df_just_usa where State like '%tx%' group by State,Shape having count(State)>=10 order b
4 | '''
5 tx_ocorrencias = pandasql.sqldf(query.lower(), locals())
6 tx_ocorrencias = tx_ocorrencias.drop(index=[3,4])
7 tx_ocorrencias
```

	State	numero de ocorrencias	Shape
0	TX	579	Light
1	TX	301	Triangle
2	TX	290	Circle
6	TX	183	Fireball

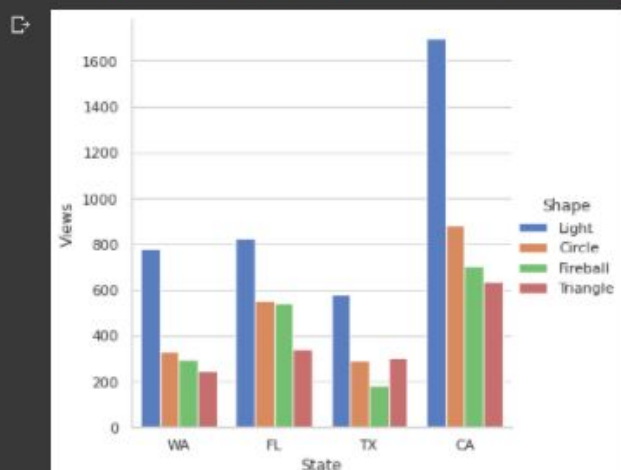
Juntando os Dataframes e atribuindo a um novo dataframe, com os 4 estados mais recorrentes e suas formas.

```
[ ] 1 state_shape = pd.concat([wa_ocorrencias,fl_ocorrencias,tx_ocorrencias,ca_ocorrencias])
    2 state_shape
```

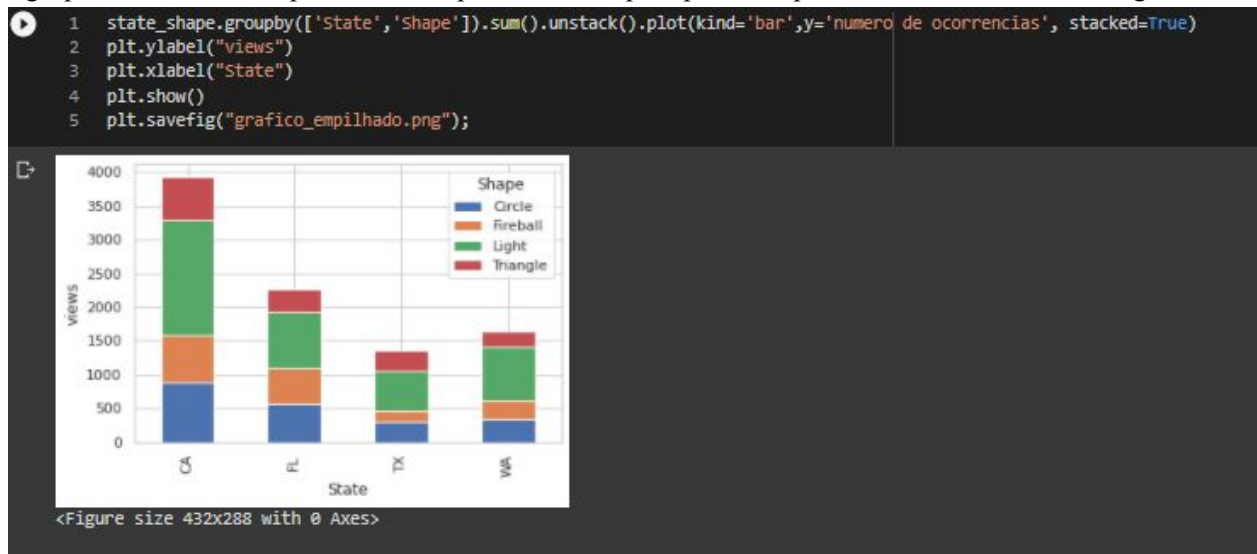
	State	numero de ocorrencias	Shape
0	WA	779	Light
1	WA	329	Circle
2	WA	296	Fireball
4	WA	245	Triangle
0	FL	824	Light
1	FL	551	Circle
2	FL	541	Fireball
3	FL	343	Triangle
0	TX	579	Light
1	TX	301	Triangle
2	TX	290	Circle
5	TX	183	Fireball
0	CA	1700	Light
1	CA	880	Circle

Gerando o gráfico de barras mostrando os estados da califórnia, washington, Flórida e Texas, com as visualização no eixo y e os estados no eixo x. E depois exportando o gráfico.

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.set(style="whitegrid")
5 g = sns.catplot(x="State", y="numero de ocorrencias", hue='Shape', data=state_shape, kind="bar", palette="muted")
6 g.set_ylabels("Views")
7 g.set_xlabels("State")
8
9 plt.savefig("grafico_barras.png", bbox_inches='tight',pad_inches=0.1)
```



Agrupando o dataframe por state e shape e somando para poder empilhar os dados e mostrar no gráfico.



Utilizando a biblioteca zipcode. E coletando os dados que a biblioteca possui e atribuindo este valor para a variável zipcode_json e depois transformando a resposta em json para um dataframe para facilitar a análise de dados.

```
1 #forma 2 de filtrar as cidades e estados dos EUA e agregar a latitude e longitude
2 #instalando a biblioteca zipcode
3 #pip install zipcodes
4 #importando a biblioteca zipcodes
5 import zipcodes
6 #listando o conteúdo da biblioteca zipcodes que está em formato json
7 zipcodes_json = zipcodes.list_all()
8 #transformando em um Dataframe a biblioteca zipcodes
9 df_zipcodes = pd.DataFrame(zipcodes_json)
10 df_zipcodes
```

Collecting zipcodes
 Downloading <https://files.pythonhosted.org/packages/df/33/da326d6d915c1ac7ca8c3232b0a4ff6c8c60a326012a978199455c437f31/zipcodes-1.1.2-py2.py3-none-any.whl> (717kB)
 727kB 3.4MB/s
 Installing collected packages: zipcodes
 Successfully installed zipcodes-1.1.2

	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
...
42627	99926	PO BOX	True	Mellakalia	[]	[]	AK	Prince of Wales-Outer Ketchikan Borough	America/Mellakalia	[907]	NA	US	55.1450	-131.5439
42628	99927	PO BOX	True	Point Baker	[]	[]	AK	Prince of Wales-Hyder Census Area	America/Sitka	[907]	NA	US	56.1513	-133.3490
42629	99928	PO BOX	True	Ward Cove	[]	[]	AK	Ketchikan Gateway Borough	America/Sitka	[907]	NA	US	55.4104	-131.7237
42630	99929	PO BOX	True	Wrangell	[]	[]	AK	Wrangell City and Borough	America/Sitka	[907]	NA	US	56.1800	-132.0304

Filtrando as colunas do dataframe zipcode que foi criado para apenas as colunas que iremos utilizar para a análise

```
[ ] 1 #Filtrando multiplas colunas
    2 df_zipcodes = df_zipcodes[['zip_code', 'city', 'state', 'lat', 'long']]
    3 df_zipcodes
```

↗

	zip_code	city	state	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

Removendo valores duplicados nas colunas city e state do dataframe df_zipcode.

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

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyError: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/10min/5min.html#setting-with-copy>
This is separate from the ipykernel package so we can avoid doing import

	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

Lendo o arquivo ovnis.csv e atribuindo a variável df_ovnis. E no bloco seguinte realizando a consulta da query, que irá filtrar os dados agrupados por state e city.

```
[ ] 1 df_ovnis = pd.read_csv("OVNIS.csv")
```

```
[ ] 1 # Executa o seu comando SQL e retorna um dataframe
2 query = '''
3 | SELECT State, city,Count(*) as Views FROM df_ovnis group by State,city order by Views desc
4 | '''
5 df_filtrado = pandasql.sqldf(query.lower(), locals())
6 df_filtrado.to_csv('filtrado_mapa.csv',index=False)
7 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
...
20565	YT	Richards Bay (KwaZulu-Natal)(South Africa)	1
20566	YT	Teslin (Canada)	1
20567	YT	Watson Lake (Canada)	1
20568	YT	Yukon City (Canada)	1

Transformando o nome das colunas para letras minúsculas para que sejam padronizadas para que depois possa se juntar como dataframe do zipcodes

```
1 # Transformando as colunas do df_filtrando em minúsculas
2 df_filtrado.columns = map(str.lower, df_filtrado.columns)
3 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
...
20565	YT	Richards Bay (KwaZulu-Natal)(South Africa)	1
20566	YT	Teslin (Canada)	1
20567	YT	Watson Lake (Canada)	1
20568	YT	Yukon City (Canada)	1
20569	YT	Yukon Territory (location undisclosed) (Canada)	1

20570 rows × 3 columns

Transformando as colunas para letras minúsculas no dataframe df_zipcode.

```
1 #Colocando as colunas do df_zipcodes em letras minusculas
2 df_zipcodes.columns = map(str.lower,df_zipcodes.columns)
3 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 x 5 columns

```
1 #Transformando Os dados do df_filtrado na coluna city em letras minusculas
2 df_filtrado["city"] = df_filtrado["city"].str.lower()
3 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
...
20565	YT	richards bay (kwazulu-natal)(south africa)	1
20566	YT	teslin (canada)	1
20567	YT	watson lake (canada)	1
20568	YT	yukon city (canada)	1
20569	YT	yukon territory (location undisclosed) (canada)	1

20570 rows x 3 columns


```

1 #Transformando Os dados do df_zipcodes na coluna city em letras minúsculas
2 df_zipcodes["city"] = df_zipcodes["city"].str.lower()
3 df_zipcodes

```

 /usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead


See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/>

	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

```

1 #Transformando Os dados do df_zipcodes na coluna state em letras minúsculas
2 df_zipcodes["state"] = df_zipcodes["state"].str.lower()
3 df_zipcodes

```

 /usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

	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

Juntando os dataframes a partir da coluna city e state. Utilizando a função merge do pandas.

```
1 #Juntando os dataframes df_filtrado e df_zipcodes nas colunas state e city
2 df_merged = df_filtrado.merge(df_zipcodes, on=['state','city'])
3 df_merged
```

	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
...
10957	wy	recluse	1	82725	44.8203	-105.7762
10958	wy	rozet	1	82727	44.1855	-105.2337
10959	wy	saratoga	1	82331	41.4684	-106.7911
10960	wy	shawnee	1	82229	42.8910	-105.1056
10961	wy	worland	1	82401	43.9071	-108.0607

4. Considerações Finais

Nesta parte de exploração de dados, houve estudos das bibliotecas utilizadas para que se fosse possível gerar os gráficos e o mapa. No processo de desenvolvimento houve alguns empecilhos, como a execução dos gráficos e do mapa. Porém com os estudos e leitura da documentação, foi possível gerá-los

Referências

Raymond. Pandas DataFrame Plot - Bar Chart. Março de 2020. Kontext. Disponível em <<https://kontext.tech/column/code-snippets/399/pandas-dataframe-plot-bar-chart/>> Acesso em 27 de Setembro de 2020

Albon,Chris. Lower Case Column Names In Pandas Dataframe. Chris Albon. 20 de dezembro de 2017. Disponível em<https://chrisalbon.com/python/data_wrangling/pandas_lowercase_column_names/>Acesso em 01 de Outubro de 2020

Kite your programming copilot. How to make a pandas DataFrame string column lowercase in Python. Disponível em <[https://www.kite.com/python/answers/how-to-make-a-pandas-dataframe-string-column-lowercase-in-python#:~:text=Use%20str.,a%20DataFrame%20string%20column%20lowercase&text=lower\(\)%20to%20make%20all.%5B%22first_column%22%5D%20lowercase/](https://www.kite.com/python/answers/how-to-make-a-pandas-dataframe-string-column-lowercase-in-python#:~:text=Use%20str.,a%20DataFrame%20string%20column%20lowercase&text=lower()%20to%20make%20all.%5B%22first_column%22%5D%20lowercase/)> Acesso em 01 de Outubro de 2020.