SHARK ATTACKS

25.723 Rows X 24 Columns

Steps followed

Read the CSV file

import pandas as pd

sharks = pd.read_csv(r"./attacks.csv", encoding='latin-1')

"Clean" the content

- Empty Rows and Duplicates
- Date
- Sex
- Species
- Age
- Fatal
- Activity
- Type

Save it as a new file

sharks_clean = pd.read_csv(r"./sharks_clean.csv")

Data Visualisation

- Pivot Table
- Plots

-

1.0.1 Read the file and first overview

```
import pandas as pd
```

```
sharks = pd.read_csv(r"./attacks.csv", encoding='latin-1')
```

```
sharks.shape
```

```
(25723, 24)
```

```
pd.set_option("display.max_columns",24)
sharks
```

```
pd.set_option("display.max_rows",None)
```

- Unnamed: 22 and Unnamed: 23 should be removed
- Date should have a Datetime format
- Age & Species & Time have many empty cells
- Many Rows with only NaN or almost all NaN

```
sharks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25723 entries, 0 to 25722
Data columns (total 24 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Case Number             8702 non-null   object 
 1   Date                    6302 non-null   object 
 2   Year                    6300 non-null   float64
 3   Type                    6298 non-null   object 
 4   Country                 6252 non-null   object 
 5   Area                    5847 non-null   object 
 6   Location                5762 non-null   object 
 7   Activity                5758 non-null   object 
 8   Name                    6092 non-null   object 
 9   Sex                     5737 non-null   object 
 10  Age                     3471 non-null   object 
 11  Injury                  6274 non-null   object 
 12  Fatal (Y/N)             5763 non-null   object 
 13  Time                    2948 non-null   object 
 14  Species                 3464 non-null   object 
 15  Investigator or Source  6285 non-null   object 
 16  pdf                     6302 non-null   object 
 17  href formula            6301 non-null   object 
 18  href                    6302 non-null   object 
 19  Case Number.1           6302 non-null   object 
 20  Case Number.2           6302 non-null   object 
 21  original order          6309 non-null   float64
 22  Unnamed: 22             1 non-null      object 
 23  Unnamed: 23             2 non-null      object 
dtypes: float64(2), object(22)
memory usage: 4.7+ MB
```

1.0.2 We delete rows: Only NaN values / Duplicate rows/ No content in at least 3

```
sharks.dropna(how = "all", inplace = True) #we delete the rows with only NaN values
```

```
sharks.shape #we go from (25723, 24) to (8703, 24)
```

```
(8703, 24)
```

```
sharks.drop_duplicates(inplace = True) #we delete the duplicate rows
```

```
sharks.shape
```

```
(6311, 24)
```

```
##we keep only the rows that have information in at least 3 columns
sharks.dropna(thresh =3, inplace = True)
```

```
sharks.shape
```

```
(6302, 24)
```

1.0.3 We drop columns 22 & 23 (only 1 or 2 non NaN)

```
sharks.drop(['Unnamed: 22','Unnamed: 23'], axis=1, inplace=True) #axis = 1 because we are deleting columns
```

```
pd.set_option("display.max_rows",None)
```

1.0.4 Year (we delete the rows with years older than 1700)

```
sharks["Year"].fillna(float(0.0), inplace = True) #3 rows only NaN
```

```
sharks['Year'] = sharks['Year'].astype(int)
```

```
sharks.sort_values("Year",ascending=False,inplace = True) # we sort it "descending"
```

```
sharks.reset_index(drop=True,inplace = True) #we reset the index
```

```
sharks.drop(sharks.index[6159:6302],inplace = True) #we delete all the rows with year 1700 or older
```

1.0.5 Extracting Date : Case Number

```
import re
```

1.0.5.1 There is only one NaN (I replace it with the right date)

```
sharks["Case Number"].fillna("1905.09.06", inplace = True)
```

1.0.5.2 General "cleaning"

```
# str.replace("\W*[a-zA-Z]", "") --> Letters by Empty Set
# str.replace("\W00",".01") --> .00 by .01
# str.replace("[.|,]","-") --> "." and "," by "-""
# str.replace("(-)$","") --> "-" at the end of the string by empty set
```

```
sharks["Case Number"] = sharks["Case Number"].str.replace("\W*[a-zA-Z]", "").str.replace("\W00",".01").str.replace("[.|,]","-").
```

1.0.5.3 Cleaning specific cases

```
#str.replace("-014","-14")
#str.replace("-013","-13")
#str.replace("-012","-12")
#str.replace("2002-06-132","2002-06-13")
#str.replace("2002-06-131","2002-06-13")
#str.replace("1884-04-38","1884-04-28")
#str.replace("1853-94-29","1853-04-29")
```

```
sharks["Case Number"] = sharks["Case Number"].str.replace("-014","-14").str.replace("-013","-13").str.replace("-012","-12").str.r
```

1.0.5.4 Change format to_datetime

```
sharks["Case Number"] = pd.to_datetime(sharks["Case Number"])
```

```
sharks.info()
```

1.0.5.5 3 New columns: Year1, Month, Day

```
import datetime as dt
```

```
sharks["Year1"] = sharks["Case Number"].dt.year.astype(int)
```

```
sharks["Month"] = sharks["Case Number"].dt.month.astype(int)
```

```
sharks["Day"] = sharks["Case Number"].dt.day.astype(int)
```

```
sharks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6159 entries, 0 to 6158
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Case Number             6159 non-null   datetime64[ns]
 1   Date                    6159 non-null   object
 2   Year                    6159 non-null   int32
 3   Type                    6155 non-null   object
 4   Country                 6116 non-null   object
 5   Area                    5737 non-null   object
 6   Location                5663 non-null   object
 7   Activity                5639 non-null   object
 8   Name                    5954 non-null   object
 9   Sex                     5604 non-null   object
 10  Age                     3457 non-null   object
 11  Injury                  6137 non-null   object
 12  Fatal (Y/N)             5623 non-null   object
 13  Time                    2941 non-null   object
 14  Species                 3425 non-null   object
 15  Investigator or Source  6142 non-null   object
 16  pdf                     6159 non-null   object
 17  href formula            6158 non-null   object
 18  href                    6159 non-null   object
 19  Case Number.1           6159 non-null   object
 20  Case Number.2           6159 non-null   object
 21  original order          6159 non-null   float64
 22  Year1                   6159 non-null   int32
 23  Month                   6159 non-null   int32
 24  Day                     6159 non-null   int32
dtypes: datetime64[ns](1), float64(1), int32(4), object(19)
```

1.0.6 Rearrange and rename columns

```
sharks = sharks.reindex(columns = ["Case Number","Year1","Month","Day","Type","Country","Area","Location","Activity","Name","Sex
```

```
sharks.rename(columns = {'Case Number':'Date','Year1':'Year','Fatal (Y/N)':'Fatal','Investigator or Source':'Source','Sex ':'Sex'
```

```
sharks.head()
```

	Date	Year	Month	Day	Type	Country	Area	Location	Activity	Name	Sex	Age	Injury	Fatal	Species	Sou
0	2018-06-25	2018	6	25	Boating	USA	California	Oceanside, San Diego County	Paddling	Julie Wolfe	F	57	No injury to occupant, outrigger canoe and pad	N	White shark	R. Collier, GS

Rearrange Columns:

1. Case Number'
2. Year1
3. Month
4. Day
5. Type
6. Country
7. Area
8. Location
9. Activity
10. Name
11. Sex
12. Age
13. Injury
14. Fatal (Y/N)
15. Species
16: Investigator or Source

Rename Columns:

- 'Case Number':'Date'
- 'Year1':'Year'
- 'Fatal (Y/N)':'Fatal'
- 'Investigator or Source':'Source'
- 'Sex ':'Sex',
- 'Species ':'Species'

1.0.7 Sex

```
sharks["Sex"].value_counts()
```

```
M      4974
F       624
N         2
M         2
lli       1
.         1
Name: Sex, dtype: int64
```

```
sharks["Sex"].fillna("Unknown",inplace=True) # we fill NaN with "Unknown"
```

```
sharks["Sex"] = sharks["Sex"].apply(lambda x: x.strip()) #we delete the empty space at the beginning and at end
```

```
sharks["Sex"] = sharks["Sex"].str.replace("N|lli|\.","Unknown").str.replace("M","Male").str.replace("F","Female")
```

```
sharks["Sex"].value_counts()
```

```
Male       4976
Female      624
Unknown     559
Name: Sex, dtype: int64
```

1.0.8 Species

```
sharks['Species'].fillna("Unknown",inplace = True) # we fill NaN with "Unknown"
```

```
sharks['Species'] = sharks['Species'].apply(lambda x: x.strip()) #we delete the empty space at the beginning and at end
```

```
sharks['Species'] = sharks['Species'].apply(lambda x: x.title()) #first letter mayus and the res minus
```

```
sharks['Species'] = sharks['Species'].str.replace('"','').str.replace(".*(White).*","White Shark").str.replace(".*(Involvement).*
```

```
sharks['Species'] = sharks['Species'].str.replace(".*(Bull).*","Bull Shark").str.replace(".*(Tiger).*","Tiger Shark").str.replace
```

```
sharks['Species'] = sharks['Species'].str.replace(".*(Black).*","Blacktip").str.replace(".*(Mako).*","Mako Shark").str.replace("
```

- str.replace("","")
- str.replace(".*(White).*","White Shark")
- str.replace(".*(Involvement).*","No Shark Involved")
- str.replace(".*(Bull).*","Bull Shark")
- str.replace(".*(Tiger).*","Tiger Shark")
- str.replace(".*(Ragged).*","Ragged Tooth Shark")
- str.replace(".*(Hammer).*","Hammer Head Shark")
- str.replace(".*(Black).*","Blacktip")
- str.replace(".*(Mako).*","Mako Shark")
- str.replace(".*(Lemon).*","Lemon Shark").
- str.replace(".*(Nurse).*","Nurse Shark").
- str.replace(".*(Small).*","Small Shark")
- str.replace(".*(Questionable).*","Questionable Incident")
- str.replace(".*(Wobbegong).*","Wobbegong Shark")
- str.replace(".*(Bronze).*","Bronze Whaler Shark")
- str.replace(".*(Blue).*","Blue")

```
sharks['Species'].value_counts().nlargest(20)
```

```
Unknown                 2735
White Shark              662
No Shark Involved        330
Tiger Shark              251
Bull Shark               179
Blacktip                 106
Invalid                  101
Nurse Shark               93
Questionable Incident     75
Small Shark               66
Bronze Whaler Shark       64
Mako Shark                55
Blue                      51
Wobbegong Shark           50
Hammer Head Shark         46
Ragged Tooth Shark        43
4' Shark                  40
6' Shark                  39
Lemon Shark               35
1.8 M [6'] Shark          35
Name: Species, dtype: int64
```

```
species = sharks["Species"].value_counts().nlargest(16).index.to_list() #we create a list with the 16 more common species
```

```
sharks["Species"] = sharks["Species"].apply(lambda x: "Others" if (x not in species) else x)
```

```
sharks = sharks.drop(sharks[sharks.Species.isin(['No Shark Involved'])].index) #we delete the rows where no shark was involved
```

```
sharks.shape
```

```
(5829, 16)
```

1.0.9 Age

```
sharks["Age"].value_counts()
```

```
18                147
17                146
19                139
15                136
20                135
                 ...
25 or 28            1
21, 34,24 & 35      1
30 & 32             1
72                  1
9 & 12              1
Name: Age, Length: 148, dtype: int64
```

```
#we guess nobody might have been older than 109

Edad = [str(i) for i in list(range(1, 110))]
```

```
#we replace NaN by 0
#(we do it because otherwise the strip function doesn't work)

sharks["Age"].fillna("0",inplace=True)
```

```
#we delete the empty space at the beginning and at end

sharks["Age"] = sharks["Age"].apply(lambda x: x.strip())
```

```
#we substitute the cells that don't have an int between 1 and 109 by an empty sell

sharks["Age"] = sharks["Age"].apply(lambda x: np.NaN if x not in Edad else int(x))
```

```
# I don't understand why the type is float

sharks["Age"].value_counts()
```

```
18.0    147
17.0    146
19.0    139
20.0    136
15.0    136
       ...
87.0      1
84.0      1
75.0      1
86.0      1
81.0      1
Name: Age, Length: 79, dtype: int64
```

```
#we fill the NaN values with the average of the rest

sharks["Age"].fillna(sharks["Age"].mean(),inplace=True)
```

```
#we change the type to int (as it didn't work before)

sharks["Age"] = sharks["Age"].astype(int)
```

1.0.10 Fatal

```
sharks['Fatal'].fillna("UNKNOWN",inplace = True)
```

```
##we delete the empty space at the beginning and at end
sharks["Fatal"] = sharks["Fatal"].apply(lambda x: x.strip())
```

```
sharks["Fatal"].value_counts()
```

```
N          4222
Y          1316
UNKNOWN     288
M             1
y             1
2017          1
Name: Fatal, dtype: int64
```

```
sharks['Fatal'] = sharks['Fatal'].str.replace("2017","UNKNOWN").str.replace("^[Yy]","Yes").str.replace("^[NnM]","No")
```

```
sharks["Fatal"] = sharks["Fatal"].apply(lambda x: x.title() if (x == "UNKNOWN") else x) #we apply title function to "UNKNOWN"
```

```
sharks["Fatal"].value_counts()
```

```
No         4223
Yes        1317
Unknown     289
Name: Fatal, dtype: int64
```

1.0.11 Activity

```
sharks['Activity'].fillna("Others",inplace = True)
```

```
sharks['Activity'] = sharks['Activity'].apply(lambda x: x.title())
```

```
sharks["Activity"].value_counts()
```

```
Surfing                                                    945
Swimming                                                   792
Others                                                     437
Fishing                                                    410
Spearfishing                                               323
                                                          ...
Fishing, Had Just Speared A Ulua                             1
Free Diving For Abalone (Submerged)                          1
Fishing With Hand Line Tied To Wrist & Was Pulled Into The Water    1
Commercial Salvage Diving                                    1
Murdered By Thai Pirates                                     1
Name: Activity, Length: 1393, dtype: int64
```

```
#We delete the empty space at the beginning and at end
sharks['Activity'] = sharks['Activity'].str.replace("\s*\s$","")
```

```
#we check the 40 most common activities and sort them by name to check if there are any "duplicates"
sharks["Activity"].value_counts(normalize=True).nlargest(40).sort_index()
```

```
Bathing                 0.026420
Body Boarding           0.011151
Body Surfing            0.008063
Boogie Boarding         0.007205
Canoeing                0.002230
Diving                  0.019386
Diving For Abalone      0.001372
Diving For Trochus      0.001372
Fell Into The Water     0.001372
Fell Overboard          0.005147
Fishing                 0.072568
Fishing For Sharks      0.001887
Floating                0.002402
Free Diving             0.004804
Freediving              0.001716
Freedom Swimming        0.001544
Kayak Fishing           0.002402
Kayaking                0.005833
Kite Surfing            0.001887
Others                  0.074970
Paddle Boarding         0.001544
Pearl Diving            0.004632
Playing                 0.001372
Rowing                  0.002402
Scuba Diving            0.013381
Sea Disaster            0.002230
Shark Fishing           0.002745
Sitting On Surfboard    0.001544
Snorkeling              0.014925
Spearfishing            0.055927
Standing                0.016812
Surf Fishing            0.001716
Surf Skiing             0.003088
Surf-Skiing             0.002059
```

```
#Here we could have used open refine
sharks['Activity'] = sharks['Activity'].str.replace("Surf-Skiing","Surf Skiing").str.replace("Freediving","Free Diving")
```

```
# we create a list with the 40 most common
activities = sharks["Activity"].value_counts().nlargest(40).index.to_list()
```

```
# if the information of a cell is not in activities we replace it by "Others"
sharks["Activity"] = sharks["Activity"].apply(lambda x: "Others" if x not in activities else x)
```

```
sharks["Activity"].value_counts(normalize=True)
```

```
Others                0.353062
Surfing               0.162978
Swimming              0.143249
Fishing               0.072568
Spearfishing          0.055927
Bathing               0.026420
Wading                0.023846
Diving                0.019386
Standing              0.016812
Snorkeling            0.014925
Scuba Diving          0.013381
```

1.0.12 Country

```
sharks['Country'].fillna("Unknown",inplace = True)
```

```
sharks["Country"].value_counts()
```

```
USA                  2081
AUSTRALIA            1246
SOUTH AFRICA          535
PAPUA NEW GUINEA      129
NEW ZEALAND           121
                     ...
MALDIVE ISLANDS         1
ANDAMAN ISLANDS         1
ANTIGUA                 1
CAPE VERDE              1
OCEAN                   1
Name: Country, Length: 203, dtype: int64
```

```
sharks['Country'] = sharks['Country'].apply(lambda x: x.strip())
```

```
sharks['Country'] = sharks['Country'].apply(lambda x: x.title())
```

```
#we replace the "elements with 2 spaces or more by one space"
sharks['Country'] = sharks['Country'].str.replace("\s{2,}"," ")
```

```
sharks["Country"].value_counts().nlargest(40).sort_index()
```

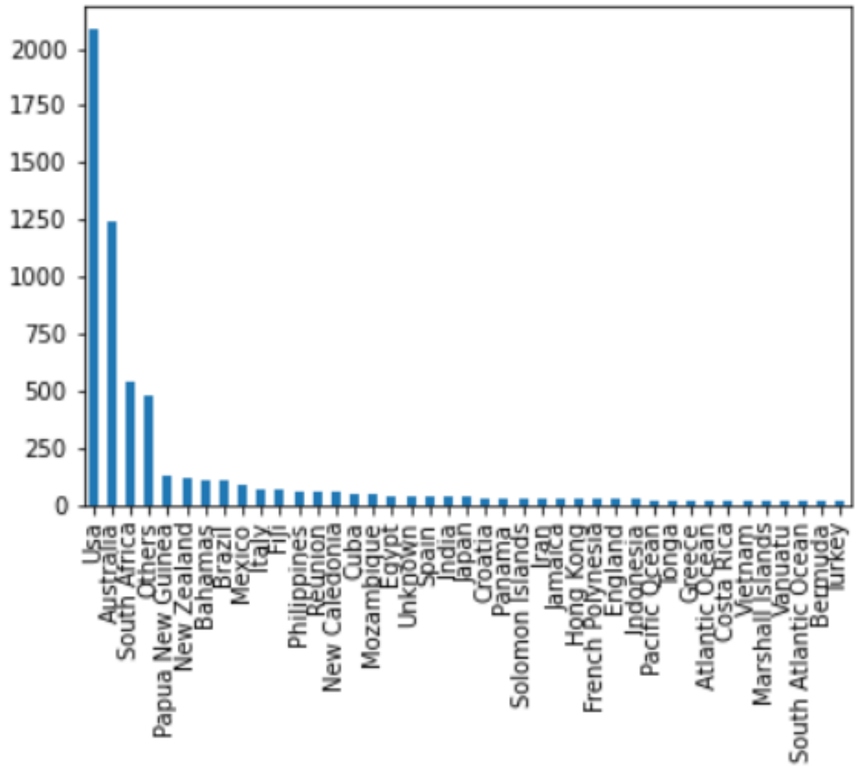
```
Atlantic Ocean           17
Australia              1246
Bahamas                 104
Bermuda                  12
```

```
# 40 most common countries
countries = sharks["Country"].value_counts().nlargest(40).index.to_list()
```

```
sharks["Country"] = sharks["Country"].apply(lambda x: "Others" if x not in countries else x)
```

```
#Here we can easily tell the countries with more accidents: USA and Australia
sharks["Country"].value_counts().plot(kind = "bar")
```

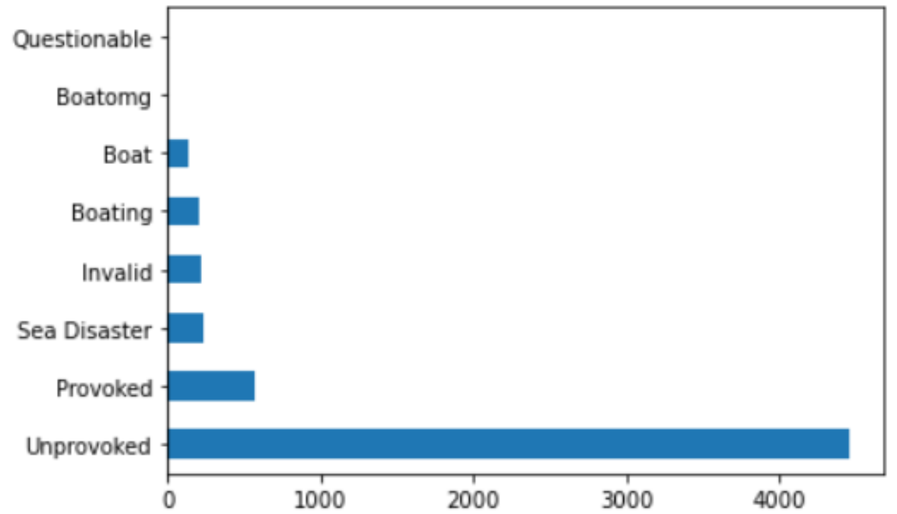
```
<AxesSubplot:>
```



1.0.13 Type

:
```
sharks["Type"].value_counts().plot(kind = "barh")
```

:
```
<AxesSubplot:>
```



:
```
sharks["Type"] = sharks["Type"].str.replace("Boatomg","Boating")
```

1.0.14 Delete Columns not Cleaned

```
sharks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5829 entries, 0 to 6158
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      5829 non-null   datetime64[ns]
 1   Year      5829 non-null   int32
 2   Month     5829 non-null   int32
 3   Day       5829 non-null   int32
 4   Type      5825 non-null   object
 5   Country   5829 non-null   object
 6   Area      5436 non-null   object
 7   Location  5362 non-null   object
 8   Activity  5829 non-null   object
 9   Name      5662 non-null   object
 10  Sex       5829 non-null   object
 11  Age       5829 non-null   int32
 12  Injury    5809 non-null   object
 13  Fatal     5829 non-null   object
 14  Species   5829 non-null   object
 15  Source    5815 non-null   object
dtypes: datetime64[ns](1), int32(4), object(11)
memory usage: 683.1+ KB
```

```
sharks.drop(['Source','Injury','Area','Location','Name'], axis=1, inplace=True)
```

```
sharks.head()
```

	Date	Year	Month	Day	Type	Country	Activity	Sex	Age	Fatal	Species
0	2018-06-25	2018	6	25	Boating	Usa	Others	Female	57	No	White Shark

1.0.15 We save the document to a new one called "sharks_clean"

```
sharks.to_csv("sharks_clean.csv", index = False)
```

1.0.16 Data Visualisation

1.0.16.1 Pivot Table

```
sharks_clean = pd.read_csv(r"./sharks_clean.csv")
```

```
speciesfatal = sharks_clean.pivot_table(index = "Species",values = "Activity",columns="Fatal",aggfunc="count",fill_value = 0)
```

```
speciesfatal.sort_values("Yes", ascending = False)
```

Fatal	No	Unknown	Yes
Species			
Unknown	1753	71	911
White Shark	499	13	150
Others	1094	24	134
Tiger Shark	182	4	65
Bull Shark	141	0	38
Blue	42	1	8
Bronze Whaler Shark	59	1	4
Mako Shark	53	0	2
Questionable Incident	2	71	2
Small Shark	63	1	2
Hammer Head Shark	45	0	1

Summary

Methods used

- dropna, drop_duplicates, fillna
- sort_values, value_counts(), nlargest()
- str.replace
- regex
- to_datetime
- apply()
- x.strip()
- x.title()
- lambda
- isin()
- plot
- pivot_table

Lessons Learned

- I should have spent more time analysing the data before starting to clean it (Ex: Case Number easier to clean than Date)
- I should have used Regex from the beginning
- I should have used less "replace" (next time I should use find)
- I should practice regex more
- I should have used group by, and more pvot tables, melt