</talentlabs>

Data Wrangling Project: Sharks Attack Dataset

Instructions

Welcome to the final project of this data wrangling module! In this project, you will get a chance to work through the entire data wrangling workflow while preparing the shark_attacks.csv file for analysis. This dataset contains very dirty data and will require a lot of work! This project is broken down into key steps of the data wrangling process to help guide you along the process. When you are finished, save the wrangled dataset as a final_project.csv file. Submit the final project as a zip folder named final_project.zip. Make sure the zipped folder has both your wrangled dataset and this word document within it. Best of luck!

Step 1: Decide which tool to use

This dataset contains around 1100 rows. Discuss which tool (BigQuery/Python/Google Sheets) is best suited for the data cleaning task for this dataset. Mention the relevant advantages and disadvantages of each tool. Finally, state which tool you think is best suited for the task and why. (6 marks)

Python and **Google Sheets** are best suited for the data cleaning task for this dataset. Below are their advantages and disadvantages:

	Google sheets	**Python**
Best for	Quick data exploration, simple calculations, basic data visualization, and small to medium-sized datasets.	Advanced data cleaning or data manipulation and working with large datasets.
Advantages	User-friendly interface, no programming required, and suitable for non-technical users.	Versatile, extensive libraries (e.g., Pandas, NumPy), automation, and reproducibility.
Limitations	Limited scalability, not ideal for complex data transformations or large datasets.	less user-friendly interface to monitor all the data in one table sheet

Why these tools are best suited in this task:

Tool	**Reason to use**
Python	Mainly use python for data cleaning and data wrangling, because: - It's reproducable and easy to navigate in form of jupyter notebook. - It's easy to clean and manipulate complex data, for example, replacing the comma symbol at end of sentence using regex pattern and replacing null values of designated columns, etc. Sometimes use python for data ingestion together with Google sheet, as it'll be easier to spot the errors in datasets with these two combinations.
Google sheet	Use Google sheet for data ingestion and data cleaning validation, because: - It's easy to navigate and investigate every data row with its user-

	friendly interface.

Step 2: Data Inspection

Inspect the dataset. In the box below, discuss the following:

- Are there any irrelevant columns? Which ones?
- Are there any duplicates?
- Which columns have missing data?
- For each column with missing data, describe what you think the best way to handle that missing data is, and why?
- Are there any errors? Describe any you find.
- Is there anything else that requires data cleaning attention?

(12 marks)

(i) Are there any irrelevant columns? Which ones?

The columns: “Case Number”, “Year”, “Investigator or Source”, “pdf”, “href formula”, “href”, “Case Number.1” and “Case Number.2” are irrelevant, because they are redundant as they duplicate information already present in other columns."

(ii) Are there any duplicates?

Yes, there are 2 rows duplicated.

	Date	Type	Country	Area	Location	Activity	Sex	Age	Fatal (Y/N)	Time	Species	original order
662	2012-04-03	Unprovoked	USA	Hawaii	Leftovers near Chun's Reef, Oahu	Surfing	M	28	N	12h38	Tiger shark, 10'	5434
958	2009-08-29	Unprovoked	SOUTH AFRICA	Western Cape Province	Glentana	Surfing	M	25	Y	15h30	White shark	5139

(iii) Which columns have missing data?

As reference to the image below, the columns: “Type”, “Country”, “Area”, “Location”, “Activity”, “Sex”, “Age”, “Fatal (Y/N)”, “Time” and “Species” have missing data.

	Missing	Missing (%)
Date	0.0	0.0000
Type	2.0	0.0017
Country	1.0	0.0009
Area	56.0	0.0484
Location	59.0	0.0510
Activity	52.0	0.0449
Sex	43.0	0.0372
Age	289.0	0.2498
Fatal (Y/N)	7.0	0.0061
Time	328.0	0.2835
Species	447.0	0.3863
original order	0.0	0.0000

(iv) For each column with missing data, describe what you think the best way to handle that missing data is, and why?

	Missing	Missing (%)
Age	289.0	0.2498
Time	328.0	0.2835
Species	447.0	0.3863

For the columns with a substantial number of missing values, for example exceeding 25% of total rows, we could consider dropping these columns if those columns are not critical for analysis.

However, I decide to flag those null values as either “Unknown” or “Other” values, because removing those columns would result in a loss of potentially valuable information. This same approach has been applied to other columns with fewer missing values, to preserve data integrity and maximize the insights that can be retrieved from the dataset.

(v) Are there any errors? Describe any you find.

Inconsistent date entry: some of the date entries are not properly formatted as “YYYY-MM-DD”. For example, there are wrong date entries like “Reported 07-Jun-2017”, “16-Aug—2021”, “11-Aug—2011”, “190Feb-2010”, “Late Jul-2008”.

```
1  df3[df3["Date"].str.contains("Reported")]
```

✓ 0.0s Python

	Date	Type	Country	Area	Location	Activity	Sex	Age	Fatal (Y/N)	Time	Species	original order
3	Reported 07-Jun-2017	Unprovoked	UNITED KINGDOM	South Devon	Bantham Beach	Surfing	M	30	N	Unknown	3m shark, probably a smooth hound	6092
10	Reported 06-May-2017	Provoked	AUSTRALIA	Queensland	Weipa	Attempting to lasso a shark	M	29	N	Unknown	9' shark	6085
37	Reported 09-Mar-2017	Unprovoked	BAHAMAS	Great Exuma	Other	Washing hands	M	58	N	Unknown	Lemon shark	6058
52	Reported 08-Jan-2017	Invalid	AUSTRALIA	Queensland	Other	Spearfishing	M	35	Unknown	Unknown	Bull shark	6043
113	Reported 14-Jul-2016	Unprovoked	BAHAMAS	Other	Tiger Beach	Scuba Diving	M	Unknown	N	Unknown	Lemon shark, 9'	5982
...	...	...	...	...	...	...	...	...	...	...	...	...
1133	Reported 19-Apr-2008	Invalid	SOUTH AFRICA	KwaZulu-Natal	Aliwal Shoal	Free-diving	M	Unknown	N	Unknown	Tiger shark, 13' female	4964
1137	Reported 09-Apr-2008	Unprovoked	FIJI	Other	Other	Spearfishing	M	Unknown	N	Unknown	Other	4960
1139	Reported 08-Apr-2008	Unprovoked	USA	Florida	1.4 miles south of Ponce de Leon Jetty, New Sm...	Surfing	Unknown	Unknown	N	Unknown	Other	4958
1150	Reported 21-Feb-2008	Unprovoked	FRENCH POLYNESIA	Society Islands	Tahiti	Spearfishing	M	26	N	Unknown	Other	4947
1157	Reported 19-Jan-2008	Invalid	NEW ZEALAND	South Island	Marfells Beach	Wading	M	Unknown	N	Unknown	No shark involvement	4940

	Date	Type	Country	Area	Location	Activity	Sex	Age	Fatal (Y/N)	Time	Species	original order
746	16-Aug--2011	Unprovoked	USA	Puerto Rico	Vieques	Floating	M	27.0	N	Night	Other	5350
749	11-Aug--2011	Unprovoked	USA	North Carolina	Beaufort Inlet	Swimming	M	54.0	N	14h00	Other	5347
897	190Feb-2010	Unprovoked	NEW ZEALAND	South Island	Tahunanui Beach	Swimming	M	29.0	N	Night	Possibly a blue shark	5199
1101	Late Jul-2008	Boat	UNITED KINGDOM	Sussex	Rock-a-Nore, Hastings	Rowing an inflatable dinghy	M	16.0	N	Unknown	Starry smoothhound shark, 1m	4996

Data type error: the “Fatal (Y/N)” column should not contain values other than “Y”, “N”, and “Unknown”.

	Date	Type	Country	Area	Location	Activity	Sex	Age	Fatal (Y/N)	Time	Species	original order
646	2012-06-10	Provoked	ITALY	Sardinia	Muravera	Attempting to rescue an injured & beached shark	M	57.0	2017	Morning	Blue shark, 2.5m	5449

Multiple error type in "Age" column: In the "Age" column, the examples of wrong entries are "40s" or "50s", "28 & 26", "18 months", "Teen" or "teen".

	Age
17	20s
47	Teen
84	60s
158	Teen
305	18 months
328	40s
362	20s
369	30s
382	50s
386	Teen
428	teen
429	teen
513	teen
558	28 & 26
603	30s
742	20s
844	Teen

Multiple error type in Time column:

Time
Shortly before 12h00
Morning
Afternoon
9h00
After noon
Late afternoon
1300
Midnight
Evening
Night
Sometime between 06h00 & 08hoo
Early afternoon
830
Just before noon
1600
Early morning
Dawn
Midday
AM
A.M.
Dusk
Lunchtime
15j45
500
Before 07h00

Spelling error: the "Area" column should not contain the comma at the end of phrase, and there should be no spaces at the start or end of phrase.

	Date	Type	Country	Area	Location	Activity	Sex	Age	Fatal (Y/N)	Time	Species	original order
9	2017-05-12	Unprovoked	UNITED ARAB EMIRATES	Sharjah,	Khor Fakkan	Spearfishing	M	41.0	N	Morning	Other	6086
1061	2008-10-06	Unprovoked	CROATIA	Split-Dalmatia Count,	Smokvina Bay, Vis Island	Spearfishing	M	43.0	N	12h00	5 m white shark	5036

Step 3: Data Cleaning

Following on from Step 2, clean the dataset. Document all the changes you make in the box below. Before data cleaning, make sure to check every column thoroughly (audit the data). List all the actions to take so that you don't overlook anything. (12 marks)

The data is cleaned using python with ***pandas*** and ***numpy*** library. Below is the code:

First, read the shark_attacks csv file using ***pandas.read_csv()*** method:

```
1  import pandas
2  import numpy
3
4  df = pandas.read_csv("shark_attacks.csv")
5  df.head(3)
```

✓ 1.7s Python

	Case Number	Date	Year	Type	Country	Area	Location	Activity	Name	Sex	...	Fatal (Y/N)	Time	Species	Investigator or Source	pdf	
0	2017.06.11	2017-06-11	2017.0	Unprovoked	AUSTRALIA	Western Australia	Point Casuarina, Bunbury	Body boarding	Paul Goff	M	...	N	08h30	White shark, 4 m	WA Today, 6/11/2017	2017.06.11-Goff.pdf	http://sharkattackf
1	2017.06.10.b	2017-06-10	2017.0	Unprovoked	AUSTRALIA	Victoria	Flinders, Mornington Penisula	Surfing	female	F	...	N	15h45	7 gill shark	NaN	2017.06.10.b-Flinders.pdf	http://sharkattackf
2	2017.06.10.a	2017-06-10	2017.0	Unprovoked	USA	Florida	Ponce Inlet, Volusia County	Surfing	Bryan Brock	M	...	N	10h00	NaN	Daytona Beach News-Journal, 6/10/2017	2017.06.10.a-Brock.pdf	http://sharkattackf

(i) Remove irrelevant columns, such as "Case Number", "Year", "Investigator or Source", "pdf", "href formula", "href", "Case Number.1" and "Case Number.2" with ***pandas.DataFrame.drop()*** method

```
# remove irrelevant columns
df1 = df.drop(columns=["Case Number","Year","Name", "Investigator or Source", "pdf",
                       "Injury", "href formula", "href", "Case Number.1", "Case Number.2"])
```

(ii) Check duplicated rows in a dataframe with ***DataFrame.duplicated()*** & remove duplicated rows with ***pandas.DataFrame.drop_duplicates()*** method

```
1  df1[df1.duplicated()]
```

✓ 0.0s Python

	Date	Type	Country	Area	Location	Activity	Sex	Age	Fatal (Y/N)	Time	Species	original order
662	2012-04-03	Unprovoked	USA	Hawaii	Leftovers near Chun's Reef, Oahu	Surfing	M	28	N	12h38	Tiger shark, 10'	5434
958	2009-08-29	Unprovoked	SOUTH AFRICA	Western Cape Province	Glentana	Surfing	M	25	Y	15h30	White shark	5139

```
1  # check and remove duplicated rows
2  print(f"{df1.duplicated().sum()} rows duplicated...")
3
4  # drop duplicated rows if any
5  print("duplicated rows are removed")
6  df1.drop_duplicates(inplace=True)
```

✓ 0.0s Python

```
2 rows duplicated...
duplicated rows are removed
```

(iii) For each column with missing data, replace null values with "Unknown" or "Other" value using ***pandas.DataFrame.replace()*** method. How the missing data are handled

are as below:

```
# replace null values with "Unknown" or "Other" value

print("Missing values of columns was filled with 'Unknown' or 'Other' value")
df3 = df2.copy(deep=True)
df3["Age"].fillna("Unknown", inplace=True)
df3["Species"] = df3["Species"].replace(numpy.nan, "Other")
df3["Time"] = df3["Time"].replace(numpy.nan, "Unknown")

df3["Type"] = df3["Type"].replace(numpy.nan, "Unknown")
df3["Sex"] = df3["Sex"].replace(numpy.nan, "Unknown")
df3["Country"] = df3["Country"].replace(numpy.nan, "Other")
df3["Location"] = df3["Location"].replace(numpy.nan, "Other")
df3["Area"] = df3["Area"].replace(numpy.nan, "Other")
df3["Activity"] = df3["Activity"].replace(numpy.nan, "Unknown")

df3["Fatal (Y/N)"] = df3["Fatal (Y/N)"].replace(numpy.nan, "Unknown")
df3["Fatal (Y/N)"] = df3["Fatal (Y/N)"].replace("UNKNOWN", "Unknown")
```

(iv) Check and deal with other errors in the dataframe

- **Inconsistent date entry**:
  In “Date” column, the examples of wrong date entries are “Reported 07-Jun-2017”, “16-Aug—2021”, “11-Aug—2011”, “190Feb-2010”, “Late Jul-2008”.

  Here are some ways to clean the incorrect date entries in “Date” column:
  - **For entries like “Reported 07-Jun-2017”**, replace “Reported” with empty string.
  - **For entries like “16-Aug—2021”**, replace “—” with “-”.
  - **For entries like “190Feb-2010**”, assume “190Feb-2010” is equivalent to “19-Feb-2010”.
  - **For entries like “Late Jul-2008”**, assume “Late Jul-2008” is equivalent to “30-Jul-2008”.

```
df4 = df3.copy(deep=True)
# replace
df4["Date"] = df4["Date"].str.replace("Reported ","")
df4["Date"] = df4["Date"].str.replace("--","-")
df4["Date"] = df4["Date"].str.replace("190Feb-2010","19-Feb-2010")
df4.drop(index=1101, inplace=True) # drop the wrong date entry with no day specified, i.e., Late Jul-2008
# convert data type of "Date" column to datetime type
df4["Date"] = pandas.to_datetime(df4["Date"])
df4["Date"]
```

- **Type error**:
  In the “Fatal (Y/N)” column, an incorrect value “2017” is present. This column should only contain “Y”, “N”, or “Unknown”.

  To remove entries like '2017', apply a condition to drop values other than 'Y', 'N', and 'Unknown' using the ***pandas.DataFrame.drop()*** method."

```
# Remove rows from df6 where the "Fatal (Y/N)" column contains values other than "Y", "N" and "Unknown"
df6 = df5.drop(index=df5[~ df5["Fatal (Y/N)"].isin(["Y","N", "Unknown"])].index)
```

- **Multiple errors in age column**:
  In the “Age” column, the examples of wrong entries are “40s” or “50s”, “28 & 26”, “18 month”, “Teen” or “teen”.

  Here are some ways to clean the incorrect entries in “Age” column:
  - **For entries like “40s” or “50s”**, simply remove the “s” at the end of word to convert

them into “40”and “50”.

- **For entries like “28 & 26”**, simply transform the value into 27 (=average of 28 and 26)
- **For entries like “18 month”**, convert its unit from month into year.
- **For entries like “teen” or “Teen”**, assume their age is equivalent to 15 years old.

```
# clean the age column
def clean_age_data(age):
    # Replace "s" with an empty string, if present
    ## Example of age data: 40s, 50s
    age = str(age).replace("s","")
    # Check if the age contains an "&" symbol (e.g., "28 & 26")
    if len(age.split("&")) > 1:
        age_list = [float(age) for age in age.split("&")]
        age = str(sum(age_list)/len(age_list))
    # Check if the age ends with "month" (e.g., "18 month")
    if age.endswith("month"):
        age = str(float((age.split(" "))[0])/12)
    # Check if the age is "teen" or "Teen"
    if age in ["teen", "Teen"]:
        # teen is around 13 - 17 years old
        return str((13+17)/2)
    return str(age)

df6["Age"] = df6['Age'].apply(clean_age_data)
```

- **Multiple errors in Time column**:
  In the “Time” column, the examples of wrong time entries are “830” or “1600”, “Morning” or “Afternoon”, “Shortly before 12h00” or “Sometime between 06h00 & 08hoo”, “15j45”.

- **For entries like “830” or “1600”**, replace the time values with formatted version like “8h30” and “16h00” using pandas.DataFrame.replace(regex=True)
- For entries in words like “Morning” or “Afternoon”, replace them with “Unknown” values.
- **For entries like “Shortly before 12h00” or “Sometime between 06h00 & 08hoo”**, replace the time values with a formatted version like “12h00” and “06h00” from the sentence using regex and ***pandas.DataFrame.extract()*** method.
- **For entries like “15j45”,** replace the “j” into “h” so that it becomes “15h45”.

```
df7 = df6.copy(deep=True)
# convert the time format from "1600" to "16h00" time format
df7["Time"] = df7["Time"].str.replace('(\d{1,2})(\d{2})', r'\1h\2', regex=True)
# replace wrong time format like "15j45" into "15h45"
df7["Time"] = df7["Time"].str.replace("(\d{1,2})\w(\d{2})", r"\1h\2", regex=True)
# extract the time from long sentence such as "Shortly before 12h00" and "Before 07h00"
df7["Time"] = df7["Time"].str.extract("(\d{1,2}h\d{2})")
# replace null values with "Unknown" value
df7["Time"] = df7["Time"].fillna("Unknown")
```

- **Spelling error**:
  In the "Area" column, the examples of wrong entries are “Sharjah,” and “ Split-Dalmatia Count,”.

  To clean this spelling error, just simply remove the spaces in front of the word, and

just remove the comma at end of the sentence.

```
# remove the comma inside "Area" column
df7["Area"] = df7["Area"].str.strip().str.replace(",$", "", regex=True)
```

Step 4: Data Cleaning Validation

Go through the data cleaning checklist and make sure there is no dirty data remaining! List below all the data validation steps you take. (3 marks)

1. Check if the Missing values have been filled in each column.
2. Check if the duplicated rows has been removed.
3. Ensure the date entries are correct in form of “YYYY-MM-DD”.
4. Check the “Fatal (Y/N)” column if it only contains “Y”, “N”, and “Unknown” value.
5. Check the “Age” column if it only contains digit string (e.g., “40”, “50”) and the “Unknown” value.
6. Check the “Time” column if it only contains values in time format like “12h00” or “14h00” or “Unknown”.
7. Double check if there are any spelling errors.

Step 5: Data Enrichment

With the dataset cleaned it’s time to enrich the data:

- Make an address column, by combining the Location, Area and Country columns together (this might affect your missing value strategy!).
- Add a new column, call it “Shark”. Extract information from the Species column. If the species text mentions the word “white”, make the “Shark” column value “Great White”. If the text mentions “bull”, make the “Shark” column value “Bull”. Otherwise, if neither of the words found, make the value “Other”. (Hint: make sure the species column is all lowercase).

Step 6: Publish the dataset

Export the data as csv file. Call it final_project.csv. Submit the file in a zip folder called final_project.zip. Make sure the zip folder contains both your wrangled dataset and this word document with your answers!