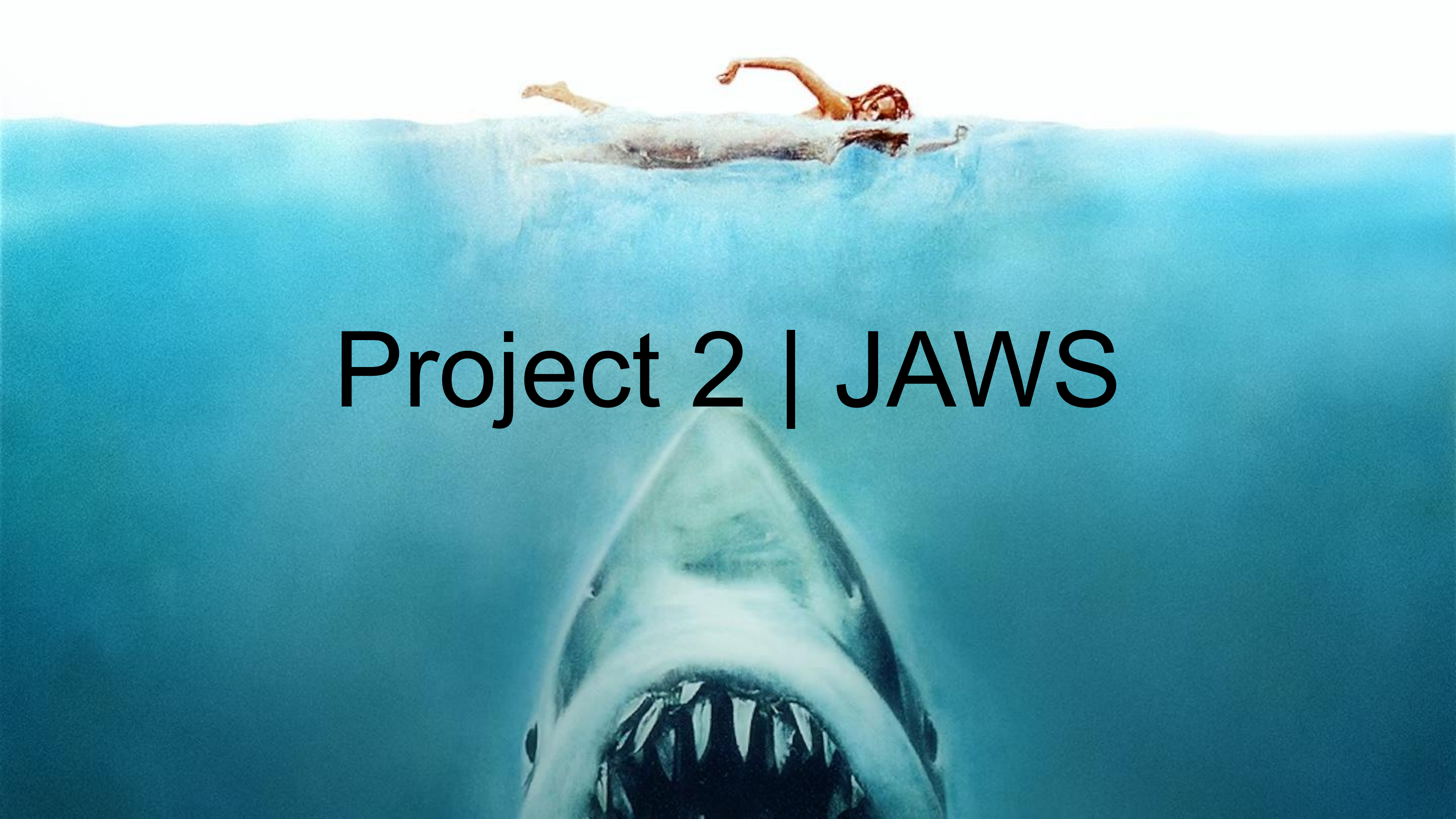
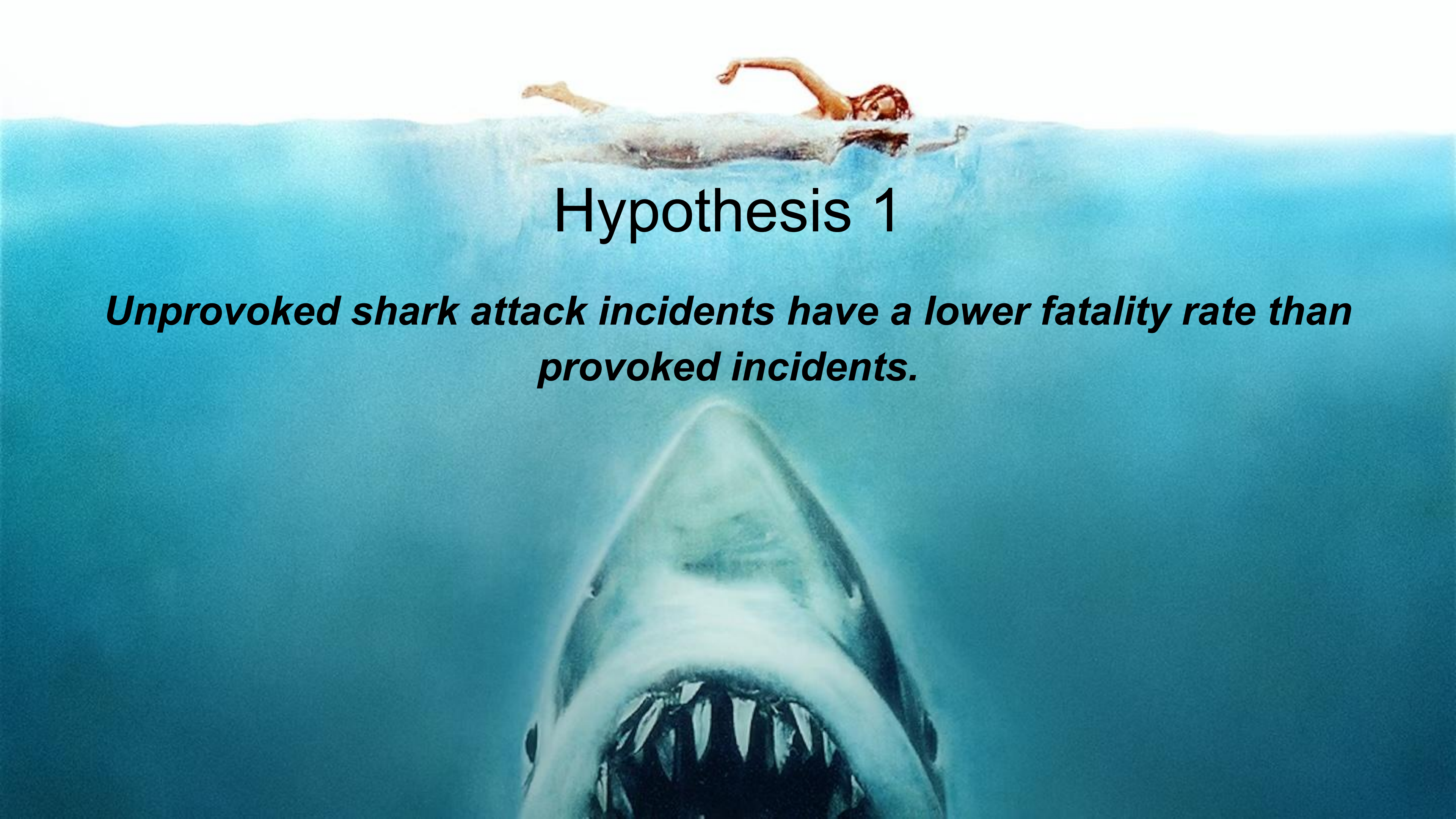
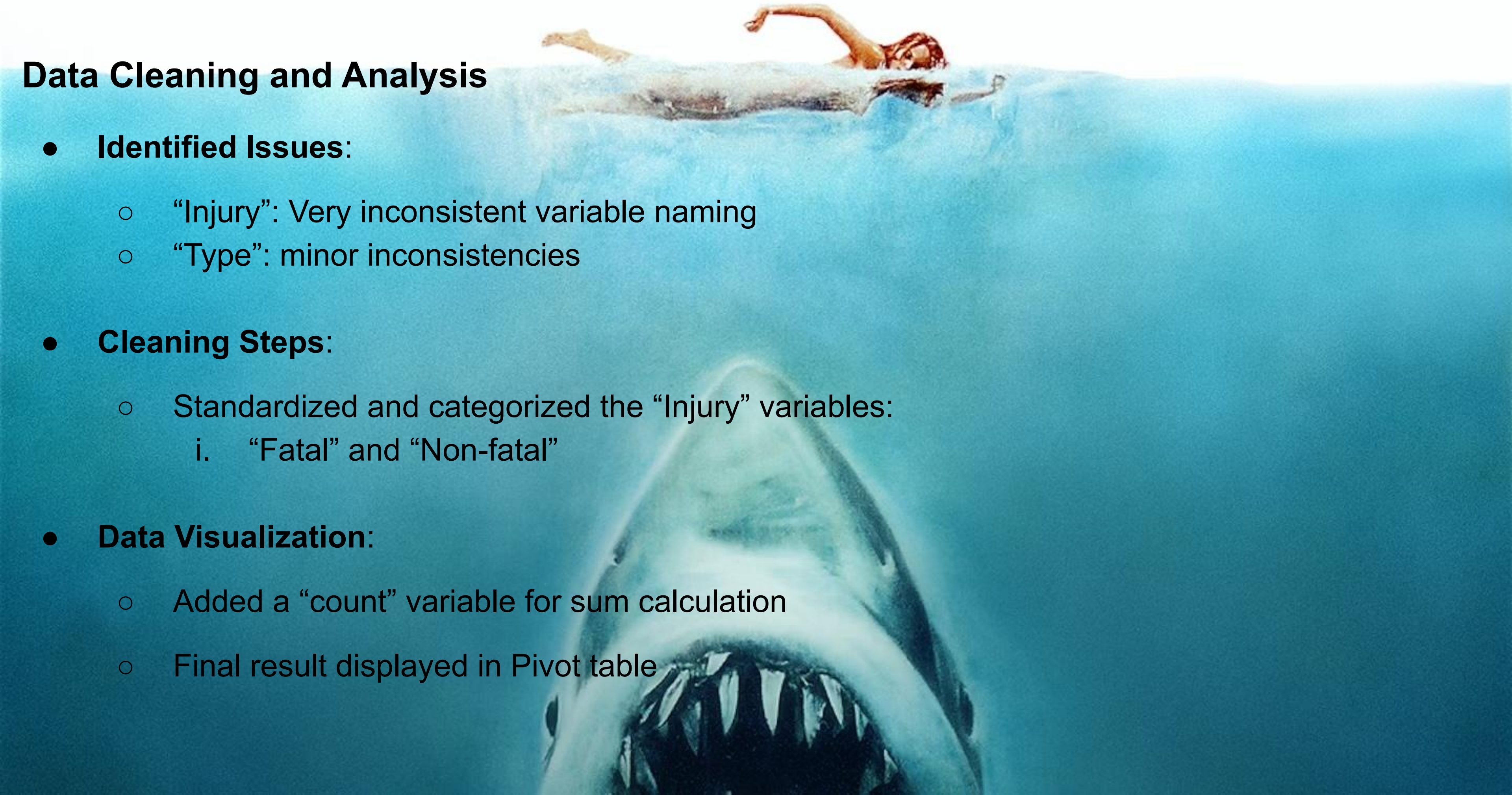


Project 2 | JAWS



Hypothesis 1

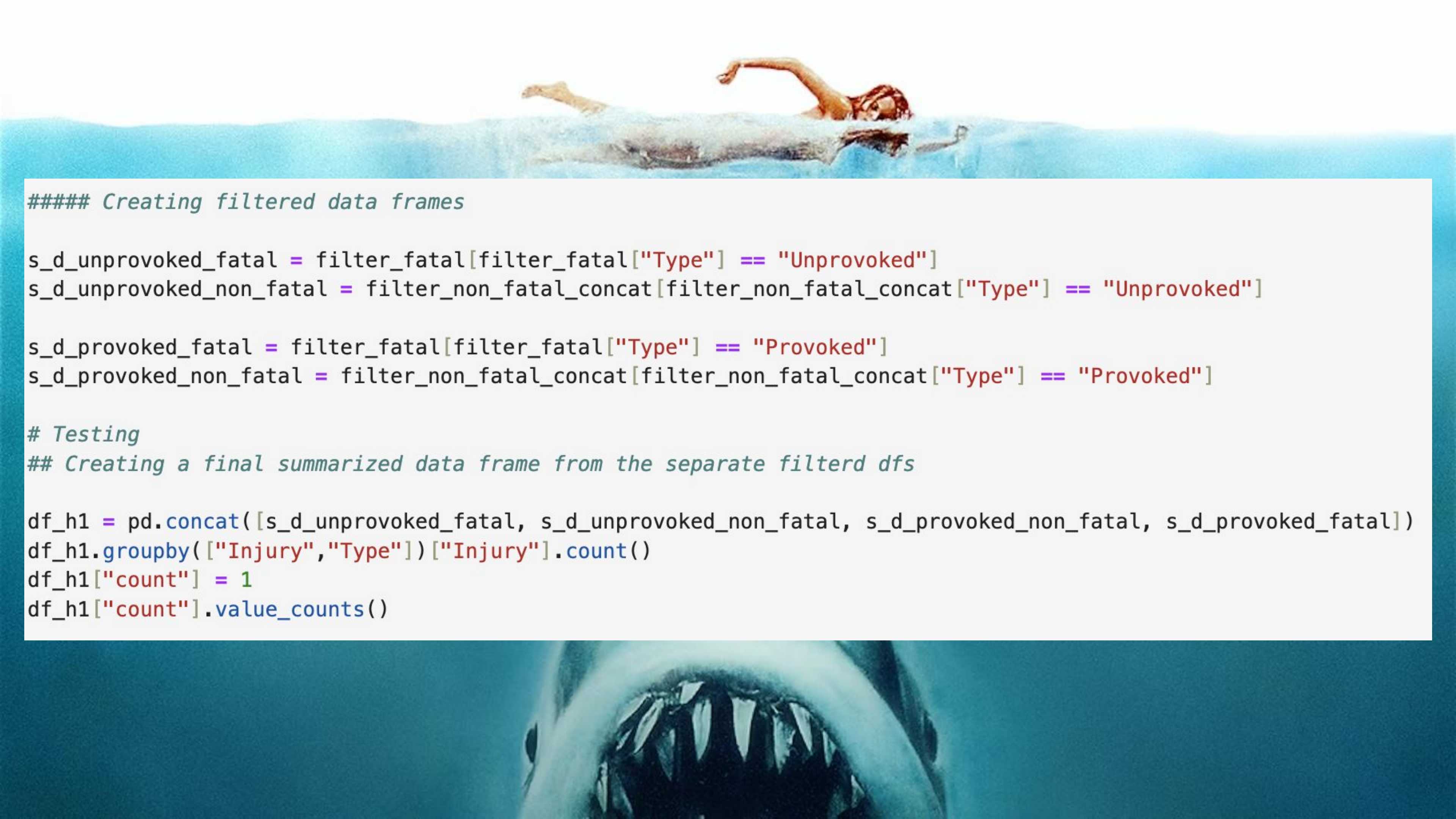
Unprovoked shark attack incidents have a lower fatality rate than provoked incidents.



Data Cleaning and Analysis

- **Identified Issues**:
    - “Injury”: Very inconsistent variable naming
    - “Type”: minor inconsistencies
- **Cleaning Steps**:
    - Standardized and categorized the “Injury” variables:
        i. “Fatal” and “Non-fatal”
- **Data Visualization**:
    - Added a “count” variable for sum calculation
    - Final result displayed in Pivot table

```python
## Creating filters to filter for "fatal" and "non-fatal"

condition1 = s_d["Injury"].str.contains("fatal") == True
condition11 = s_d["Injury"].str.contains("non-fatal") != True
condition12 = s_d["Injury"].str.contains("not confirmed") != True
condition13 = s_d["Injury"].str.contains("unconfirmed") != True

condition2 =  s_d["Injury"].str.contains("fatal") != True
condition21 =  s_d["Injury"].str.contains("unknown") != True

condition3 = s_d["Injury"].str.contains("non-fatal") == True

### Creating filtered data frames

filter_fatal = s_d[condition1 & condition11 & condition12 & condition13]
filter_non_fatal = s_d[condition2 & condition21]
filter_non_fatal2 = s_d[condition3]

s_d_unprovoked = s_d[s_d["Type"] == "Unprovoked"]
s_d_provoked = s_d[s_d["Type"] == "Provoked"]
```

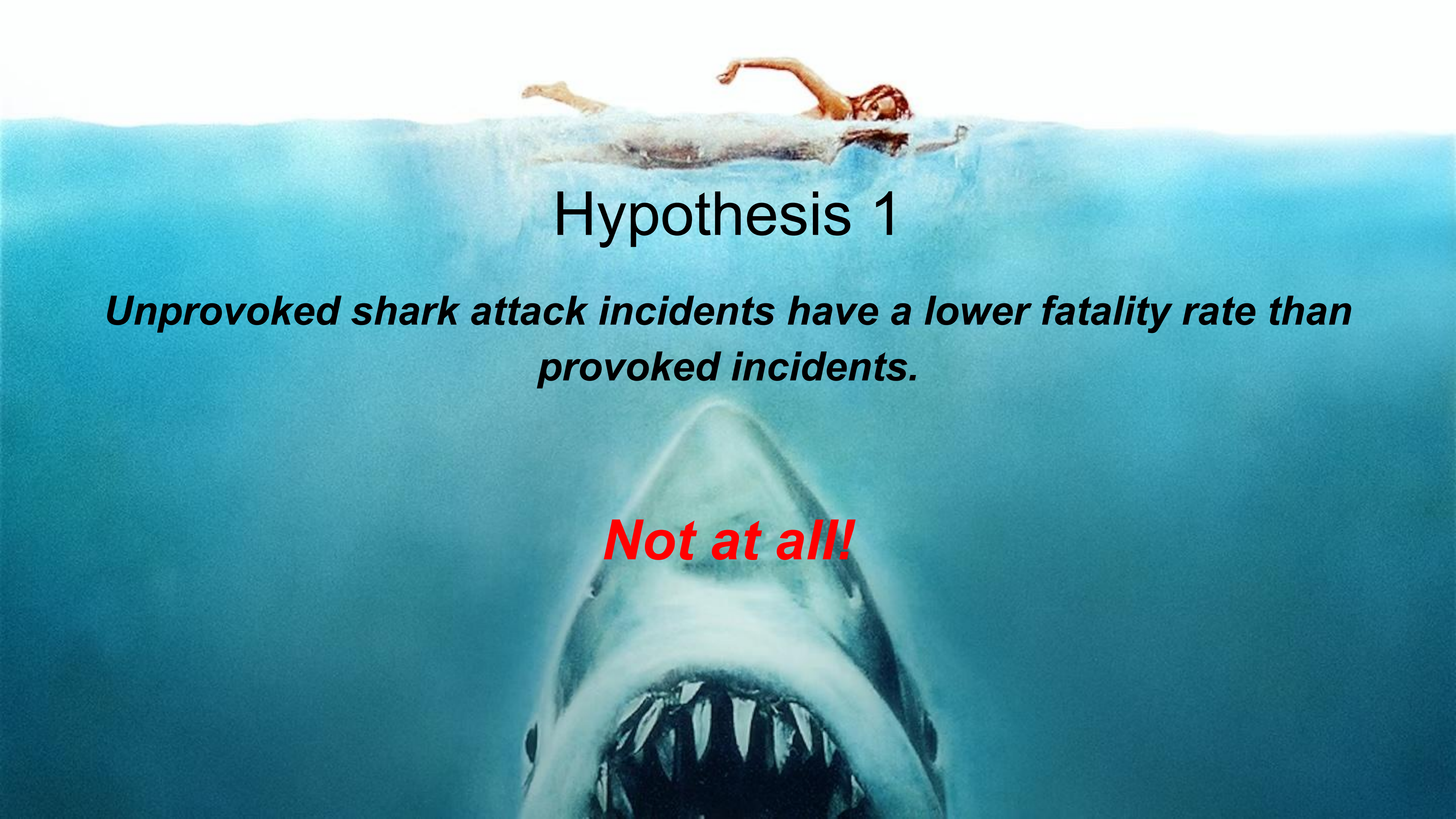


```python
##### Creating filtered data frames

s_d_unprovoked_fatal = filter_fatal[filter_fatal["Type"] == "Unprovoked"]
s_d_unprovoked_non_fatal = filter_non_fatal_concat[filter_non_fatal_concat["Type"] == "Unprovoked"]

s_d_provoked_fatal = filter_fatal[filter_fatal["Type"] == "Provoked"]
s_d_provoked_non_fatal = filter_non_fatal_concat[filter_non_fatal_concat["Type"] == "Provoked"]

# Testing
## Creating a final summarized data frame from the separate filterd dfs

df_h1 = pd.concat([s_d_unprovoked_fatal, s_d_unprovoked_non_fatal, s_d_provoked_non_fatal, s_d_provoked_fatal])
df_h1.groupby(["Injury","Type"])["Injury"].count()
df_h1["count"] = 1
df_h1["count"].value_counts()
```

Type / Injury	Provoked	Unprovoked
fatal	17	1225
non-fatal	619	3874

Fatality Rate:

Provoked:
17/(619+17) ~ 2,67%
Unprovoked:
1225/(1225+3874) ~ 24,02%



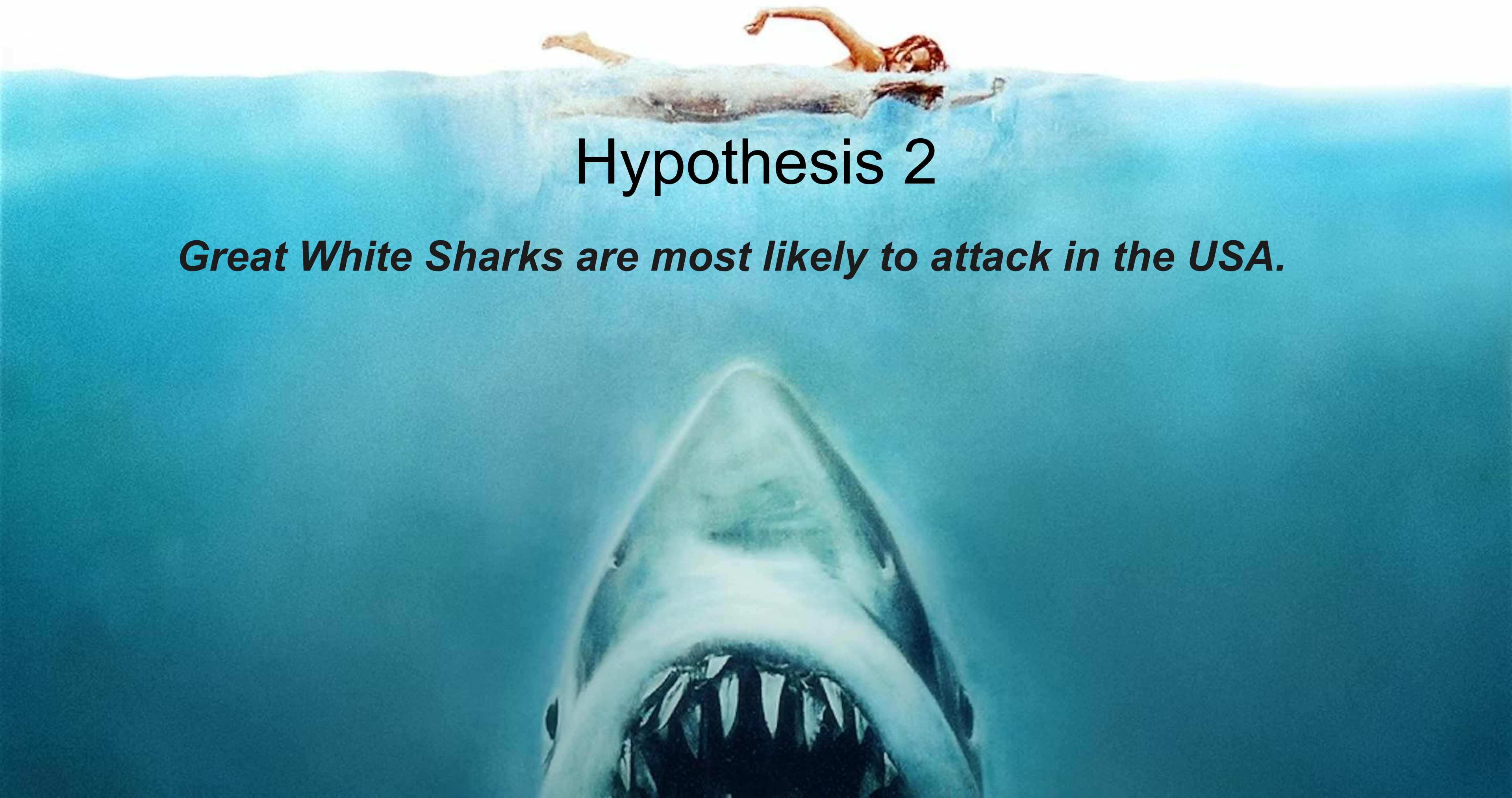
Hypothesis 1

Unprovoked shark attack incidents have a lower fatality rate than provoked incidents.

Not at all!

Hypothesis 2

Great White Sharks are most likely to attack in the USA.

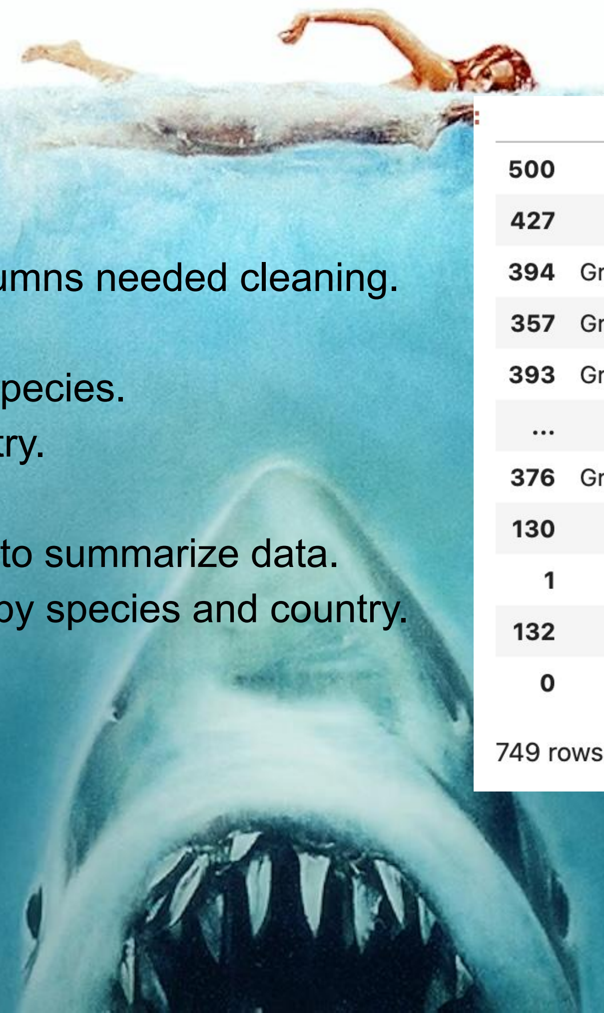


Data Cleaning and Analysis

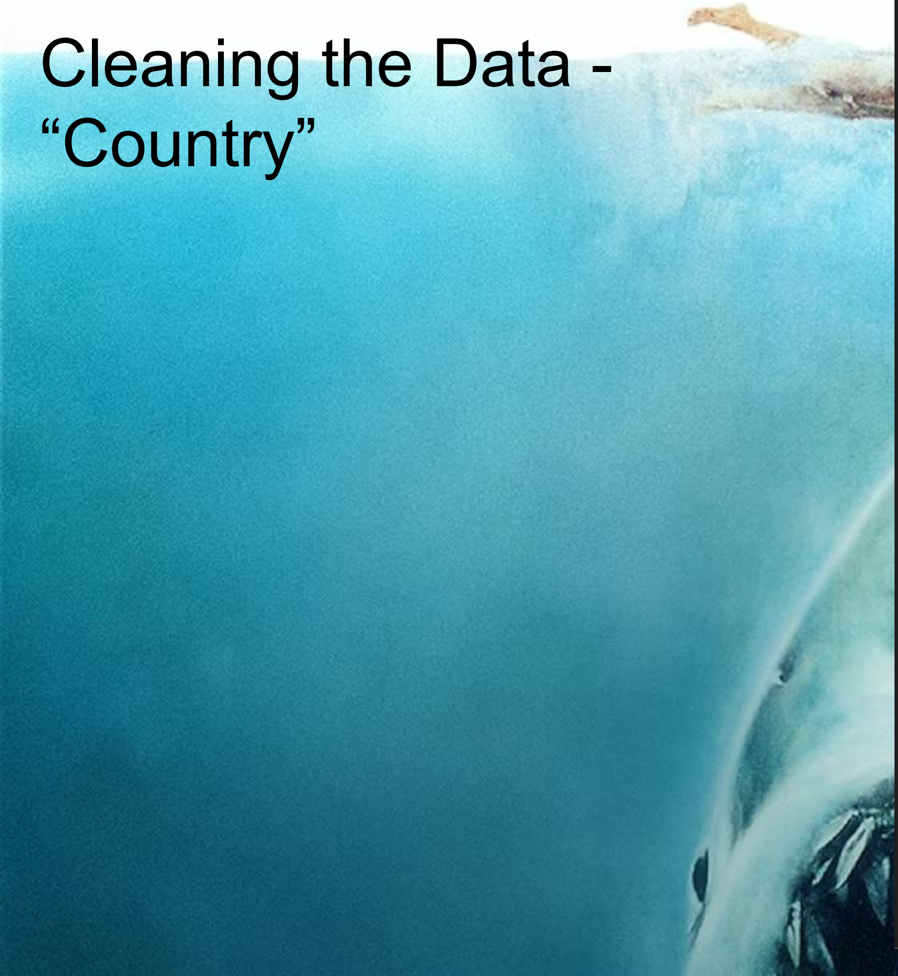
- **Identified Issues**:
  - Species and Country columns needed cleaning.
- **Cleaning Steps**:
  - Standardized names in Species.
  - Corrected errors in Country.
- **Data Visualization**:
  - Used groupby and count to summarize data.
  - Visualized shark attacks by species and country.

	Species	Country	Count
500	Invalid	USA	685
427	Invalid	AUSTRALIA	335
394	Great White Shark	USA	229
357	Great White Shark	AUSTRALIA	184
393	Great White Shark	UNSPECIFIED COUNTRY	179
...	...	...	...
376	Great White Shark	MONTENEGRO	1
130	2.4 m shark	AUSTRALIA	1
1		BRAZIL	1
132	2.5 m shark	MADAGASCAR	1
0		AUSTRALIA	1

749 rows × 3 columns



Cleaning the Data - “Country”

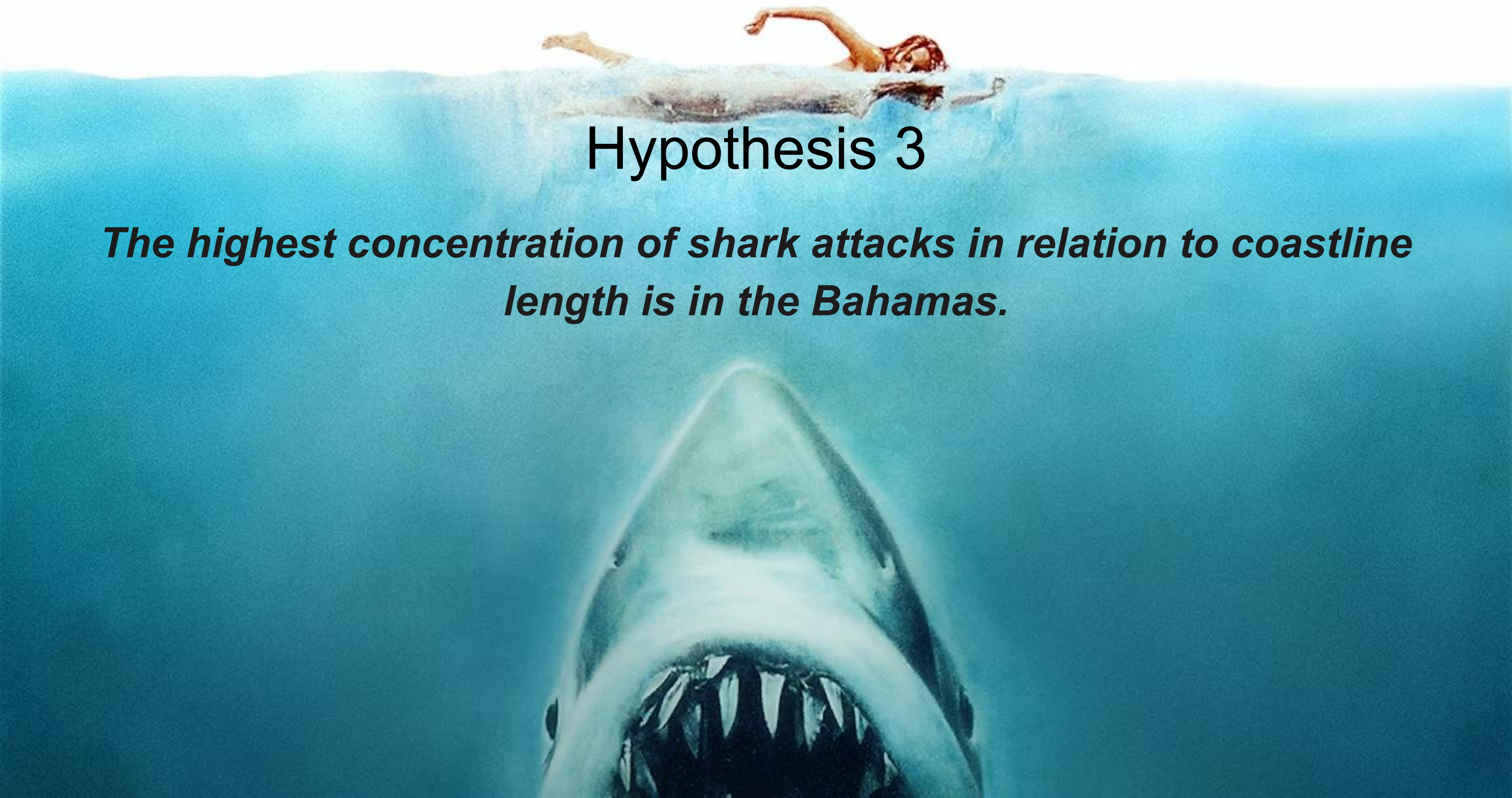


```python
# Remove question marks and strip whitespace
country = country.replace('?', '').strip()

# Handle cases where multiple countries are listed
country = country.replace('IRAN / IRAQ', 'IRAN') \
                 .replace('SOLOMON ISLANDS / VANUATU', 'SOLOMON ISLANDS') \
                 .replace('EQUATORIAL GUINEA / CAMEROON', 'CAMEROON') \
                 .replace('CEYLON (SRI LANKA)', 'SRI LANKA') \
                 .replace('EGYPT / ISRAEL', 'EGYPT') \
                 .replace('ITALY / CROATIA', 'ITALY') \
                 .replace('BETWEEN PORTUGAL & INDIA', 'UNSPECIFIED COUNTRY') \
                 .replace('DIEGO GARCIA', 'UNSPECIFIED COUNTRY')

# Replace "/" with "and" in specific countries
country = country.replace('ANDAMAN / NICOBAR ISLANDS', 'ANDAMAN AND NICOBAR ISLANDS') \
                 .replace('ST KITTS / NEVIS', 'ST KITTS AND NEVIS')

# Mapping replacements for specific island entries
replacements = {
    'UNITED ARAB EMIRATES (UAE)': 'UNITED ARAB EMIRATES',
    'NEW GUINEA / PAPUA NEW GUINEA': 'PAPUA NEW GUINEA',
    'SOLOMON ISLANDS / VANUATU': 'SOLOMON ISLANDS',
    'MALDIVE ISLANDS': 'MALDIVES',
    'ST. MAARTIN': 'ST. MARTIN',  # Correct spelling, merge duplicates
    'KOREA': 'SOUTH KOREA'        # Replace KOREA with SOUTH KOREA, merge duplicates
}

# Apply replacements
for key, value in replacements.items():
    if country == key:
        country = value

# Assign 'UNSPECIFIED COUNTRY' to values containing sea/ocean-related terms
sea_terms = ['sea', 'SEA', 'OCEAN', 'ocean', 'Ocean', 'Sea', 'BAY OF BENGAL', 'AFRICA']
for term in sea_terms:
    if term in country:
        return 'UNSPECIFIED COUNTRY'

# Remove invalid countries or regions
invalid_countries = ['Diego Garcia', 'GULF OF ADEN', 'THE BALKANS', 'BRITISH ISLES', 'PERSIAN GULF', 'JOHNSTON ISLAND',
                     'JAVA', 'ROTAN', 'SAN DOMINGO', 'ST. MARTIN', 'NEVIS', 'GRAND CAYMAN', 'NETHERLANDS ANTILLES',
                     'NORTHERN MARIANA ISLANDS', 'ASIA']

if country in invalid_countries:
    return 'UNSPECIFIED COUNTRY'

# Convert all to uppercase
country = country.upper()

return country
```

```
shark_data.rename(columns={"Species ": "Species"}, inplace=True) #rename species column

#shark_data["Species"].drop(shark_data[(shark_data == 0).any(axis=1)].index, inplace=True) # removes 0 value
#shark_data.dropna(subset=['Species'], inplace=True) #removes NaN


shark_data["Species"] = shark_data["Species"].apply(lambda x: "Tiger Shark" if "tiger" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Bull Shark" if "bull" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Blue Pointer" if "blue" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Great White Shark" if "white" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Hammerhead Shark" if "hammer" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Catshark" if "cat" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Hammerhead Shark" if "hammer" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Brown Shark" if "brown" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Blacktip" if "black" in str(x).lower() else x)

shark_data["Species"] = shark_data["Species"].apply(lambda x: "Invalid" if "NaN" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Invalid" if "questionable" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Invalid" if "involvement" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Invalid" if "'" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Invalid" if "''" in str(x).lower() else x)
shark_data["Species"] = shark_data["Species"].apply(lambda x: "Invalid" if "[]" in str(x).lower() else x)

#shark_data["Species"].nunique()
shark_data["Species"].value_counts().head(50)
```

Hypothesis 3

The highest concentration of shark attacks in relation to coastline length is in the Bahamas.



Data Gathering, Cleaning and Analysis

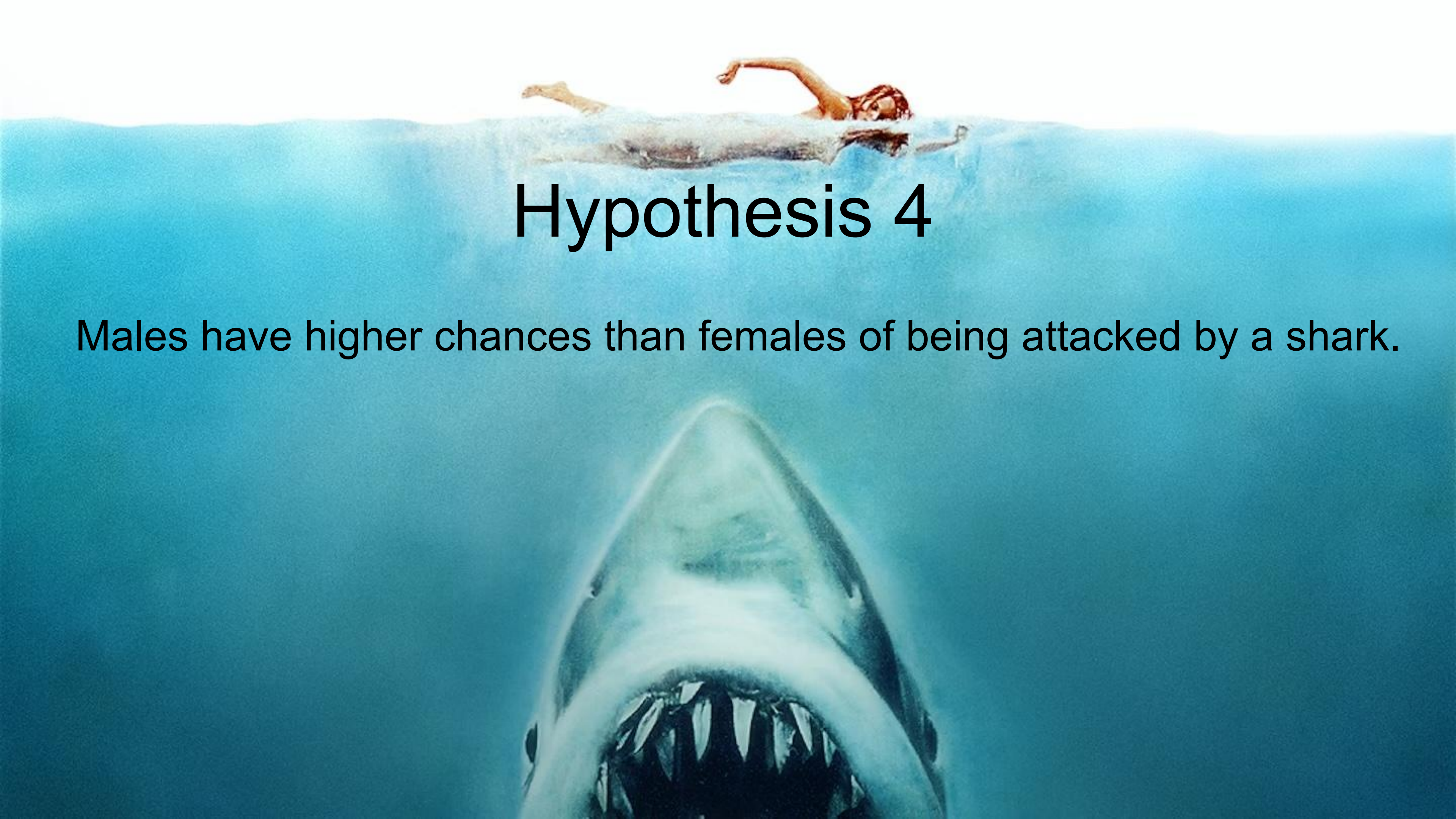
- **Identified Issues**:
    - Import CSV table for coastline length found (with two reliable sources)
    - Country cleaning for imported CSV file
    - Country cleaning shark_data (taken from H2)
    - Combining Country, Country_counts & coast length
- **Problem**:
    - Country cleaning for the CSV file needed a whole different approach
    - Not enough time to do it for each countries

```python
#Type Import Coast Length
country_series = s_d['Country'].value_counts()
country_series[100:]
coastline_df = pd.read_csv('https://raw.githubusercontent.com/LSYS/country-coastline-distance/master/coastlines.csv')
coastline_df.head()
```

	country	iso2	area_wf	coastline_wf	coast_to_area_wf	coastline_wri	coast_to_area_wri
0	Afghanistan	AF	652230.0	0.0	0.000000	0.0	0.000000
1	Albania	AL	28748.0	362.0	1.259218	649.0	2.257548
2	Algeria	DZ	2381740.0	998.0	0.041902	1557.0	0.065372
3	American Samoa	AS	224.0	116.0	51.785714	NaN	NaN
4	Andorra	AD	468.0	0.0	0.000000	NaN	NaN

Hypothesis 4

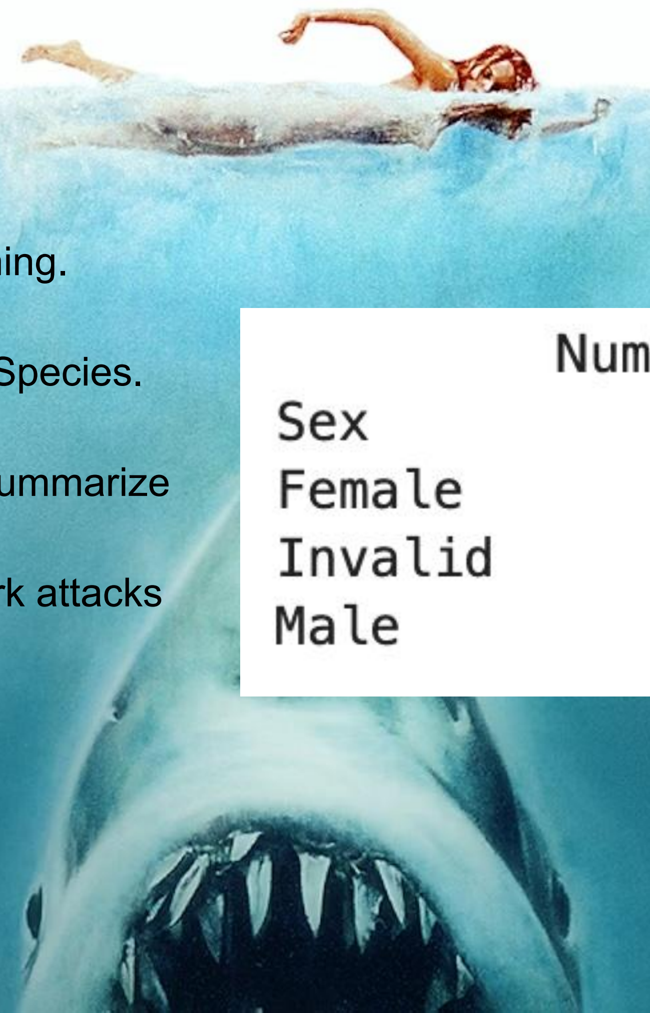
Males have higher chances than females of being attacked by a shark.

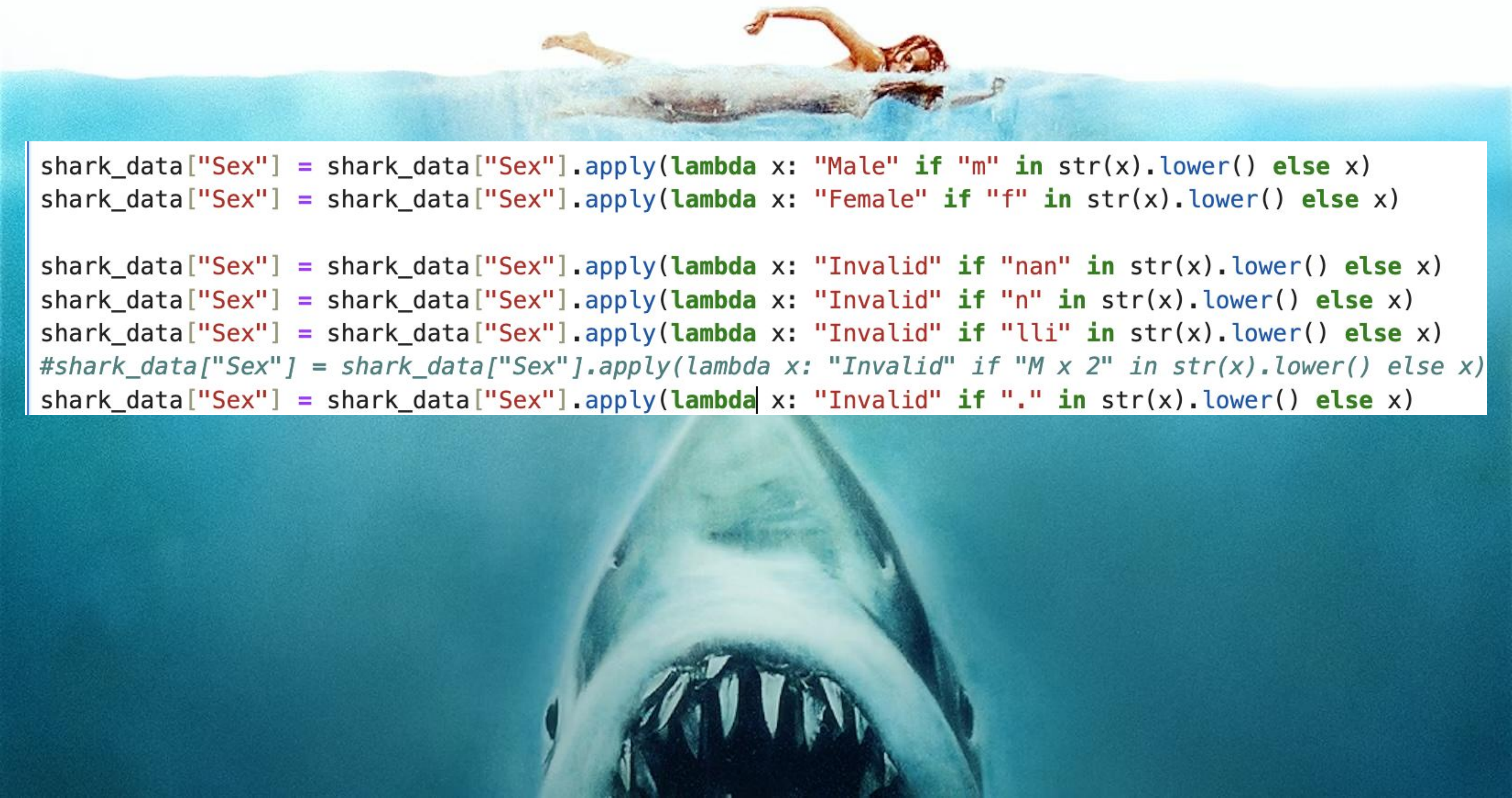


Data Cleaning and Analysis

- **Identified Issues**:
  - Sex column needed cleaning.
- **Cleaning Steps**:
  - Standardized F and M in Species.
- **Data Visualization**:
  - Used pivot and count to summarize data.
  - Visualized number of shark attacks by Sex.
- **Outcome**:

Sex	Number of Attacks
Female	744
Invalid	580
Male	5474





```python
shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Male" if "m" in str(x).lower() else x)
shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Female" if "f" in str(x).lower() else x)

shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Invalid" if "nan" in str(x).lower() else x)
shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Invalid" if "n" in str(x).lower() else x)
shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Invalid" if "lli" in str(x).lower() else x)
#shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Invalid" if "M x 2" in str(x).lower() else x)
shark_data["Sex"] = shark_data["Sex"].apply(lambda x: "Invalid" if "." in str(x).lower() else x)
```

Biggest Mistakes

- Deleting Duplicates in Country Column
- GroupBy Syntax
- Errors from Overwriting Names
- Issues:
    - Data loss
    - Type mismatches
- Following wrong approach for too long
- Messing up the sequence of cleaning steps, leading to errors and incorrect data

Technical Challenges

- Overwriting DataFrames
- Creating Unison Dataframe
- Find good source to import external data


A mini project by balsamico bizeps, marc, sunmax44, ironhack.
Github repository
Thank you for your attention.