1. Setting Up the Environment and Importing Data

In [ ]:
```
#!pip install fiona
#!pip install pyproj
#!pip install geopandas
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import datetime
import calendar
import geopandas as gpd
```

In [3]:
```
filepath = 'data/Airplane_Crashes_and_Fatalities_Since_1908.csv'
df = pd.read_csv(filepath)
```

2. Exploring the Dataset

In [4]:
```
# Preview the firs 5 rows of data
df.head()
```

Out[4]:

	Date	Time	Location	Operator	Flight #	Route	Type	Registration	cn/In
0	09/17/1908	17:18	Fort Myer, Virginia	Military - U.S. Army	NaN	Demonstration	Wright Flyer III	NaN	1
1	07/12/1912	06:30	AtlantiCity, New Jersey	Military - U.S. Navy	NaN	Test flight	Dirigible	NaN	NaN
2	08/06/1913	NaN	Victoria, British Columbia, Canada	Private	-	NaN	Curtiss seaplane	NaN	NaN
3	09/09/1913	18:30	Over the North Sea	Military - German Navy	NaN	NaN	Zeppelin L-1 (airship)	NaN	NaN
4	10/17/1913	10:30	Near Johannisthal, Germany	Military - German Navy	NaN	NaN	Zeppelin L-2 (airship)	NaN	NaN

In [5]:
```
# Preview the last 5 rows of data
```

```
df.tail()
```

Out[5]:

	Date	Time	Location	Operator	Flight #	Route	Type	Registra
5263	05/20/2009	06:30	Near Madiun, Indonesia	Military - Indonesian Air Force	NaN	Jakarta - Maduin	Lockheed C-130 Hercules	A-
5264	05/26/2009	NaN	Near Isiro, DemocratiRepubliCongo	Service Air	NaN	Goma - Isiro	Antonov An-26	9Q-
5265	06/01/2009	00:15	AtlantiOcean, 570 miles northeast of Natal, Br...	Air France	447	Rio de Janeiro - Paris	Airbus A330-203	F-(
5266	06/07/2009	08:30	Near Port Hope Simpson, Newfoundland, Canada	Strait Air	NaN	Lourdes de BlanSablon - Port Hope Simpson	Britten-Norman BN-2A-27 Islander	C
5267	06/08/2009	NaN	State of Arunachal Pradesh, India	Military - Indian Air Force	NaN	Mechuka for Jorhat	Antonov An-32	

In [6]:
```
print(str("Dataset consist of " + str(df.shape[0]) + " observations (crashes) and "
```

```
Dataset consist of 5268 observations (crashes) and 13 features.
```

Features are following:

- Date : The date on which the flight crashed
- Time : The time at which flight crashed
- Location : Location of the crash
- Operator : The name of the flight operator
- Flight : Flight Number of the airplane that crashed
- Route : The Route of the flight
- Type : The type of flight carrier
- Registration
- cn/ln
- Abroad : The number of passenger on board
- Fatalities : The number of deaths
- Ground : The number of deaths on the ground caused by the plane crash

- Summary : Brief summary of the case

In [7]:
```
# Column names
print('Columns names' ,list(df.columns))
```

```
Columns names ['Date', 'Time', 'Location', 'Operator', 'Flight #', 'Route', 'Typ
e', 'Registration', 'cn/In', 'Aboard', 'Fatalities', 'Ground', 'Summary']
```

In [8]:
```
# remove unnecessary columns
df.drop(columns=['Flight #','Registration','cn/In','Summary'] ,inplace=True)
```

In [9]:
```
# more informations about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5268 entries, 0 to 5267
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        5268 non-null   object
 1   Time        3049 non-null   object
 2   Location    5248 non-null   object
 3   Operator    5250 non-null   object
 4   Route       3562 non-null   object
 5   Type        5241 non-null   object
 6   Aboard      5246 non-null   float64
 7   Fatalities  5256 non-null   float64
 8   Ground      5246 non-null   float64
dtypes: float64(3), object(6)
memory usage: 370.5+ KB
```

In [10]:
```
# Descriptive statistics
df.describe().rename({'50%': 'median'}).T
```

Out[10]:

	count	mean	std	min	25%	median	75%	max
Aboard	5246.0	27.554518	43.076711	0.0	5.0	13.0	30.0	644.0
Fatalities	5256.0	20.068303	33.199952	0.0	3.0	9.0	23.0	583.0
Ground	5246.0	1.608845	53.987827	0.0	0.0	0.0	0.0	2750.0

In [11]:
```
# Check if duplicated row exist
df.duplicated().sum()
```

Out[11]:
```
0
```

3. Questions

1. What is the trend of airplane crashes over the years, months?
2. What is the trend of fatalities over the years, months?
3. Which locations have the highest number of airplane crashes?
4. Which operators have the most incidents?

5. What types of aircraft are most involved in crashes?

4. Data preprocessing

In [12]:
```
data = df.copy()
```

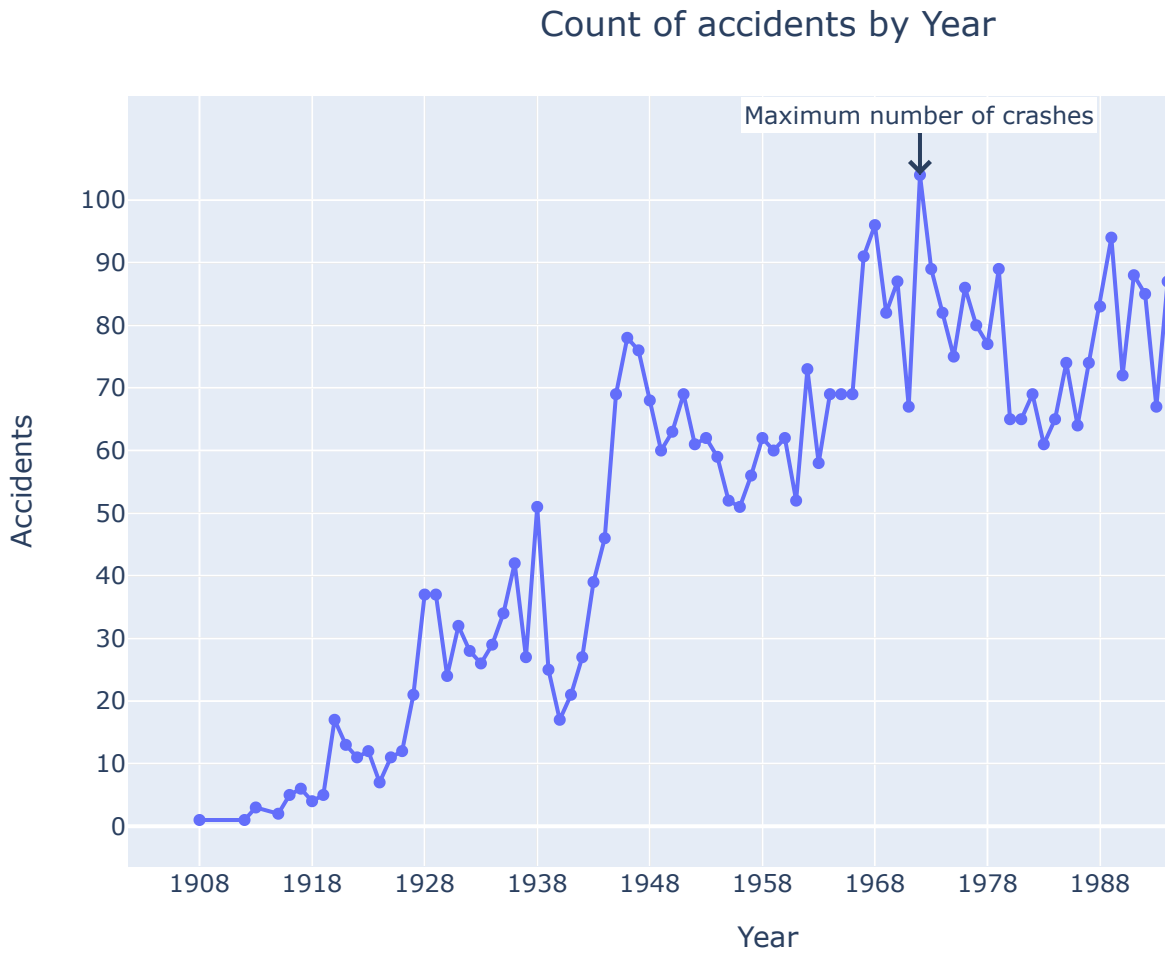
In [13]:
```
# Transform the type of column date from string to Date
data.Date = pd.to_datetime(data.Date)

# Extracting the year, month and day out of the Date column
data["Year"] = data.Date.apply(lambda x: x.year)
data["Month"] = data.Date.apply(lambda x: x.month)
data["Day"] = data.Date.apply(lambda x: x.day)
```

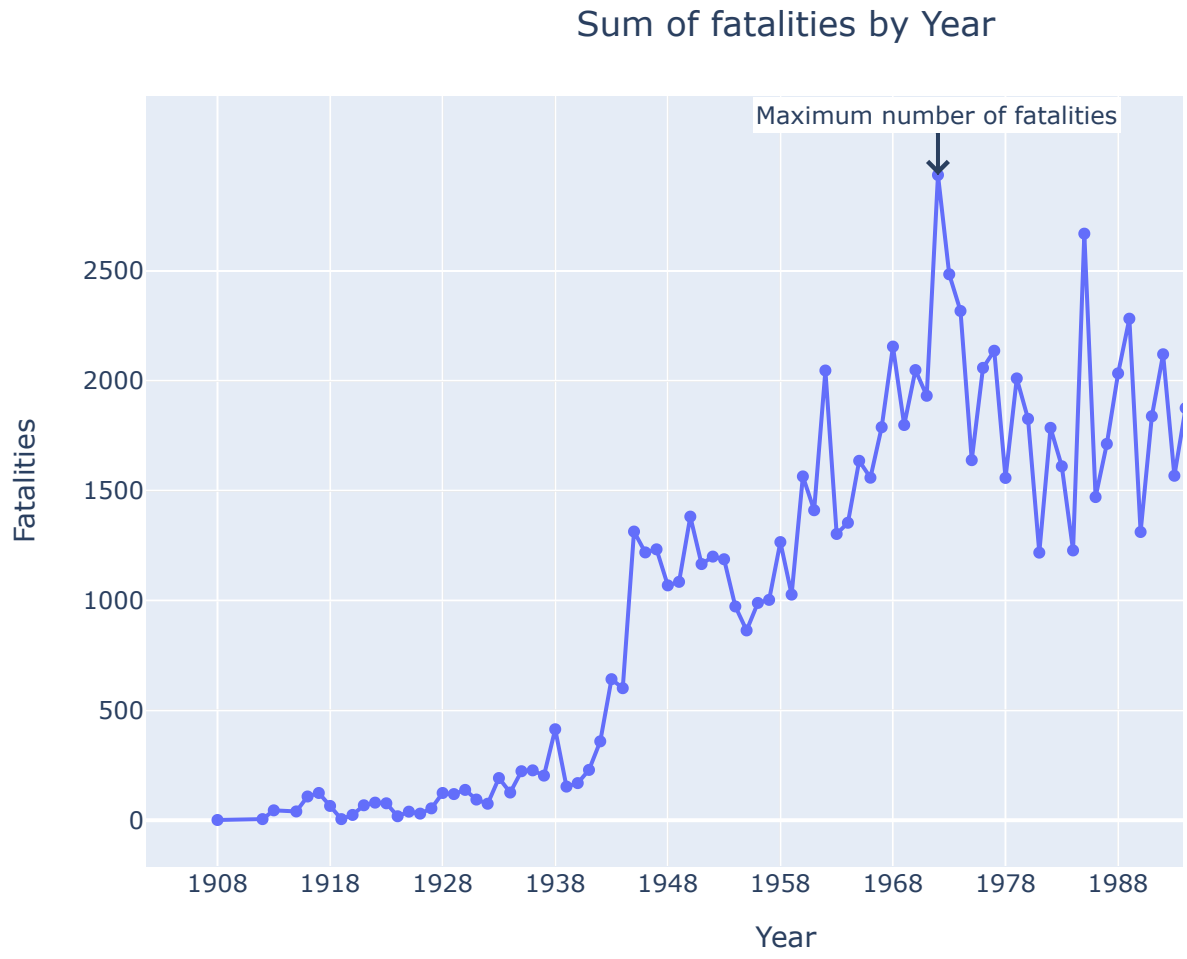
In [14]:
```
data.dropna(subset=['Aboard'], inplace=True)
data.dropna(subset=['Fatalities'], inplace=True)
```

5. Data analysis and visualization

A. Crach trend and Fatality Percent by year and month

In [15]:
```
# Group data by 'Year' and count the number of occurrences in each year
Temp1 = data.groupby('Year')[['Date']].count().rename(columns={"Date": "Count"}).re

# Group data by 'Year' and calculate the sum of 'Fatalities' in each year
Temp2 = data.groupby('Year')[['Fatalities']].sum().rename(columns={"Fatalities": "S
Temp2.Sum = Temp2.Sum.astype('int64')

# Create a line plot for the count of accidents by year
fig1 = px.line(Temp1, x="Year", y="Count", width=800, markers=True)
fig1.add_annotation(text='Maximum number of crashes',x=Temp1.loc[Temp1['Count'].idx
                    bgcolor='#fff', showarrow=True, arrowhead=4, arrowwidth=2, ax=0
# Edit the layout
fig1.update_layout(title='Count of accidents by Year',title_x=0.5,
                    xaxis=dict(tickmode='array',tickvals=list(range(Temp1.Year.min(
                    yaxis=dict(tickmode='array',tickvals=list(range(0, Temp1.Count.

# Create a line plot for the sum of fatalities by year
fig2 = px.line(Temp2, x="Year", y="Sum", width=800 ,markers=True)
fig2.add_annotation(text='Maximum number of fatalities', x=Temp2.loc[Temp2['Sum'].i
                    bgcolor='#fff', showarrow=True, arrowhead=4, arrowwidth=2, ax=0
# Edit the layout
fig2.update_layout(title='Sum of fatalities by Year', title_x=0.5,
                    xaxis=dict(tickmode='array', tickvals=list(range(Temp2.Year.min
                    yaxis=dict(tickmode='array', tickvals=list(range(0, Temp2.Sum.m

fig1.show()
fig2.show()
```

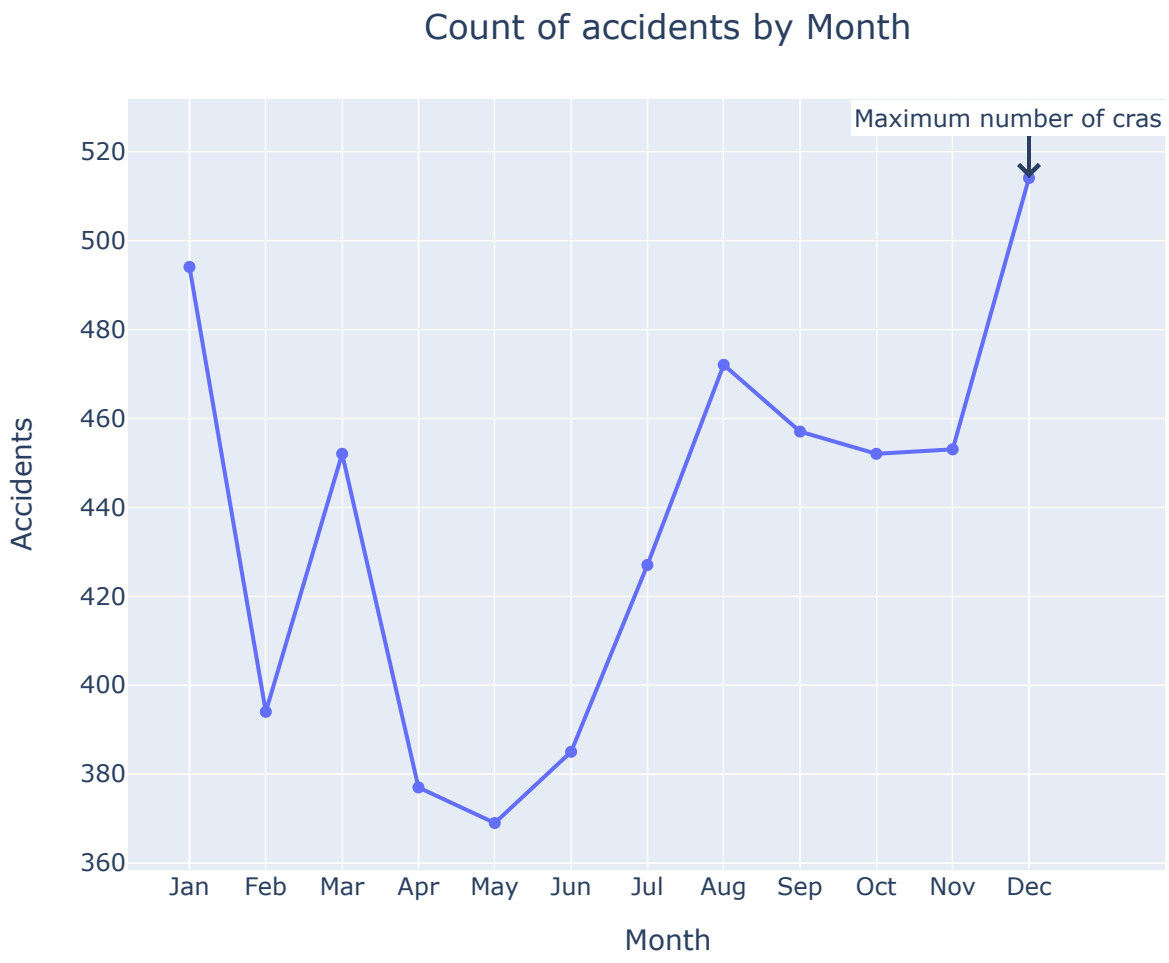

Count of accidents by Year
Maximum number of crashes
Accidents
Year
0
10
20
30
40
50
60
70
80
90
100
1908
1918
1928
1938
1948
1958
1968
1978
1988

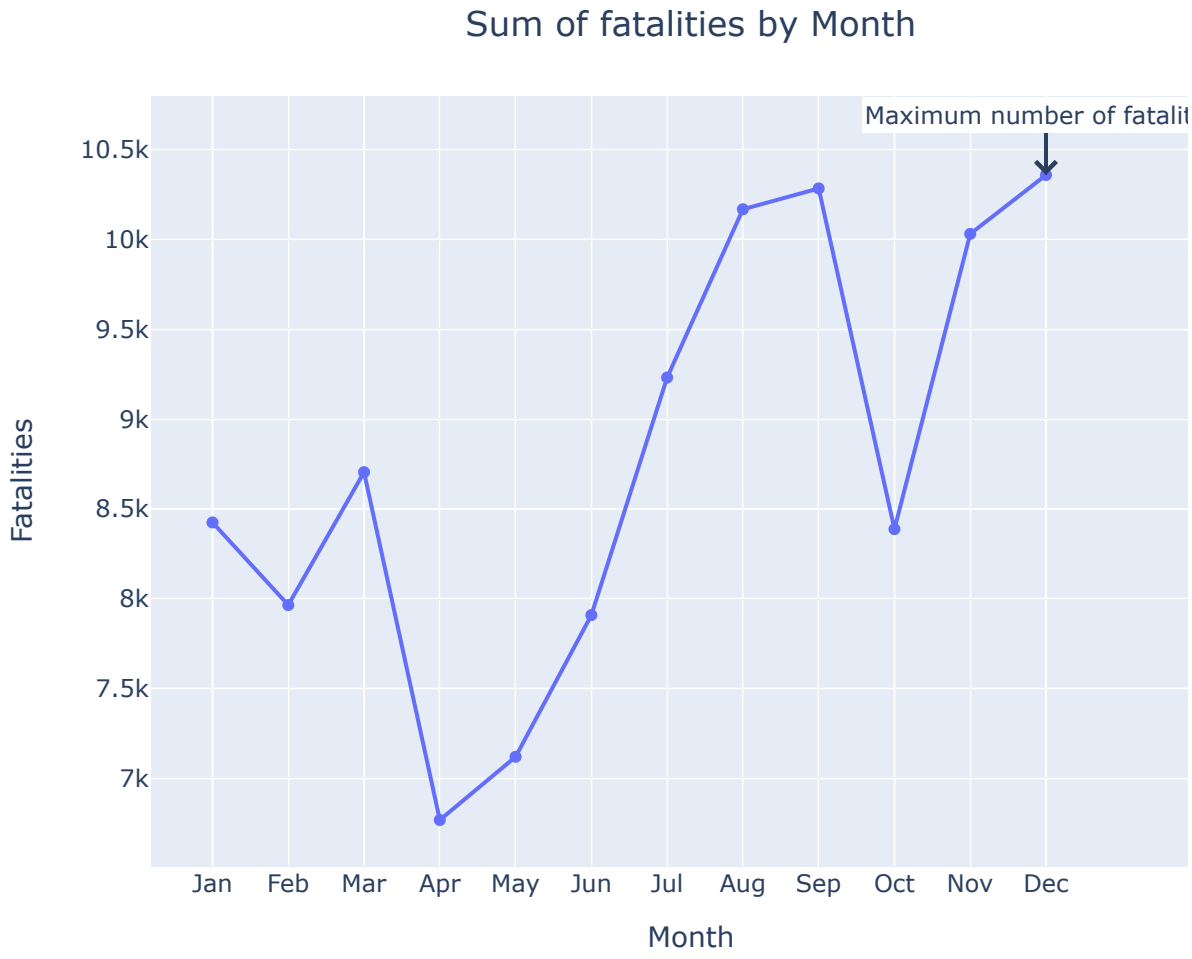


In [16]:

```
# Group data by 'Year' and count the number of occurrences in each year
Temp1 = data.groupby(data.Month)[['Date']].count().rename(columns={"Date": "Count"}

# Group data by 'Year' and calculate the sum of 'Fatalities' in each year
Temp2 = data.groupby(data.Month)[['Fatalities']].sum().rename(columns={"Fatalities"
Temp2.Sum = Temp2.Sum.astype('int64')

# Max value: crashes and Fatalities
x_max1 = Temp1.loc[Temp1['Count'].idxmax(), 'Month']
y_max1 = Temp1.loc[Temp1['Count'].idxmax(), 'Count']

x_max2 = Temp2.loc[Temp2['Sum'].idxmax(), 'Month']
y_max2 = Temp2.loc[Temp2['Sum'].idxmax(), 'Sum']

# Chart one
fig1 = px.line(Temp1, x="Month", y="Count", width=700 ,markers=True)
fig1.add_annotation(text='Maximum number of crashes',x=x_max1,y=y_max1, bgcolor='#f

month_num = list(range(1, 13))
month = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov
fig1.update_layout(title='Count of accidents by Month',title_x=0.5,
                   xaxis=dict(tickmode='array',tickvals=month_num,ticktext=month,ti
                   yaxis=dict(title='Accidents'))
```

```
fig1.show()

# Chart two
fig2 = px.line(Temp2, x="Month", y="Sum", width=700 ,markers=True)
fig2.add_annotation(text='Maximum number of fatalities',x=x_max2,y=y_max2, bgcolor=
fig2.update_layout(title='Sum of fatalities by Month',title_x=0.5,
                   xaxis=dict(tickmode='array',tickvals=month_num,ticktext=month,ti
                   yaxis=dict(title='Fatalities'))

fig2.show()
```





B. Airecraft Oparators

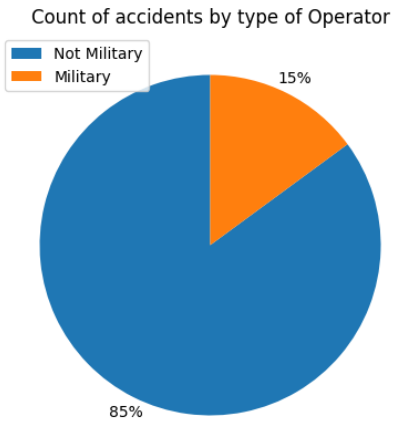
In [17]:
```
# Function to check if 'Operator' contains 'Military'
def is_military(operator):
    try:
        return 'Yes' if 'Military' in operator else 'No'
    except:
        return np.nan

data['IsMilitary'] = data['Operator'].apply(is_military)
data.head(5)
```

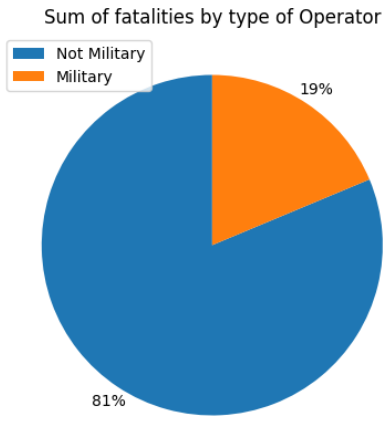
Out[17]:

	Date	Time	Location	Operator	Route	Type	Aboard	Fatalities	Ground	Yea
0	1908-09-17	17:18	Fort Myer, Virginia	Military - U.S. Army	Demonstration	Wright Flyer III	2.0	1.0	0.0	1908
1	1912-07-12	06:30	AtlantiCity, New Jersey	Military - U.S. Navy	Test flight	Dirigible	5.0	5.0	0.0	1912
2	1913-08-06	NaN	Victoria, British Columbia, Canada	Private	NaN	Curtiss seaplane	1.0	1.0	0.0	1913
3	1913-09-09	18:30	Over the North Sea	Military - German Navy	NaN	Zeppelin L-1 (airship)	20.0	14.0	0.0	1913
4	1913-10-17	10:30	Near Johannisthal, Germany	Military - German Navy	NaN	Zeppelin L-2 (airship)	30.0	30.0	0.0	1913

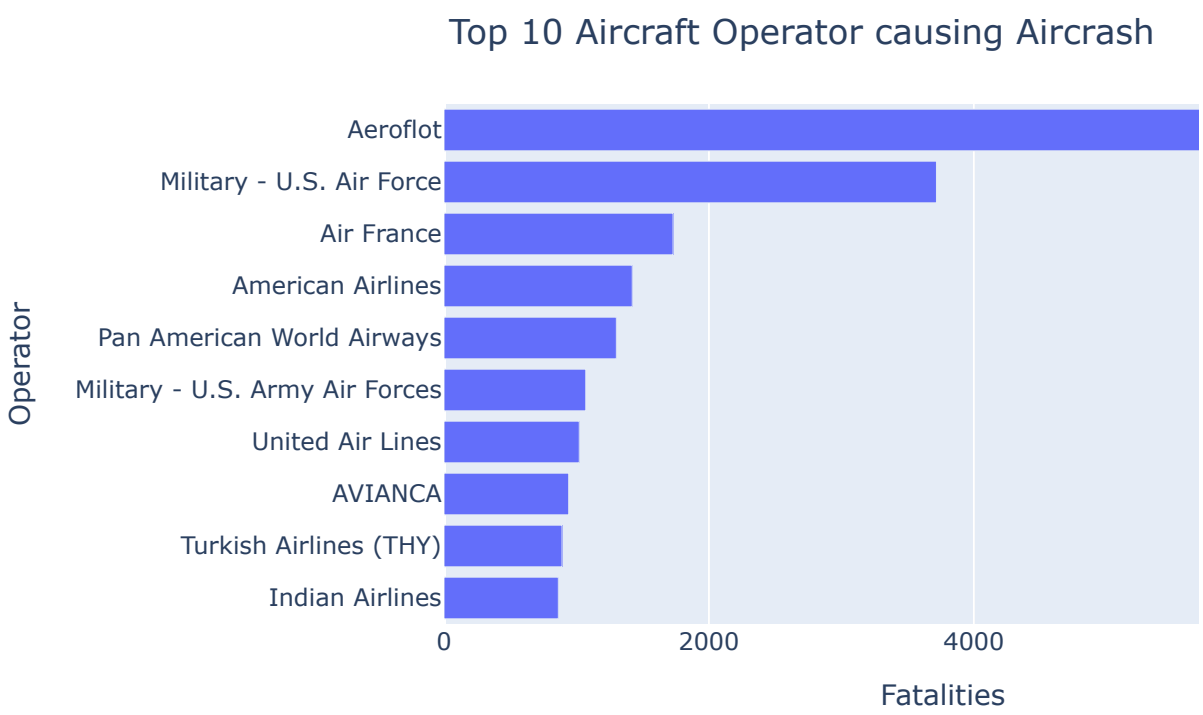
In [18]:
```
# Drop rows where 'Operator' is NaN and create a copy of the DataFrame
data1 = data.dropna(subset=['Operator']).copy()

# Group the filtered data by 'IsMilitary' and count the occurrences
Temp1 = data1.groupby('IsMilitary')[['IsMilitary']].count().rename(columns={"IsMili

# Group the filtered data by 'IsMilitary' and calculate the sum of 'Fatalities'
Temp2 = data1.groupby('IsMilitary')[['Fatalities']].sum().rename(columns={"Fataliti
```

In [19]:
```
# Create a figure with two subplots(count accidents and sum of fatalities by operat
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(16, 5))

# Chart one
patches, texts , autotexts = axes[0].pie(Temp1.Count.values,startangle=90,autopct='
axes[0].legend(patches, ['Not Military', 'Military'], loc="upper left", fontsize=10
axes[0].set_title('Count of accidents by type of Operator')

# Chart Two
patches1, texts1 , autotexts1 = axes[1].pie(Temp2.Sum.values,startangle=90,autopct=
axes[1].legend(patches1, ['Not Military', 'Military'], loc="upper left", fontsize=1
axes[1].set_title('Sum of fatalities by type of Operator')

# Show the plot
plt.show()
```





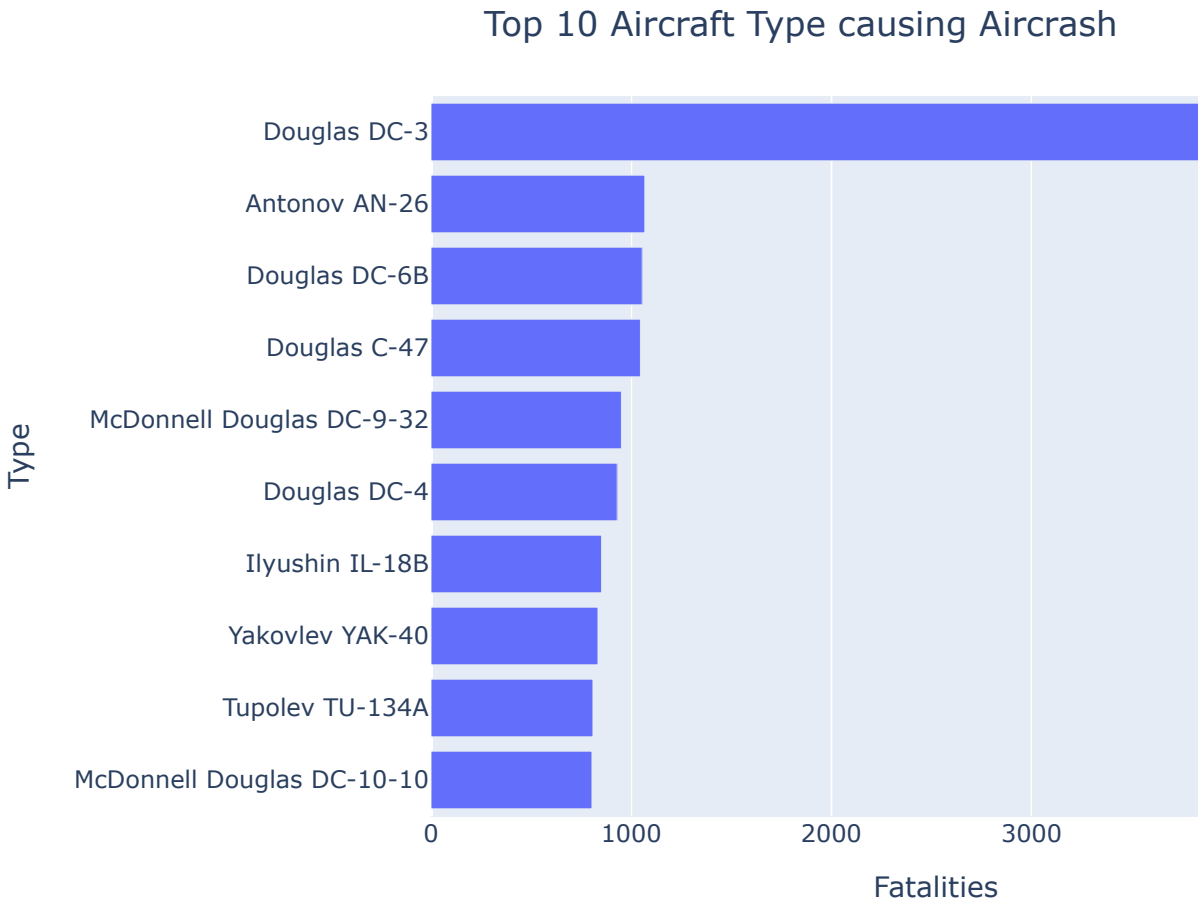
In [20]:
```
# Group the filtered data by 'Operator', calculate the sum of 'Fatalities'
Temp = data1.groupby('Operator')[['Fatalities']].sum().reset_index().sort_values(by

fig1 = px.bar(Temp.iloc[:10][::-1], y="Operator", x="Fatalities", width=800, height
fig1.update_layout(title='Top 10 Aircraft Operator causing Aircrash',title_x=0.5,
                   yaxis=dict(title='Operator'),
                   xaxis=dict(title='Fatalities'))
```



C. Airecraft Type

In [21]:
```
# Drop rows where 'Type' is NaN and create a copy of the DataFrame
data1 = data.dropna(subset=('Type')).copy()

# Group the filtered data by 'Type', calculate the sum of 'Fatalities'
Temp = data1.groupby('Type')['Fatalities'].sum().reset_index().sort_values(by='Fata
```

```
fig1 = px.bar(Temp.iloc[:10][::-1], y="Type", x="Fatalities", width=800, height=500
fig1.update_layout(title='Top 10 Aircraft Type causing Aircrash',title_x=0.5,
                   yaxis=dict(title='Type'),
                   xaxis=dict(title='Fatalities'),
                       )
```



D. Crach Locations

In [22]:
```
# Extract Country from LOcation column
data['Country'] = data.Location.apply(lambda x: x.split(',')[-1].strip() if not pd.
data[['Location', 'Country']].head()
```

Out[22]:

	Location	Country
0	Fort Myer, Virginia	Virginia
1	AtlantiCity, New Jersey	New Jersey
2	Victoria, British Columbia, Canada	Canada
3	Over the North Sea	Over the North Sea
4	Near Johannisthal, Germany	Germany

In [23]:
```
# Drop rows where 'Country' is NaN and create a copy of the DataFrame
data1 = data.dropna(subset=('Country')).copy()

# Group the filtered data by 'Country', count the occurrences
Temp = data1.groupby('Country').size().reset_index(name='Accident_Count').sort_valu
```

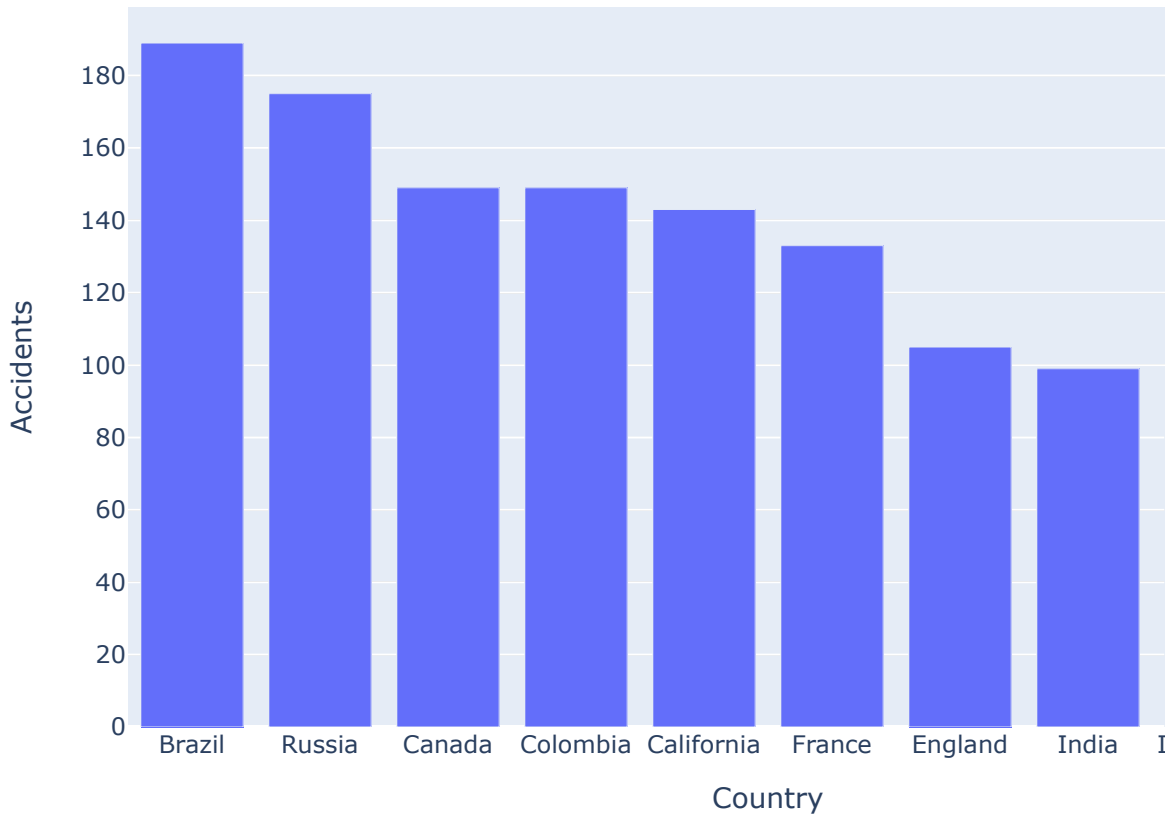
In [24]:
```
# Sort countries by number of accidents
accidents_by_country = data.groupby('Country').size().reset_index(name='Accident_Co
accidents_by_country.head(12)
```

Out[24]:

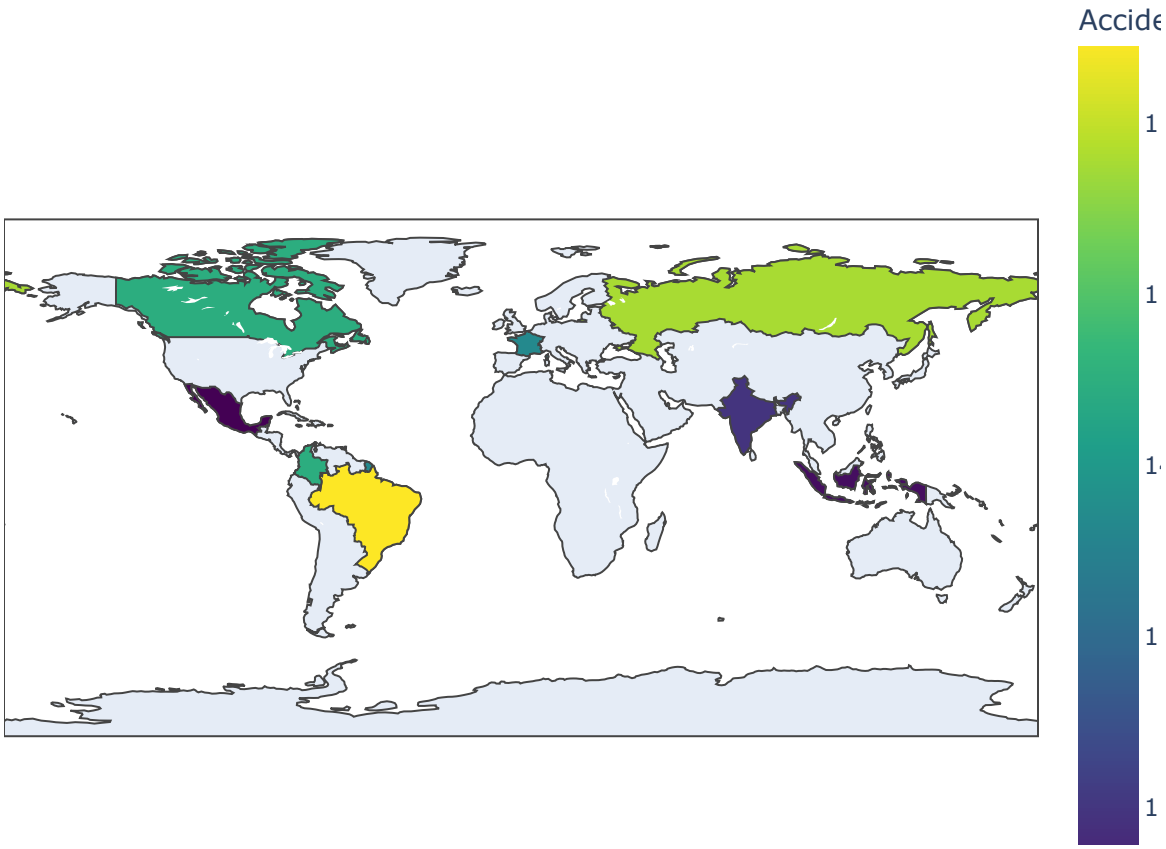
	Country	Accident_Count
57	Brazil	189
10	Alaska	177
379	Russia	175
73	Canada	149
87	Colombia	149
68	California	143
133	France	133
122	England	105
181	India	99
185	Indonesia	87
242	Mexico	83
85	China	81

In [25]:
```
# Drop States
accidents_by_country.drop([10,70], inplace=True)

# Get top 10 countries by number of accidents
top_10_countries = accidents_by_country.iloc[:10]
accidents_by_country.loc[125, 'Country'] = 'U.K. of Great Britain and Northern Irel
```

In [26]:
```
fig1 = px.bar(top_10_countries, x="Country", y="Accident_Count", width=800, height=
fig1.update_layout(title='Top 10 Countries by Number of Accidents',title_x=0.5,
                   xaxis=dict(title='Country'),
                   yaxis=dict(title='Accidents'))
```



In [27]:
```
plt.figure(figsize=(100,10))
fig = px.choropleth(top_10_countries, locations='Country', locationmode='country na
        color_continuous_scale="Viridis", title='Countries on Map with Values')
fig.update_layout(title='Map of Top 10 Countries by Number of Accidents',margin={"r
fig.show()
```




```
<Figure size 10000x1000 with 0 Axes>
```

Conclusion :

Temporal Patterns Analysis:

- Between 1960 and 1980, Aeroflot-operated flights were involved in a significant number of accidents.

- In December, August, and January, a notable increase in aviation accidents was observed, possibly linked to increased holiday travel.

- In 1972, a notable increase in fatalities was followed by a subsequent decrease, suggesting improvements in aviation safety.

- The highest number of deaths occurred in December and September, which raises the question of whether it was related to more people traveling during the holidays.

- Accidents commonly reach a peak in the middle of the month.

Operator Classification Impact:

- Military operators (15% of accidents) have a comparatively lower impact on fatalities, while non-military operators (85% of accidents) significantly influence both accident frequency and severity. This underscores the importance of operator classification in aviation safety analysis.

Type-Specific Insights:

- The 'Douglas DC-3' type stands out with the highest fatalities.

High-Incident Countries:

- The geographical analysis of aviation accidents has identified Canada, Brazil, and Russia as regions with higher accident rates.

Next Steps :

As a next step, we plan to leverage Natural Language Processing (NLP) techniques to analyze the accident summaries in our dataset. By extracting key information and causes from these narratives, we aim to gain deeper insights into the underlying factors contributing to aviation accidents.