Aviation_Accident_Database_Project

Introduction

This presentation will explore the main reasons behind aviation accidents dating back to 1948. We aim to identify patterns and causes to provide suggestions for improving the safety of air travel.

objectives

- to clean up the data
- to find out the number of accidents that have happened
- to find out how many passangers were injured or killed
- to find out the cause of the crashes
- what is the majour contribution to accidents (pilot error or other)
- airline with most accidents
- country with most accidents

Loading files

```
#importing the necesarry libraries
import pandas as pd
import numpy as np
import json
import csv
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#checking the files in the folder
!ls
```

```
AviationData.csv
Aviation_Accident_Database_Project.ipynb
Project_powerpoint_pdf
README.md
USState_Codes.csv
tableau worksheet
~tableau worksheet__11420.twbr
```

```
#loading the first file ("AviationData.csv")
with open("AviationData.csv") as f:
    reader = csv.reader(f)
    data = list(reader)
df = pd.DataFrame(data)
df.head()
# most of the columns appear to be empty but do not contain the NaN
```

	0	1	2	3	4	5	6	7	8	9
0	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
1	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States				
2	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States				
3	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056		
4	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States				

5 rows × 31 columns

```
#loading the second file
codes = pd.read_csv("USState_Codes.csv")
codes.head()
# since the first data set has ['Location'] and we are focusing on the country of the incidents there is no need to further look into ea
```

	US_State	Abbreviation
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA

Analysing the dataset and looking for missing values

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88890 entries, 0 to 88889
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       88890 non-null  object
 1   1       88890 non-null  object
 2   2       88890 non-null  object
 3   3       88890 non-null  object
 4   4       88890 non-null  object
 5   5       88890 non-null  object
 6   6       88890 non-null  object
 7   7       88890 non-null  object
 8   8       88890 non-null  object
 9   9       88890 non-null  object
 10  10      88890 non-null  object
 11  11      88890 non-null  object
 12  12      88890 non-null  object
 13  13      88890 non-null  object
 14  14      88890 non-null  object
 15  15      88890 non-null  object
 16  16      88890 non-null  object
 17  17      88890 non-null  object
 18  18      88890 non-null  object
 19  19      88890 non-null  object
 20  20      88890 non-null  object
 21  21      88890 non-null  object
 22  22      88890 non-null  object
 23  23      88890 non-null  object
 24  24      88890 non-null  object
 25  25      88890 non-null  object
 26  26      88890 non-null  object
 27  27      88890 non-null  object
 28  28      88890 non-null  object
 29  29      88890 non-null  object
 30  30      88890 non-null  object
dtypes: object(31)
memory usage: 21.0+ MB
```

```
df.shape
```

```
(88890, 31)
```

Aviation data seems to not have a propper title, hence i will have to make the first(0) row into a title row

```
df.columns #cecking initial column names to confirm
```

```
RangeIndex(start=0, stop=31, step=1)
```

```
df.columns = df.iloc[0] #replacing the column names with the first row
df.columns #calling to confirm the changes made
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Description',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object', name=0)
```

```
df = df.drop(0) #dropping the first row as it contained the column titles
```

```
df.isna().sum() #doesnt work as the sells contain ' ' and is  non-null
```

```
0
Event.Id                  0
Investigation.Type        0
Accident.Number           0
Event.Date                0
Location                  0
Country                   0
Latitude                  0
Longitude                 0
Airport.Code              0
Airport.Name              0
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Registration.Number       0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type               0
FAR.Description           0
Schedule                  0
Purpose.of.flight         0
Air.carrier               0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Broad.phase.of.flight     0
Report.Status             0
Publication.Date          0
dtype: int64
```

```
#looping each column to find empty cells containing '' as they do not contribute to the dataset
results = {}
for col in df.columns:
    count = df[col].str.contains(' ').sum()
    results[col] = count
results
```

```
{'Event.Id': 0,
 'Investigation.Type': 0,
 'Accident.Number': 0,
 'Event.Date': 0,
 'Location': 88692,
 'Country': 83485,
 'Latitude': 0,
 'Longitude': 0,
 'Airport.Code': 5,
 'Airport.Name': 41222,
 'Injury.Severity': 0,
 'Aircraft.damage': 0,
 'Aircraft.Category': 91,
 'Registration.Number': 21,
 'Make': 12388,
 'Model': 10933,
 'Amateur.Built': 0,
 'Number.of.Engines': 0,
 'Engine.Type': 10197,
 'FAR.Description': 7737,
 'Schedule': 0,
 'Purpose.of.flight': 8584,
 'Air.carrier': 15997,
 'Total.Fatal.Injuries': 0,
 'Total.Serious.Injuries': 0,
 'Total.Minor.Injuries': 0,
 'Total.Uninjured': 0,
 'Weather.Condition': 0,
 'Broad.phase.of.flight': 0,
 'Report.Status': 80303,
 'Publication.Date': 0}
```

```
#dropping mostly empty columns
df.drop(columns=['Number.of.Engines','Latitude','Airport.Name','Airport.Code','Airport.Code','Longitude','Air.carrier', 'Aircraft.Catego
#keeping country and location as it is important in my analysis
```

```
df.shape  #previous shape = (88890, 31)
```

```
(88889, 17)
```

```
df.describe()
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Ama
count	88889	88889	88889	88889	88889	88889	88889	88889	
unique	87951	2	88863	14782	27759	220	110	5	
top	20001212X19172	Accident	DCA22WA089	1984-06-30	ANCHORAGE, AK	United States	Non-Fatal	Substantial	
freq	3	85015	2	25	434	82248	67357	64148	

```
df.head() #confirming changes
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Injury.Severity	Aircraft.damage	Amateur.
1	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	Destroyed	
2	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	Destroyed	
3	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	Fatal(3)	Destroyed	
4	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Fatal(2)	Destroyed	
5	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	Fatal(1)	Destroyed	

Analysis starts

```
minimum_date = df['Event.Date'].min()[:4] #makes the code neat and short---finding the date where the records begin
print(f"there have been {len(df['Accident.Number'].value_counts())} accidents since {minimum_date}")
```

```
there have been 88863 accidents since 1948
```

```
df["Injury.Severity"].unique #finding the different types of injuries we might find in this dataset
```

```
<bound method Series.unique of 1         Fatal(2)
2         Fatal(4)
3         Fatal(3)
4         Fatal(2)
5         Fatal(1)
            ...
88885        Minor
88886
88887    Non-Fatal
88888
88889        Minor
Name: Injury.Severity, Length: 88889, dtype: object>
```

Documenting the different types of injuries

```
fatal = df[df["Injury.Severity"] == "Fatal"]
len(fatal)
```

```
5262
```

```
Minor = df[df["Injury.Severity"] == "Minor"]
len(Minor)
```

```
218
```

```
Non_Fatal = df[df["Injury.Severity"] == "Non-Fatal"]
len(Non_Fatal)
```

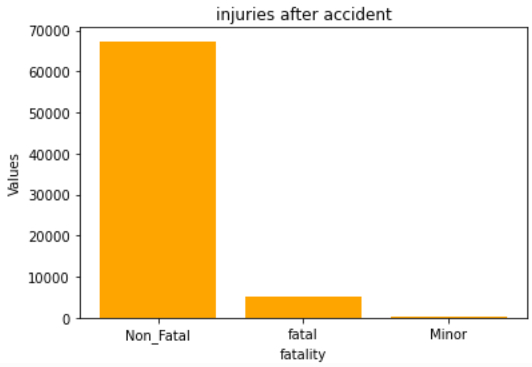
```
67357
```

```
#combining the data for a smooth flowing statement
print(f"there were {len(fatal)} fatal incidents, {len(Minor)} minor injuries and  {len(Non_Fatal)} non fatal injuries in total")
```

```
there were 5262 fatal incidents, 218 minor injuries and  67357 non fatal injuries in total
```

```
#creating a dicionary to plot the results in one graph
fatality_dict = { 'Non_Fatal': len(Non_Fatal),'fatal': len(fatal), 'Minor': len(Minor)}
```

```
# creating values for each axis
fatality = list(fatality_dict.keys())
values = list(fatality_dict.values())
plt.bar(fatality, values, color='orange')
plt.xlabel('fatality')
plt.ylabel('Values')
plt.title('injuries after accident')
plt.show();
#most of the injuries apear to be non-fatal
```



Finding different causes of the crashes

```
#cause of crash
df['Investigation.Type'].unique()
```

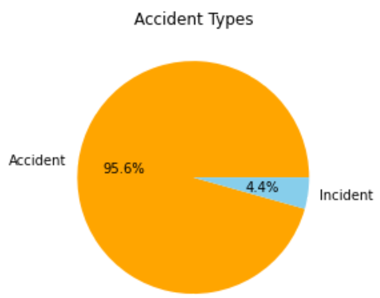
```
array(['Accident', 'Incident'], dtype=object)
```

```
accident = len(df[df['Investigation.Type'] == 'Accident'])
incident = len(df[df['Investigation.Type'] == 'Incident'])
```

```
#making the dataset cleaner by combining the information into a singe line
print(f"there are {len(df[df['Investigation.Type'] == 'Accident'])} accidents and {len(df[df['Investigation.Type'] == 'Incident'])} inc:
```

```
there are 85015 accidents and 3874 incidents
```

```
# Creating dictionary with data
Investigation_type = { 'Accident': accident, 'Incident': incident }
# Creating values for the pie chart
labels = list(Investigation_type.keys())
values = list(Investigation_type.values())
plt.pie(values, labels=labels, colors=['orange', 'skyblue'], autopct='%1.1f%%')
plt.title('Accident Types')
plt.show();
```



Investigating factors that could cause the accident

```
df["Weather.Condition"].unique()
```

```
array(['UNK', 'IMC', 'VMC', '', 'Unk'], dtype=object)
```

```
VMC = len(df[df["Weather.Condition"] == 'VMC'])
#"VMC" stands for Visual Meteorological Conditions. This indicates that the weather conditions are good enough for pilots to operate
```

```
IMC = len(df[df["Weather.Condition"] == 'IMC'])
#"IMC" stands for Instrument Meteorological Conditions. These conditions occur when visibility is poor, typically due to weather factor:
```

```
UNK = len(df[df["Weather.Condition"] == 'UNK' ]) #unk = unknown or not recorded
```

```
Unk = len(df[df["Weather.Condition"] == 'Unk']) #unk = unknown or not recorded
```

```
blank = len(df[df["Weather.Condition"] == ''])
blank #blank information
```

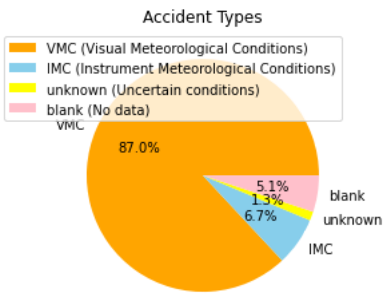
```
4492
```

```
unknown = UNK + Unk # adding the unknowns
unknown
```

```
1118
```

```
# pie chart of data
# Creating dictionary with data
Investigation_type = { 'VMC': VMC, 'IMC': IMC, 'unknown': unknown, 'blank': blank}
# Creating values for the pie chart
labels = list(Investigation_type.keys())
values = list(Investigation_type.values())
plt.pie(values, labels=labels, colors=['orange', 'skyblue', 'yellow', 'pink'], autopct='%1.1f%%')
plt.title('Accident Types')
plt.legend(['VMC (Visual Meteorological Conditions)',
            'IMC (Instrument Meteorological Conditions)',
            'unknown (Uncertain conditions)',
            'blank (No data)'],
           loc="best")
plt.show();
```

```
#finding out why most of the accidents happen during visible conditions
#mandate all pilots/airlines to record flight data
```



Investigating other factors

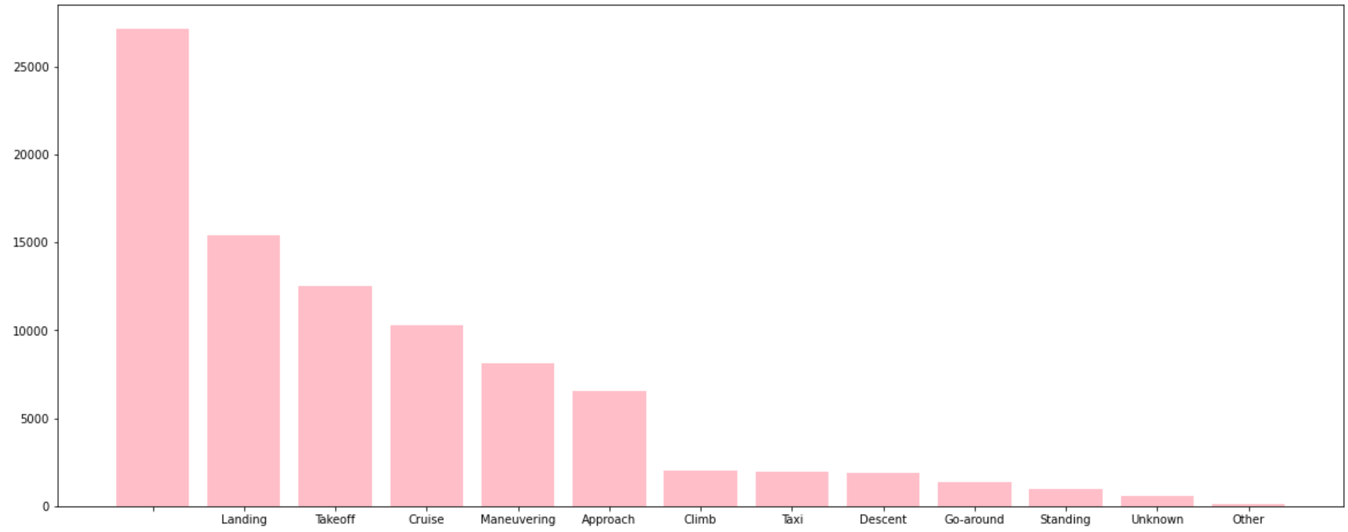
```
weather_investigation = df[df["Weather.Condition"] == "VMC"]["Broad.phase.of.flight"]
# tying to identify which phase of flight has the highest rate of accidents during normal conditions
```

```
df["Broad.phase.of.flight"].value_counts()
#majour accidents caused at landing
```

```
              27165
Landing       15428
Takeoff       12493
Cruise        10269
Maneuvering    8144
Approach       6546
Climb          2034
Taxi           1958
Descent        1887
Go-around      1353
Standing        945
Unknown         548
Other           119
Name: Broad.phase.of.flight, dtype: int64
```

```
#plotting a bargraph for phase of flight vs accident
fig, ax = plt.subplots(figsize=(20,8))
```

```
phase = df["Broad.phase.of.flight"].value_counts().keys()
value = df["Broad.phase.of.flight"].value_counts().values
plt.bar(phase, value, color = 'pink')
plt.show();
# recomendations
# train pilots on landing
#record more inflight weather conditions
#27165 entries are left blank
```




```
df['Report.Status'].value_counts()
#What Is Probable Cause
#Probable cause is legal justification for a police officer to make an arrest, obtain a warrant, or search a person or his property.
```

```
Probable Cause
61754

6380
Foreign
1999
<br /><br />
167
Factual
145

...
The pilot's failure to maintain sufficient airspeed following a loss of engine power, which resulted in the airplane exceeding its critical angle of attack and an aerodynamic stall. Contributing to the accident was the pilot's inadvertent placement of the fuel selector in the "Off" position before takeoff, which resulted in fuel starvation and a total loss of engine power.        1
Failure of the pilot to maintain a safe clearance from the terrain while manuevering for the landing surface.
1
The pilot's inadequate preflight which resulted in insufficient fuel on board to complete the flight.  A factor associated with the accident was the soft terrain encountered during the forced landing.
1
A total loss of engine power due to fuel starvation for undetermined reasons.\n
1
The pilot's improper decision to attempt an off-airport departure from rough terrain without inspecting the takeoff area, which resulted in a collision with a shallow rut during takeoff and the nose landing gear collapsing. \n
1
Name: Report.Status, Length: 17077, dtype: int64
```

```
probable = len(df[df['Report.Status'] == 'Probable Cause'])
# finding the number of accidents reported as probable
```

```
foreign = len(df[df['Report.Status'] == 'Foreign'])
# finding the number of accidents reported as foreign
```

```
len(df[df['Report.Status'] == '<br /><br />'])
# finding the number of accidents reported incorectly
```

```
167
```

```
factual = len(df[df['Report.Status'] == 'Factual'])
# finding the number of accidents reported as factual
```

Start coding or generate with AI.

```
to_delete = []
for i, row in df.iterrows():
    if '<br /><br />' in row.values:
        to_delete.append(i)

# Dropping rows with the identified indices
df_cleaned = df.drop(index=to_delete)
```

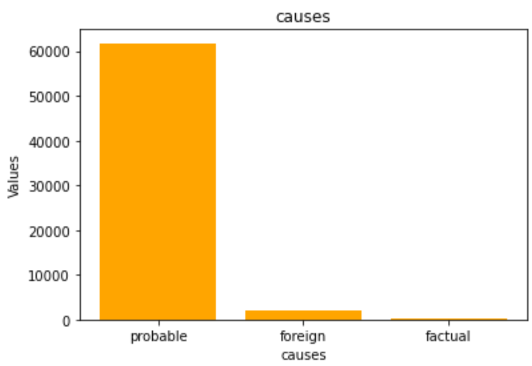
```
df.shape
```

```
(88889, 17)
```

```
df_cleaned.shape
```

```
(88722, 17)
```

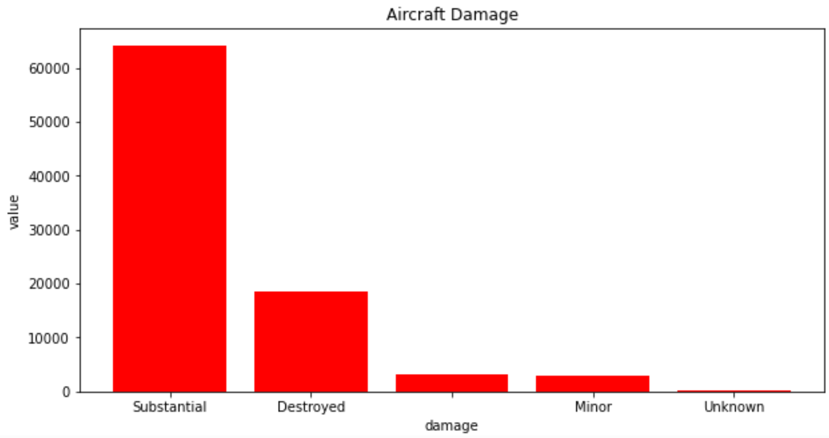
```
#creating a dicionary to plot the results in one graph
causes_dict = { 'probable': probable,'foreign': foreign, 'factual': factual}

# creating values for each axis
cause = list(causes_dict.keys())
values = list(causes_dict.values())
plt.bar(cause, values, color='orange')
plt.xlabel('causes')
plt.ylabel('Values')
plt.title('causes')
plt.show();

#most accidents are caused due to pilot error
#possible solution is to increade training to reduce errors
#each report status statement is different each statement has to be read individually to trully understand the cause of the crash
```




```
#conditions of the planes after the crash
df['Aircraft.damage'].value_counts()
```

```
Substantial    64148
Destroyed      18623
                3194
Minor           2805
Unknown          119
Name: Aircraft.damage, dtype: int64
```

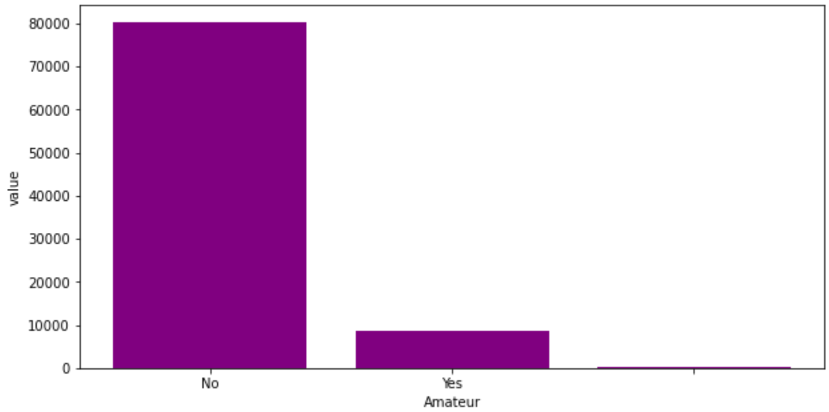
```
fig, ax = plt.subplots(figsize=(10,5))
damage = df['Aircraft.damage'].value_counts().keys()
value = df['Aircraft.damage'].value_counts().values
plt.bar(damage, value, color = 'red')
plt.xlabel('damage')
plt.ylabel('value')
plt.title('Aircraft Damage')
plt.show();
```




```
#does ameture built planes cause more accidents?
df['Amateur.Built'].value_counts()
```

```
No      80312
Yes      8475
          102
Name: Amateur.Built, dtype: int64
```

```
fig, ax = plt.subplots(figsize=(10,5))
Amateu = df['Amateur.Built'].value_counts().keys()
value = df['Amateur.Built'].value_counts().values
plt.bar(Amateu, value, color = 'purple')
plt.xlabel('Amateur')
plt.ylabel('value')
plt.title('')
plt.show();
```

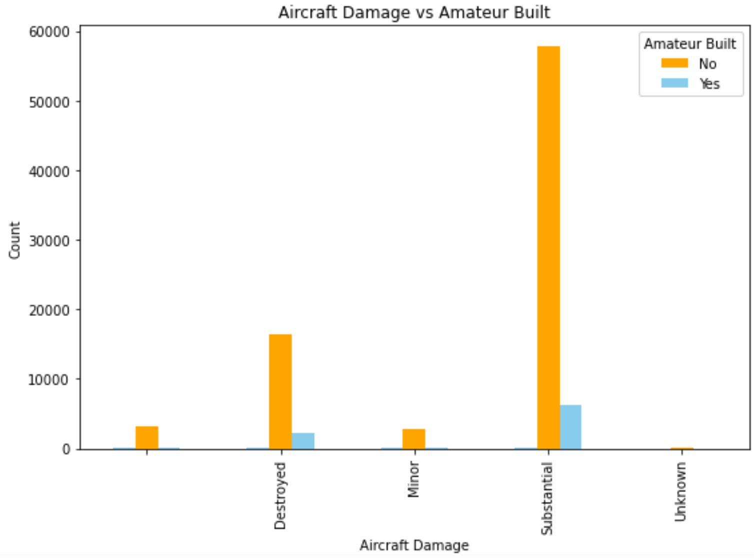



```
#method to show tht ametaur build does not coorelate with accident proneness
ametaur_damage = df[['Aircraft.damage', 'Amateur.Built']]
ametaur_damage
```

	Aircraft.damage	Amateur.Built
1	Destroyed	No
2	Destroyed	No
3	Destroyed	No
4	Destroyed	No
5	Destroyed	No
...	...	...
88885		No
88886		No
88887	Substantial	No
88888		No
88889		No

88889 rows × 2 columns

```
grouped_data = df.groupby(['Aircraft.damage', 'Amateur.Built']).size().unstack()
grouped_data.plot(kind='bar', figsize=(8, 6), color=['skyblue', 'orange'])
plt.title('Aircraft Damage vs Amateur Built')
plt.xlabel('Aircraft Damage')
plt.ylabel('Count')
plt.legend(title='Amateur Built')
plt.tight_layout()
#majoritty of the crashes are done by profesional built aircrafts and most of the damage is substantial
```

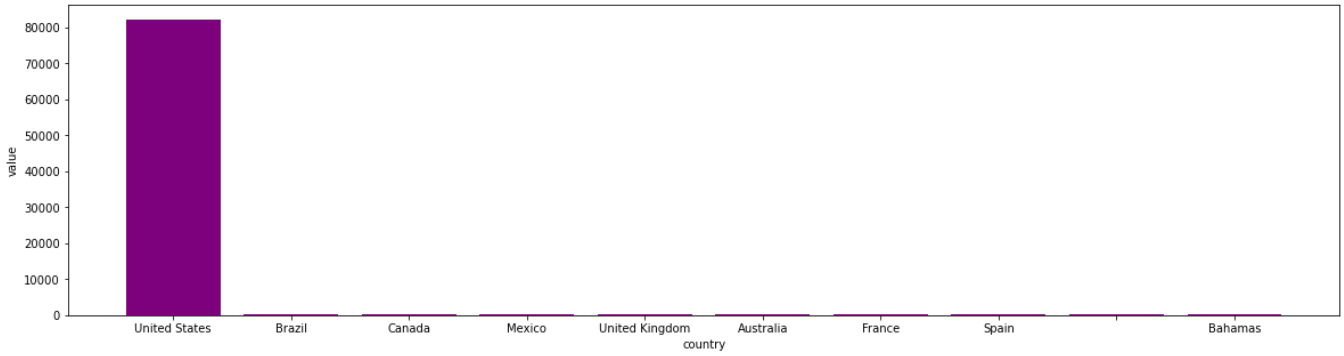


which country has the most accidents?

```
df['Country'].value_counts()[:10]
```

```
United States     82248
Brazil              374
Canada              359
Mexico              358
United Kingdom      344
Australia           300
France              236
Spain               226
                    226
Bahamas             216
Name: Country, dtype: int64
```

```
#the data is more in favour of the USA seems like there is missing information with the other countries
#we will focus on USA
fig, ax = plt.subplots(figsize=(20,5))
country = df['Country'].value_counts()[:10].keys()
value = df['Country'].value_counts()[:10].values
plt.bar(country, value, color = 'purple')
plt.xlabel('country')
plt.ylabel('value')
plt.title('')
plt.show();
```



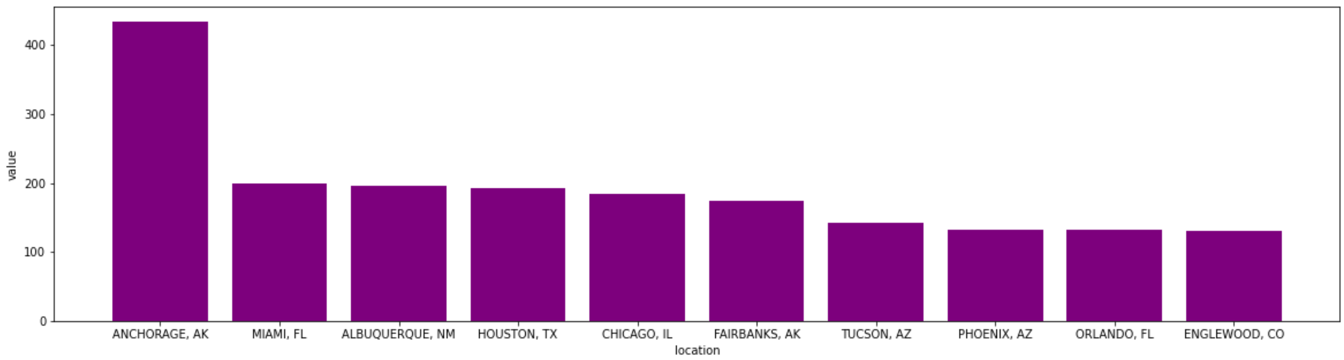
```
len(df[(df['Country'] == 'United States') & (df['Weather.Condition'] == 'VMC')])
#weather conditions are mostly clear so weather is not a main reason to crashes
```

```
75317
```

```
df['Location'].value_counts()
```

```
ANCHORAGE, AK       434
MIAMI, FL           200
ALBUQUERQUE, NM     196
HOUSTON, TX         193
CHICAGO, IL         184
                   ...
Walker, CA            1
DALE, IN              1
Windsor, CO           1
Ormond beach, FL      1
Brownsville, CA       1
Name: Location, Length: 27759, dtype: int64
```

```
#to find out which state has the highest accident
fig, ax = plt.subplots(figsize=(20,5))
location = df['Location'].value_counts()[:10].keys()
value = df['Location'].value_counts()[:10].values
plt.bar(location, value, color = 'purple')
plt.xlabel('location')
plt.ylabel('value')
plt.title('')
plt.show();
```




```
len(df[(df['Location'] == 'ANCHORAGE, AK') & (df['Weather.Condition'] == 'VMC')])
#414 of the accidents happened during VMC
```

```
414
```

Analysis

The data shows that 95.6% of crashes are accidents, which happen more often than incidents. Most accidents are caused by pilot error, but there are other factors too, like overconfidence, busy air traffic, distractions, runway problems, and miscommunications with traffic control. Surprisingly, many accidents happen in clear weather. Landing is the riskiest part of a flight because it involves a lot of factors and little time to fix mistakes. However, as technology improves, airplanes have become safer, and the number of injuries has gone down.

Recomendations

- Train pilots more on landing and takeoff emergency scenarios
- Increase their mandatory flight hours on a training simulator as Prospective commercial pilots typically log at least 250 hours of flying time to earn their license
- Optimize air traffic management systems with AI-based solutions for smoother coordination.
- Improve communication tools between pilots and air traffic controllers.
- Expand infrastructure to handle increasing air traffic efficiently.
- Standardize communication protocols to reduce ambiguity.
- Invest in reliable backup systems for uninterrupted communication.

Links

- Powerpoint presentation: https://docs.google.com/presentation/d/1ytu83b6jLZPcnG6btw22G82pGBaod7TEUOXxrFlFVCs/edit?usp=sharing

- tableau public Tableau public: Aviation Accident Project | Tableau Public

```
Start coding or generate with AI.
```