Final Project Submission

Please fill out:

- Student name: George Mochama Edwin
- Student pace: Part time
- Scheduled project review date/time: 08/09/2024
- Instructor name: Mr William Okomba
- Blog post URL: https://github.com/mojay6111/dsc-phase-1-project_1/tree/master

OVERVIEW

Business Problem

Your company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating airplanes for commercial and private enterprises, but do not know anything about the potential risks of aircraft. You are charged with determining which aircraft are the lowest risk for the company to start this new business endeavor. You must then translate your findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase.

Business Understanding

As we expand into the aviation industry, I recognize the importance of thoroughly understanding the risks associated with purchasing and operating aircraft for both commercial and private enterprises. To ensure our success and minimize potential hazards, I will use the CRISP-DM methodology to systematically analyze aviation risks. My focus will be on identifying the aircraft with the lowest risk, predicting which models are more likely to be involved in accidents, and examining the environmental and operational factors that contribute to these risks.

To achieve this, I will gather and analyze data on aircraft specifications, safety records, incident reports, and more. By applying advanced modeling techniques, I'll be able to provide actionable insights that will guide our decisions on which aircraft to purchase and how to operate them safely. Ultimately, my goal is to deliver clear, data-driven recommendations that will help us confidently enter the aviation industry while ensuring the highest level of safety and operational efficiency.

Objectives

- To determine which aircraft/make have the lowest risk for our company.
- To identify aircraft models most likely to be involved in accidents.
- To pinpoint locations or environments where incidents happen more frequently to guide our operational decisions.
- To evaluate whether we should invest in amateur-built aircraft.
- To analyze the dataset to find aircraft types with the lowest incident and accident rates.

- To determine the key factors that contribute to aviation risks, such as weather, flight phase, and aircraft age.
- To examine the purpose of the flight to see how it affects risk levels.

Questions To Consider

- What specific criteria define "low risk" for aircraft, and how should these criteria be prioritized.
- How do environmental factors like weather and geography influence the likelihood of an accident for certain aircraft types?
- What impact does the purpose of the flight (e.g., commercial, cargo, private) have on the overall risk profile of different aircraft models?

Data Understanding and Analysis

Importing necessary Python Libraries

In [64]:

```
# Numpy for numerical operations, especially for working with arrays
import numpy as np

# Pandas  for data manipulation and analysis, especially for working with DataFrames
import pandas as pd

# Seaborn for data visualization, building on top of Matplotlib to provide enhanced 
import seaborn as sns

# Matplotlib's pyplot for creating static, animated, and interactive visualizations
import matplotlib.pyplot as plt

# %matplotlib inline to display plots inline in Jupyter notebooks.
%matplotlib inline
```

Loading the data

In [65]:

```
# Loading the data.
data = pd.read_csv('AviationData.csv',  encoding='ISO-8859-1')
```

```
<ipython-input-65-b29267527608>:2: DtypeWarning: Columns (6,7,28) have mixed types. S
pecify dtype option on import or set low_memory=False.
  data = pd.read_csv('AviationData.csv',  encoding='ISO-8859-1')
```

Dataset Overview and Validation

In [66]:

```
# Display the first few rows of the dataset to get an overview of its structure
data.head()
```

Out[66]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latituc
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Na

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latituc
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Na
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.92222
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Na
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	Na

In [67]:
```
# Display the last few rows of the dataset to check the final entries
data.tail()
```

Out[67]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitu
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	N
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	N
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	34152
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	N
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	N

In [68]:
```
# Display the list of column names in the dataset
data.columns
```

Out[68]:
```
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')
```

In [69]:
```
# Count the total number of columns in the dataset, ensuring no duplicates
data.columns.value_counts().sum()
```

Out[69]:
```
31
```

In [70]:
```
# Checking for any missing values in the dataset
data.isnull().sum()
```

Out[70]:
```
Event.Id                  0
Investigation.Type        0
Accident.Number           0
Event.Date                0
```

```
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38640
Airport.Name              36099
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1317
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7077
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6381
Publication.Date          13771
dtype: int64
```

In [71]:

```python
# Get summary statistics for numerical columns
data.describe()
```

Out[71]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjure
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.00000
mean	1.146585	0.647855	0.279881	0.357061	5.32544
std	0.446510	5.485960	1.544084	2.235625	27.91363
min	0.000000	0.000000	0.000000	0.000000	0.00000
25%	1.000000	0.000000	0.000000	0.000000	0.00000
50%	1.000000	0.000000	0.000000	0.000000	1.00000
75%	1.000000	0.000000	0.000000	0.000000	2.00000
max	8.000000	349.000000	161.000000	380.000000	699.00000

In [72]:

```python
# Get summary statistics for numerical columns, transposed for easier reading
data.describe().T
```

Out[72]:

	count	mean	std	min	25%	50%	75%	max
Number.of.Engines	82805.0	1.146585	0.446510	0.0	1.0	1.0	1.0	8.0
Total.Fatal.Injuries	77488.0	0.647855	5.485960	0.0	0.0	0.0	0.0	349.0
Total.Serious.Injuries	76379.0	0.279881	1.544084	0.0	0.0	0.0	0.0	161.0
Total.Minor.Injuries	76956.0	0.357061	2.235625	0.0	0.0	0.0	0.0	380.0
Total.Uninjured	82977.0	5.325440	27.913634	0.0	0.0	1.0	2.0	699.0

In [73]:
```
# Check the data types of each column
data.dtypes
```

Out[73]:
```
Event.Id                  object
Investigation.Type        object
Accident.Number           object
Event.Date                object
Location                  object
Country                   object
Latitude                  object
Longitude                 object
Airport.Code              object
Airport.Name              object
Injury.Severity           object
Aircraft.damage           object
Aircraft.Category         object
Registration.Number       object
Make                      object
Model                     object
Amateur.Built             object
Number.of.Engines        float64
Engine.Type               object
FAR.Description           object
Schedule                  object
Purpose.of.flight         object
Air.carrier               object
Total.Fatal.Injuries     float64
Total.Serious.Injuries   float64
Total.Minor.Injuries     float64
Total.Uninjured          float64
Weather.Condition         object
Broad.phase.of.flight     object
Report.Status             object
Publication.Date          object
dtype: object
```

In [74]:
```
# Display the number of unique values for each column
data.nunique()
```

Out[74]:
```
Event.Id                 87951
Investigation.Type           2
Accident.Number          88863
Event.Date               14782
Location                 27758
Country                    219
Latitude                 25592
Longitude                27156
Airport.Code             10375
Airport.Name             24871
Injury.Severity            109
Aircraft.damage              4
Aircraft.Category           15
Registration.Number      79105
Make                      8237
Model                    12318
Amateur.Built                2
Number.of.Engines            7
Engine.Type                 13
FAR.Description             31
Schedule                     3
Purpose.of.flight           26
Air.carrier              13590
Total.Fatal.Injuries       125
Total.Serious.Injuries      50
Total.Minor.Injuries        57
Total.Uninjured            379
Weather.Condition            4
Broad.phase.of.flight       12
```

```
Report.Status            17075
Publication.Date          2924
dtype: int64
```

In [75]:
```
# Check for any duplicate rows in the dataset
data.duplicated().sum()
```

Out[75]:
```
0
```

DATA CLEANING

In [76]:
```
# Calculate the percentage of missing values for each column
missing_data_percentage = data.isnull().mean() * 100
missing_data_percentage.count()
```

Out[76]:
```
31
```

In [77]:
```
columns_with_missing_data = missing_data_percentage[missing_data_percentage > 0]
columns_with_missing_data
```

Out[77]:
```
Location                   0.058500
Country                    0.254250
Latitude                  61.320298
Longitude                 61.330423
Airport.Code              43.469946
Airport.Name              40.611324
Injury.Severity            1.124999
Aircraft.damage            3.593246
Aircraft.Category         63.677170
Registration.Number        1.481623
Make                       0.070875
Model                      0.103500
Amateur.Built              0.114750
Number.of.Engines          6.844491
Engine.Type                7.961615
FAR.Description           63.974170
Schedule                  85.845268
Purpose.of.flight          6.965991
Air.carrier               81.271023
Total.Fatal.Injuries      12.826109
Total.Serious.Injuries    14.073732
Total.Minor.Injuries      13.424608
Total.Uninjured            6.650992
Weather.Condition          5.053494
Broad.phase.of.flight     30.560587
Report.Status              7.178616
Publication.Date          15.492356
dtype: float64
```

In [78]:
```
# Displaying columns with more than a 40% percentage of missing data
high_missing_data = missing_data_percentage[missing_data_percentage > 40].index
high_missing_data
```

Out[78]:
```
Index(['Latitude', 'Longitude', 'Airport.Code', 'Airport.Name',
       'Aircraft.Category', 'FAR.Description', 'Schedule', 'Air.carrier'],
      dtype='object')
```

Dropping columns with too much missing data(more that 40 % and not useful in data)

In [79]:
```
# Dropping the,mkz specified columns from the DataFrame
data.drop(columns=['Latitude', 'Longitude', 'Air.carrier', 'Airport.Code', 'Publicat
```

In [80]:
```
# Checking the remaing columns
data.columns.value_counts()
```

Out[80]:
```
Event.Id                 1
Investigation.Type       1
Broad.phase.of.flight    1
Weather.Condition        1
Total.Uninjured          1
Total.Minor.Injuries     1
Total.Serious.Injuries   1
Total.Fatal.Injuries     1
Purpose.of.flight        1
Schedule                 1
FAR.Description          1
Engine.Type              1
Number.of.Engines        1
Amateur.Built            1
Model                    1
Make                     1
Registration.Number      1
Aircraft.Category        1
Aircraft.damage          1
Injury.Severity          1
Airport.Name             1
Country                  1
Location                 1
Event.Date               1
Accident.Number          1
Report.Status            1
dtype: int64
```

In [81]:
```
data.head()
```

Out[81]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airport.
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	

In [82]:
```
# Fill null values in the 'Location' column with 'Unknown'
data['Location'].fillna('Unknown', inplace=True)

# Verifying that the null values have been replaced
print(data['Location'].isnull().sum())
```

```
0
```

In [83]:
```
# Fill null values in the 'Country' column with 'Unknown'
data['Country'].fillna('Unknown', inplace=True)

# Verifying that the null values have been replaced
print(data['Country'].isnull().sum())
```

```
0
```

In [84]:
```
# Fill null values in the 'Airport.Name' column with 'Unknown'
data['Airport.Name'].fillna('Unknown', inplace=True)
```

```
# Verifying that the null values have been replaced
print(data['Airport.Name'].isnull().sum())
```

```
0
```

In [85]:
```
# Forward fill missing values in Aircraft.Damage column
data['Aircraft.damage'] = data['Aircraft.damage'].ffill()

# Verifying that the null values have been replaced
print(data['Aircraft.damage'].isnull().sum())
```

```
0
```

In [86]:
```
# Finding the mode (most frequent value) of the 'Aircraft.Category' column
#mode_value = data['Aircraft.Category'].mode()[0]
#mode_value = data['Aircraft.Category'].mode(dropna=True)

# Filling the null values in the 'Aircraft.Category' column with the mode value
data['Aircraft.Category'].fillna('Unknown', inplace=True)

# Verifying that the null values have been replaced
print(data['Aircraft.Category'].isnull().sum())
```

```
0
```

To make the values in the Injury.Severity column uniform (i.e., removing the numbers in brackets and ensuring the values read just "Fatal" or "Nonfatal"), I will use a regular expression to remove any text within parentheses.

In [87]:
```
# Import regular expressions module
import re

# Remove any numbers or text within parentheses in 'Injury.Severity' column
data['Injury.Severity'] = data['Injury.Severity'].replace(to_replace=r'Fatal.*', val

# Verify the changes
print(data['Injury.Severity'].unique())
```

```
['Fatal' 'Non-Fatal' 'Incident' 'Unavailable' nan 'Minor' 'Serious']
```

In [88]:
```
# Replacing null values in the 'Injury.Severity' column with 'Unknown'
data['Injury.Severity'].fillna('Unknown', inplace=True)

# Verifying that the null values have been replaced
print(data['Injury.Severity'].isnull().sum())
```

```
0
```

In [89]:
```
# Replacing null values in the 'Registration.Number' column with 'Unknown'
data['Registration.Number'].fillna('Unknown', inplace=True)

# Verifying that the null values have been replaced
print(data['Registration.Number'].isnull().sum())
```

```
0
```

In [90]:
```
# Capitalizing all values in the 'Make' column
data['Make'] = data['Make'].str.upper()

# Verifying the changes by displaying unique values
print(data['Make'].unique())
```

```
['STINSON' 'PIPER' 'CESSNA' ... 'JAMES R DERNOVSEK' 'ORLICAN S R O'
 'ROYSE RALPH L']
```

To ensure that all values in the Make column that start with or contain the phrase "CESSNA" remain as just "CESSNA" (removing any text before or after it), I will use a regular expression to match and replace the relevant entries.

In [91]:
```
# Replacing any value that contains 'CESSNA' with just 'CESSNA'
data['Make'] = data['Make'].replace(to_replace=r'.*CESSNA.*', value='CESSNA', regex=

# Verifying the changes by displaying unique values
print(data['Make'].unique())
```

```
['STINSON' 'PIPER' 'CESSNA' ... 'JAMES R DERNOVSEK' 'ORLICAN S R O'
 'ROYSE RALPH L']
```

In [92]:
```
# Replacing 'CESNA' with 'CESSNA' in the 'Make' column
data['Make'] = data['Make'].replace('CESNA', 'CESSNA')

# Verifying the changes by displaying unique values
print(data['Make'].unique())
```

```
['STINSON' 'PIPER' 'CESSNA' ... 'JAMES R DERNOVSEK' 'ORLICAN S R O'
 'ROYSE RALPH L']
```

In [93]:
```
# Counting occurrences of each unique value in the 'Make' column
make_counts = data['Make'].value_counts()

# Displaying the counts
print(make_counts)
```

```
CESSNA           27216
PIPER            14870
BEECH             5372
BOEING            2745
BELL              2722
                 ...
LUTES                1
IZATT                1
MINCE                1
DANA A. MOORE        1
ROYSE RALPH L        1
Name: Make, Length: 7571, dtype: int64
```

In [94]:
```
# Checking for null values in the 'Make' column
null_count_make = data['Make'].isnull().sum()

# Displaying the count of null values
print(null_count_make)
```

```
63
```

In [95]:
```
# Replacing null values in the 'Make' column with 'UNKNOWN'
data['Make'].fillna('UNKNOWN', inplace=True)

# Verifying that the null values are being replaced
print(data['Make'].isnull().sum())
```

```
0
```

In [96]:
```
# Checking for null values in the 'Model' column
null_count_model = data['Model'].isnull().sum()

null_count_model
```

Out[96]:
```
92
```

In [97]:
```python
# Replacing null values in the 'Model' column with 'Unknown'
data['Model'].fillna('Unknown', inplace=True)

# Verifying that the null values are being replaced
print(data['Model'].isnull().sum())
```

```
0
```

In [98]:
```python
# Finding the mode value of the 'Amateur.Built' column
mode_value = data['Amateur.Built'].mode()[0]

# Replacing null values in the 'Amateur.Built' column with the mode value
data['Amateur.Built'].fillna(mode_value, inplace=True)

# Verifying that the null values are being replaced
print(data['Amateur.Built'].isnull().sum())
```

```
0
```

In [99]:
```python
# Finding the mode value of the 'Number.of.Engines' column
mode_value = data['Number.of.Engines'].mode()[0]

# Replacing null values with the mode value
data['Number.of.Engines'].fillna(mode_value, inplace=True)

# Verifying that the null values are being replaced
print(data['Number.of.Engines'].isnull().sum())
```

```
0
```

In [100…
```python
# Finding the mode value of the 'Engine.Type' column
mode_value = data['Engine.Type'].mode()[0]

# Replacing null values in the 'Engine.Type' column with the mode value
data['Engine.Type'].fillna(mode_value, inplace=True)

# Verifying that the null values are being replaced
print(data['Engine.Type'].isnull().sum())
```

```
0
```

In [101…
```python
pd.set_option('display.max_columns', None)
data.head(10)
```

Out[101…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airpoı
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	U
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	U
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	U
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	U
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	U
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	United States	U
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States	U

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airpo
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	BLA A
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States	HA
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	United States	JACKS

In [102…

```
# Replacing null values with 'Unknown' in the FAR.Description
data['FAR.Description'].fillna('Unknown', inplace=True)

# Verifying that the null values are being replaced
print(data['FAR.Description'].isnull().sum())
```

```
0
```

In [103…

```
# Replacing null values in the 'Schedule' column with 'Unknown'
data['Schedule'].fillna('Unknown', inplace=True)

# Verifying that the null values are being replaced
print(data['Schedule'].isnull().sum())
```

```
0
```

In [104…

```
# Replacing null values in the 'Purpose of Flight' column with 'Unknown'
data['Purpose.of.flight'].fillna('Unknown', inplace=True)

# Verifying that the null values are being replaced
print(data['Purpose.of.flight'].isnull().sum())
```

```
0
```

In [105…

```
# Inferring at least 1 fatality if severity is 'Fatal'
data.loc[(data['Total.Fatal.Injuries'].isnull()) & (data['Injury.Severity'] == 'Fata

# Setting 0 fatality if severity is 'Non-Fatal'
data.loc[(data['Total.Fatal.Injuries'].isnull()) & (data['Injury.Severity'] == 'Non-

# Replacing null values in 'Total.Fatal.Injuries' with 'Unknown'
data['Total.Fatal.Injuries'].fillna('Unknown', inplace=True)

# Verifying the changes
print(data['Total.Fatal.Injuries'].isnull().sum())
```

```
0
```

In [106…

```
# Replacing null values in 'Total.Serious.Injuries' with 'Unknown'
data['Total.Serious.Injuries'].fillna('Unknown', inplace=True)

# Replacing null values in 'Total.Minor.Injuries' with 'Unknown'
data['Total.Minor.Injuries'].fillna('Unknown', inplace=True)

# Replacing null values in 'Total.Uninjured' with 'Unknown'
data['Total.Uninjured'].fillna('Unknown', inplace=True)

# Verifying the changes
print(data[['Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].is
```

```
Total.Serious.Injuries     0
Total.Minor.Injuries       0
```

```
Total.Uninjured          0
dtype: int64
```

In [107…

```python
# Replacing null values in 'Weather.Condition' with 'UNK'
data['Weather.Condition'].fillna('UNK', inplace=True)
# Replacing 'Unk' with 'UNK' in the Weather.Condition column
data['Weather.Condition'].replace('Unk', 'UNK', inplace=True)

# Replacing null values in 'Broad.phase.of.flight' with 'Unknown'
data['Broad.phase.of.flight'].fillna('Unknown', inplace=True)

# Verifying the changes
print(data[['Weather.Condition', 'Broad.phase.of.flight']].isnull().sum())
```

```
Weather.Condition        0
Broad.phase.of.flight    0
dtype: int64
```

In [108…

```python
# Replacing null values in 'Report.Status' with 'Not Reported'
data['Report.Status'].fillna('Not Reported', inplace=True)

# Verifying the changes
print(data['Report.Status'].isnull().sum())
```

```
0
```

In [109…

```python
# Displaying the total number of null values in each column
data.isnull().sum()
```

Out[109…

```
Event.Id                  0
Investigation.Type        0
Accident.Number           0
Event.Date                0
Location                  0
Country                   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
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
dtype: int64
```

In [110…

```python
data.head()
```

Out[110…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airport.
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	Unl
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	Unl

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airport.
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	Unl
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	Unl
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	Unl

In [111…
```
# Saving the cleaned DataFrame to a CSV file
data.to_csv('cleaned_aircraft_data.csv', index=False)
```

In [112…
```
# Saving the cleaned DataFrame to an Excel file
#data.to_excel('cleaned_aircraft_data.xlsx', index=False)
```

EXPLORATORY DATA ANALYSIS

In [113…
```
data.describe().T
```

Out[113…

	count	mean	std	min	25%	50%	75%	max
Number.of.Engines	88889.0	1.136552	0.432545	0.0	1.0	1.0	1.0	8.0

In [114…
```
# Generating descriptive statistics for numerical columns
summary_statistics = data.describe(include=[np.number])

# Displaying the summary statistics
print(summary_statistics)
```

```
       Number.of.Engines
count       88889.000000
mean            1.136552
std             0.432545
min             0.000000
25%             1.000000
50%             1.000000
75%             1.000000
max             8.000000
```

In [115…
```
# Generating descriptive statistics for categorical columns
categorical_columns = data.select_dtypes(include=['object']).columns

# Displaying the count of unique values for each categorical column
for column in categorical_columns:
    print(f"--- {column} ---")
    print(data[column].value_counts(dropna=False))
    print("\n")
```

```
--- Event.Id ---
20001212X19172    3
20001214X45071    3
20220730105623    2
20051213X01965    2
20001212X16765    2
                 ..
20001211X14216    1
20001211X14239    1
20001211X14207    1
20001211X14204    1
```

```
20221230106513    1
Name: Event.Id, Length: 87951, dtype: int64


--- Investigation.Type ---
Accident    85015
Incident     3874
Name: Investigation.Type, dtype: int64


--- Accident.Number ---
CEN22LA149    2
WPR23LA041    2
WPR23LA045    2
DCA22WA214    2
DCA22WA089    2
             ..
LAX92FA065    1
ANC92T#A12    1
MIA92LA049    1
NYC92LA048    1
ERA23LA097    1
Name: Accident.Number, Length: 88863, dtype: int64


--- Event.Date ---
1984-06-30    25
1982-05-16    25
2000-07-08    25
1983-08-05    24
1984-08-25    24
              ..
2014-03-16     1
2014-03-15     1
2014-03-12     1
2014-03-10     1
2022-12-29     1
Name: Event.Date, Length: 14782, dtype: int64


--- Location ---
ANCHORAGE, AK       434
MIAMI, FL           200
ALBUQUERQUE, NM     196
HOUSTON, TX         193
CHICAGO, IL         184
                   ...
MALLARDS LDG, GA      1
LODGEPOLE, MT         1
VERNILLION, SD        1
MCMECHEN, WV          1
Brasnorte,            1
Name: Location, Length: 27758, dtype: int64


--- Country ---
United States                       82248
Brazil                                374
Canada                                359
Mexico                                358
United Kingdom                        344
                                    ...
Saint Vincent and the Grenadines        1
Cambodia                                1
Malampa                                 1
AY                                      1
Turks and Caicos Islands                1
Name: Country, Length: 219, dtype: int64
```

```
--- Airport.Name ---
Unknown                            36106
Private                              240
PRIVATE                              224
Private Airstrip                     153
NONE                                 146
                                   ...  
WESTCHESTER COUNTY ARPT                1
IL VALLEY PARACHUTE CLUB               1
LAUGHLIN/BULLHEAD                      1
Otsego County Airport                  1
WICHITA DWIGHT D EISENHOWER NT         1
Name: Airport.Name, Length: 24871, dtype: int64


--- Injury.Severity ---
Non-Fatal      67357
Fatal          17826
Incident        2219
Unknown         1000
Minor            218
Serious          173
Unavailable       96
Name: Injury.Severity, dtype: int64


--- Aircraft.damage ---
Substantial    66154
Destroyed      19631
Minor           2976
Unknown          128
Name: Aircraft.damage, dtype: int64


--- Aircraft.Category ---
Unknown              56616
Airplane             27617
Helicopter            3440
Glider                 508
Balloon                231
Gyrocraft              173
Weight-Shift           161
Powered Parachute       91
Ultralight              30
WSFT                     9
Powered-Lift             5
Blimp                    4
UNK                      2
Rocket                   1
ULTR                     1
Name: Aircraft.Category, dtype: int64


--- Registration.Number ---
Unknown    1320
NONE        344
UNREG       126
None         65
UNK          13
           ... 
N93478        1
N519UA        1
N8840W        1
N21040        1
N9026P        1
Name: Registration.Number, Length: 79105, dtype: int64
```

```
--- Make ---
CESSNA           27216
PIPER            14870
BEECH             5372
BOEING            2745
BELL              2722
                 ...
LUTES                1
IZATT                1
MINCE                1
DANA A. MOORE        1
ROYSE RALPH L        1
Name: Make, Length: 7571, dtype: int64


--- Model ---
152             2367
172             1756
172N            1164
PA-28-140        932
150              829
                ...
GC-1-A             1
737-3S3            1
MBB-BK117-B2       1
GLASSAIR GL25      1
M-8 EAGLE          1
Name: Model, Length: 12318, dtype: int64


--- Amateur.Built ---
No     80414
Yes     8475
Name: Amateur.Built, dtype: int64


--- Engine.Type ---
Reciprocating      76607
Turbo Shaft         3609
Turbo Prop          3391
Turbo Fan           2481
Unknown             2051
Turbo Jet            703
None                  19
Geared Turbofan       12
Electric              10
LR                     2
NONE                   2
Hybrid Rocket          1
UNK                    1
Name: Engine.Type, dtype: int64


--- FAR.Description ---
Unknown                            56888
091                                18221
Part 91: General Aviation           6486
NUSN                                1584
NUSC                                1013
137                                 1010
135                                  746
121                                  679
Part 137: Agricultural               437
UNK                                  371
Part 135: Air Taxi & Commuter        298
PUBU                                 253
129                                  246
Part 121: Air Carrier                165
133                                  107
```

```
Part 129: Foreign                   100
Non-U.S., Non-Commercial             97
Non-U.S., Commercial                 93
Part 133: Rotorcraft Ext. Load       32
Public Use                           19
091K                                 14
ARMF                                  8
125                                   5
Part 125: 20+ Pax,6000+ lbs           5
107                                   4
Public Aircraft                       2
103                                   2
Part 91 Subpart K: Fractional         1
Armed Forces                          1
Part 91F: Special Flt Ops.            1
437                                   1
Name: FAR.Description, dtype: int64


--- Schedule ---
Unknown    76307
NSCH        4474
UNK         4099
SCHD        4009
Name: Schedule, dtype: int64


--- Purpose.of.flight ---
Personal                     49448
Unknown                      12994
Instructional                10601
Aerial Application            4712
Business                      4018
Positioning                   1646
Other Work Use                1264
Ferry                          812
Aerial Observation             794
Public Aircraft                720
Executive/corporate            553
Flight Test                    405
Skydiving                      182
External Load                  123
Public Aircraft - Federal      105
Banner Tow                     101
Air Race show                   99
Public Aircraft - Local         74
Public Aircraft - State         64
Air Race/show                   59
Glider Tow                      53
Firefighting                    40
Air Drop                        11
ASHO                             6
PUBS                             4
PUBL                             1
Name: Purpose.of.flight, dtype: int64


--- Total.Fatal.Injuries ---
0.0      70346
1.0       8883
2.0       5173
3.0       1589
4.0       1103
         ...
156.0        1
68.0         1
31.0         1
115.0        1
176.0        1
```

```
Name: Total.Fatal.Injuries, Length: 126, dtype: int64


--- Total.Serious.Injuries ---
0.0        63289
Unknown    12510
1.0         9125
2.0         2815
3.0          629
4.0          258
5.0           78
6.0           41
7.0           27
9.0           16
8.0           13
10.0          13
13.0           9
11.0           6
12.0           5
26.0           5
14.0           5
20.0           3
25.0           3
28.0           3
59.0           2
47.0           2
21.0           2
50.0           2
17.0           2
53.0           1
67.0           1
34.0           1
33.0           1
125.0          1
35.0           1
137.0          1
19.0           1
27.0           1
88.0           1
161.0          1
41.0           1
44.0           1
63.0           1
55.0           1
23.0           1
43.0           1
39.0           1
45.0           1
18.0           1
16.0           1
60.0           1
106.0          1
81.0           1
15.0           1
22.0           1
Name: Total.Serious.Injuries, dtype: int64


--- Total.Minor.Injuries ---
0.0        61454
Unknown    11933
1.0        10320
2.0         3576
3.0          784
4.0          372
5.0          129
6.0           67
7.0           59
9.0           22
```

```
8.0           20
13.0          14
10.0          11
12.0          11
14.0          10
11.0           9
17.0           8
19.0           6
18.0           6
24.0           5
22.0           5
25.0           4
16.0           4
15.0           4
33.0           4
20.0           3
21.0           3
26.0           3
23.0           3
32.0           3
27.0           3
50.0           2
30.0           2
36.0           2
31.0           2
28.0           2
42.0           2
38.0           2
57.0           1
65.0           1
84.0           1
43.0           1
35.0           1
380.0          1
47.0           1
68.0           1
200.0          1
71.0           1
58.0           1
171.0          1
39.0           1
96.0           1
29.0           1
69.0           1
62.0           1
45.0           1
125.0          1
40.0           1
Name: Total.Minor.Injuries, dtype: int64


--- Total.Uninjured ---
0.0        29879
1.0        25101
2.0        15988
Unknown     5912
3.0         4313
           ...
558.0          1
412.0          1
338.0          1
401.0          1
455.0          1
Name: Total.Uninjured, Length: 380, dtype: int64


--- Weather.Condition ---
VMC    77303
IMC     5976
```

```
UNK     5610
Name: Weather.Condition, dtype: int64


--- Broad.phase.of.flight ---
Unknown        27713
Landing        15428
Takeoff        12493
Cruise         10269
Maneuvering     8144
Approach        6546
Climb           2034
Taxi            1958
Descent         1887
Go-around       1353
Standing         945
Other            119
Name: Broad.phase.of.flight, dtype: int64


--- Report.Status ---
Probable Cause
61754
Not Reported
6381
Foreign
1999
<br /><br />
167
Factual
145

...
The pilot's incapacitation due to a ruptured berry aneurysm during takeoff.
1
The unauthorized operation of the helicopter by a non-certificated and unqualified in
dividual who failed to maintain helicopter control.                                  1
A loss of engine power due to the pilot's failure to utilize carburetor heat while ma
neuvering.\r\n.                                                                      1
The pilot's failure to maintain adequate separation behind a corporate jet, which res
ulted in an encounter with wake turbulence and a subsequent loss of control.         1
The pilot�s loss of control due to a wind gust during landing.
1
Name: Report.Status, Length: 17076, dtype: int64
```

To determine the best aircraft for the company based on historical data, I will use several visualizations to uncover trends, identify patterns, and provide insights. Let me focus on key aspects like aircraft safety, injury distribution, and incident factors using matplotlib.pyplot and other visualization tools like seaborn.

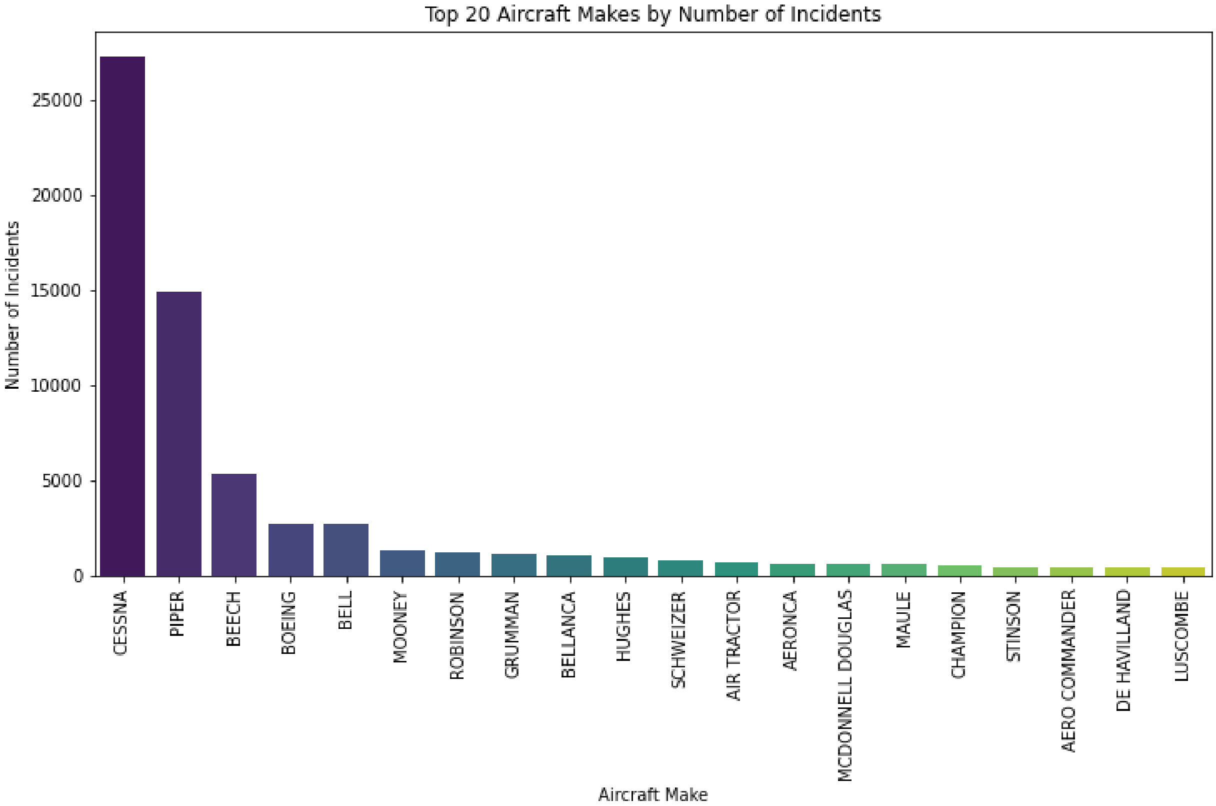
1. Distribution of Incidents by Aircraft Make

I want to see how incidents are distributed across different aircraft makes, which can help identify if any particular make has a high risk.

In [116…

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the frequency of incidents by aircraft make
plt.figure(figsize=(12,6))
make_counts = data['Make'].value_counts().head(20)  # Top 20 aircraft makes
sns.barplot(x=make_counts.index, y= make_counts.values, palette='viridis')
```

```
plt.xticks(rotation=90)
plt.title('Top 20 Aircraft Makes by Number of Incidents')
plt.xlabel('Aircraft Make')
plt.ylabel('Number of Incidents')
plt.show()
```

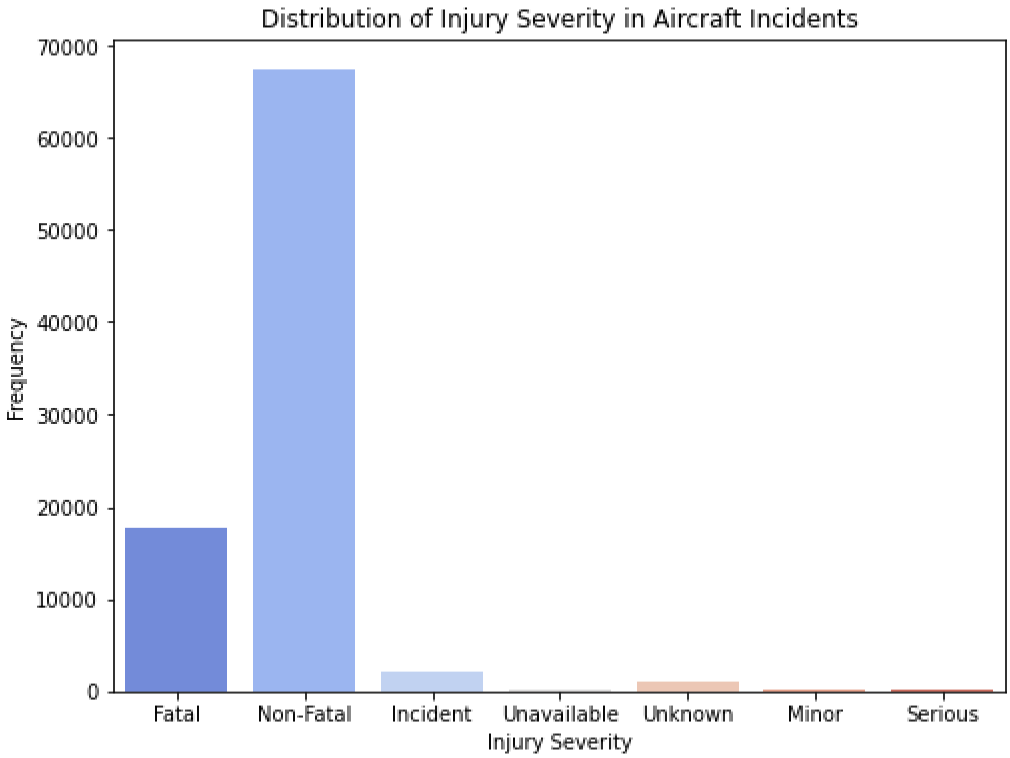


2. Injury Severity Distribution

I want to analyze how injury severities (fatal, serious, minor, uninjured) are distributed across the dataset to understand the most common outcomes in incidents.

In [117…

```
# Plot the distribution of injury severity
plt.figure(figsize=(8,6))
sns.countplot(data=data, x='Injury.Severity', palette='coolwarm')
plt.title('Distribution of Injury Severity in Aircraft Incidents')
plt.xlabel('Injury Severity')
plt.ylabel('Frequency')
plt.show()
```



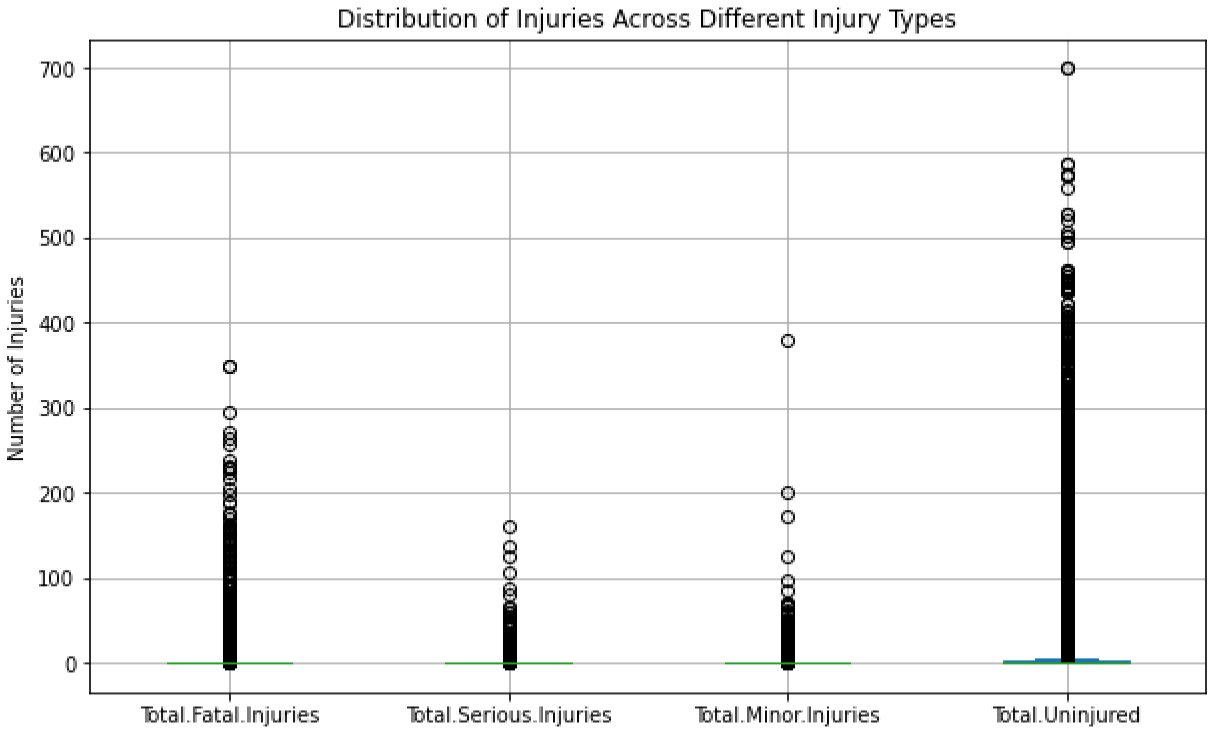
3. Total Injuries Distribution (Fatal, Serious, Minor, Uninjured)

Visualizing the distribution of total injuries can give insight into the overall safety of aircraft types.

In [118…

```
# Select injury-related columns
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Inj

# Ensure the injury columns are numeric
data[injury_columns] = data[injury_columns].apply(pd.to_numeric, errors='coerce')

# Plot frequency of injury columns
plt.figure(figsize=(12,6))
data[injury_columns].plot(kind='box', figsize=(10,6), grid=True)
plt.title('Distribution of Injuries Across Different Injury Types')
plt.ylabel('Number of Injuries')
plt.show()
```

```
<Figure size 864x432 with 0 Axes>
```



4. Incidents by Weather Condition

Since weather conditions play a key role in aviation risk, I can analyze how incidents vary with different weather conditions.

In [119…

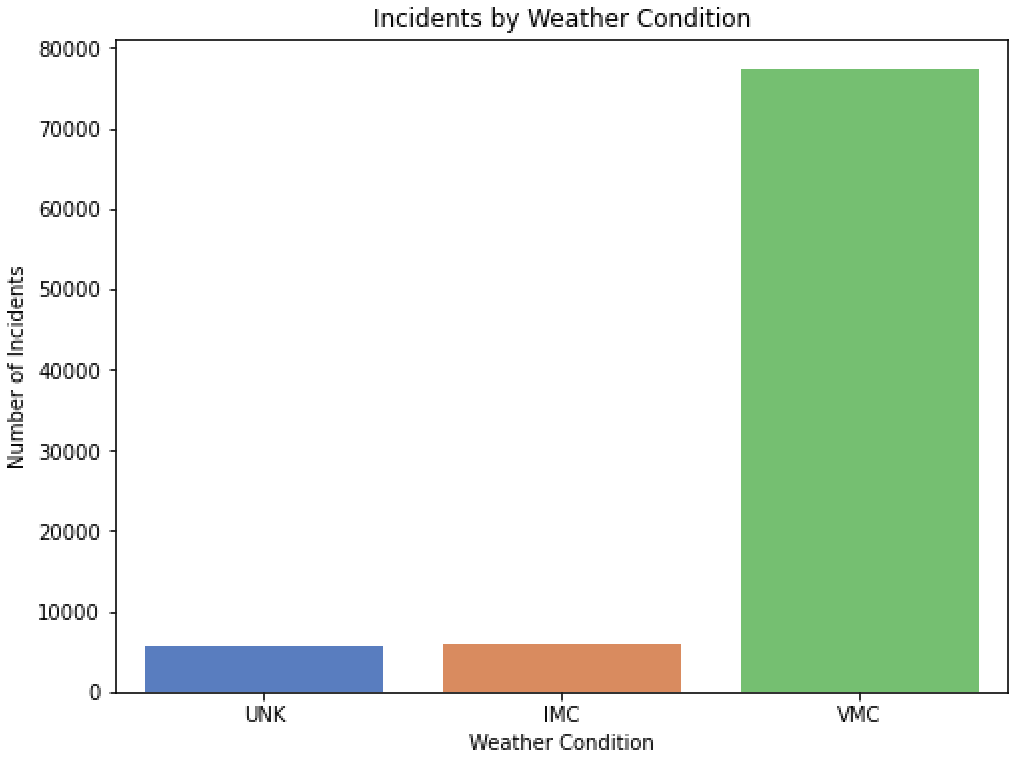
```
# Displaying unique values in the Weather.Condition column
data['Weather.Condition'].unique()
```

Out[119…

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

In [120…

```
# Plot incidents based on weather conditions
plt.figure(figsize=(8,6))
sns.countplot(data=data, x='Weather.Condition', palette='muted')
plt.title('Incidents by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Incidents')
plt.show()
```

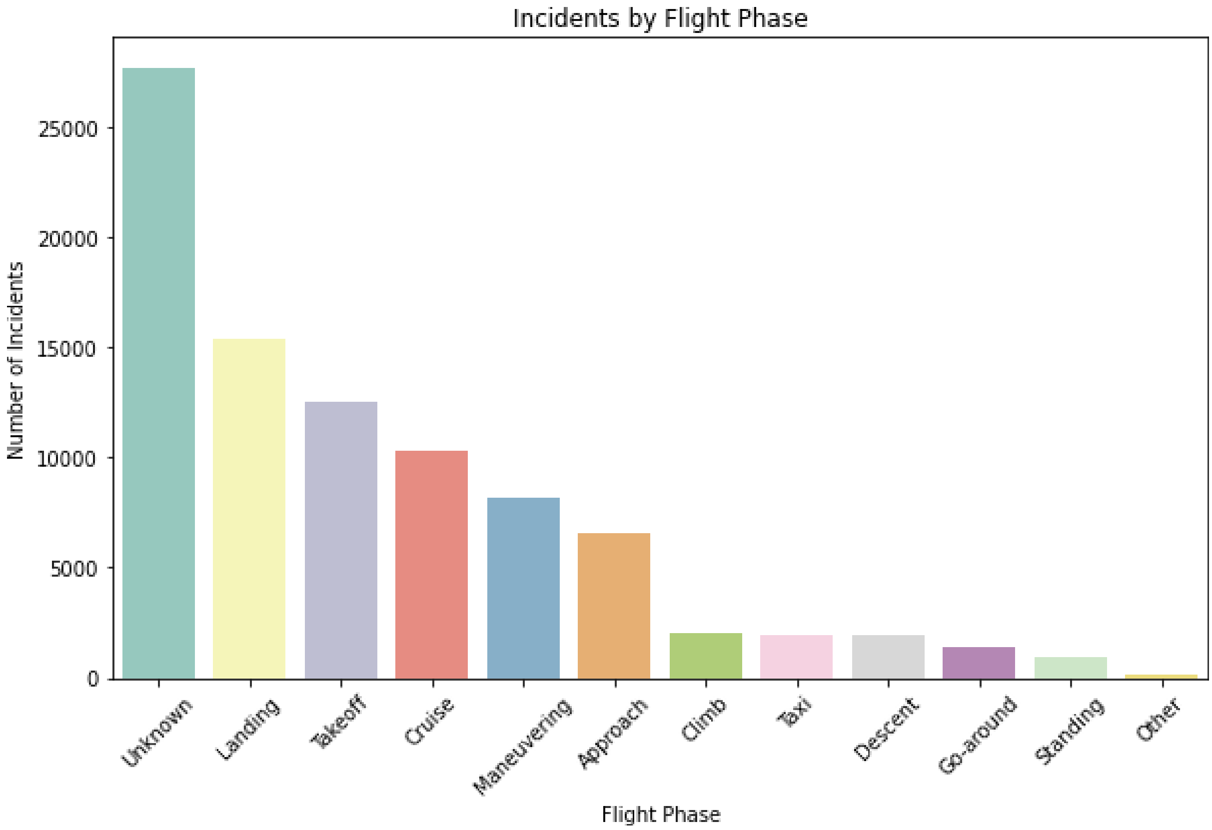


5. Incidents by Flight Phase

The phase of flight during which an incident occurs can highlight operational risks (e.g., takeoff, landing).

In [121…

```
# Plot incidents based on flight phase
plt.figure(figsize=(10,6))
sns.countplot(data=data, x='Broad.phase.of.flight', palette='Set3', order=data['Broa
plt.xticks(rotation=45)
plt.title('Incidents by Flight Phase')
plt.xlabel('Flight Phase')
plt.ylabel('Number of Incidents')
plt.show()
```

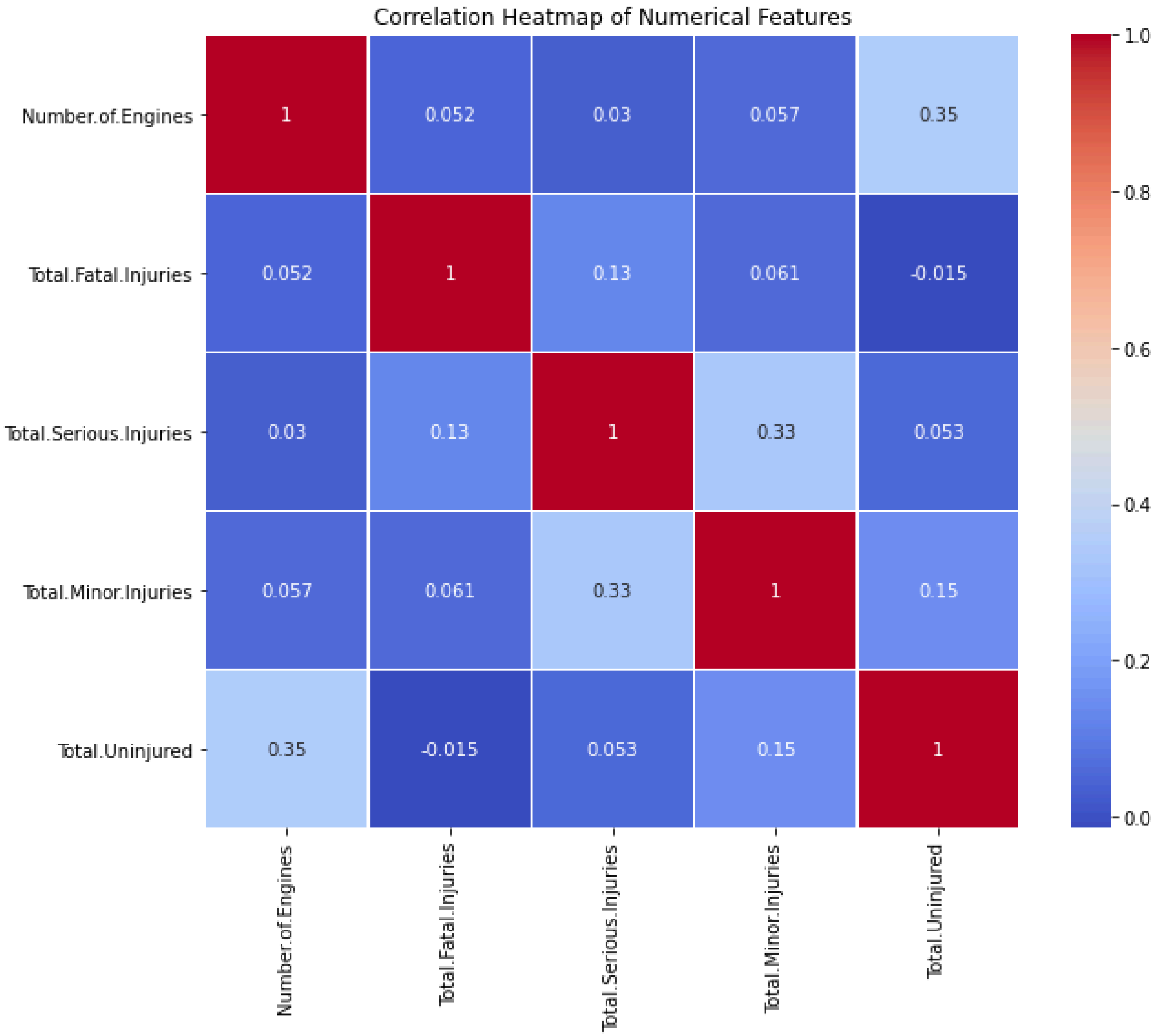


6. Heatmap of Correlations between Features

A heatmap can show correlations between numerical variables, helping identify factors that are strongly related to the risk of incidents.

In [122…

```
# Create a heatmap of the correlation matrix
plt.figure(figsize=(10,8))
corr = data.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```



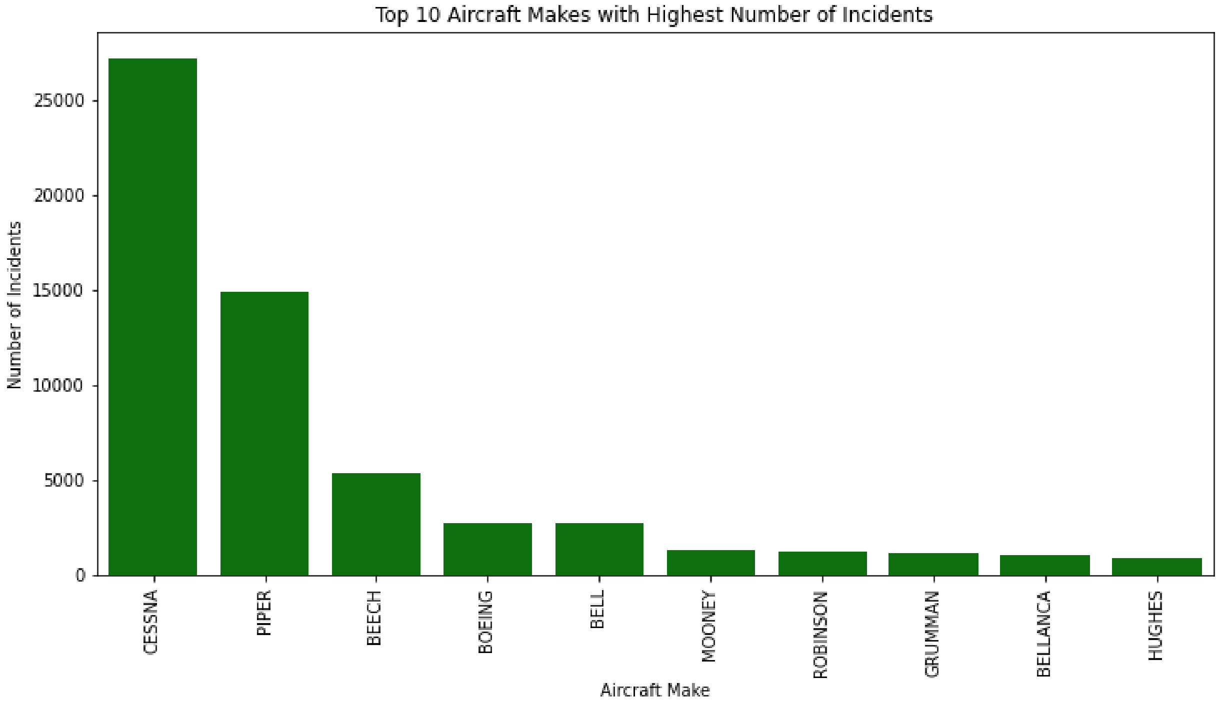
DETAILED ANALYSIS

1. Detailed Analysis of Incidents by Aircraft Make

This analysis will help determine which specific aircraft types are more frequently involved in incidents, focusing on key makes.

In [123…

```
# Plot the top 10 aircraft makes with the highest number of incidents
plt.figure(figsize=(12,6))
top_makes = data['Make'].value_counts().head(10)  # Top 10 makes
sns.barplot(x=top_makes.index, y=top_makes.values, color='green')
plt.xticks(rotation=90)
plt.title('Top 10 Aircraft Makes with Highest Number of Incidents')
plt.xlabel('Aircraft Make')
plt.ylabel('Number of Incidents')
plt.show()
```

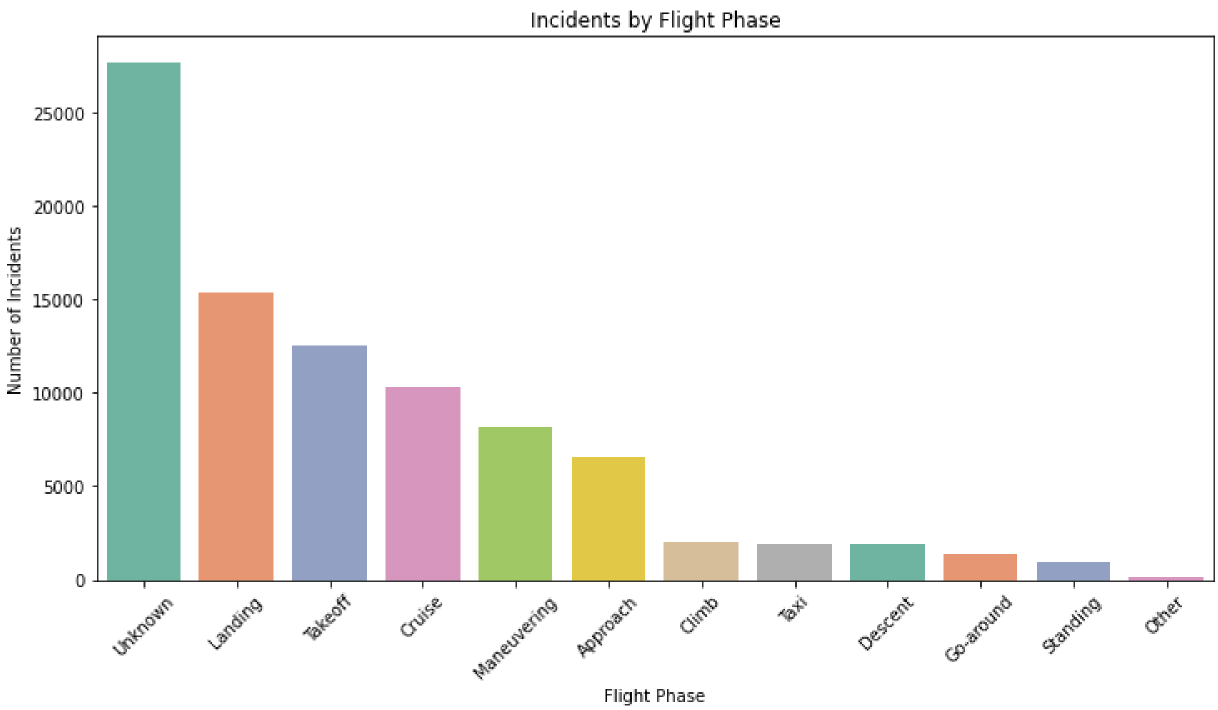


2. Incidents by Flight Phase

Next, I will analyze which phases of flight carry the most risk, focusing on the broad phases of flight such as landing, takeoff, or cruise.

In [124…

```
# Plot incidents by flight phase
plt.figure(figsize=(12,6))
flight_phase_counts = data['Broad.phase.of.flight'].value_counts()

sns.barplot(x=flight_phase_counts.index, y=flight_phase_counts.values, palette='Set2
plt.xticks(rotation=45)
plt.title('Incidents by Flight Phase')
plt.xlabel('Flight Phase')
plt.ylabel('Number of Incidents')
plt.show()
```

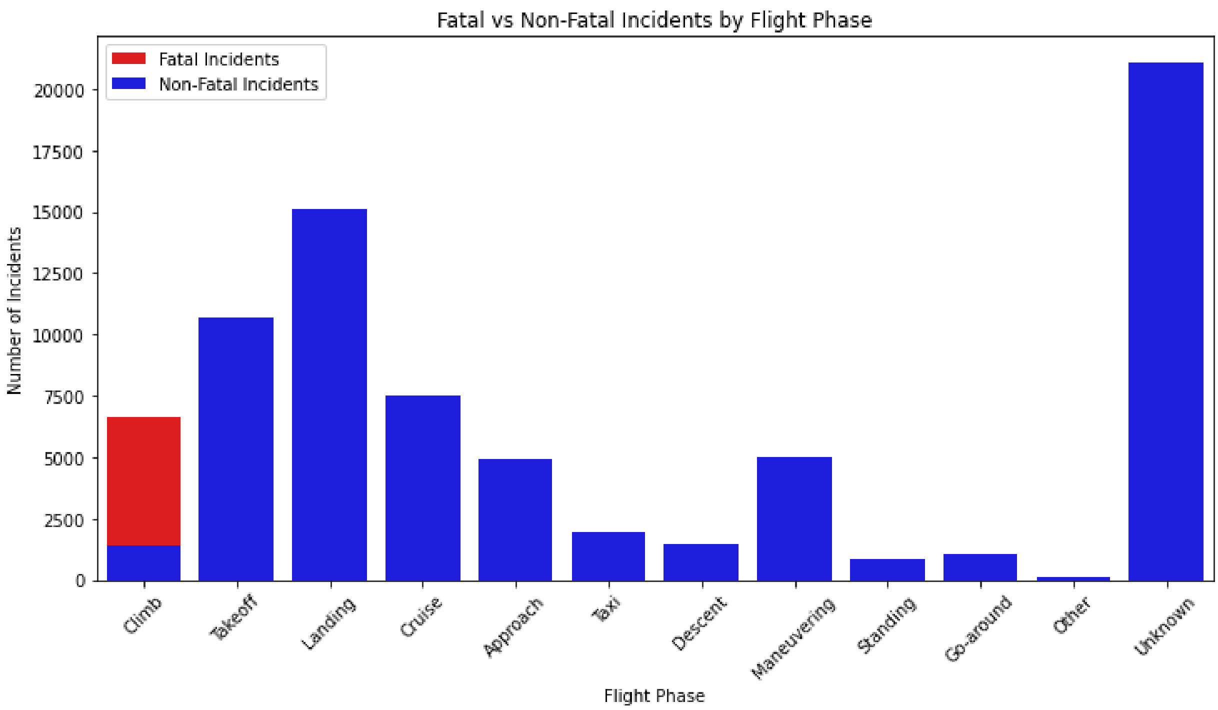


3. Comparison of Incident Types (Fatal vs Non-fatal) by Flight Phase

I can analyze how different flight phases contribute to either fatal or non-fatal incidents.

In [548…

```
# Filter non-fatal incidents
non_fatal_incidents = data[data['Injury.Severity'] != 'Fatal']

# Plot fatal vs non-fatal incidents by flight phase
plt.figure(figsize=(12,6))
sns.countplot(data=fatal_incidents, x='Broad.phase.of.flight', color='red', label='F
sns.countplot(data=non_fatal_incidents, x='Broad.phase.of.flight', color='blue', lab
plt.xticks(rotation=45)
plt.legend()
plt.title('Fatal vs Non-Fatal Incidents by Flight Phase')
plt.xlabel('Flight Phase')
plt.ylabel('Number of Incidents')
plt.show()
```



In [551…

```
import pandas as pd

# Group by 'Make' and 'Model', count the number of incidents (rows)
accident_frequency = data.groupby(['Make', 'Model']).size().reset_index(name='Accide

# Sort the results by 'Accident_Frequency' to find models with the highest incident
accident_frequency_sorted = accident_frequency.sort_values(by='Accident_Frequency',

# Display the top 20 aircraft models with the highest accident frequency
print(accident_frequency_sorted.head(20))
```

```
         Make      Model  Accident_Frequency
4670   CESSNA        152                2367
4695   CESSNA        172                1754
4746   CESSNA       172N                1164
13539   PIPER  PA-28-140                 932
4643   CESSNA        150                 829
4744   CESSNA       172M                 798
4749   CESSNA       172P                 689
```

```
4803    CESSNA        182                   659
4779    CESSNA        180                   621
4669    CESSNA       150M                   585
13430    PIPER      PA-18                   578
13549    PIPER  PA-28-180                   572
13440    PIPER  PA-18-150                   571
13548    PIPER  PA-28-161                   565
13556    PIPER  PA-28-181                   529
2573      BELL       206B                   516
3210    BOEING        737                   489
13696    PIPER  PA-38-112                   468
4668    CESSNA       150L                   460
2106     BEECH        A36                   419
```

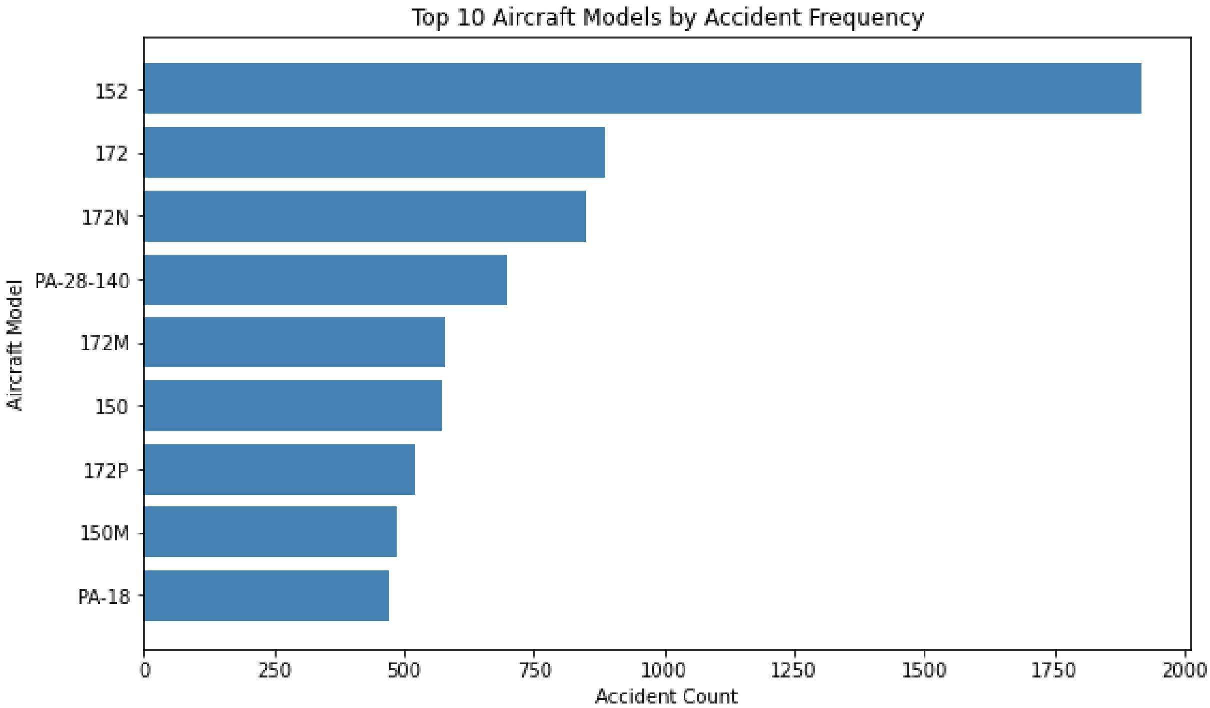
In [552…

```
# Grouping by Make and Model to calculate accident frequency
accident_freq_make_model = data.groupby(['Make', 'Model']).size().reset_index(name='

# Sorting to get the top aircraft makes with the highest accident frequencies
top_makes = accident_freq_make_model.groupby('Make')['Accident Count'].sum().sort_va

# Displaying the top aircraft makes by accident count
print(top_makes)
```

```
Make
CESSNA      27216
PIPER       14870
BEECH        5372
BOEING       2745
BELL         2722
MOONEY       1334
ROBINSON     1230
GRUMMAN      1172
BELLANCA     1045
HUGHES        932
Name: Accident Count, dtype: int64
```

In [553…

```
# Grouping by Make, Model, and Aircraft Category to calculate accident frequency
accident_frequency = data.groupby(['Make', 'Model', 'Aircraft.Category']).size().res

# Sorting the aircraft models with the highest accident counts
accident_frequency_sorted = accident_frequency.sort_values(by='Accident Count', asce

# Plotting accident frequency for the top 10 aircraft models
plt.figure(figsize=(10,6))
plt.barh(accident_frequency_sorted['Model'], accident_frequency_sorted['Accident Cou
plt.xlabel('Accident Count')
plt.ylabel('Aircraft Model')
plt.title('Top 10 Aircraft Models by Accident Frequency')
plt.gca().invert_yaxis()
plt.show()
```



In [554…

```
# Grouping by Make, Model, and Aircraft.Category to calculate accident frequency
accident_frequency = data.groupby(['Make', 'Model', 'Aircraft.Category']).size().res

# Sorting the aircraft models with the highest accident counts
accident_frequency_sorted = accident_frequency.sort_values(by='Accident Count', asce

# Displaying the top 10 high-risk aircraft models based on accident frequency
print(accident_frequency_sorted.head(10))
```

```
         Make      Model Aircraft.Category  Accident Count
5387   CESSNA        152           Unknown            1916
5420   CESSNA        172           Unknown             886
5419   CESSNA        172          Airplane             868
5491   CESSNA       172N           Unknown             848
15255   PIPER  PA-28-140           Unknown             700
5488   CESSNA       172M           Unknown             581
5346   CESSNA        150           Unknown             573
5495   CESSNA       172P           Unknown             522
5385   CESSNA       150M           Unknown             485
15096   PIPER      PA-18           Unknown             470
```

In [555…

```
# Creating a Severity Index based on injury columns
data['Severity Index'] = (data['Total.Fatal.Injuries'].fillna(0) * 3) + \
                         (data['Total.Serious.Injuries'].fillna(0) * 2) + \
                         (data['Total.Minor.Injuries'].fillna(0) * 1)

# Grouping by Make and Model to calculate average severity per aircraft
severity_index = data.groupby(['Make', 'Model'])['Severity Index'].mean().reset_inde

# Sorting the aircraft models with the highest severity scores
severity_index_sorted = severity_index.sort_values(by='Severity Index', ascending=Fa

# Displaying the top 10 models based on severity index
print(severity_index_sorted.head(10))
```

```
                     Make        Model  Severity Index
3352               BOEING      747-168          1047.0
17382             TUPOLEV       TU-154          1047.0
3492               BOEING   767-366-ER           651.0
3501               BOEING    777 - 206           534.0
11675  MCDONNELL DOUGLAS       DC-8-62           522.0
922     AIRBUS INDUSTRIE         A 310           517.0
960     AIRBUS INDUSTRIE      A310-300           490.0
```

```
852              AIRBUS    A320 - 216          486.0
7229            EMBRAER  E135 Legacy          462.0
3655             BOEING        MD-83          459.0
```

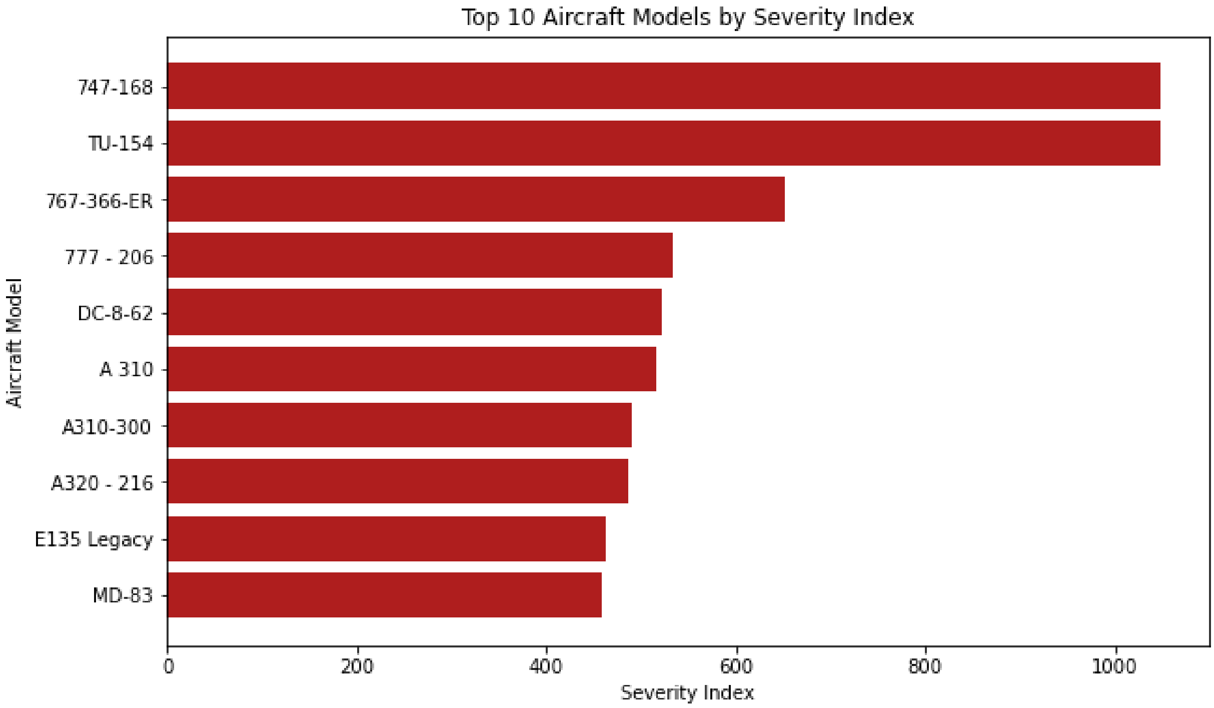
In [556…

```
# Grouping by Make, Model, and Aircraft.Category to calculate accident frequency
accident_frequency = data.groupby(['Make', 'Model', 'Aircraft.Category']).size().res

# Sorting the aircraft models with the highest accident counts
accident_frequency_sorted = accident_frequency.sort_values(by='Accident Count', asce

# Displaying the top 10 high-risk aircraft models based on accident frequency
print(accident_frequency_sorted.head(10))
```

```
         Make      Model Aircraft.Category  Accident Count
5387   CESSNA        152           Unknown            1916
5420   CESSNA        172           Unknown             886
5419   CESSNA        172          Airplane             868
5491   CESSNA       172N           Unknown             848
15255   PIPER  PA-28-140           Unknown             700
5488   CESSNA       172M           Unknown             581
5346   CESSNA        150           Unknown             573
5495   CESSNA       172P           Unknown             522
5385   CESSNA       150M           Unknown             485
15096   PIPER      PA-18           Unknown             470
```

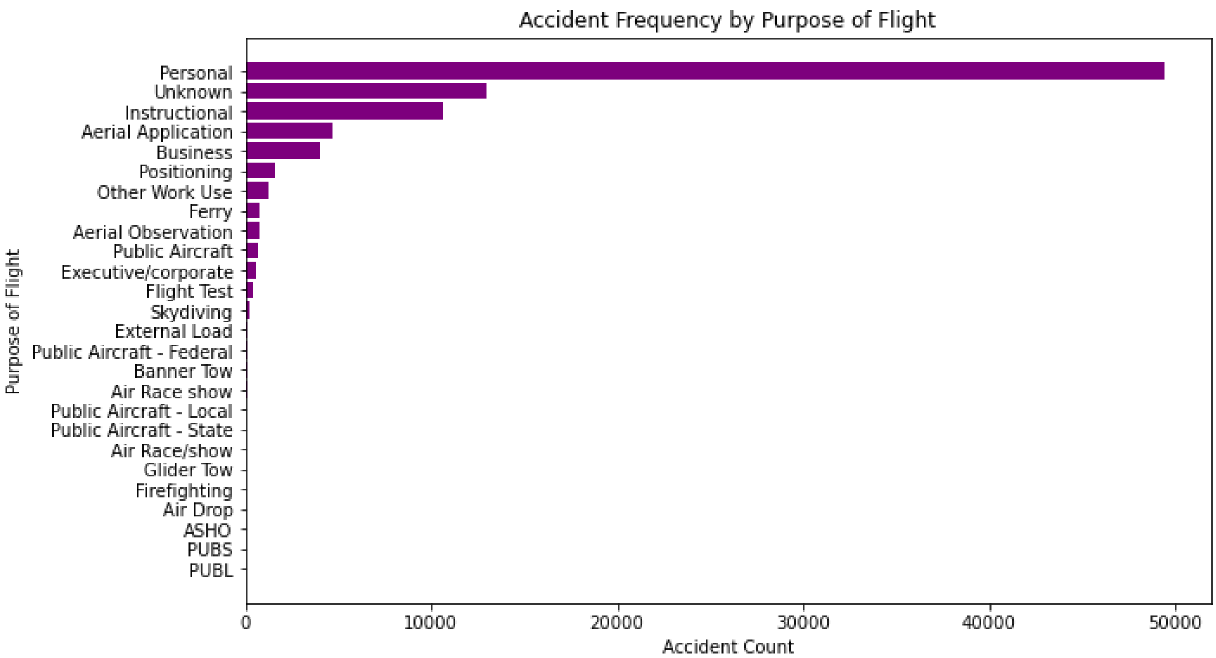
In [557…

```
# Creating the Severity Index as described
data['Severity Index'] = (data['Total.Fatal.Injuries'].fillna(0) * 3) + \
                         (data['Total.Serious.Injuries'].fillna(0) * 2) + \
                         (data['Total.Minor.Injuries'].fillna(0) * 1)

# Grouping by Make and Model to calculate average severity per aircraft
severity_index = data.groupby(['Make', 'Model'])['Severity Index'].mean().reset_inde

# Sorting the aircraft models with the highest severity scores
severity_index_sorted = severity_index.sort_values(by='Severity Index', ascending=Fa

# Plotting the severity index for the top 10 aircraft models
plt.figure(figsize=(10,6))
plt.barh(severity_index_sorted['Model'], severity_index_sorted['Severity Index'], co
plt.xlabel('Severity Index')
plt.ylabel('Aircraft Model')
plt.title('Top 10 Aircraft Models by Severity Index')
plt.gca().invert_yaxis()
plt.show()
```



In [558…

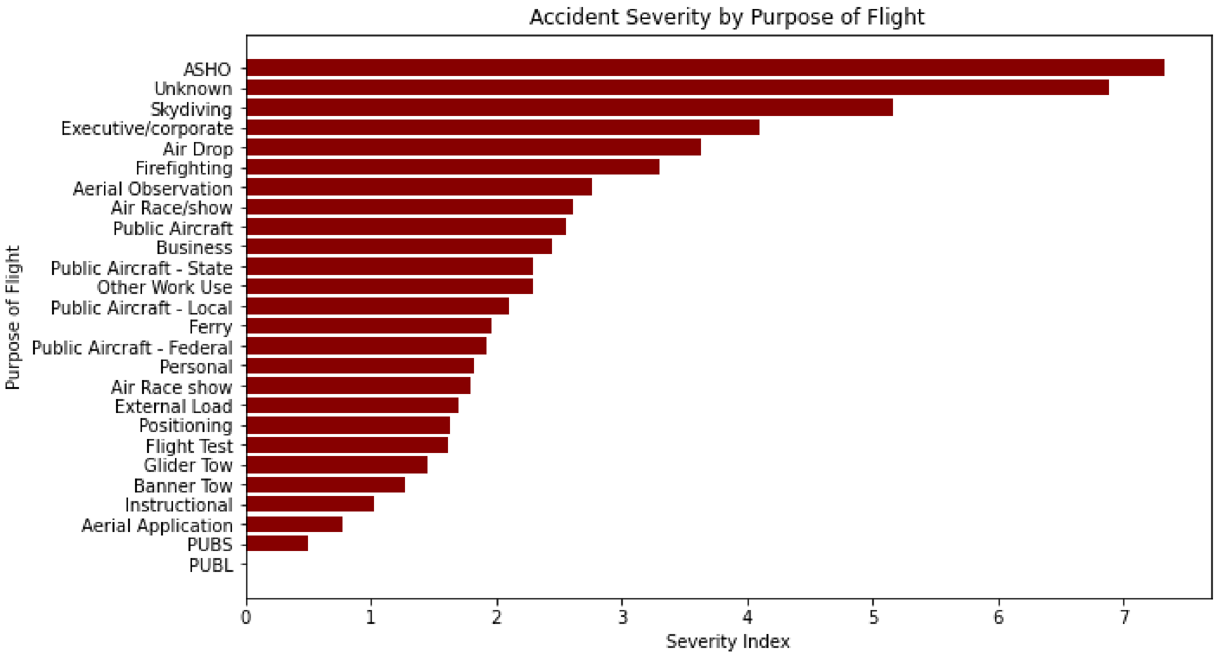
```
# Grouping by Purpose of Flight to calculate accident frequency
flight_purpose_accidents = data.groupby('Purpose.of.flight').size().reset_index(name

# Sorting by accident count to focus on the most frequent accident purposes
flight_purpose_accidents_sorted = flight_purpose_accidents.sort_values(by='Accident

# Plotting accident frequency by Purpose of Flight
plt.figure(figsize=(10,6))
plt.barh(flight_purpose_accidents_sorted['Purpose.of.flight'], flight_purpose_accide
plt.xlabel('Accident Count')
plt.ylabel('Purpose of Flight')
plt.title('Accident Frequency by Purpose of Flight')
plt.gca().invert_yaxis()
plt.show()
```



In [559…

```
# Calculating the Severity Index for each Purpose of Flight
flight_purpose_severity = data.groupby('Purpose.of.flight')['Severity Index'].mean()

# Sorting by Severity Index to focus on the purposes with the highest accident sever
```

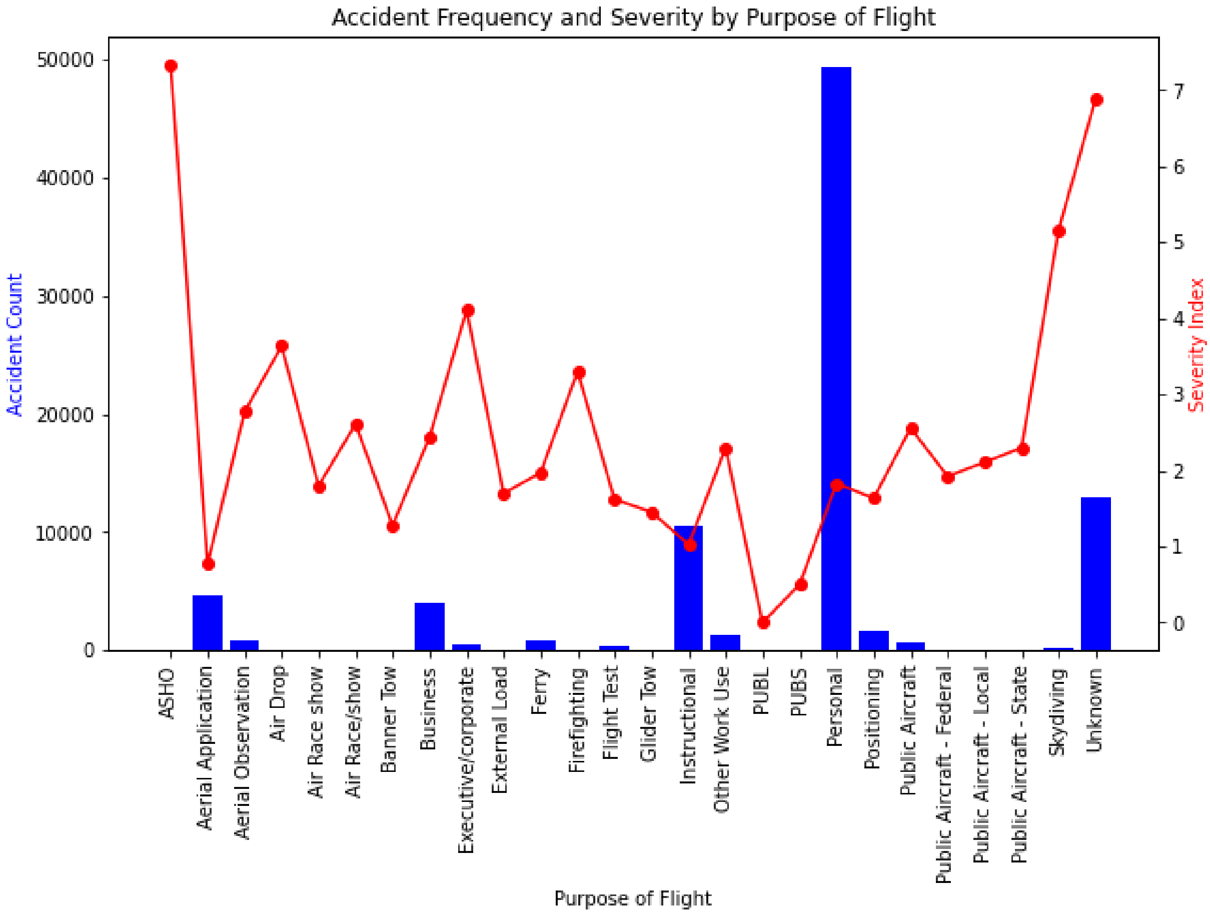
```
flight_purpose_severity_sorted = flight_purpose_severity.sort_values(by='Severity In

# Plotting accident severity by Purpose of Flight
plt.figure(figsize=(10,6))
plt.barh(flight_purpose_severity_sorted['Purpose.of.flight'], flight_purpose_severit
plt.xlabel('Severity Index')
plt.ylabel('Purpose of Flight')
plt.title('Accident Severity by Purpose of Flight')
plt.gca().invert_yaxis()
plt.show()
```



In [560…

```
# Merging the accident frequency and severity data
combined_flight_purpose = pd.merge(flight_purpose_accidents, flight_purpose_severity

# Plotting accident frequency and severity on a dual-axis plot
fig, ax1 = plt.subplots(figsize=(10,6))

ax2 = ax1.twinx()
ax1.bar(combined_flight_purpose['Purpose.of.flight'], combined_flight_purpose['Accid
ax2.plot(combined_flight_purpose['Purpose.of.flight'], combined_flight_purpose['Seve

ax1.set_xlabel('Purpose of Flight')
ax1.set_ylabel('Accident Count', color='blue')
ax2.set_ylabel('Severity Index', color='red')

# Rotating the x-axis labels by 90 degrees
ax1.set_xticklabels(combined_flight_purpose['Purpose.of.flight'], rotation=90)

plt.title('Accident Frequency and Severity by Purpose of Flight')
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-560-ad5640c0371a>:16: UserWarning: FixedFormatter should only be used
together with FixedLocator
  ax1.set_xticklabels(combined_flight_purpose['Purpose.of.flight'], rotation=90)
```



In [561…

```
data.isnull().sum()
```

Out[561…

```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                      0
Country                       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
Total.Fatal.Injuries        730
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition             0
Broad.phase.of.flight         0
Report.Status                 0
Severity Index                0
dtype: int64
```

In [562…

```
data.isnull()
```

Out[562…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airport.Name
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
88884	False	False	False	False	False	False	False
88885	False	False	False	False	False	False	False
88886	False	False	False	False	False	False	False
88887	False	False	False	False	False	False	False
88888	False	False	False	False	False	False	False

88889 rows × 27 columns

In [ ]: