Phase 1 Project

Introduction

This project is in fact meant to help a firm, which having considered the opportunity to venture into aviation, requires guidance on which aircraft to purchase. The outcomes of the project will be based on the analysis of the "AviationData.csv" and "USState_Codes.csv" datasets to reveal fundamental tendencies of aviation incidents concerning aircraft characteristics, incidences, states, etc. This means that after carrying out the advanced data cleaning and analysis, it will be possible for the company to establish which aircraft models are the most reliable, safer and best suited for the particular company's operations and hence make future investments with informed decisions.

Dataset explanation

For this project, I have two datasets; "AviationData.csv" which I acquired from Kaggle and "USState_Codes.csv". With this data, I am required to clean and analyze data to obtain conclusions and recommendations. The findings here will assist the head of a new division in the aviation industry select the right aircraft to buy. In regard to the evaluation of the outcome of the study consideration will be given to patterns, trends, and factors that relate to the performance and safety of the required aircraft.

Questions I intend to answer:

1. What aircraft makes is or is not suitable to fly due to numerous accidents?
2. What aircraft models are buyers advised to look at and have low fatality rates?
3. Which engines have the lowest or risk of fatalities per accident?
4. What should we do particularly in IMC conditions when encountering various forms of weather risks?
5. Are there risks involved with an amateur built aircraft and, if so, should they be permitted?
6. In what way do regional trends in terms of aircraft incidents affect the fleet?
7. Are aviation incidences increasing or decreasing?
8. What makes and models are best to buy?
9. What makes and models should be least considered when purchasing aircraft?

Objectives

1. Describe the data
2. Clean the data
3. Perform statistical analysis of the data
4. Find emerging trends and patterns in the cleaned data

Description of the datasets

Importing Modules

```
##Importing pandas, matplotlib.pyplot, numpy, seaborn using aliases
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

%matplotlib inline
```

Loading Datasets

```
#Load AviationData.csv as a dataframe "df", change encoding, add
memory
df = pd.read_csv('AviationData.csv', encoding='Latin',
low_memory=False)
#Display the dataframe head
df.head()
```

```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country   Latitude   Longitude Airport.Code
\
0  MOOSE CREEK, ID  United States        NaN         NaN          NaN

1   BRIDGEPORT, CA  United States        NaN         NaN          NaN

2    Saltville, VA  United States  36.922223  -81.878056          NaN

3       EUREKA, CA  United States        NaN         NaN          NaN

4       Canton, OH  United States        NaN         NaN          NaN


  Airport.Name  ... Purpose.of.flight Air.carrier Total.Fatal.Injuries
\
0          NaN  ...          Personal         NaN                  2.0

1          NaN  ...          Personal         NaN                  4.0

2          NaN  ...          Personal         NaN                  3.0

3          NaN  ...          Personal         NaN                  2.0
```

```
4          NaN  ...         Personal        NaN                     1.0


  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0

  Weather.Condition  Broad.phase.of.flight   Report.Status
Publication.Date
0               UNK                 Cruise  Probable Cause
NaN
1               UNK                Unknown  Probable Cause       19-
09-1996
2               IMC                 Cruise  Probable Cause       26-
02-2007
3               IMC                 Cruise  Probable Cause       12-
09-2000
4               VMC               Approach  Probable Cause       16-
04-1980

[5 rows x 31 columns]
```

Information about the dataframe

In this analysis, I will explore the dataset structure using df.columns and determine its size with df.shape. I will also apply df.describe() to generate summary statistics, providing insights into key numeric values like fatalities and injuries, helping assess and prepare the data for further analysis.

```
#Find the columns in the dataframe
df.columns

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')
```

```python
#Finding the shape, number of columns and rows
df.shape
```

```
(88889, 31)
```

```python
#Finding key statistics of the dataframe
df.describe
```

```
<bound method NDFrame.describe of               Event.Id
Investigation.Type Accident.Number  Event.Date  \
0      20001218X45444           Accident      SEA87LA080  1948-10-24

1      20001218X45447           Accident      LAX94LA336  1962-07-19

2      20061025X01555           Accident      NYC07LA005  1974-08-30

3      20001218X45448           Accident      LAX96LA321  1977-06-19

4      20041105X01764           Accident      CHI79FA064  1979-08-02

...               ...                ...             ...         ...

88884  20221227106491           Accident      ERA23LA093  2022-12-26

88885  20221227106494           Accident      ERA23LA095  2022-12-26

88886  20221227106497           Accident      WPR23LA075  2022-12-26

88887  20221227106498           Accident      WPR23LA076  2022-12-26

88888  20221230106513           Accident      ERA23LA097  2022-12-29


              Location        Country   Latitude   Longitude
Airport.Code  \
0      MOOSE CREEK, ID  United States        NaN         NaN
NaN
1       BRIDGEPORT, CA  United States        NaN         NaN
NaN
2        Saltville, VA  United States  36.922223  -81.878056
NaN
3           EUREKA, CA  United States        NaN         NaN
NaN
4           Canton, OH  United States        NaN         NaN
NaN
...                ...            ...        ...         ...
...
88884    Annapolis, MD  United States        NaN         NaN
NaN
88885      Hampton, NH  United States        NaN         NaN
NaN
```

```
88886       Payson, AZ  United States    341525N    1112021W
PAN
88887       Morgan, UT  United States        NaN         NaN
NaN
88888       Athens, GA  United States        NaN         NaN
NaN

      Airport.Name  ... Purpose.of.flight         Air.carrier  \
0              NaN  ...          Personal                 NaN
1              NaN  ...          Personal                 NaN
2              NaN  ...          Personal                 NaN
3              NaN  ...          Personal                 NaN
4              NaN  ...          Personal                 NaN
...            ...  ...               ...                 ...
88884          NaN  ...          Personal                 NaN
88885          NaN  ...               NaN                 NaN
88886       PAYSON  ...          Personal                 NaN
88887          NaN  ...          Personal  MC CESSNA 210N LLC
88888          NaN  ...          Personal                 NaN

      Total.Fatal.Injuries Total.Serious.Injuries Total.Minor.Injuries
\
0                      2.0                    0.0                  0.0

1                      4.0                    0.0                  0.0

2                      3.0                    NaN                  NaN

3                      2.0                    0.0                  0.0

4                      1.0                    2.0                  NaN

...                    ...                    ...                  ...

88884                  0.0                    1.0                  0.0

88885                  0.0                    0.0                  0.0

88886                  0.0                    0.0                  0.0

88887                  0.0                    0.0                  0.0

88888                  0.0                    1.0                  0.0


      Total.Uninjured Weather.Condition  Broad.phase.of.flight  \
0                 0.0               UNK                 Cruise
1                 0.0               UNK                Unknown
2                 NaN               IMC                 Cruise
3                 0.0               IMC                 Cruise
4                 0.0               VMC               Approach
```

```
...             ...              ...                    ...
88884           0.0              NaN                    NaN
88885           0.0              NaN                    NaN
88886           1.0              VMC                    NaN
88887           0.0              NaN                    NaN
88888           1.0              NaN                    NaN

        Report.Status Publication.Date
0      Probable Cause              NaN
1      Probable Cause       19-09-1996
2      Probable Cause       26-02-2007
3      Probable Cause       12-09-2000
4      Probable Cause       16-04-1980
...               ...              ...
88884             NaN       29-12-2022
88885             NaN              NaN
88886             NaN       27-12-2022
88887             NaN              NaN
88888             NaN       30-12-2022

[88889 rows x 31 columns]>
```

```
#Finding the data types of columns in the dataset
df.dtypes
```

```
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
```

What have I learnt?

The dataset contains 88,889 aviation incidents with 31 columns detailing event dates, locations, aircraft info, and injury severities. It has missing values in key fields like geographic data and injuries, and mixed data types that require cleaning. The dataset contains different columns which have data of two data types, objects, and float values,

Cleaning the datasets

At this stage, null values and duplicates will be checked for. I intend to improve the quality and reliability of the dataset by removing duplicates and handling null values. This process ensures that redundant or incomplete data is eliminated, reducing potential biases and errors. Addressing missing data will be caried out either by removal or imputation, and eliminating duplicate records.

Finding with Null Values

```python
# Calculating and displaying null values per column as a percentage of the entire DataFrame
null_percentage = (df.isnull().sum() / len(df)) * 100
print(null_percentage)
```

```
Event.Id                   0.000000
Investigation.Type         0.000000
Accident.Number            0.000000
Event.Date                 0.000000
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
```

```python
#Finding duplicated rows in the dataset
df.duplicated().sum()
```

```
0
```

Based on a 30% missing data threshold recommended by vast data cleaning information sources, the columns that should be removed include Latitude (61.32%), Longitude (61.33%), Airport.Code (43.47%), Airport.Name (40.61%), Aircraft.Category (63.68%), FAR.Description (63.97%), Schedule (85.85%), and Air.carrier (81.27%). Additionally, columns like Total.Fatal.Injuries (12.83%), Total.Serious.Injuries (14.07%), Total.Minor.Injuries (13.42%), and Publication.Date (15.49%) may be considered for removal depending on their importance to the analysis. Removing these columns improves data quality by addressing significant gaps.

```python
# Identifying columns with more than 30% missing data
missing_threshold = 0.30
columns_to_drop = df.columns[df.isnull().mean() > missing_threshold]

# Dropping the columns from the DataFrame
df = df.drop(columns=columns_to_drop)

# Display the columns that were dropped
print("Columns to be dropped due to missing more than 30% of data: ",
[element for element in columns_to_drop])

# Display the cleaned DataFrame (first few rows)
df.head()
```

```
Columns to be dropped due to missing more than 30% of data:
['Latitude', 'Longitude', 'Airport.Code', 'Airport.Name',
'Aircraft.Category', 'FAR.Description', 'Schedule', 'Air.carrier',
'Broad.phase.of.flight']
```

```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
```

```
3  20001218X45448          Accident      LAX96LA321  1977-06-19
4  20041105X01764          Accident      CHI79FA064  1979-08-02

          Location        Country Injury.Severity Aircraft.damage  \
0  MOOSE CREEK, ID  United States        Fatal(2)       Destroyed
1   BRIDGEPORT, CA  United States        Fatal(4)       Destroyed
2    Saltville, VA  United States        Fatal(3)       Destroyed
3       EUREKA, CA  United States        Fatal(2)       Destroyed
4       Canton, OH  United States        Fatal(1)       Destroyed

  Registration.Number      Make  ... Number.of.Engines    Engine.Type
\
0              NC6404   Stinson  ...               1.0  Reciprocating

1              N5069P     Piper  ...               1.0  Reciprocating

2              N5142R    Cessna  ...               1.0  Reciprocating

3              N1168J  Rockwell  ...               1.0  Reciprocating

4               N15NY    Cessna  ...               NaN            NaN


   Purpose.of.flight Total.Fatal.Injuries Total.Serious.Injuries  \
0           Personal                  2.0                    0.0
1           Personal                  4.0                    0.0
2           Personal                  3.0                    NaN
3           Personal                  2.0                    0.0
4           Personal                  1.0                    2.0

   Total.Minor.Injuries  Total.Uninjured  Weather.Condition
Report.Status  \
0                   0.0              0.0                UNK  Probable
Cause
1                   0.0              0.0                UNK  Probable
Cause
2                   NaN              NaN                IMC  Probable
Cause
3                   0.0              0.0                IMC  Probable
Cause
4                   NaN              0.0                VMC  Probable
Cause

  Publication.Date
0              NaN
1       19-09-1996
2       26-02-2007
3       12-09-2000
4       16-04-1980
```

```
[5 rows x 22 columns]
```

For numeric columns like Total.Fatal.Injuries, Total.Serious.Injuries, Total.Minor.Injuries, and Total.Uninjured, median imputation is used to replace missing values with the median, which helps handle outliers. For categorical columns such as Injury.Severity, Aircraft.damage, Weather.Condition, and others, mode imputation is applied, filling missing values with the most frequent value in each column. Additionally, "Unknown" is used for the Location and Country columns, ensures that these gaps are visible and won't skew the analysis which could misrepresent the data. This process helps prepare the dataset for analysis by addressing missing data in a contextually relevant way while retaining the data integrity.

```
# Mean/Median imputation for numeric columns
df['Total.Fatal.Injuries'].fillna(df['Total.Fatal.Injuries'].median(),
inplace=True)
df['Total.Serious.Injuries'].fillna(df['Total.Serious.Injuries'].media
n(), inplace=True)
df['Total.Minor.Injuries'].fillna(df['Total.Minor.Injuries'].median(),
inplace=True)
df['Total.Uninjured'].fillna(df['Total.Uninjured'].median(),
inplace=True)

# Mode imputation for categorical columns
df['Injury.Severity'].fillna(df['Injury.Severity'].mode()[0],
inplace=True)
df['Aircraft.damage'].fillna(df['Aircraft.damage'].mode()[0],
inplace=True)
df['Weather.Condition'].fillna(df['Weather.Condition'].mode()[0],
inplace=True)
df['Registration.Number'].fillna(df['Registration.Number'].mode()[0],
inplace=True)
df['Make'].fillna(df['Make'].mode()[0], inplace=True)
df['Model'].fillna(df['Model'].mode()[0], inplace=True)
df['Amateur.Built'].fillna(df['Amateur.Built'].mode()[0],
inplace=True)
df['Number.of.Engines'].fillna(df['Number.of.Engines'].mode()[0],
inplace=True)
df['Engine.Type'].fillna(df['Engine.Type'].mode()[0], inplace=True)
df['Purpose.of.flight'].fillna(df['Purpose.of.flight'].mode()[0],
inplace=True)
df['Report.Status'].fillna(df['Report.Status'].mode()[0],
inplace=True)
df['Publication.Date'].fillna(df['Publication.Date'].mode()[0],
inplace=True)

# Remove redundancies due to case, make them uniform
df['Make'].replace({'boeing': 'BOEING', 'Boeing': 'BOEING', 'Cessna':
'CESSNA'}, inplace=True)
```

```
# Calculate the mode of the 'Weather.Condition' column
# Strip leading/trailing spaces and standardize to 'Unknown' for any
variations
df['Weather.Condition'] =
df['Weather.Condition'].str.strip().str.capitalize()

# Calculate the mode of the 'Weather.Condition' column (excluding
'Unknown')
weather_mode = df[df['Weather.Condition'] != 'Unknown']
['Weather.Condition'].mode()[0]

# Replace 'Unknown' with the mode
df['Weather.Condition'] = df['Weather.Condition'].replace('Unknown',
weather_mode)

# Replace 'Unknown' with the mode
df['Weather.Condition'] = df['Weather.Condition'].replace('Unknown',
weather_mode)



# Fill missing values in 'Location' and 'Country' with 'Unknown'
df['Location'].fillna('Unknown', inplace=True)
df['Country'].fillna('Unknown', inplace=True)
```

Looking to see if there are any null values in the dataframe

```
#Finding null values
df.isnull().values.sum()

0
```

Familiarize with the cleaned data

What unique Values exist in the dataset? What are we dealing with? The following columns will be investigated for unique values: Investigation.Type, Locations, Countries, Make, Models, Engine.Type, Purpose.Of.Flight

```
# Unique engines
print(f'There are {df["Engine.Type"].nunique()} unique engines: ')
df['Engine.Type'].unique()

There are 13 unique engines:

array(['Reciprocating', 'Turbo Fan', 'Turbo Shaft', 'Unknown',
       'Turbo Prop', 'Turbo Jet', 'None', 'Electric', 'Hybrid Rocket',
       'Geared Turbofan', 'LR', 'NONE', 'UNK'], dtype=object)
```

```python
# Unique Makes
print(f'There are {df["Make"].nunique()} distinct aircraft makes: ')
df['Make'].unique()
```

```
There are 8235 distinct aircraft makes:
```

```
array(['Stinson', 'Piper', 'CESSNA', ..., 'JAMES R DERNOVSEK',
       'ORLICAN S R O', 'ROYSE RALPH L'], dtype=object)
```

```python
# Unique Models
print(f'There are {df["Model"].nunique()} distinct aircraft models: ')
df['Model'].unique()
```

```
There are 12318 distinct aircraft models:
```

```
array(['108-3', 'PA24-180', '172M', ..., 'ROTORWAY EXEC 162-F',
       'KITFOX S5', 'M-8 EAGLE'], dtype=object)
```

During an incident, what investigations are carried out in the aircraft industry?

```python
# Investigation Types
print(f'There are {df["Investigation.Type"].nunique()} investigation
types: ')
df['Investigation.Type'].unique()
```

```
There are 2 investigation types:
```

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

What locations and countries did these incidents occur?

```python
# Countries where these incidents took place
print(f'These incidents took place in {df["Country"].nunique()}
Countries: ')
df['Country'].unique()
```

```
These incidents took place in 219 Countries:
```

```
array(['United States', 'Unknown', 'GULF OF MEXICO', 'Puerto Rico',
       'ATLANTIC OCEAN', 'HIGH ISLAND', 'Bahamas', 'MISSING',
'Pakistan',
       'Angola', 'Germany', 'Korea, Republic Of', 'Martinique',
       'American Samoa', 'PACIFIC OCEAN', 'Canada', 'Bolivia',
'Mexico',
       'Dominica', 'Netherlands Antilles', 'Iceland', 'Greece',
'Guam',
       'Australia', 'CARIBBEAN SEA', 'West Indies', 'Japan',
       'Philippines', 'Venezuela', 'Bermuda', 'San Juan Islands',
       'Colombia', 'El Salvador', 'United Kingdom',
       'British Virgin Islands', 'Netherlands', 'Costa Rica',
       'Mozambique', 'Jamaica', 'Panama', 'Guyana', 'Norway', 'Hong
```

```
Kong',
       'Portugal', 'Malaysia', 'Turks And Caicos Islands',
       'Northern Mariana Islands', 'Dominican Republic', 'Suriname',
       'Honduras', 'Congo', 'Belize', 'Guatemala', 'Anguilla',
'France',
       'St Vincent And The Grenadines', 'Haiti', 'Montserrat',
       'Papua New Guinea', 'Cayman Islands', 'Sweden', 'Taiwan',
       'Senegal', 'Barbados', 'BLOCK 651A', 'Brazil', 'Mauritius',
       'Argentina', 'Kenya', 'Ecuador', 'Aruba', 'Saudi Arabia',
'Cuba',
       'Italy', 'French Guiana', 'Denmark', 'Sudan', 'Spain',
       'Federated States Of Micronesia', 'St Lucia', 'Switzerland',
       'Central African Republic', 'Algeria', 'Turkey', 'Nicaragua',
       'Marshall Islands', 'Trinidad And Tobago', 'Poland', 'Belarus',
       'Austria', 'Malta', 'Cameroon', 'Solomon Islands', 'Zambia',
       'Peru', 'Croatia', 'Fiji', 'South Africa', 'India', 'Ethiopia',
       'Ireland', 'Chile', 'Antigua And Barbuda', 'Uganda', 'China',
       'Cambodia', 'Paraguay', 'Thailand', 'Belgium', 'Gambia',
'Uruguay',
       'Tanzania', 'Mali', 'Indonesia', 'Bahrain', 'Kazakhstan',
'Egypt',
       'Russia', 'Cyprus', "Cote D'ivoire", 'Nigeria', 'Greenland',
       'Vietnam', 'New Zealand', 'Singapore', 'Ghana', 'Gabon',
'Nepal',
       'Slovakia', 'Finland', 'Liberia', 'Romania', 'Maldives',
       'Antarctica', 'Zimbabwe', 'Botswana', 'Isle of Man', 'Latvia',
       'Niger', 'French Polynesia', 'Guadeloupe', 'Ivory Coast',
       'Tunisia', 'Eritrea', 'Gibraltar', 'Namibia', 'Czech Republic',
       'Benin', 'Bosnia And Herzegovina', 'Israel', 'Estonia',
       'St Kitts And Nevis', 'Sierra Leone', 'Corsica', 'Scotland',
       'Reunion', 'United Arab Emirates', 'Afghanistan', 'Ukraine',
       'Hungary', 'Bangladesh', 'Morocco', 'Iraq', 'Jordan', 'Qatar',
       'Madagascar', 'Malawi', 'Central Africa', 'South Sudan',
       'Saint Barthelemy', 'Micronesia', 'South Korea', 'Kyrgyzstan',
       'Turks And Caicos', 'Eswatini', 'Tokelau', 'Sint Maarten',
'Macao',
       'Seychelles', 'Rwanda', 'Palau', 'Luxembourg', 'Lebanon',
       'Bosnia and Herzegovina', 'Libya', 'Guinea',
       'Saint Vincent and the Grenadines', 'UN', 'Iran', 'Lithuania',
       'Malampa', 'Antigua and Barbuda', 'AY', 'Chad', 'Cayenne',
       'New Caledonia', 'Yemen', 'Slovenia', 'Nauru', 'Niue',
'Bulgaria',
       'Republic of North Macedonia', 'Virgin Islands', 'Somalia',
       'Pacific Ocean', 'Obyan', 'Mauritania', 'Albania', 'Wolseley',
       'Wallis and Futuna', 'Saint Pierre and Miquelon', 'Georgia',
       "Côte d'Ivoire", 'South Korean', 'Serbia', 'MU', 'Guernsey',
       'Great Britain', 'Turks and Caicos Islands'], dtype=object)
```

```python
# Locations where these incidents took place
print(f'These incidents took place in {df["Location"].nunique()}
```

```
locations: ')
df['Location'].unique()
```

```
These incidents took place in 27758 locations:

array(['MOOSE CREEK, ID', 'BRIDGEPORT, CA', 'Saltville, VA', ...,
       'San Manual, AZ', 'Auburn Hills, MI', 'Brasnorte, '],
dtype=object)
```

What is the purpose for flight

```
# Unique purposes of flight
print(f'There are {df["Purpose.of.flight"].nunique()} purposes of
flight: ')
df['Purpose.of.flight'].unique()
```

```
There are 26 purposes of flight:

array(['Personal', 'Business', 'Instructional', 'Unknown', 'Ferry',
       'Executive/corporate', 'Aerial Observation', 'Aerial
Application',
       'Public Aircraft', 'Skydiving', 'Other Work Use',
'Positioning',
       'Flight Test', 'Air Race/show', 'Air Drop',
       'Public Aircraft - Federal', 'Glider Tow',
       'Public Aircraft - Local', 'External Load',
       'Public Aircraft - State', 'Banner Tow', 'Firefighting',
       'Air Race show', 'PUBS', 'ASHO', 'PUBL'], dtype=object)
```

With this information, I will look to answer the following questions:

Statistical analysis of the dataset

Performing statistical analysis is essential to understand the dataset, detect data quality issues like outliers, and uncover relationships between variables. It helps turn raw data into actionable insights, guiding decision-making and allowing me to test hypotheses with data-driven evidence. In this project, it will help identify trends in aviation incidents, supporting informed decisions about aircraft purchases, safety measures, and current trends.

In this section, I intend to perform the following:

```
1. Find distributions in the numerical data
2. Perform correlation analysis
3. Perform frequency counts for categorical data
4. Find data outliers
5. Conduct trend analysis
6. Carry out geographical analysis
7. Compare all the above metrics and come up with insights
```

Analysis of the numerical data

Find numerical columns in the dataframe:

```
#numerical columns in the dataframe
numeric_columns = [col for col in df.columns if df[col].dtype in
['float64', 'int64']]
numeric_columns
```

```
['Number.of.Engines',
 'Total.Fatal.Injuries',
 'Total.Serious.Injuries',
 'Total.Minor.Injuries',
 'Total.Uninjured']
```

Find the count, mean, standard deviation, minimum value, quantiles, and maximum values of the numerical columns in the dataframe using the .describe() method. The mean gives the average value, which shows the central tendency of a dataset, similar to the quantiles. The min and max values on the other hand give the outliers of the dataset.

```
df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
    'Total.Minor.Injuries', 'Total.Uninjured']].describe()
```

```
       Number.of.Engines  Total.Fatal.Injuries  Total.Serious.Injuries
\
count       88889.000000          88889.000000            88889.000000

mean            1.136552              0.564761                0.240491

std             0.432545              5.126649                1.434614

min             0.000000              0.000000                0.000000

25%             1.000000              0.000000                0.000000

50%             1.000000              0.000000                0.000000

75%             1.000000              0.000000                0.000000

max             8.000000            349.000000              161.000000


       Total.Minor.Injuries  Total.Uninjured
count          88889.000000     88889.000000
mean               0.309127         5.037755
std                2.083715        26.990914
min                0.000000         0.000000
25%                0.000000         0.000000
50%                0.000000         1.000000
```

```
75%                    0.000000         2.000000
max                  380.000000       699.000000
```

Find the median of numerical values in the dataset. The median represents the middle value. Comparing the two helps understand if the data is skewed. For example, if the mean is higher than the median, it suggests the presence of higher values pulling the average up.

```
median = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
    'Total.Minor.Injuries', 'Total.Uninjured']].median()

print(f'\n\nThe median of the numerical columns are:\n\n{median}')
```

```


The median of the numerical columns are:

Number.of.Engines         1.0
Total.Fatal.Injuries      0.0
Total.Serious.Injuries    0.0
Total.Minor.Injuries      0.0
Total.Uninjured           1.0
dtype: float64
```

Finding the variance and standard deviation of numerical values in the dataset. The variance measures how spread out the data is, and the standard deviation gives this spread in the same units as the data. These are crucial for understanding the variability within the dataset, helping to identify whether most accidents result in similar outcomes or if there is wide fluctuation.

```
variance = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
    'Total.Minor.Injuries', 'Total.Uninjured']].var()

standard_deviation = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
    'Total.Minor.Injuries', 'Total.Uninjured']].std()

print(f'The variance of the numerical columns are:\n\n{variance}')

print(f'\n\nThe standard deviation of the numerical columns are:\n\
n{standard_deviation}')
```

```
The variance of the numerical columns are:

Number.of.Engines           0.187095
Total.Fatal.Injuries       26.282529
Total.Serious.Injuries      2.058118
Total.Minor.Injuries        4.341866
Total.Uninjured           728.509420
dtype: float64
```

```
The standard deviation of the numerical columns are:

Number.of.Engines         0.432545
Total.Fatal.Injuries      5.126649
Total.Serious.Injuries    1.434614
Total.Minor.Injuries      2.083715
Total.Uninjured          26.990914
dtype: float64
```

Create a correlation matrix. The covariance matrix is a measure of how two variables change together. It helps quantify the direction of the linear relationship between variables.

```
correlation_matrix = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
                         'Total.Minor.Injuries',
'Total.Uninjured']].corr()
print(correlation_matrix)
```

```
                        Number.of.Engines  Total.Fatal.Injuries  \
Number.of.Engines                1.000000              0.050789
Total.Fatal.Injuries             0.050789              1.000000
Total.Serious.Injuries           0.028226              0.108066
Total.Minor.Injuries             0.052285              0.035698
Total.Uninjured                  0.344710             -0.015009

                        Total.Serious.Injuries
Total.Minor.Injuries  \
Number.of.Engines                     0.028226              0.052285

Total.Fatal.Injuries                  0.108066              0.035698

Total.Serious.Injuries                1.000000              0.216400

Total.Minor.Injuries                  0.216400              1.000000

Total.Uninjured                       0.042116              0.098340


                        Total.Uninjured
Number.of.Engines              0.344710
Total.Fatal.Injuries          -0.015009
Total.Serious.Injuries         0.042116
Total.Minor.Injuries           0.098340
Total.Uninjured                1.000000
```

Skewness measures the asymmetry of the data distribution. Kurtosis measures the "tailedness" or the presence of extreme values (outliers). These metrics help in understanding the distribution's shape and whether outliers dominate.

```
# Calculating skewness using pandas
skewness = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
               'Total.Minor.Injuries', 'Total.Uninjured']].skew()

# Calculating kurtosis using pandas
kurtosis = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
               'Total.Minor.Injuries', 'Total.Uninjured']].kurt()

print("Skewness:\n", skewness)
print("Kurtosis:\n", kurtosis)

Skewness:
 Number.of.Engines          2.706529
Total.Fatal.Injuries      35.318019
Total.Serious.Injuries    53.005957
Total.Minor.Injuries      93.380236
Total.Uninjured            9.413431
dtype: float64
Kurtosis:
 Number.of.Engines             13.149635
Total.Fatal.Injuries       1552.209819
Total.Serious.Injuries     4315.727391
Total.Minor.Injuries      14193.993494
Total.Uninjured             112.312668
dtype: float64
```

Correlation measures the strength of the relationship between two variables. Strong correlations can highlight factors that might be contributing to higher injury or fatality rates, helping in understanding key drivers of safety outcomes.

```
# Correlation matrix for the numerical columns
correlation_matrix = df[['Number.of.Engines', 'Total.Fatal.Injuries',
'Total.Serious.Injuries',
                         'Total.Minor.Injuries',
'Total.Uninjured']].corr()

print("Correlation Matrix:\n", correlation_matrix)

Correlation Matrix:
                         Number.of.Engines  Total.Fatal.Injuries  \
Number.of.Engines                1.000000              0.050789
Total.Fatal.Injuries             0.050789              1.000000
Total.Serious.Injuries           0.028226              0.108066
Total.Minor.Injuries             0.052285              0.035698
Total.Uninjured                  0.344710             -0.015009

                        Total.Serious.Injuries
Total.Minor.Injuries  \
```

```
Number.of.Engines                    0.028226              0.052285

Total.Fatal.Injuries                 0.108066              0.035698

Total.Serious.Injuries               1.000000              0.216400

Total.Minor.Injuries                 0.216400              1.000000

Total.Uninjured                      0.042116              0.098340


                        Total.Uninjured
Number.of.Engines              0.344710
Total.Fatal.Injuries          -0.015009
Total.Serious.Injuries         0.042116
Total.Minor.Injuries           0.098340
Total.Uninjured                1.000000
```

Insights from statistical analysis the numerical data

Number of Engines

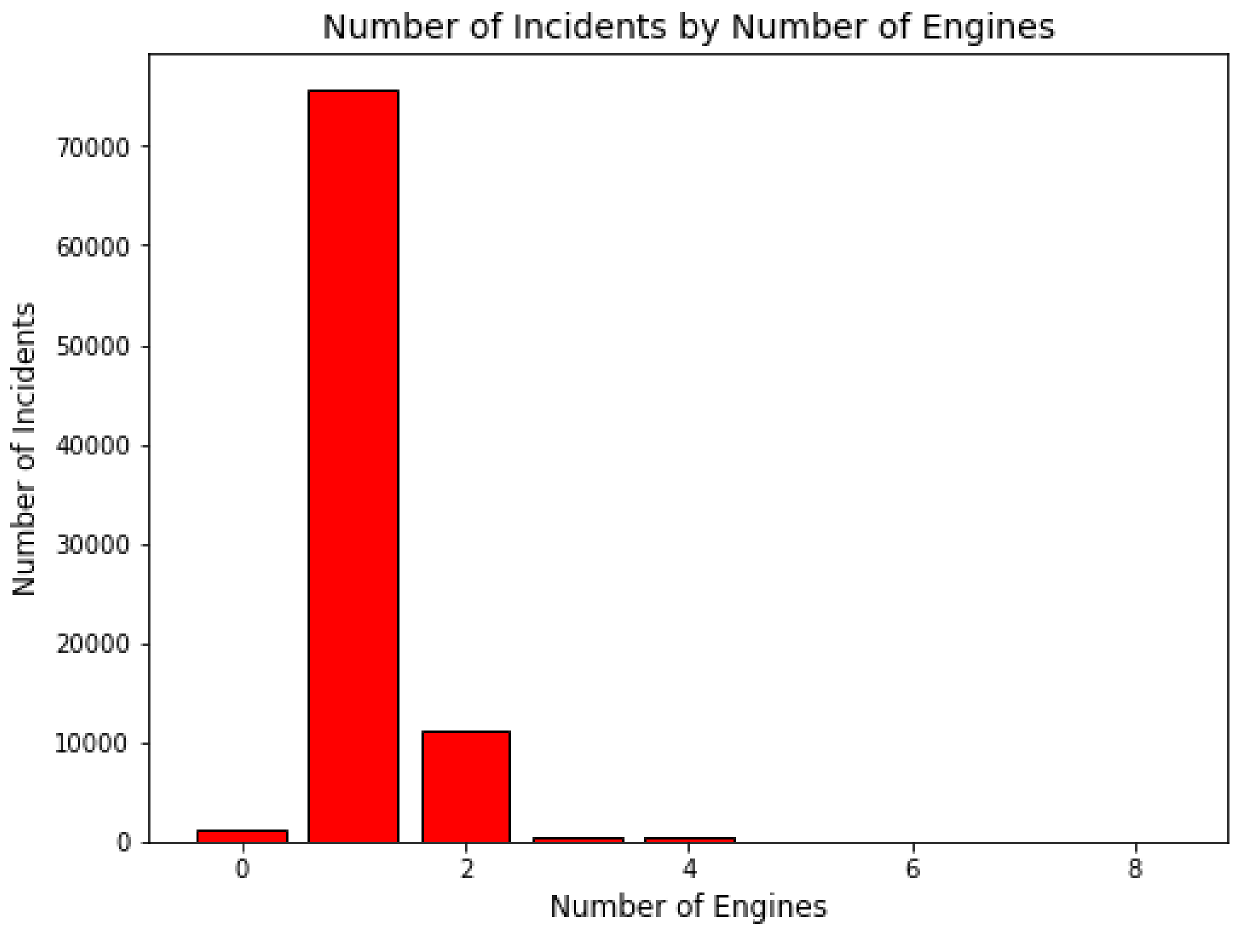
From the measures of central tendencies in the .describe(), We can gather that:

The mean engines used in aircrafts is 1.14, slightly higher than the median of 1, indicating a lot of planes with 1 engine are involved in incidences, with the slight difference between mean and median indicating some planes with more than 1 engine are involved in incidences.

Plot to support the same

A plot of number of engines against number of incidences can help prove this

```python
# Counting the number of incidents for each number of engines
engine_incidents = df['Number.of.Engines'].value_counts().sort_index()

# Plotting the bar graph
plt.figure(figsize=(8, 6))
plt.bar(engine_incidents.index, engine_incidents.values, color='red',
edgecolor='black')

# Adding titles and labels
plt.title('Number of Incidents by Number of Engines', fontsize=14)
plt.xlabel('Number of Engines', fontsize=12)
plt.ylabel('Number of Incidents', fontsize=12)

# Show plot
plt.show()
```

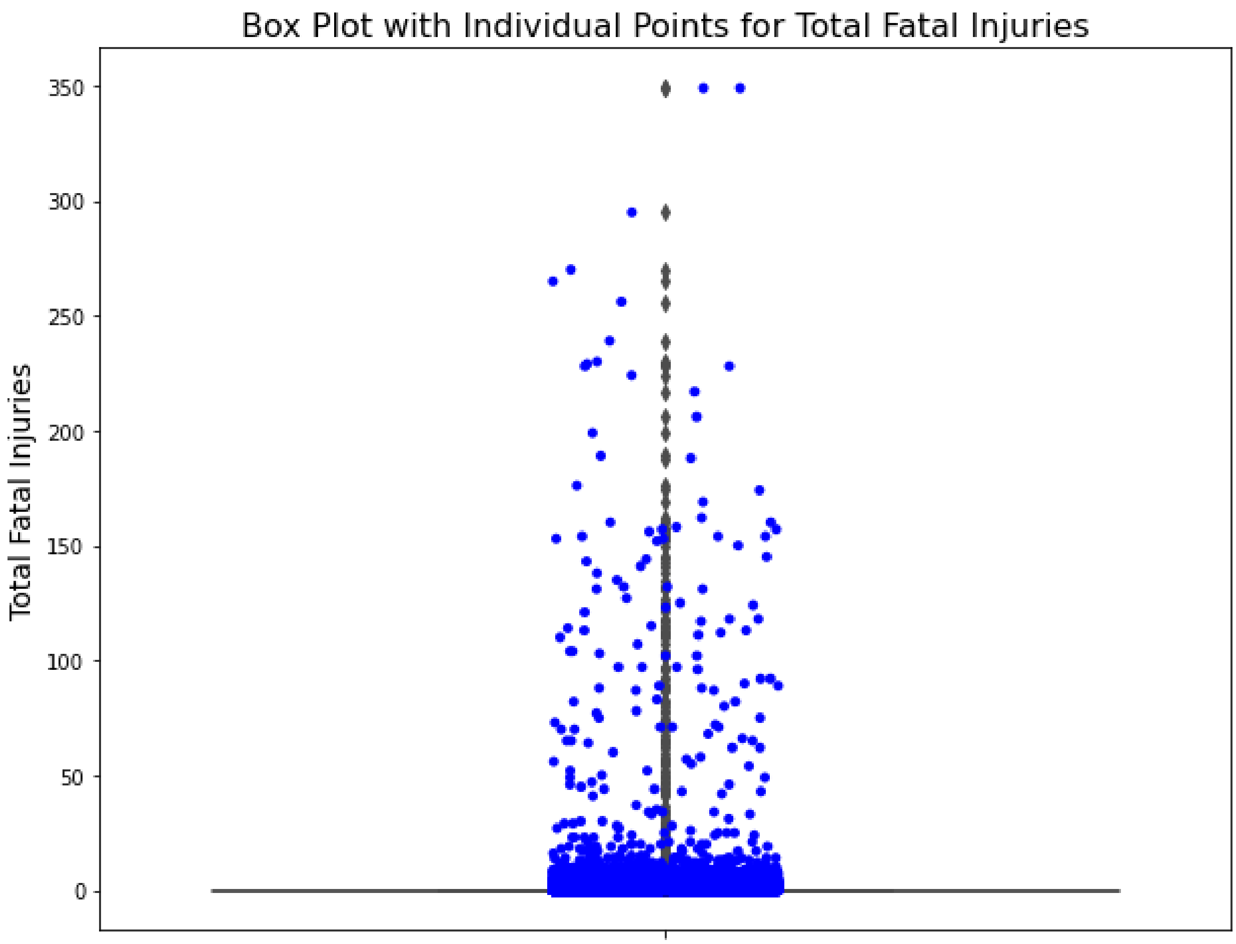


Total Fatal Injuries

With a mean of 0.56 and a median of 0, it indicates that most incidents do not have fatalities, but have big outliers, indicating when incidents occur, they claim a lot of fatalities

```
# Wider Box plot with strip plot for Total Fatal Injuries
plt.figure(figsize=(10, 8))  # Increase the figure size for a wider
plot

# Create the box plot
sns.boxplot(y=df['Total.Fatal.Injuries'], color='red')

# Overlay a strip plot to show individual data points
sns.stripplot(y=df['Total.Fatal.Injuries'], color='blue', size=5,
jitter=True)

# Adding titles and labels
plt.title('Box Plot with Individual Points for Total Fatal Injuries',
fontsize=16)
plt.ylabel('Total Fatal Injuries', fontsize=14)
```

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

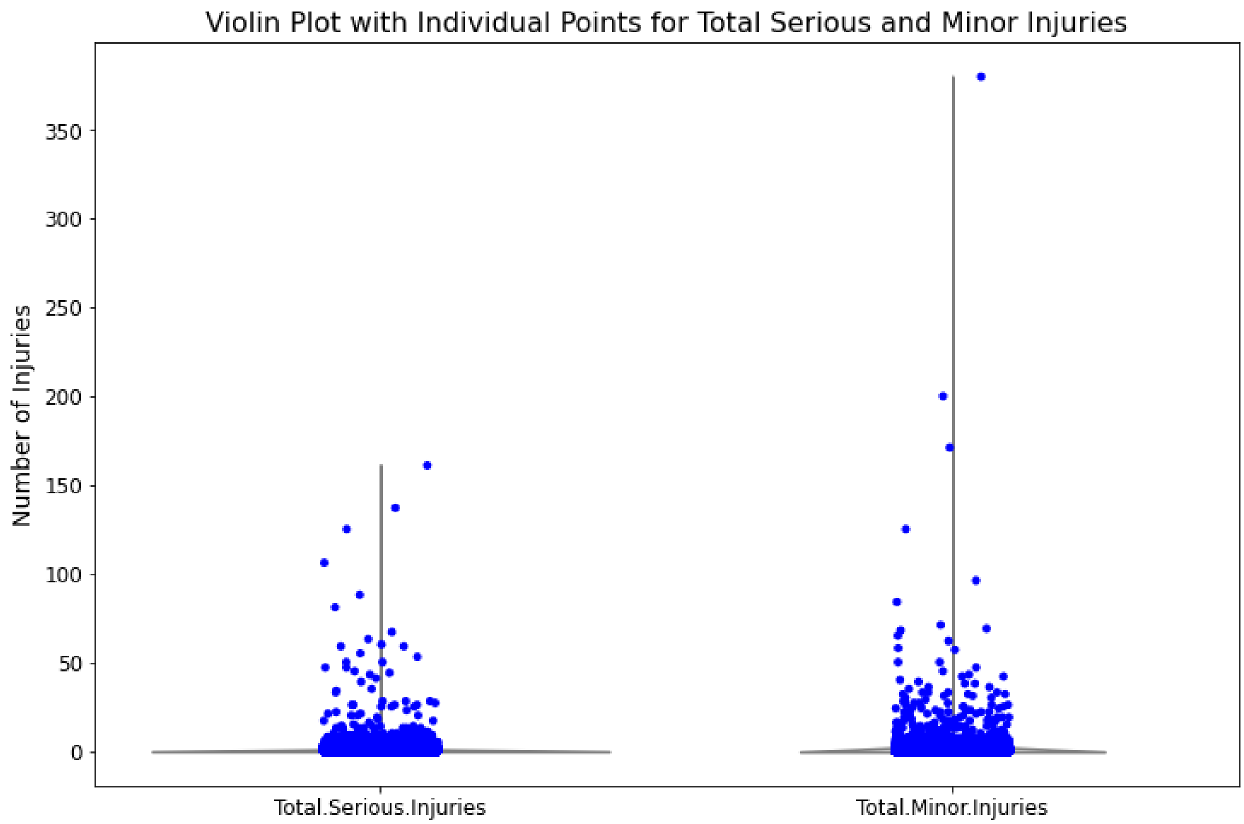


Total serious and Minor Injuries

They both have a median of 0, which shows accidents rarely result in any injuries, but with a mean of 0.24 for serious injuries and 0.31 for minor injuries, it is probable that when incidents occur they could lead to significant injuries

```
# Wider Violin plot with strip plot for Total Serious and Minor
Injuries
plt.figure(figsize=(12, 8))  # Increase the figure size for a wider
plot

# Create the violin plot
sns.violinplot(data=df[['Total.Serious.Injuries',
'Total.Minor.Injuries']], inner=None, color='lightgray')

# Overlay a strip plot to show individual data points with jitter for
visibility
```

```
sns.stripplot(data=df[['Total.Serious.Injuries',
'Total.Minor.Injuries']], color='blue', size=5, jitter=True)

plt.title('Violin Plot with Individual Points for Total Serious and
Minor Injuries', fontsize=16)
plt.ylabel('Number of Injuries', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



Total Uninjured

There mean of total uninjured people is 5.4, way higher than the mode of 1 which means accidents involve a single uninjured person, but there are a few incidents where many passengers escape unscathed

Variance and Standard deviations

First, let us look at the Total Fatal Injuries variable and its coefficient of variation which stands at 26.28 and the standard deviation of 5.13, this means that many times there are incidences with many fatalities.

The Total Uninjured also has a large coefficient of variation which is 26.99, meaning there is a high variability of passengers escape with out any injuries. This means that there are certain accidents where everyone survives an accident while in other accidents fewer individuals live survive.

Corelation, Skewness and Kurtosis

1. Number of Engines is weakly correlated with all injury-related metrics, indicating that having more engines does not strongly influence the number of fatalities or injuries.However from the mean, they have been involved in a lot more incidences.
2. Total Minor Injuries has a modest correlation with Total Serious Injuries (0.216), suggesting that accidents with serious injuries are somewhat more likely to also involve minor injuries.
3. All numerical columns exhibit positive skewness, meaning the distributions are heavily skewed to the right. This suggests that most accidents involve low or no fatalities/injuries, but there are a few extreme cases where many injuries or fatalities occur.
4. Kurtosis is extremely high, especially for Total.Fatal.Injuries (1552.21), Total.Serious.Injuries (4315.73), and Total.Minor.Injuries (14193.99). This indicates the presence of significant outliers—major accidents with high numbers of fatalities and injuries are very rare but highly impactful when they occur.

Insights from statistical analysis categorical and numerical data

What planes are most and least involved in accidents, based on their makes, models, and engine types?

```
# Group by Make, Model, and Engine Type to sum up the number of
incidents, fatalities, and injuries
safety_stats = df.groupby(['Make', 'Model', 'Engine.Type']).agg({
    'Total.Fatal.Injuries': 'sum',
    'Total.Serious.Injuries': 'sum',
    'Total.Minor.Injuries': 'sum',
    'Total.Uninjured': 'sum',
    'Event.Id': 'count'  # Counting the number of incidents
}).reset_index()

# Rename the columns for better readability
safety_stats.rename(columns={'Event.Id': 'Total.Incidents'},
inplace=True)

# Calculate the total number of injuries (fatal, serious, and minor
combined)
safety_stats['Total.Injuries'] = safety_stats['Total.Fatal.Injuries']
+ safety_stats['Total.Serious.Injuries'] +
safety_stats['Total.Minor.Injuries']

# Sort by total injuries (for discouraged) and reverse for recommended
recommended_aircraft = safety_stats.sort_values(by='Total.Injuries',
ascending=True).head(10)
```

```
discouraged_aircraft = safety_stats.sort_values(by='Total.Injuries',
ascending=False).head(10)

# Displaying the top 5 recommended aircraft combinations (low
injuries)
print("Top 5 Recommended Aircraft (low injuries):")
print(recommended_aircraft[['Make', 'Model', 'Engine.Type',
'Total.Incidents', 'Total.Injuries']])

# Displaying the top 5 discouraged aircraft combinations (high
injuries)
print("\nTop 5 Discouraged Aircraft (high injuries):")
print(discouraged_aircraft[['Make', 'Model', 'Engine.Type',
'Total.Incidents', 'Total.Injuries']])
```

```
Top 5 Recommended Aircraft (low injuries):
                    Make            Model    Engine.Type
Total.Incidents  \
10547  HARRITY WILLIAM V   GLASAIR (SH2F)  Reciprocating
1
5726              CESSNA             310A  Reciprocating
2
5725              CESSNA            310-R  Reciprocating
1
5724              CESSNA            310-L  Reciprocating
1
5723              CESSNA            310-J  Reciprocating
1
5722              CESSNA            310-H  Reciprocating
1
5721              CESSNA            310-D  Reciprocating
1
12685             Levick  RAF 2000 GTX-SE  Reciprocating
1
5719              CESSNA               31  Reciprocating
1
5718              CESSNA             305F  Reciprocating
1

       Total.Injuries
10547             0.0
5726              0.0
5725              0.0
5724              0.0
5723              0.0
5722              0.0
5721              0.0
12685             0.0
5719              0.0
5718              0.0
```

```
Top 5 Discouraged Aircraft (high injuries):
         Make      Model    Engine.Type  Total.Incidents
Total.Injuries
2624   BOEING        737  Reciprocating              440
1596.0
5418   CESSNA        172  Reciprocating             1752
1100.0
5392   CESSNA        152  Reciprocating             2410
1093.0
5476   CESSNA       172N  Reciprocating             1163
975.0
15847   Piper  PA-28-140  Reciprocating              807
874.0
5472   CESSNA       172M  Reciprocating              792
656.0
15868   Piper  PA-28-181  Reciprocating              474
628.0
5479   CESSNA       172P  Reciprocating              688
564.0
2665   BOEING    737-200  Reciprocating               12
555.0
3024   BOEING  777 - 206  Reciprocating                3
534.0
```

The Top 5 recommended aircraft involve mostly Cessna models, C208B and C207 with reciprocating engines. Some of these aircrafts have low to no cases of injuries or incidents thereby making them safer. On the other hand, the Top 5 discouraged aircraft are; Boeing 737 with reciprocating engines, Cessna aircrafts, Piper aircrafts all reciprocating engine type which recorded higher incidents and injuries as well. Despite this, these aircraft have been implicated in many accidents, meaning that selecting them increases the risk based on information from the mishaps.

What engines are most common in this flight report?

```
# Grouping by Engine Type to sum the fatal incidents
fatality_by_engine = df[df['Injury.Severity'] ==
'Fatal'].groupby('Engine.Type').size()

# Sorting and displaying the engine types with the highest number of
fatal incidents
fatality_by_engine_sorted =
fatality_by_engine.sort_values(ascending=False)

# Displaying the top 10 engine types with the most fatal incidents
fatality_by_engine_sorted.head(10)
```

```
Engine.Type
Reciprocating    4648
Turbo Prop        275
```

```
Turbo Shaft       243
Turbo Fan          58
Turbo Jet          27
Unknown             7
Electric            2
Hybrid Rocket       1
None                1
dtype: int64
```

The Engine Type data shows that Reciprocating engines are the most common, with 4,648 occurrences, followed by Turbo Prop (275) and Turbo Shaft (243). Less common engine types include Turbo Fan (58) and Turbo Jet (27). Rare engine types like Electric, Hybrid Rocket, and None are almost negligible, each with only one or two occurrences, while Unknown engines appear in 7 incidents.

In the countries that appear most, what makes and models are most common

Here we rank countries that appear most in the dataframe, group makes and models

```
# Grouping by Country to get the total number of incidents per country
incidents_by_country = df.groupby('Country').size()

# Sorting the countries by number of incidents in descending order and
selecting the top 5
top_5_countries =
incidents_by_country.sort_values(ascending=False).head(5)

# Grouping by Country, Make, and Model to analyze accident frequency
incidents_by_country_model = df.groupby(['Country', 'Make',
'Model']).size()
print("The following countries have the most accident incidences in
the world: \n")
# Loop through each of the top 5 countries and display the top 10
models for each
for country in top_5_countries.index:
    print(f"Top 10 aircraft models with the most incidents in
{country}:")

    # Filter for the specific country
    country_incidents = incidents_by_country_model.loc[country]

    # Sort the results and display the top 10 models with the highest
number of incidents
    country_incidents_sorted =
country_incidents.sort_values(ascending=False).head(10)

    # Displaying the result
```

```
    print(country_incidents_sorted)
    print("\n")
```

```
The following countries have the most accident incidences in the
world:

Top 10 aircraft models with the most incidents in United States:
Make    Model
CESSNA  152           2332
        172           1635
        172N          1136
Piper   PA-28-140      798
CESSNA  150            790
        172M           773
        172P           680
        180            616
        182            589
        150M           578
dtype: int64


Top 10 aircraft models with the most incidents in Brazil:
Make      Model
ROBINSON  R44      19
BOEING    737      10
PIPER     PA25     10
CESSNA    210      10
BEECH     58        8
BELL      206       7
CESSNA    182       7
BEECH     C90       6
CESSNA    208       5
          188       5
dtype: int64


Top 10 aircraft models with the most incidents in Canada:
Make          Model
ROBINSON      R44      9
CESSNA        172      8
BOEING        737      8
Bell          206B     7
CESSNA        A185F    6
              208B     6
              208      5
              182      5
BOEING        767      4
DE HAVILLAND  DHC2     3
dtype: int64
```

```
Top 10 aircraft models with the most incidents in Mexico:
Make      Model
BOEING    737      24
CESSNA    182      13
PIPER     PA25     11
AIRBUS    A320      6
CESSNA    210       5
          150       5
          206       5
          208       5
ROBINSON  R44       4
Bell      407       4
dtype: int64


Top 10 aircraft models with the most incidents in United Kingdom:
Make      Model
BOEING    737        28
          777         7
          747         6
          747-400     5
SIKORSKY  S92         4
BOEING    767         4
PIPER     PA28        4
BOEING    757         4
BEECH     200         4
BOEING    787         4
dtype: int64
```

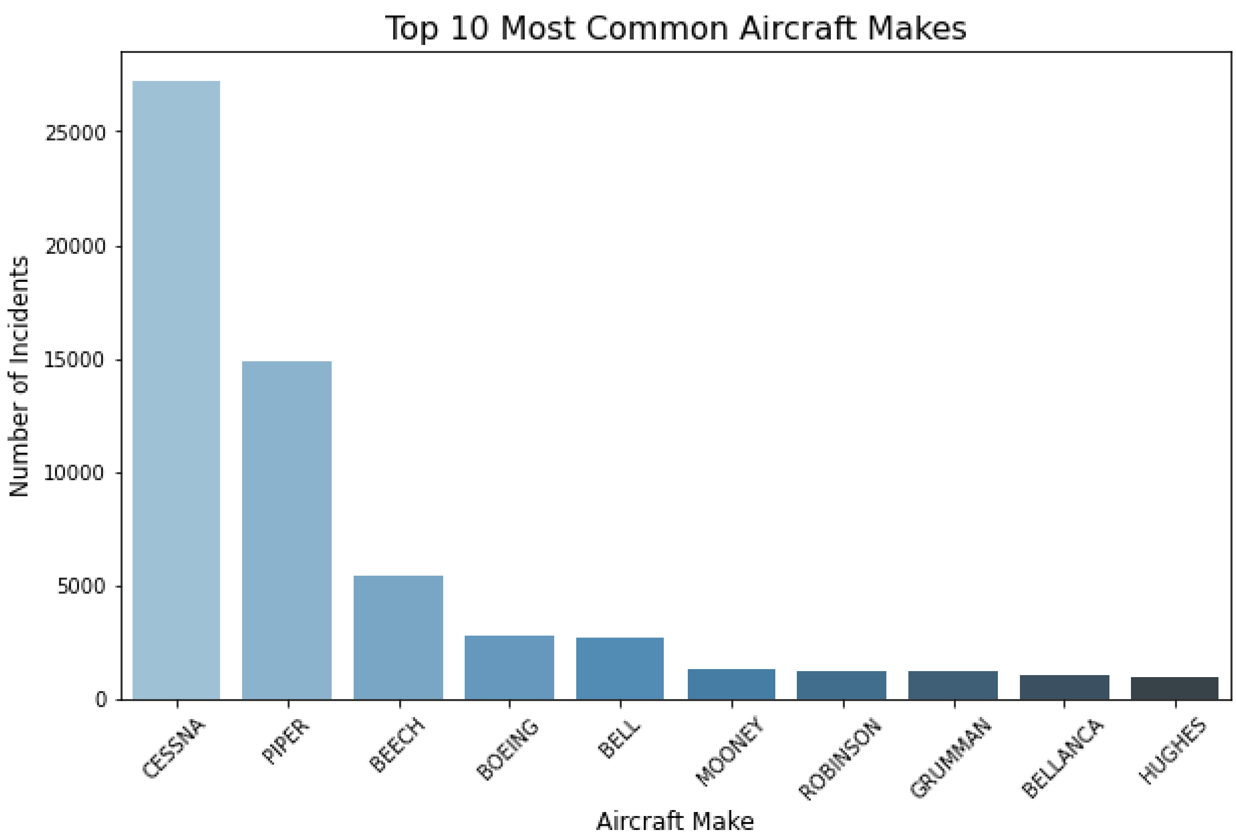
The Boeing 737 appears frequently across multiple countries, including the United States, Brazil, Canada, Mexico, and the United Kingdom, indicating that it is involved in a high number of incidents globally. Given its widespread use as a commercial aircraft, its presence in these incident reports suggests it is not only one of the most commonly flown planes but also one that consistently features in accident data, likely due to its high operational volume rather than an inherent safety issue.

What models and makes are most prone to incidents that lead to fatalities

We are plotting a bar graph of the 10 most accident prone aircraft in the dataframe, ranked from highest to lowest.

```
# Group by 'Make' and count the occurrences
most_common_makes = df['Make'].value_counts().head(10)  # Top 10 makes
by count
```

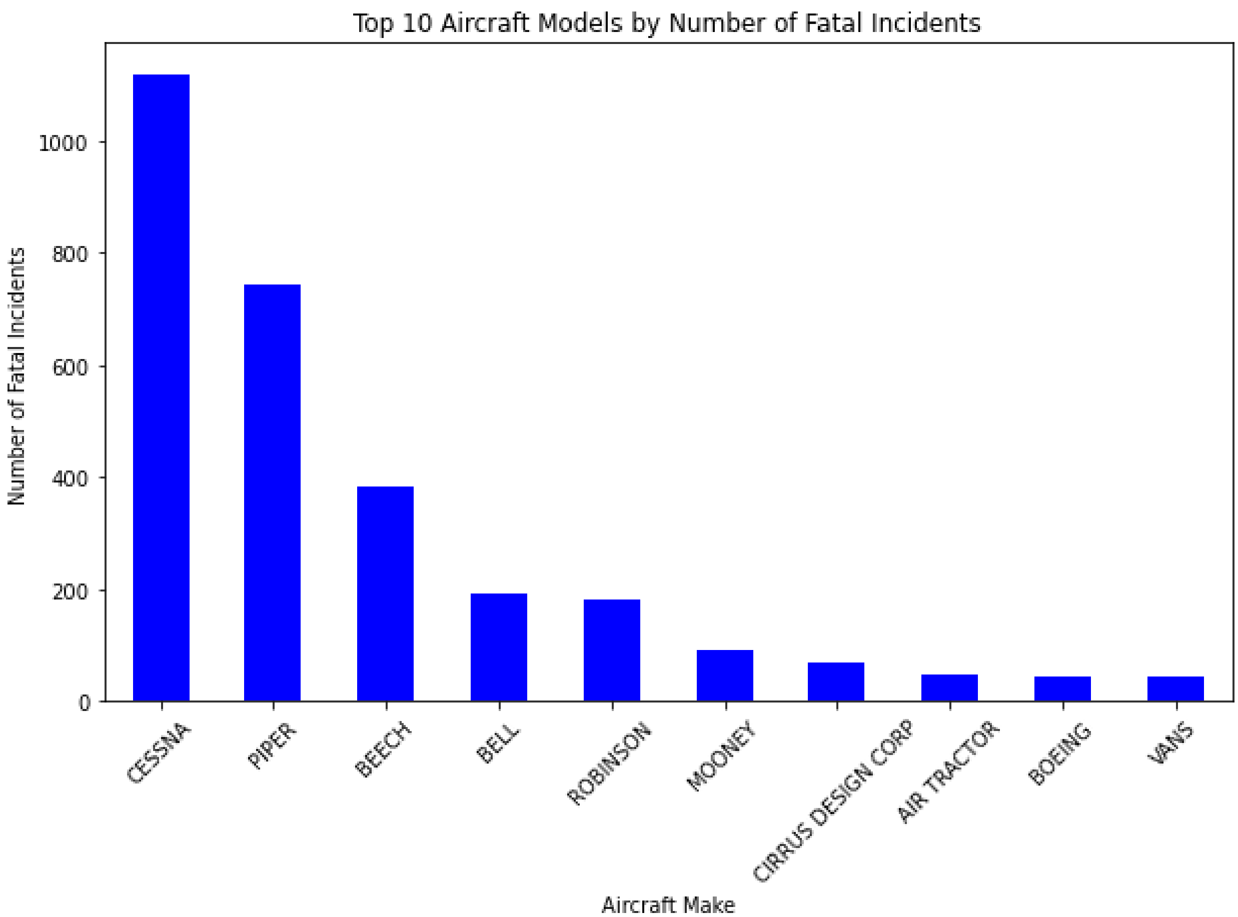
```
# Plotting the bar chart for most common makes
plt.figure(figsize=(10, 6))
sns.barplot(x=most_common_makes.index, y=most_common_makes.values,
palette='Blues_d')

# Adding titles and labels
plt.title('Top 10 Most Common Aircraft Makes', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Number of Incidents', fontsize=12)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show plot
plt.show()
```



The above is the number of aircrafts in the dataset according to make. This will offer context when we determine if an aircraft causes fatalities because of it's production numbers or the insufficience of it's safety systems.

```
# Group by Make and count the number of fatal incidents
fatal_incidents_by_model = df[df['Injury.Severity'] == 'Fatal']
['Make'].value_counts().head(10)
```

```
# Plot the result
plt.figure(figsize=(10, 6))
fatal_incidents_by_model.plot(kind='bar', color='blue')
plt.title('Top 10 Aircraft Models by Number of Fatal Incidents')
plt.xlabel('Aircraft Make')
plt.ylabel('Number of Fatal Incidents')
plt.xticks(rotation=45)
plt.show()
```



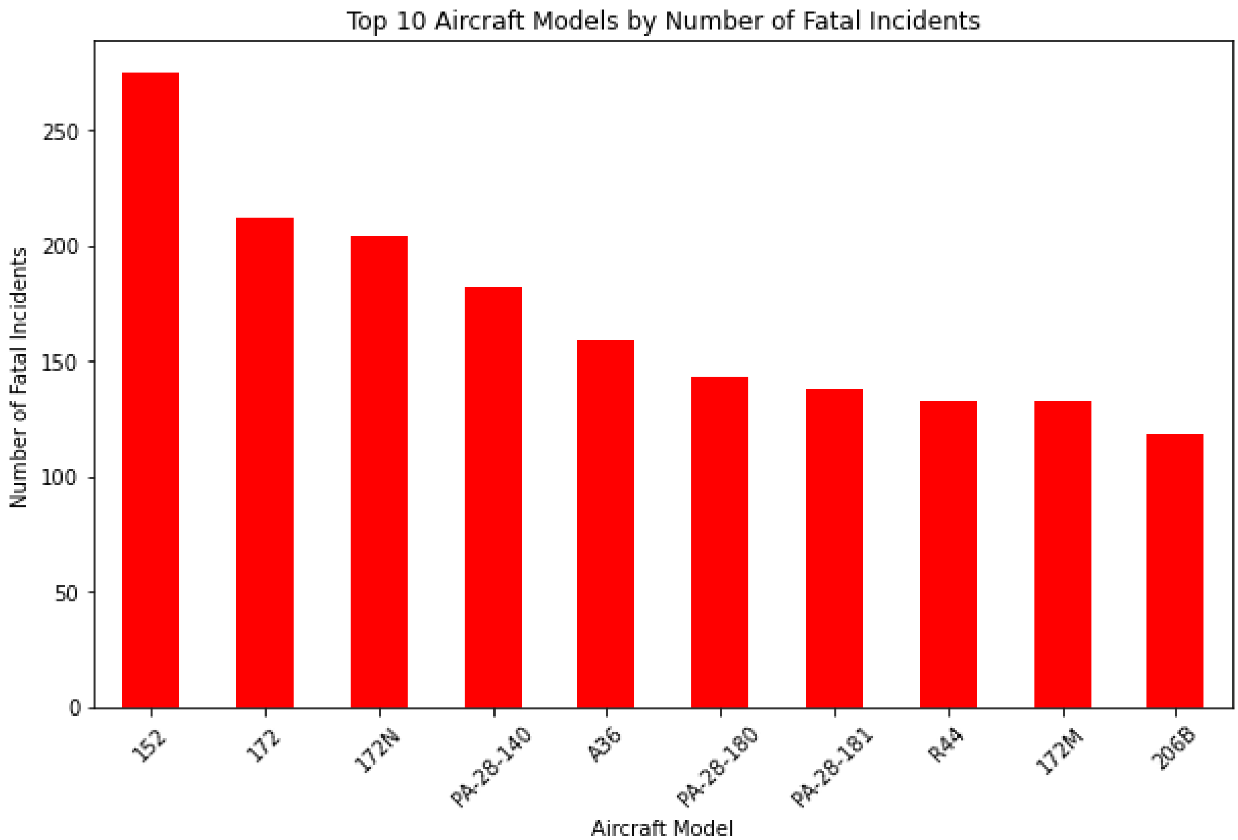
The CESSNA has appeared the most in this list, with its models also featuring a lot in the list of the top 5 countries with the most accidents. This could be the case because CESSNA is widely used in many parts of the world and in huge numbers.

Robinsons has moved up the list, possibly indicating questionable safety features to prevent casualties in cases of accidents. Boeing on the other hand has moved down the list, indicating high production volumes but good safety features that prevent casualities in cases of accidents.

Individual models that cause the most fatalities

```
# Filter dataset for incidents that involved fatalities
fatal_accidents = df[df['Total.Fatal.Injuries'] > 0]
```

```
# Group by Model and count the number of fatal incidents
fatal_incidents_by_model = 
fatal_accidents['Model'].value_counts().head(10)  # Top 10 models by 
fatal accident count

# Plot the result
plt.figure(figsize=(10, 6))
fatal_incidents_by_model.plot(kind='bar', color='red')
plt.title('Top 10 Aircraft Models by Number of Fatal Incidents')
plt.xlabel('Aircraft Model')
plt.ylabel('Number of Fatal Incidents')
plt.xticks(rotation=45)
plt.show()
```

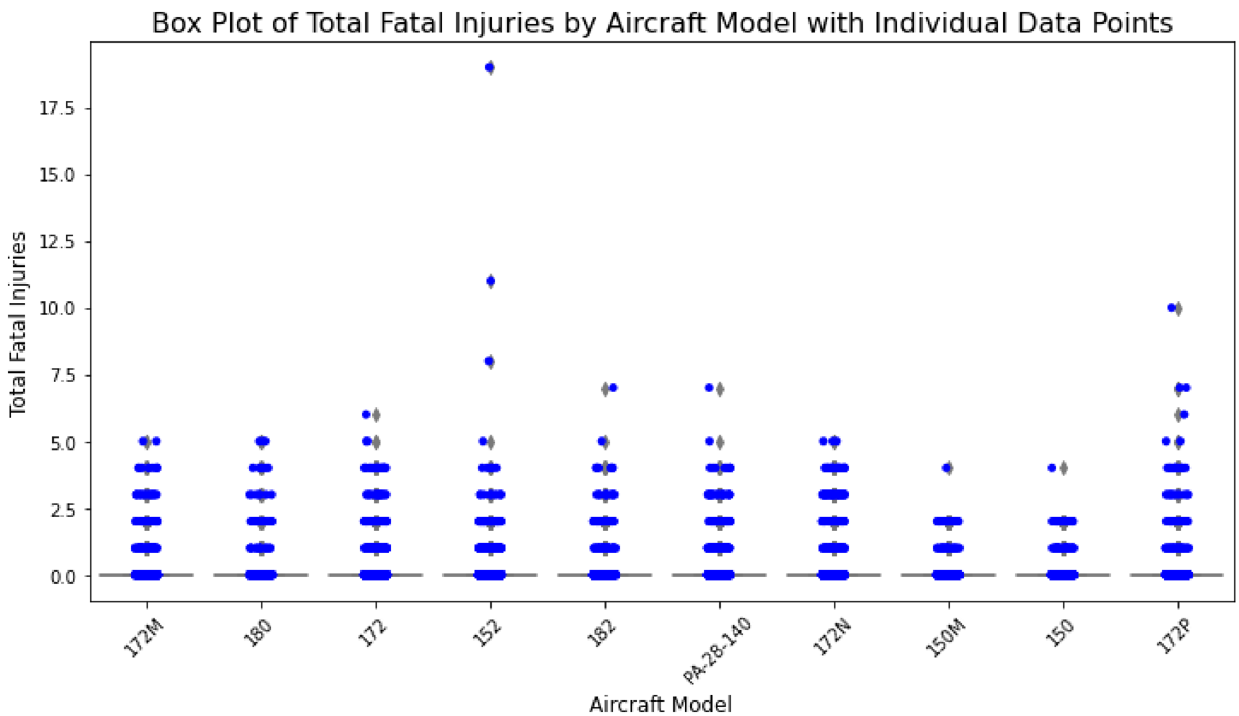


The 152 model leads to fatalities in accidents often, with 172, 172N, PA-28-140, A36, PA-28-180, PA-28-181, R44, 172M, 206B taking the rest of the positions in the list of 10 planes that cause casualties the most in cases they are involved incidents. This could be an indicator of how much safety equipment is installed in these plane models.

Boxplot representing the most common models and fatalities in accident cases

What are the most common aircraft models with incidents?

```
# Filter data for models with more than 5 incidents for meaningful
comparison
top_models = df['Model'].value_counts().head(10).index
df_filtered = df[df['Model'].isin(top_models)]

# Box plot for Total Fatal Injuries by Aircraft Model
plt.figure(figsize=(12, 6))

# Create the box plot
sns.boxplot(x='Model', y='Total.Fatal.Injuries', data=df_filtered,
color='lightblue')

# Overlay a strip plot to show individual data points
sns.stripplot(x='Model', y='Total.Fatal.Injuries', data=df_filtered,
color='blue', size=5, jitter=True)

# Customize titles and labels
plt.title('Box Plot of Total Fatal Injuries by Aircraft Model with
Individual Data Points', fontsize=16)
plt.xlabel('Aircraft Model', fontsize=12)
plt.ylabel('Total Fatal Injuries', fontsize=12)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=45)

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



Jitters were added to know the weight of plots in the boxplot.

What engines are regarded as most reliable or least reliable?

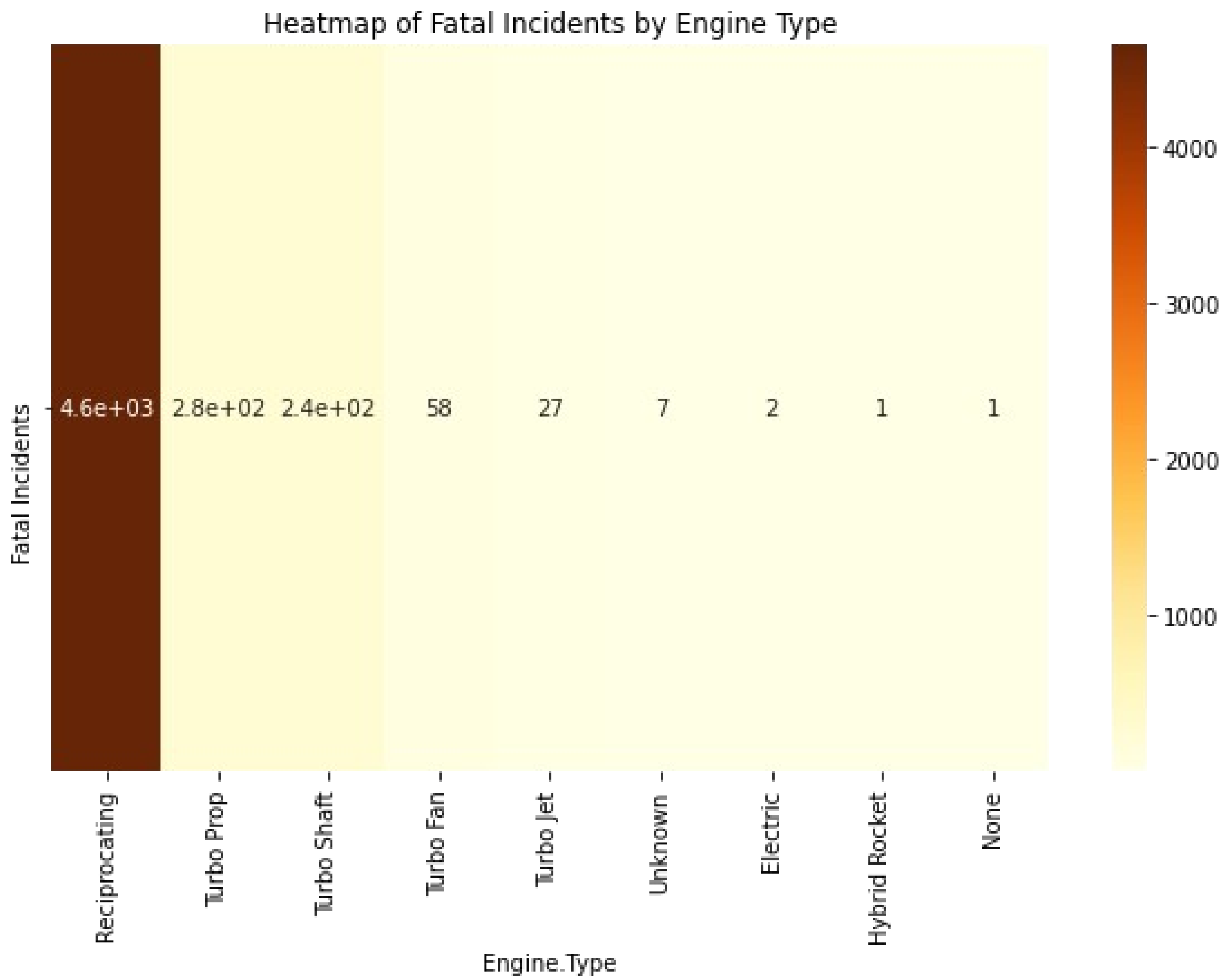
To do this, we have to find the number of engines in the dataset to get a good overview.

```
engines_count = df['Engine.Type'].value_counts()

# Display the result
print(engines_count)
```

```
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
```

```
# Group by Engine Type and Injury Severity (for Fatal incidents)
fatal_by_engine_type = df[df['Injury.Severity'] ==
```

```
'Fatal'].groupby('Engine.Type').size()

# Convert to DataFrame and sort
fatal_by_engine_type =
fatal_by_engine_type.reset_index().rename(columns={0: 'Fatal
Incidents'}).sort_values(by='Fatal Incidents',
ascending=False).head(10)
# Plot a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(fatal_by_engine_type.set_index('Engine.Type').T,
annot=True, cmap="YlOrBr")
plt.title('Heatmap of Fatal Incidents by Engine Type')
plt.show()
```




```
# Group by engine type and count fatal incidents (after cleaning the
data)
fatal_by_engine_type = df[df['Injury.Severity'] ==
'Fatal'].groupby('Engine.Type').size()
```

```
# Count the total number of engines by type
engines_count = df['Engine.Type'].value_counts()

# Now, calculate the fatality rate
fatality_rate = {}

for engine_type in fatal_by_engine_type.index:
    if engine_type in engines_count:
        fatality_rate[engine_type] =
(fatal_by_engine_type[engine_type] / engines_count[engine_type]) * 100

# Display the fatality rates
for engine, rate in fatality_rate.items():
    print(f'{engine}: {rate:.2f}%')
```

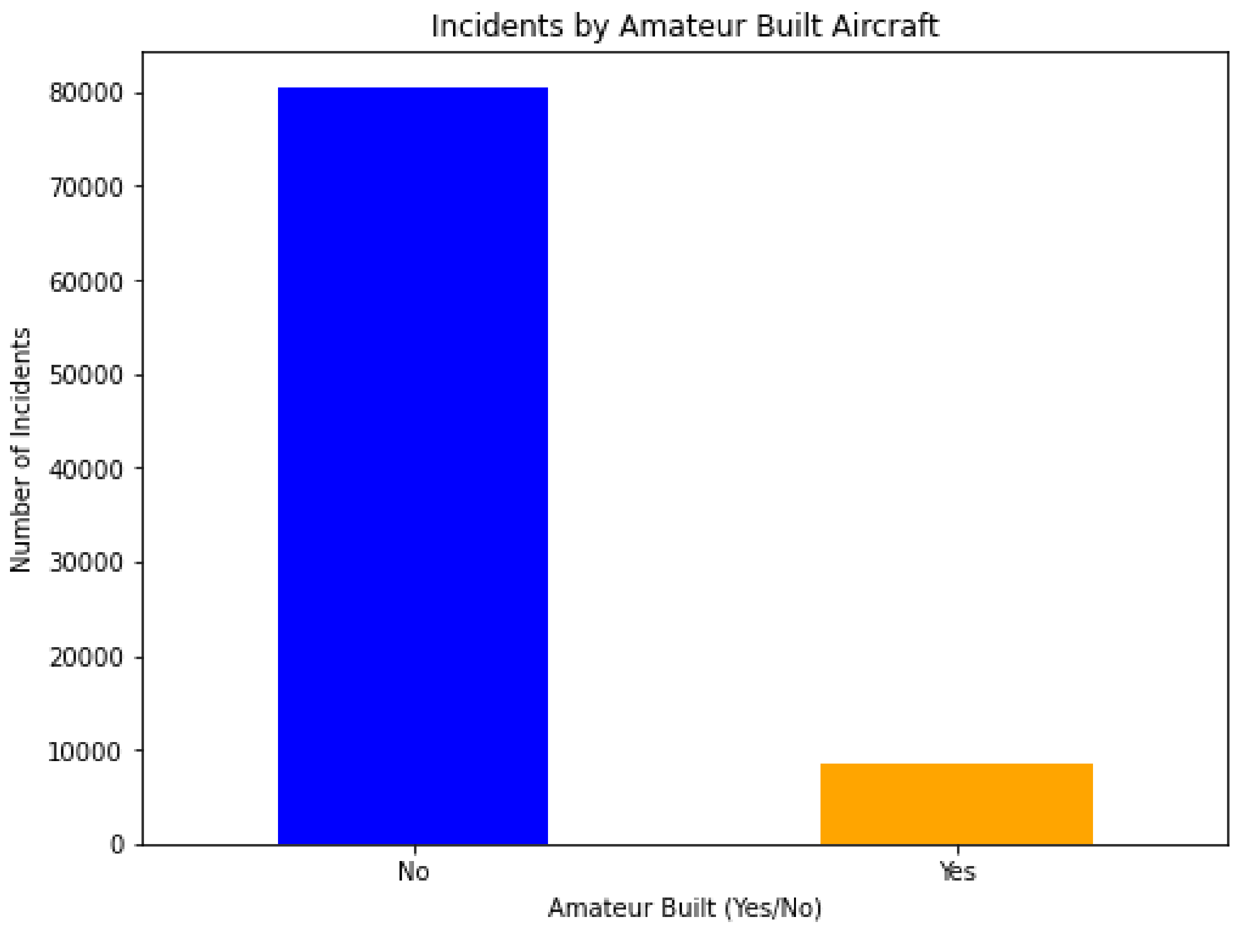
```
Electric: 20.00%
Hybrid Rocket: 100.00%
None: 5.26%
Reciprocating: 6.07%
Turbo Fan: 2.34%
Turbo Jet: 3.84%
Turbo Prop: 8.11%
Turbo Shaft: 6.73%
Unknown: 0.34%
```

From the above percentages, reciprocating engines are far more reliable than the number of incidents and fatalities indicate. Hybrid Rockets, and Turbo prop engines are far less reliable.

Are amateur built aircrafts reliable

The following is a bar comparison between amateur and professional made crafts in the incidence dataframe.

```
# Group by Amateur.Built and count the number of incidents
amateur_built_counts = df['Amateur.Built'].value_counts()

# Plot the bar chart
plt.figure(figsize=(8, 6))
amateur_built_counts.plot(kind='bar', color=['blue', 'orange'])
plt.title('Incidents by Amateur Built Aircraft')
plt.xlabel('Amateur Built (Yes/No)')
plt.ylabel('Number of Incidents')
plt.xticks(rotation=0)
plt.show()
```



Without information of production numbers, it can be realized that amateur crafts are less involved in accidents.

```
# Step 1: Group by 'Amateur.Built' and count total incidents
total_incidents = df['Amateur.Built'].value_counts()

# Step 2: Filter for fatal incidents and count fatal incidents by
'Amateur.Built'
fatal_incidents = df[df['Injury.Severity'] == 'Fatal']
['Amateur.Built'].value_counts()

# Step 3: Calculate the fatality rate for amateur and professional
aircraft
fatality_rate = (fatal_incidents / total_incidents) * 100

# Display the results
print("Fatality rate for amateur-built and professional aircraft:")
print(fatality_rate)
```

```
Fatality rate for amateur-built and professional aircraft:
No     5.669411
```

```
Yes    8.294985
Name: Amateur.Built, dtype: float64
```

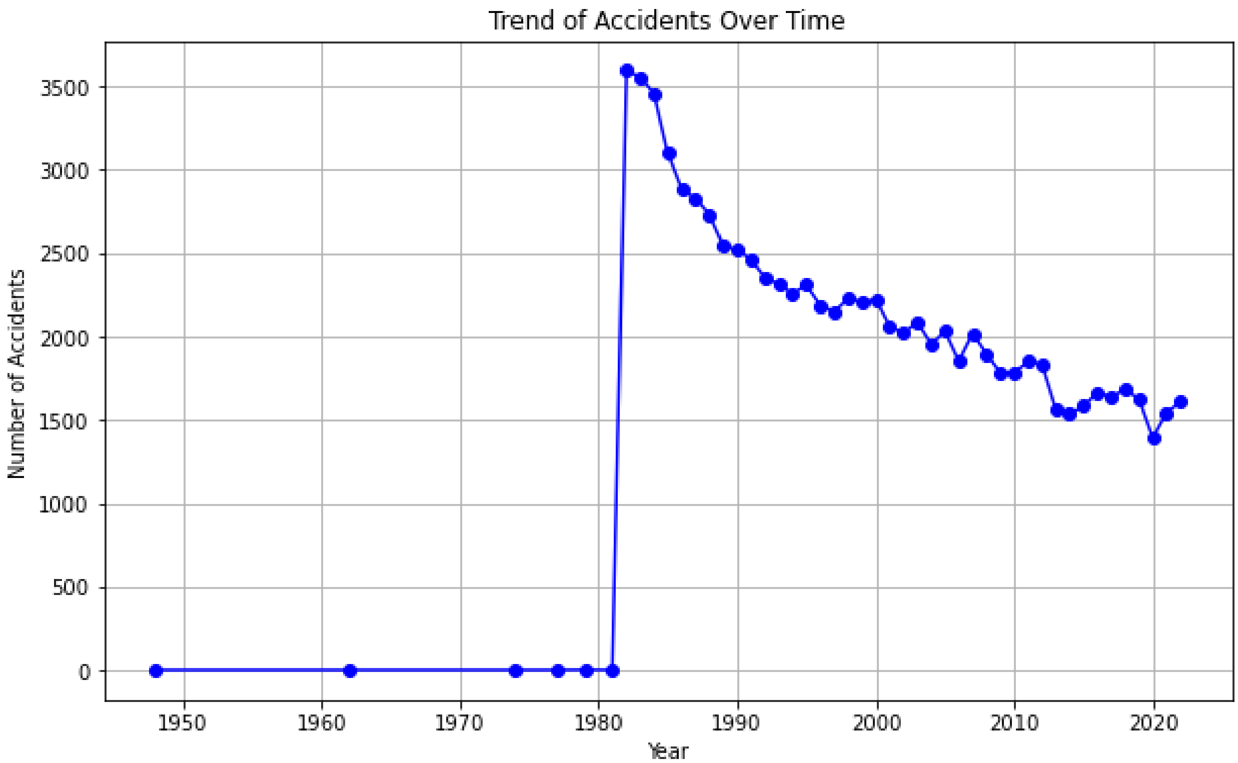
However, amateur built crafts cause 3% more fatalities than professionally made aircrafts.

```
df.columns

Index(['Event.Id', 'Investigation.Type', 'Accident.Number',
'Event.Date',
       'Location', 'Country', 'Injury.Severity', 'Aircraft.damage',
       'Registration.Number', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Report.Status', 'Publication.Date'],
      dtype='object')
```

Trends of airplane incidences over time

The below code investigates the trend of accidents over time, to get an accurate insight of if air travel is becoming safer.

```
# Convert Event.Date to datetime format (if not already done)
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')

# Extract the year from the Event.Date column
df['Year'] = df['Event.Date'].dt.year

# Group by year and count the number of incidents per year
accidents_per_year = df.groupby('Year').size()

# Plot the trend of accidents over time
plt.figure(figsize=(10, 6))
accidents_per_year.plot(kind='line', color='blue', marker='o')
plt.title('Trend of Accidents Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.grid(True)
plt.show()
```

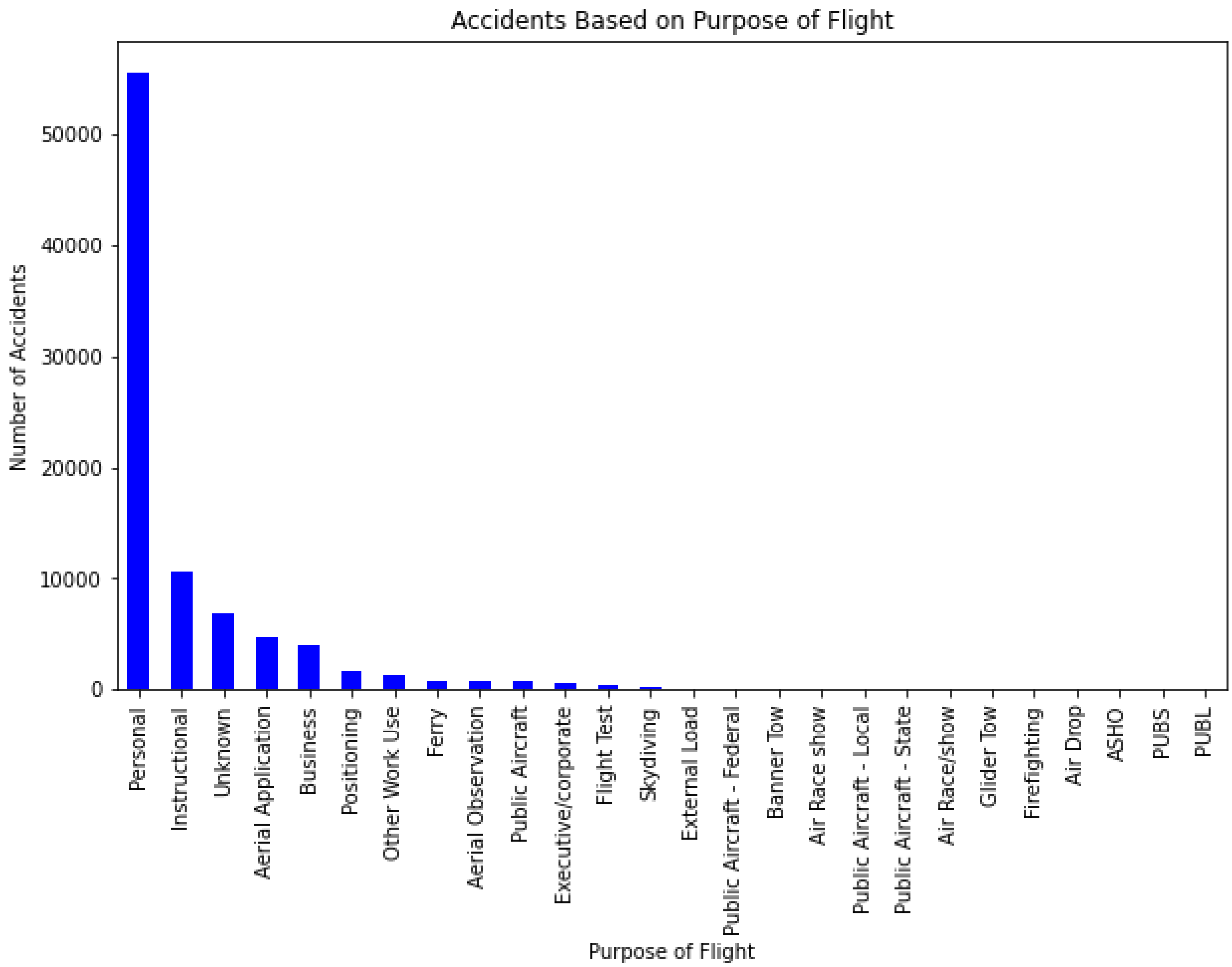


Generally, it can be noted that the number of aircraft involved in accidents has been on a steady decline since 1982, possibly indicating improvement in safety standards of planes and improvement of plane engineering. Modern planes can be assumed to be more reliable therefore.

Purpose of flight in relation to number of accidents

To find the distribution of accidents in the purpose of flight, I will plot a bar graph of the same.

```
# Group by Purpose.of.flight and count the number of incidents
accidents_by_purpose = df['Purpose.of.flight'].value_counts()

# Plot a bar chart to show accidents based on the purpose of flight
plt.figure(figsize=(10, 6))
accidents_by_purpose.plot(kind='bar', color='blue')
plt.title('Accidents Based on Purpose of Flight')
plt.xlabel('Purpose of Flight')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=90)
plt.show()
```



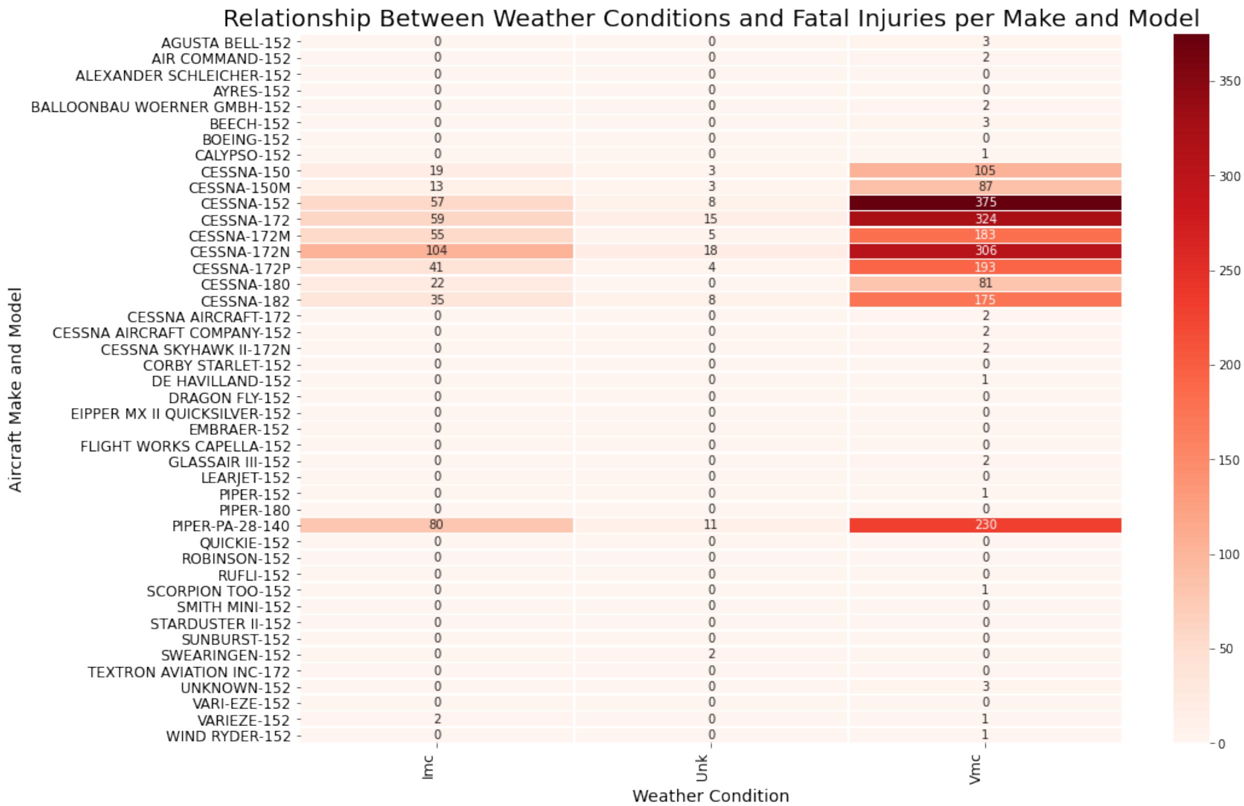
It is clear that personal flights have the highest number of accidents, followed by instructional and aerial application flights.

The relationship between weather and accidents

Using a heatmap, an analysis of how weather relates to the number of fatalities will be done. This is important in finding which weather is the most suitable, or unsuitable for flying.

```
# Filter data for models with more than 5 incidents for meaningful
comparison
top_models = df['Model'].value_counts().head(10).index
df_filtered = df[df['Model'].isin(top_models)]

# Group by Make, Model, and Weather Condition, then sum up the
casualties
weather_casualties = df_filtered.groupby(['Make', 'Model',
'Weather.Condition'])[['Total.Fatal.Injuries',
'Total.Serious.Injuries', 'Total.Minor.Injuries']].sum().reset_index()

# Pivot the table to get a better format for visualization (Total
```

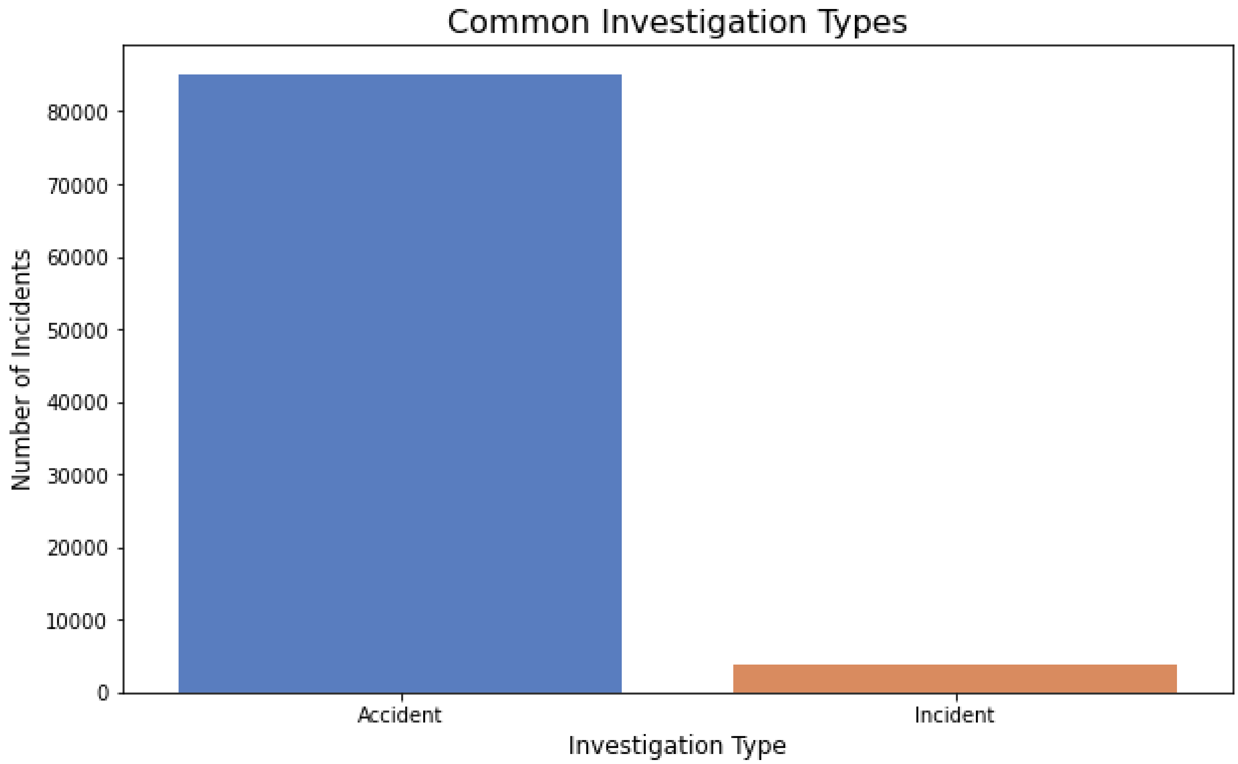
```
Fatal Injuries in this case)
weather_casualties_pivot =
weather_casualties.pivot_table(index=['Make', 'Model'],
columns='Weather.Condition', values='Total.Fatal.Injuries',
fill_value=0)

# Set up a larger figure size for more space
plt.figure(figsize=(16, 10))

# Plotting the heatmap for the relationship between weather conditions
and fatalities
sns.heatmap(weather_casualties_pivot, cmap='Reds', annot=True,
fmt='g', linewidths=.5)

# Adding titles and labels with larger font sizes
plt.title('Relationship Between Weather Conditions and Fatal Injuries
per Make and Model', fontsize=20)
plt.xlabel('Weather Condition', fontsize=14)
plt.ylabel('Aircraft Make and Model', fontsize=14)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=90, ha='right', fontsize=12)
plt.yticks(fontsize=12)

# Add more space between the elements
plt.tight_layout(pad=3)

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



The weather condition with the highest number of fatal injuries is IMC (Instrument Meteorological Conditions), followed by VMC (Visual Meteorological Conditions) and unk (unknown).The heatmap also shows that the Cessna 172N has the highest number of fatal injuries, followed by the Cessna 172P and the Cessna 172.

Nature of incidents that happen in aviation

```
# Group by 'Investigation.Type' to count the number of occurrences for each type
investigation_counts = df['Investigation.Type'].value_counts()

# plotting a bar graph
plt.figure(figsize=(10, 6))
sns.barplot(x=investigation_counts.index, y=investigation_counts.values, palette='muted')

# add all labels and titles
plt.title('Common Investigation Types', fontsize=16)
plt.xlabel('Investigation Type', fontsize=12)
plt.ylabel('Number of Incidents', fontsize=12)

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



The bar chart shows that most investigations are for accidents, likely due to their severity and legal requirements. Incidents, while still important, may not warrant the same level of investigation. This could be attributed to factors like the extent of harm, insurance claims, and public safety concerns.

Recommendations from findings

1. Aircraft Models and Makes

The statistics reveal that some of the most common model types to be implicated in accidents include Cessna 152, Cessna 172, Piper PA-28 and that some of the worst accidents have occurred in regions such as the United States of America and Brazil. The Boeing 737 is also seen frequently in the multiple countries.

Recommendation:

a. Conduct a Risk Assessment Before Purchase: While Cessna 152, 172 and Boeing 737 models are common in use, the high frequency in accidents leads to the recommendation before purchase that careful risk analysis be done. Remember always to assess the particular safety profiles, the maintenance histories, and the usage environments in which these models have been implemented.

b. Consider Alternative Models: As for the safer options the ratings for Cessna C208B and C207 are less fatal incidents report as compared to the previous models. These models are linked with safer risks, such that they can be used for operations with high-frequency.

c. Encourage Pilot Familiarity: A lot of them could be attributed to operational factors, or the fact that the models are relatively new and have not been flown by experienced pilots. Select the aircraft based on certain pilot competencies, or you can provide supplemental training to pilots on those specific aircraft models to lower the essential human mistakes.

d. Avoid Models with High Maintenance Costs: Many occurrences could mean that it experiences more usage or likely to be older machine such as the Boeing 737. Take into account the lifetime cost and instead of getting a highly likely to fail in the long run model, go for a new or a modern model equipment.

2. Engine Types

Those used most are the Reciprocating engines, but these cause the most fatal incidents on an average. Turbo Prop and Turbo Shaft on the other hand has very low ranking of fatal occurrence compared to other aircraft.

Recommendation:

Prioritize planes that have more than 2 engines. These are less likely to be involved in incidences.

Prioritize Turbo Prop and Turbo Shaft Engines: Due to the fact that they are more safer, ensure you acquire aircraft with Turbo Prop and Turbo Shaft engines; especially for commercial use.

Limit Use of Reciprocating Engines in High-Risk Conditions: Although reciprocating engines are common, the use appears to be associated with a higher risk of fatal crashes. Restrict the employment of such planes in activities that might experience difficult weather conditions, lengthy flights, or areas where malfunction could be disastrous.

Implement Rigorous Maintenance for Reciprocating Engines: If reciprocating engines are necessary, then proper and regular examinations of the engines should minimally be considered to decrease the possibilities of an engine break down. Pilots and mechanics should be encouraged to report early any deviations that they note in the expectation of preventing major problems arising.

Explore Hybrid and Electric Engine Options: Even though it is less frequent, there are promising new engine technologies such as Electric and Hybrid Rocket engines. It is time to begin analyzing these options for performing specialized functions with less of an environmental footprint and lower continued costs for business while maintaining a technological advantage.

3. Temporal Distribution and Occurrence of Incidents

When comparing the results of the evaluation with the numbers of accidents happening over time, it is observed that there is a cyclic pattern of incidents. Some years have more accidents, may be as a result to base or extraneous influences such as operational population density, shift in regulations or sometimes weather influences.

Recommendation:

Implement Predictive Maintenance Schedules: For efficient maintenance and safety check up schedules, design from trends analysis methods. By identifying when there are increased possibilities of incidents in a period of many flights, then there won’t be any wrong mechanical failures and operational hitches.

Monitor Regulatory Changes: This is noted whereby modifications in aviation rules may occur alongside changes in the number ofincidents, because of implementation of new rules or procedures. Pay attention to regulatory bodies and ensure your fleet and training programs meet the updates.

Track Incident Patterns by Model Age: Some models of aircraft used may show a general increase in the rate of incidents over the years. For the older models, it is recommended that you should either phase-out such vehicles of replace them in order to enhance the safety of your overall vehicle fleet.

Use Incident Data for Forecasting: Develop models that could identify probable risky periods for a certain extent to allow planning in advance for any harm susceptible to occur in those periods with high risk for a given type of aircraft.

4. Weather Conditions

Incidents under Instrument Meteorological Conditions (IMC) have a higher fatality rate compared to Visual Meteorological Conditions (VMC). Poor visibility and adverse weather contribute significantly to severe accidents.

Recommendation:

Invest in Advanced Avionics for IMC Flights: To minimize the dangers associated with IMC, outfit your plane with enhanced avionics systems including EGPWSs and TAWSs. These technologies will help pilots to disorient themselves while flying in adverse weather conditions.

Prioritize Aircraft with Strong IMC Safety Records: Select aircraft models that have had prior performance in IMC environments. Incident data should be used to evaluate which models work best under adverse weather conditions, and aircrafts with advanced avionics should be selected.

Enhance IMC Pilot Training: Make sure all pilots go through relevant training which regards difficult weather conditions. This entails imitation of IMC training and repeat performances of how to get through low visibility conditions.

Delay Flights in Severe Weather: Set up higher standards of procedure for when a flight is precisely scheduled for when the climate is bad for travel. When IMC is expected, operational schedules should be subordinate to safety considerations.

5. Technology Comparing Amateur and Professional Aircraft

The fatality rate of the amateur aircraft (8.29%) is higher than that of the professionally built aircraft (5.67%) and therefore the didactic shows that the amateur aircraft is more dangerous to fly.

Recommendation:

Limit Use of Amateur-Built Aircraft for Commercial Operations: Since hobby built aircraft have higher fatal accident rates than their certified counterparts, they should not be put to commercial or high-risk uses. This indicates that these types of aircrafts are more suitable for leisure or non-commercial flights where safety standards are relaxed.

Improve Inspection and Certification for Amateur Aircraft: For organizations that own amateur built aircraft, it is important to involve a more technical assessment and accreditation process to ensure that such planes are meeting the safety standards as those of professional planes.

Focus on Professional Aircraft for Long-Term Investments: In the case of long-term investment in the fleet, purchase professional aircraft models that are safe in the industry. These aircrafts are more safer and has less risk of fatal ocurrence.

6. Regional Trends

Some areas, including the United States, Brazil and Canada show many occurrences across certain models such as Cessna 152, Piper PA-28, Boeing 737.

Recommendation:

Region-Specific Aircraft Selection: Regional considerations: The environment in which your operations are carried out will determine the kind of aircraft that provides greater safety. For instance, do not consider models that have high incidence rate in certain countries but rather consider models that have good record in other similar country locations.

Adapt Maintenance Schedules to Regional Needs: There may also be regional conditions that might vary the general experience of HAs and cause influence the durability and safety of an aircraft (such as high humidity, freezing temperature, or high altitude). Adapt the company's levels of maintenance and programs of renewal of fleets depending on the operating conditions of the regions.

7. Additional Recommendations

Data-Driven Fleet Management: Sustaining the data analysis of aircraft performance, operation, and maintenance as well as accident reports. The following information can be used for effective decisions regarding purchases of new fleets, the retirement of old fleets, and improvements to existing fleets.

Diversify Fleet with Safety in Mind: Do not rely on any particular aircraft type. This means that by having a wide variation of fleets, you cannot have a problem that would endanger the whole business. Introduce Automation and AI for Safety: Discuss the application of other Artificial Intelligence systems that might be used for maintenance prediction, pilot evaluation, and safety improvement. These systems can further minimise human factors and business dangers.

Export data for plotting

```
df.to_csv('clean_data.csv', header=True, index=True)
```