# AVIATION RISK ANALYSIS: IDENTIFYING LOW-RISK AIRCRAFT FOR BUSINESS EXPANSION


## Author: Ian Bett


## Overview

The aviation industry presents significant opportunities for business expansion, but it also comes with inherent risks. This project aims to provide actionable insights into aircraft safety to guide informed decisions for investing in commercial and private aviation. Using a dataset from the National Transportation Safety Board (NTSB), which includes detailed records of aviation accidents, I analyzed key factors such as aircraft make, model, engine type, number of engines and weather conditions. By leveraging statistical methods and data visualizations, we identified trends and risk factors associated with different aircraft types. Our findings offers data-driven recommendations to minimize risk and optimize investment decisions, focusing on the safest and most reliable aircraft makes and models.

# Business Problem

My company is expanding into the aviation industry to diversify its portfolio. The goal is to identify the safest and lowest-risk aircraft for both commercial and private operations. The aviation industry carries significant safety risks, and understanding the patterns and causes of accidents is critical to making informed decisions about which aircraft to purchase. This analysis will help the company reduce risks and ensure a successful entry into the new industry.

#### Business Pain Points

- **Safety Concerns**: Aviation accidents can result in significant financial losses and damage to reputation. The company needs to minimize the risk of accidents by selecting reliable and safe aircraft.
- **Knowledge Gap**: The company lacks expertise in evaluating aircraft safety and understanding which factors contribute to higher accident risks.
- **Decision Support**: Without data-driven insights, choosing the right aircraft could be a costly and risky endeavor.

#### Data Analysis Questions

The following questions were chosen to address the business's needs:

1. **Which aircraft makes and models have the highest and lowest accident counts?**
    - Helps identify safer aircraft and avoid those with a history of frequent accidents.
2. **What are the patterns of accidents based on engine type and number of engines?**
    - Provides insights into the reliability of different engine configurations and helps assess risks associated with specific aircraft types.
3. **How do meteorological conditions (VMC vs. IMC) impact accident counts?**
    - Helps evaluate risks related to operating conditions and aids in planning operational strategies.
4. **Which aircraft make or model is associated with the most or least severe outcomes, such as fatalities or serious injuries?**
    - Identifies aircraft with better safety.

#### Importance to the Business

Answering these questions is critical for making informed decisions about aircraft purchases. By understanding accident trends and identifying low-risk aircraft:

- The company can mitigate potential losses and liabilities.
- It ensures better operational safety and reliability.
- The findings can shape future policies, such as pilot training or maintenance schedules, based on aircraft-specific risks.

These insights provide the foundation for data-driven recommendations that align with the company’s goal of entering the aviation industry successfully and safely.

# Data Understanding

### Data Source

The dataset used for this project was sourced from [Kaggle (https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses)](https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses). It contains detailed information about aviation accidents, which is critical for analyzing risks associated with different aircraft types.

### Data Representation

The data represent aviation accident records, primarily including information about aircraft make, model, engine type, number of engines, and accident outcomes such as injuries and fatalities.

Key variables included in the dataset are:

- **Make and Model**: Specifies the manufacturer and model of the aircraft.
- **Injury Severity**: Indicates whether the accident resulted in to fatal or unfatal injuries.
- **Total Fatal Injuries**: The number of fatalities per accident.
- **Total Serious Injuries**: The number of serious injuries per accident.
- **Meteorological Conditions**: Specifies whether the accident occurred under Visual Meteorological Conditions (VMC) or Instrument Meteorological Conditions (IMC).
- **Engine Type**: Type of engine powering the aircraft (e.g., Reciprocating, Turbo Shaft, Turbo Prop, Turbo Fan).
- **Number of Engines**: The number of engines on the aircraft.

### Target Variable

The target variable for this analysis is the **"Total fatal injuries"** column, which represents the total number of fatalities in an accident. This variable is crucial for identifying high-risk aircraft and assessing safety.

### Variable Properties

The key variables intended for use have the following properties:

- **Categorical Variables**:
  - `investigation type` : We'll focus on accidents only rather than incidents.
  - `make & model` : Combines aircraft make and model for detailed analysis.
  - `engine type` : Identifies the type of engine powering the aircraft.
  - `number of engines` : Categorizes aircraft by the number of engines.
  - `injury severity` : Classifies accidents based on injury outcomes.
- **Numerical Variables**:
  - `Total fatal injuries` : Represents the total number of fatalities.
  - `Total serious injuries` : Represents the total number of serious injuries.

In [1]:
```
#importing the packages I will be using for this project
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

# Aviation accident data

We start by exploring the data through typical data exploration method and attributes. We take a look at the first rows of data and the data types of columns.

In [2]:
```
# reading the csv file
avi_df = pd.read_csv("Data\AviationData.csv", encoding="latin1", low_memory=Fa
lse )

# previewing the DataFrame
avi_df.head()
```

Out[2]:

| | Event.Id | Investigation.Type | Accident.Number | Event.Date | Location | Country | La |
|---|---|---|---|---|---|---|---|
| **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 | 36.9 |
| **3** | 20001218X45448 | Accident | LAX96LA321 | 1977-06-19 | EUREKA, CA | United States | |
| **4** | 20041105X01764 | Accident | CHI79FA064 | 1979-08-02 | Canton, OH | United States | |

5 rows × 31 columns

In [3]:
```
# List of data's columns
avi_df.columns
```

Out[3]:
```
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.Descriptio
n',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injurie
s',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [4]:
```
# shape of the dataset
avi_df.shape
```

Out[4]:
```
(88889, 31)
```

In [5]:
```
# Lists column names, data types, and non-null counts.
avi_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50249 non-null  object 
 9   Airport.Name            52790 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87572 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81812 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82508 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [6]:
```
# Summary statistics of the data
avi_df.describe()
```

Out[6]:

| | Number.of.Engines | Total.Fatal.Injuries | Total.Serious.Injuries | Total.Minor.Injuries | Total.Uninj |
|---|---|---|---|---|---|
| **count** | 82805.000000 | 77488.000000 | 76379.000000 | 76956.000000 | 82977.00 |
| **mean** | 1.146585 | 0.647855 | 0.279881 | 0.357061 | 5.32 |
| **std** | 0.446510 | 5.485960 | 1.544084 | 2.235625 | 27.91 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **25%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **50%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.00 |
| **75%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 2.00 |
| **max** | 8.000000 | 349.000000 | 161.000000 | 380.000000 | 699.00 |

In [7]:
```
#Finding the percentages of null columns
avi_df.isna().sum() / len(avi_df) * 100
```

Out[7]:
```
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
```

# Data Preparation

To prepare the dataset for analysis, several steps were undertaken to ensure the data was clean, relevant, and aligned with the business problem of identifying low-risk aircraft. Each key step has a title.

## Dropping irrelevant columns

Dropping irrelevant columns unrelated to safety and risk assessment and columns with a higher percentage of null values

In [8]:
```
# Creating a list of the columns to drop
cols_to_drop = ['Accident.Number','Latitude', 'Longitude', 'Airport.Code', 'Ai
rport.Name','Aircraft.Category',
                'FAR.Description', 'Schedule','Registration.Number',
                'Air.carrier','Amateur.Built','Broad.phase.of.flight',
                'Publication.Date', 'Report.Status']
```

In [9]:
```
# Copying the original dataframe
avi_df_clean = avi_df.drop(columns=cols_to_drop)
```

In [10]:
```
# Checking the remaining columns
avi_df_clean.columns
```

Out[10]:
```
Index(['Event.Id', 'Investigation.Type', 'Event.Date', 'Location', 'Country',
       'Injury.Severity', 'Aircraft.damage', 'Make', 'Model',
       'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition'],
      dtype='object')
```

In [11]:
```
# Checking the sum of null values in each column
avi_df_clean.isna().sum()
```

Out[11]:
```
Event.Id                      0
Investigation.Type            0
Event.Date                    0
Location                     52
Country                     226
Injury.Severity            1000
Aircraft.damage            3194
Make                         63
Model                        92
Number.of.Engines          6084
Engine.Type                7077
Purpose.of.flight          6192
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
dtype: int64
```

In [12]:
```
# Checking the dataframe
avi_df_clean.head()
```

Out[12]:

| | Event.Id | Investigation.Type | Event.Date | Location | Country | Injury.Severity | Aircraf |
|---|---|---|---|---|---|---|---|
| **0** | 20001218X45444 | Accident | 1948-10-24 | MOOSE CREEK, ID | United States | Fatal(2) | |
| **1** | 20001218X45447 | Accident | 1962-07-19 | BRIDGEPORT, CA | United States | Fatal(4) | |
| **2** | 20061025X01555 | Accident | 1974-08-30 | Saltville, VA | United States | Fatal(3) | |
| **3** | 20001218X45448 | Accident | 1977-06-19 | EUREKA, CA | United States | Fatal(2) | |
| **4** | 20041105X01764 | Accident | 1979-08-02 | Canton, OH | United States | Fatal(1) | |

# Renaming the columns to a consistent and conscise format

In [13]:
```
# Dictionary to rename old column names to new column names
column_rename = {
    "Event.Id": "event_id",
    "Investigation.Type": "investigation_type",
    "Event.Date": "event_date",
    "Location": "location",
    "Country": "country",
    "Injury.Severity": "injury_severity",
    "Aircraft.damage": "aircraft_damage",
    "Make": "make",
    "Model": "model",
    "Number.of.Engines": "number_of_engines",
    "Engine.Type": "engine_type",
    "Purpose.of.flight": "purpose_of_flight",
    "Total.Fatal.Injuries": "total_fatal_injuries",
    "Total.Serious.Injuries": "total_serious_injuries",
    "Total.Minor.Injuries": "total_minor_injuries",
    "Total.Uninjured": "total_uninjured",
    "Weather.Condition": "weather_condition"
}

# Renaming the columns
avi_df_clean.rename(columns=column_rename, inplace=True)

# Display the new column names
print(avi_df_clean.columns)
```

```
Index(['event_id', 'investigation_type', 'event_date', 'location', 'country',
       'injury_severity', 'aircraft_damage', 'make', 'model',
       'number_of_engines', 'engine_type', 'purpose_of_flight',
       'total_fatal_injuries', 'total_serious_injuries',
       'total_minor_injuries', 'total_uninjured', 'weather_condition'],
      dtype='object')
```

# Filtering the investigation type column to accidents only

## Reasons

Accidents typically involve more severe outcomes such as fatalities, injuries, or substantial damage, making them critical to assessing risks. Concentrating on accidents aligns better with identifying safer aircraft for investment, as the business seeks to minimize operational risks.

In [14]:
```
# Checking the unique value counts in investigation type
avi_df_clean['investigation_type'].value_counts()
```

Out[14]:
```
Accident    85015
Incident     3874
Name: investigation_type, dtype: int64
```

In [15]:
```
# Filter for accidents only
avi_df_clean = avi_df_clean[avi_df_clean['investigation_type'] == 'Accident']
```

In [16]:
```
# Rechecking the unique value counts
avi_df_clean['investigation_type'].value_counts()
```

Out[16]:
```
Accident    85015
Name: investigation_type, dtype: int64
```

In [17]:
```
# Checking the dataframe
avi_df_clean.head()
```

Out[17]:

| | event_id | investigation_type | event_date | location | country | injury_severity | aircra |
|---|---|---|---|---|---|---|---|
| **0** | 20001218X45444 | Accident | 1948-10-24 | MOOSE CREEK, ID | United States | Fatal(2) | |
| **1** | 20001218X45447 | Accident | 1962-07-19 | BRIDGEPORT, CA | United States | Fatal(4) | |
| **2** | 20061025X01555 | Accident | 1974-08-30 | Saltville, VA | United States | Fatal(3) | |
| **3** | 20001218X45448 | Accident | 1977-06-19 | EUREKA, CA | United States | Fatal(2) | |
| **4** | 20041105X01764 | Accident | 1979-08-02 | Canton, OH | United States | Fatal(1) | |

## Cleaning injury_severity and total_fatal_injuries columns

I realized that the values in parenthesis in injury_severity column are actually in the total_fatal_injuries column, which contains a lot of null values. Therefore I created a new column `totalfatal_injuries` that extracts the numerical values from injury_severity column.

In [18]:
```
avi_df_clean.loc[:,["injury_severity", "total_fatal_injuries"]]
```

Out[18]:

| | injury_severity | total_fatal_injuries |
|---|---|---|
| **0** | Fatal(2) | 2.0 |
| **1** | Fatal(4) | 4.0 |
| **2** | Fatal(3) | 3.0 |
| **3** | Fatal(2) | 2.0 |
| **4** | Fatal(1) | 1.0 |
| **...** | ... | ... |
| **88884** | Minor | 0.0 |
| **88885** | NaN | 0.0 |
| **88886** | Non-Fatal | 0.0 |
| **88887** | NaN | 0.0 |
| **88888** | Minor | 0.0 |

85015 rows × 2 columns

In [19]:
```
# Creating a new column
# Extracting numbers from the 'injury_severity' column using regex & Fill miss
ing values in 'totalfatal_injuries' with 0
avi_df_clean['totalfatal_injuries'] = avi_df_clean['injury_severity'].str.extr
act(r'\((\d+)\)').fillna(0).astype(int)

print(avi_df_clean['totalfatal_injuries'].head())
```

```
0    2
1    4
2    3
3    2
4    1
Name: totalfatal_injuries, dtype: int32
```

In [20]:

```
# Verifying the assumption
avi_df_clean.loc[10060:10100, ["injury_severity", "totalfatal_injuries"]]
```

Out[20]:

| | injury_severity | totalfatal_injuries |
|---|---|---|
| **10060** | Fatal(2) | 2 |
| **10061** | Fatal(2) | 2 |
| **10062** | Non-Fatal | 0 |
| **10063** | Non-Fatal | 0 |
| **10064** | Non-Fatal | 0 |
| **10065** | Non-Fatal | 0 |
| **10066** | Non-Fatal | 0 |
| **10067** | Non-Fatal | 0 |
| **10068** | Fatal(4) | 4 |
| **10069** | Non-Fatal | 0 |
| **10070** | Non-Fatal | 0 |
| **10071** | Non-Fatal | 0 |
| **10072** | Non-Fatal | 0 |
| **10073** | Non-Fatal | 0 |
| **10074** | Non-Fatal | 0 |
| **10075** | Non-Fatal | 0 |
| **10076** | Non-Fatal | 0 |
| **10077** | Non-Fatal | 0 |
| **10079** | Fatal(1) | 1 |
| **10080** | Non-Fatal | 0 |
| **10081** | Non-Fatal | 0 |
| **10082** | Non-Fatal | 0 |
| **10083** | Fatal(1) | 1 |
| **10086** | Non-Fatal | 0 |
| **10087** | Non-Fatal | 0 |
| **10088** | Non-Fatal | 0 |
| **10089** | Non-Fatal | 0 |
| **10090** | Non-Fatal | 0 |
| **10091** | Fatal(1) | 1 |
| **10092** | Fatal(2) | 2 |
| **10093** | Non-Fatal | 0 |
| **10094** | Non-Fatal | 0 |
| **10095** | Non-Fatal | 0 |
| **10096** | Fatal(1) | 1 |
| **10097** | Non-Fatal | 0 |
| **10098** | Fatal(1) | 1 |

| | injury_severity | totalfatal_injuries |
|---|---|---|
| **10099** | Non-Fatal | 0 |
| **10100** | Fatal(3) | 3 |

In [21]:
```
# Remove values in parentheses from 'injury_severity' to remain with unique va
lues
avi_df_clean['injury_severity'] = avi_df_clean['injury_severity'].str.replace
(r'\(.*?\)', '', regex=True).str.strip()

# Displaying the updated DataFrame to verify
print(avi_df_clean['injury_severity'])
```

```
0            Fatal
1            Fatal
2            Fatal
3            Fatal
4            Fatal
           ...
88884        Minor
88885          NaN
88886    Non-Fatal
88887          NaN
88888        Minor
Name: injury_severity, Length: 85015, dtype: object
```

In [22]:
```
# Checking the value counts
avi_df_clean["injury_severity"].value_counts()
```

Out[22]:
```
Non-Fatal      66380
Fatal          17825
Minor            215
Serious          172
Unavailable       96
Name: injury_severity, dtype: int64
```

# Creating a new column to Categorize injury severity into broader groups

For easier understanding of the data

In [23]:
```
# Creating a function to categorize injury severity into broader groups
def categorize_severity(severity):
    if severity == 'Fatal':
        return 'Fatal'
    elif severity in ['Non-Fatal', 'Incident', 'Minor', 'Serious']:
        return 'Non-Fatal'
    else:
        return 'Unavailable'

# Creating a new column to categorize injury severity into broader groups by a
pplying the function
avi_df_clean['severity_category'] = avi_df_clean['injury_severity'].apply(cate
gorize_severity)
```

In [24]:
```
# Checking the value counts of the new column
avi_df_clean['severity_category'].value_counts()
```

Out[24]:
```
Non-Fatal      66767
Fatal          17825
Unavailable      423
Name: severity_category, dtype: int64
```

In [25]:
```
# For showing the columns null values
avi_df_clean.isna().sum()
```

Out[25]:
```
event_id                    0
investigation_type          0
event_date                  0
location                   40
country                   206
injury_severity           327
aircraft_damage          1460
make                       36
model                      60
number_of_engines        4900
engine_type              5841
purpose_of_flight        4327
total_fatal_injuries    10756
total_serious_injuries  11863
total_minor_injuries    11309
total_uninjured          5768
weather_condition        3133
totalfatal_injuries         0
severity_category           0
dtype: int64
```

In [26]:
```
# Droping total_fatal_injuries col since I created a new column containing its
values
avi_df_clean.drop(columns=['total_fatal_injuries'], inplace=True)
```

In [27]:

```python
# For showing the columns null values
avi_df_clean.isna().sum()
```

Out[27]:

```
event_id                    0
investigation_type          0
event_date                  0
location                   40
country                   206
injury_severity           327
aircraft_damage          1460
make                       36
model                      60
number_of_engines        4900
engine_type              5841
purpose_of_flight        4327
total_serious_injuries  11863
total_minor_injuries    11309
total_uninjured          5768
weather_condition        3133
totalfatal_injuries         0
severity_category           0
dtype: int64
```

## Cleaning the make column

It seems it contains both uppercased and lowercased values

In [28]:

```python
# Checking the value counts in make column
avi_df_clean["make"].value_counts()
```

Out[28]:

```
Cessna                        21973
Piper                         11885
CESSNA                         4820
Beech                          4170
PIPER                          2799
                              ...
Ronald D. Murray                  1
Kemp                              1
Spezio                            1
ENGLISH ELECTRIC LIGHTNING        1
EASLER KELLY                      1
Name: make, Length: 8170, dtype: int64
```

In [29]:
```python
# Convert the 'make' column to lowercase to improve our analysis
avi_df_clean['make'] = avi_df_clean['make'].str.capitalize()

avi_df_clean['make'].value_counts()
```

Out[29]:
```
Cessna                             26793
Piper                              14684
Beech                               5177
Bell                                2662
Boeing                              1362
                                   ...  
Excel jet                              1
Tamarack helicopters inc               1
Western international avia inc         1
Qac                                    1
Bowlin                                 1
Name: make, Length: 7529, dtype: int64
```

In [30]:
```python
# Checking for categorical columns

avi_df_clean.dtypes
```

Out[30]:
```
event_id                   object
investigation_type         object
event_date                 object
location                   object
country                    object
injury_severity            object
aircraft_damage            object
make                       object
model                      object
number_of_engines         float64
engine_type                object
purpose_of_flight          object
total_serious_injuries    float64
total_minor_injuries      float64
total_uninjured           float64
weather_condition          object
totalfatal_injuries         int32
severity_category          object
dtype: object
```

## Filling null values in categorical columns with the mode

In [31]:
```
# Creating a list of the categorical columns
categorical_columns = [ 'injury_severity', 'aircraft_damage',
                        'make', 'model', 'engine_type', 'purpose_of_flight',
'weather_condition']


# Filling null values in each categorical column with the mode
for column in categorical_columns:
    mode_value = avi_df_clean[column].mode()[0]
    avi_df_clean[column].fillna(mode_value, inplace=True)
```

## Combining make and model into a single column for better analysis

In [32]:
```
# Creating a new column that contains both make and model
avi_df_clean['make_model'] = avi_df_clean['make'] + " " + avi_df_clean['mode
l']
```

## Dropping rows with null values in the remaining categorical columns

In [33]:
```
#Dropping rows with null values in the remaining categorical columns
avi_df_clean = avi_df_clean.dropna(subset=['location', 'country'])
```

In [34]:
```
# For showing the columns null values
avi_df_clean.isna().sum()
```

Out[34]:
```
event_id                      0
investigation_type            0
event_date                    0
location                      0
country                       0
injury_severity               0
aircraft_damage               0
make                          0
model                         0
number_of_engines          4876
engine_type                   0
purpose_of_flight             0
total_serious_injuries    11846
total_minor_injuries      11291
total_uninjured            5756
weather_condition             0
totalfatal_injuries           0
severity_category             0
make_model                    0
dtype: int64
```

## Handling missing values numerical columns

In [35]:
```
# Finding the mean of number of engines
no_of_engines_mean = avi_df_clean['number_of_engines'].mean()
no_of_engines_mean = round(no_of_engines_mean, 0)
no_of_engines_mean
```

Out[35]:
```
1.0
```

In [36]:
```
avi_df_clean['number_of_engines'].fillna(value = no_of_engines_mean, inplace=T
rue )
```

In [37]:
```
# Convert the 'number_of_engines' column to integers
avi_df_clean['number_of_engines'] = avi_df_clean['number_of_engines'].astype(i
nt)

# Verify the conversion
print(avi_df_clean['number_of_engines'].dtype)
```

```
int32
```

In [38]:
```
# For showing the numerical columns null values
avi_df_clean.isna().sum()
```

Out[38]:
```
event_id                     0
investigation_type           0
event_date                   0
location                     0
country                      0
injury_severity              0
aircraft_damage              0
make                         0
model                        0
number_of_engines            0
engine_type                  0
purpose_of_flight            0
total_serious_injuries   11846
total_minor_injuries     11291
total_uninjured           5756
weather_condition            0
totalfatal_injuries          0
severity_category            0
make_model                   0
dtype: int64
```

In [39]:
```
# Filling the remaining numerical columns with their various modes
avi_df_clean['total_serious_injuries'].fillna(avi_df_clean['total_serious_inju
ries'].median(), inplace=True)
avi_df_clean['total_minor_injuries'].fillna(avi_df_clean['total_minor_injurie
s'].median(), inplace=True)
avi_df_clean['total_uninjured'].fillna(avi_df_clean['total_uninjured'].median
(), inplace=True)
```

## Cleaning the weather condition column

It seems like the unknown value in written in both capital letters and small letters. Lets solve that.

In [40]:
```
#Checking the value counts of weather condition
avi_df_clean['weather_condition'].value_counts()
```

Out[40]:
```
VMC    78114
IMC     5712
UNK      729
Unk      215
Name: weather_condition, dtype: int64
```

In [41]:
```
# Combine 'UNK' and 'Unk' into a single category
avi_df_clean['weather_condition'] = avi_df_clean['weather_condition'].replace
({'Unk': 'UNK'})

# Rechecking the value counts of weather condition
avi_df_clean['weather_condition'].value_counts()
```

Out[41]:
```
VMC    78114
IMC     5712
UNK      944
Name: weather_condition, dtype: int64
```

## Now all the null values are dealt with

In [42]:
```
# Rechecking for any null values
avi_df_clean.isna().sum()
```

Out[42]:
```
event_id                  0
investigation_type        0
event_date                0
location                  0
country                   0
injury_severity           0
aircraft_damage           0
make                      0
model                     0
number_of_engines         0
engine_type               0
purpose_of_flight         0
total_serious_injuries    0
total_minor_injuries      0
total_uninjured           0
weather_condition         0
totalfatal_injuries       0
severity_category         0
make_model                0
dtype: int64
```

In [43]:
```
# Converting the 'event_date' data type column from object to date
avi_df_clean['event_date'] = pd.to_datetime(avi_df_clean['event_date'])
```

In [44]:
```
# Check the data type of the 'event_date' column
print(avi_df_clean['event_date'].dtypes)
```

```
datetime64[ns]
```

In [45]:
```
# Creating a new column that contains only the years
avi_df_clean['year'] = avi_df_clean['event_date'].dt.year
```

# Data Analysis

To address these questions I had posed at the onset of this project, various charts and graphs were used to uncover patterns and provide actionable insights:

1. **Line Charts**:
   - **Trends in Accidents Over Time**: Visualized the number of accidents from 1948 to 2022, revealing historical trends and changes in safety.
2. **Bar Charts**:
   - **Accident Severity**: Compared the total counts of fatal and non-fatal injuries over the years, offering a clear picture of accident outcomes.
   - **Accident Counts by Aircraft Make and Model**: Identified aircraft makes and models with the highest and lowest accident counts to guide risk assessments.
   - **Accident Patterns by Engine Type and Number of Engines**: Explored the relationship between engine configurations and accident frequency, shedding light on aircraft reliability.
   - **Accident Counts by Meteorological Conditions**: Highlighted differences in accident rates under Visual Meteorological Conditions (VMC) and Instrument Meteorological Conditions (IMC), informing strategies for safer operations.

---

## Tools and Methods

- **Data Processing**: Used **pandas** to clean, aggregate, and analyze the dataset effectively.
- **Visualization**: Employed **Matplotlib** and **Seaborn** to create clear and impactful visualizations tailored for non-technical stakeholders.

These analyses laid the groundwork for actionable recommendations, ensuring that the business's aviation expansion is based on robust, data-driven insights.

In [46]:
```
# Summary Statistics

avi_df_clean.describe()
```

Out[46]:

| | number_of_engines | total_serious_injuries | total_minor_injuries | total_uninjured | totalfatal_in |
|---|---|---|---|---|---|
| **count** | 84770.000000 | 84770.000000 | 84770.000000 | 84770.000000 | 84770.0 |
| **mean** | 1.109001 | 0.247422 | 0.308482 | 3.065165 | 0.4 |
| **std** | 0.384232 | 1.424762 | 1.645538 | 19.058674 | 4.4 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| **25%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| **50%** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.0 |
| **75%** | 1.000000 | 0.000000 | 0.000000 | 2.000000 | 0.0 |
| **max** | 8.000000 | 161.000000 | 200.000000 | 699.000000 | 349.0 |

In [47]:
```
# Checking the summary statistics for categorical columns
avi_df_clean.describe(include='object')
```

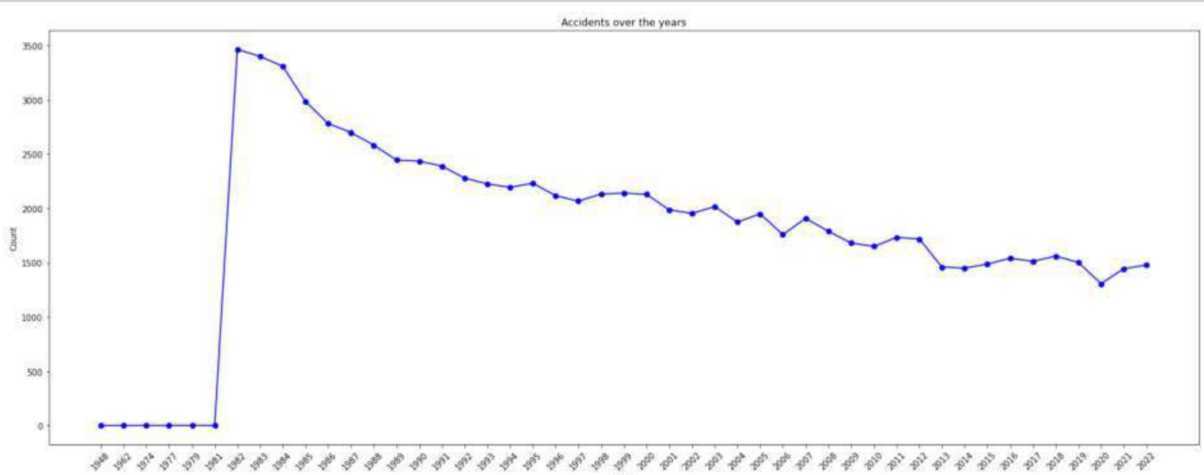
Out[47]:

| | event_id | investigation_type | location | country | injury_severity | aircraft_dama |
|---|---|---|---|---|---|---|
| **count** | 84770 | 84770 | 84770 | 84770 | 84770 | 847 |
| **unique** | 83947 | 1 | 26767 | 206 | 5 | |
| **top** | 20001212X19172 | Accident | ANCHORAGE, AK | United States | Non-Fatal | Substan |
| **freq** | 3 | 84770 | 405 | 79895 | 66545 | 654 |

# Trend Analysis

The below line chart shows the count of accidents over the years, with the most count of accidents being in 1982.

In [48]:
```
# Extracting the value counts of the years which represent the count of accide
nt in a specific year.
year_trend= avi_df_clean["year"].value_counts().sort_index()
x_= year_trend.index.astype(str)
y_ = year_trend.values
# Customizing the plot
fig,ax = plt.subplots(figsize= (20,8))
ax.plot(x_, y_ ,marker="o", color='blue')
ax.set_xlabel('Years')
plt.xticks(x_, rotation=45)
ax.set_ylabel('Count')
ax.set_title('Accidents over the years')
plt.tight_layout()
plt.show()
```



## Bar plot showing the count of total fatal and non-fatal injuries through out the years

In [49]:
```
# Grouping by year and severity category
severity_counts = avi_df_clean.groupby(['year', 'severity_category']).size().reset_index(name='count')
severity_counts.sort_values(by='count', ascending=False)
```

Out[49]:

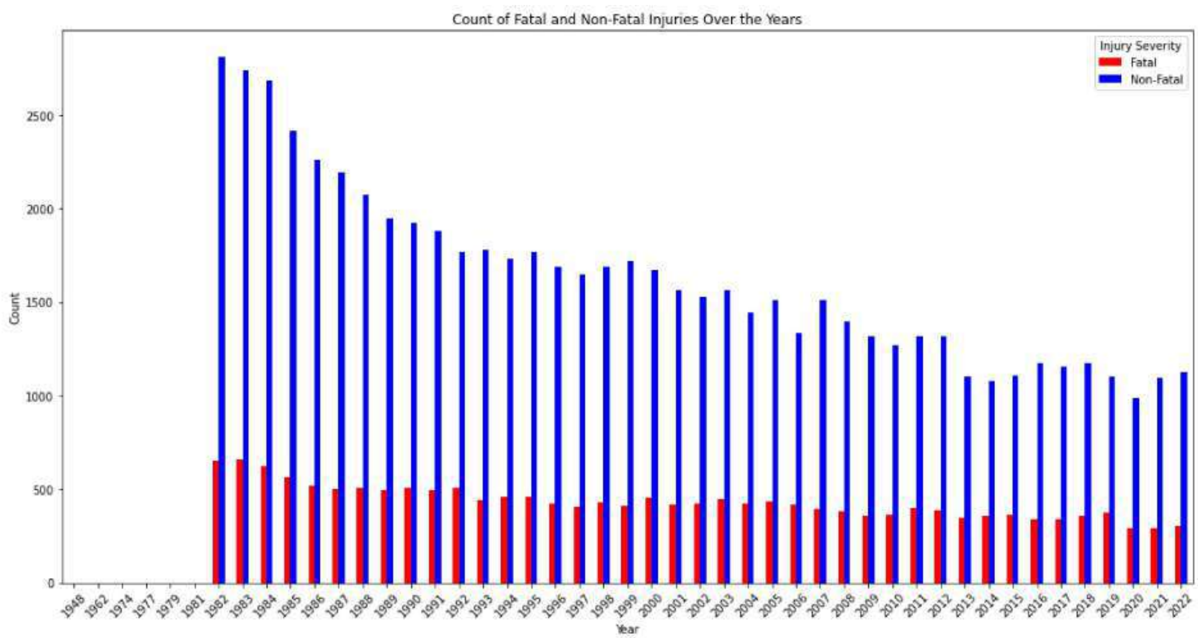
| | year | severity_category | count |
|---|---|---|---|
| **8** | 1982 | Non-Fatal | 2813 |
| **10** | 1983 | Non-Fatal | 2742 |
| **12** | 1984 | Non-Fatal | 2685 |
| **14** | 1985 | Non-Fatal | 2419 |
| **17** | 1986 | Non-Fatal | 2258 |
| **...** | ... | ... | ... |
| **5** | 1979 | Non-Fatal | 1 |
| **4** | 1979 | Fatal | 1 |
| **3** | 1977 | Fatal | 1 |
| **2** | 1974 | Fatal | 1 |
| **0** | 1948 | Fatal | 1 |

124 rows × 3 columns

In [50]:
```
# Pivoting to easen plotting and for filling null values in the pivot table
severity_pivot = severity_counts.pivot(index='year', columns='severity_category', values='count').fillna(0)
```

In [51]:
```
#Plotting and customizing the plot

plt.figure(figsize=(15, 8))
severity_pivot[['Fatal', 'Non-Fatal']].plot(kind='bar', figsize=(15, 8), color
=['red', 'blue'])
plt.title('Count of Fatal and Non-Fatal Injuries Over the Years')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Injury Severity')
plt.tight_layout()
plt.show();
```

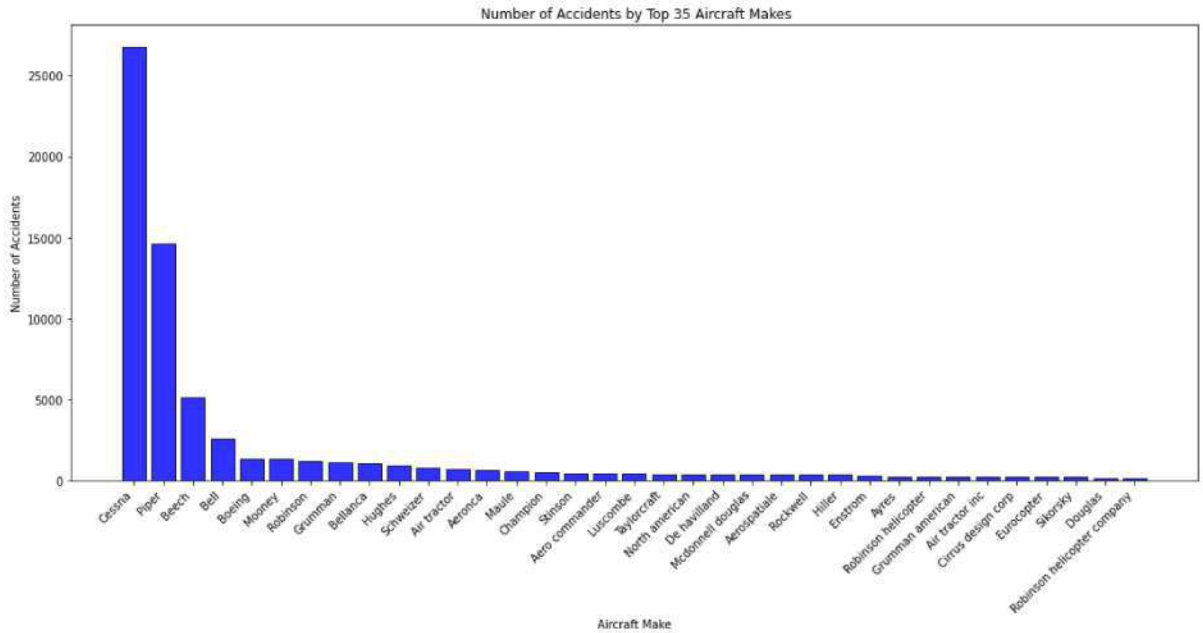
```
<Figure size 1080x576 with 0 Axes>
```




# Bar plot showing the count of accidents by top 35 aircaft make

In [52]:
```
# Group by aircraft make and count the number of accidents
make_counts = avi_df_clean['make'].value_counts().head(35)

# Extracting labels
x_ = make_counts.index
y_ = make_counts.values

# Plotting the bar chart
fig, ax = plt.subplots(figsize=(15, 8))
ax.bar(x_, y_, color='blue', edgecolor='black', alpha=0.8)

# Customizing the plot
```

```
ax.set_xlabel('Aircraft Make')
ax.set_ylabel('Number of Accidents')
ax.set_title(' Number of Accidents by Top 35 Aircraft Makes')
plt.xticks(rotation=45, fontsize=10, ha='right')
plt.tight_layout()
plt.show();
```




## Insights

- **Cessna and Piper dominate** the accident dataset. This is likely a result of their popularity and widespread usage in general aviation and training, which increases exposure to accidents.
- The presence of manufacturers like **Boeing** and **Bell** suggests the dataset includes a mix of commercial airliners and specialized aircraft like helicopters.
- The smaller accident numbers for less common manufacturers, such as **Taylorcraft** and **Luscombe**, could be due to their limited production and operation scale.

Overall, the distribution of accident counts aligns with the operational scale and popularity of these aircraft, with more commonly used makes experiencing higher accident counts.

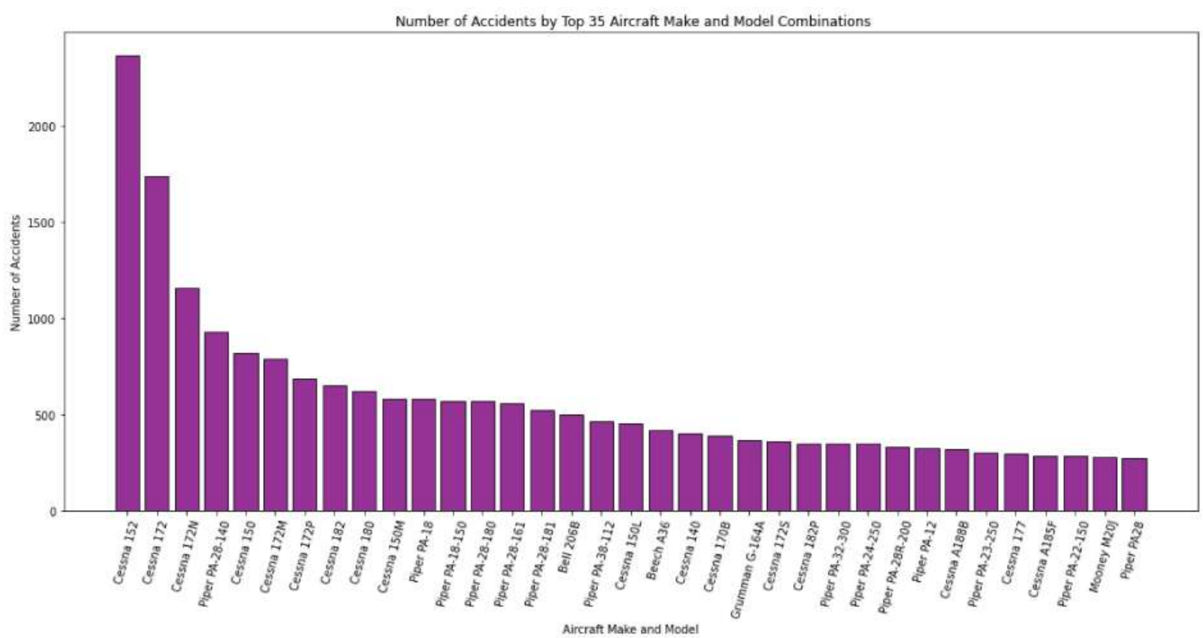
In [53]:
```
# Showcasing the accident counts values of the top 35 aircraft makes
make_counts
```

Out[53]:
```
Cessna                          26780
Piper                           14654
Beech                            5165
Bell                             2592
Boeing                           1355
Mooney                           1308
Robinson                         1213
Grumman                          1156
Bellanca                         1039
Hughes                            920
Schweizer                         769
Air tractor                       689
Aeronca                           632
Maule                             587
Champion                          517
Stinson                           436
Aero commander                    420
Luscombe                          414
Taylorcraft                       383
North american                    377
De havilland                      375
Mcdonnell douglas                 359
Aerospatiale                      348
Rockwell                          348
Hiller                            347
Enstrom                           289
Ayres                             236
Robinson helicopter               229
Grumman american                  224
Air tractor inc                   217
Cirrus design corp                212
Eurocopter                        212
Sikorsky                          205
Douglas                           192
Robinson helicopter company       190
Name: make, dtype: int64
```

## Bar chart showing the top 35 aircraft make and model combinations with the highest number of accidents

In [54]:

```
# Extracting the make_model and count the number of accidents
make_model_counts = avi_df_clean['make_model'].value_counts().head(35)

# Extracting labels
x_ = make_model_counts.index
y_ = make_model_counts.values

# Plotting the bar chart
fig, ax = plt.subplots(figsize=(15, 8))
ax.bar(x_, y_, color='purple', edgecolor='black', alpha=0.8)

# Customizing the plot
ax.set_xlabel('Aircraft Make and Model')
ax.set_ylabel('Number of Accidents')
ax.set_title(' Number of Accidents by Top 35 Aircraft Make and Model Combinati
ons')
plt.xticks(rotation=75)  # Rotating x-axis ticks for better readability

plt.tight_layout()
plt.show();
```



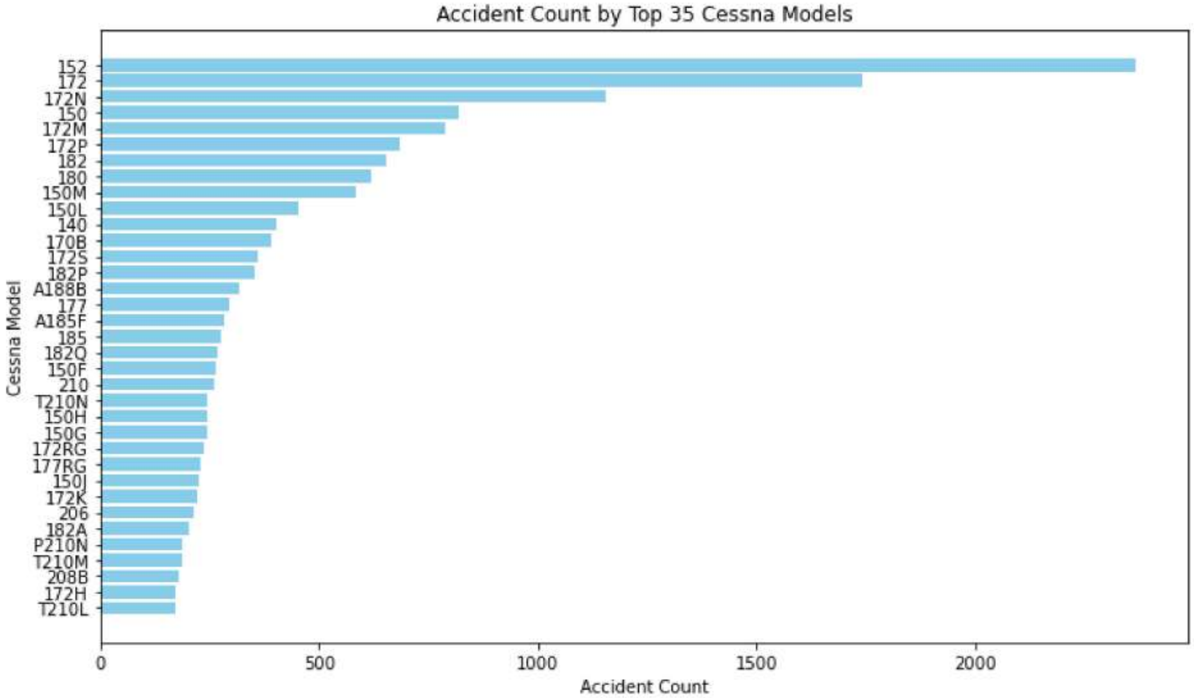
In [55]:
```
# Showing the values
make_model_counts
```

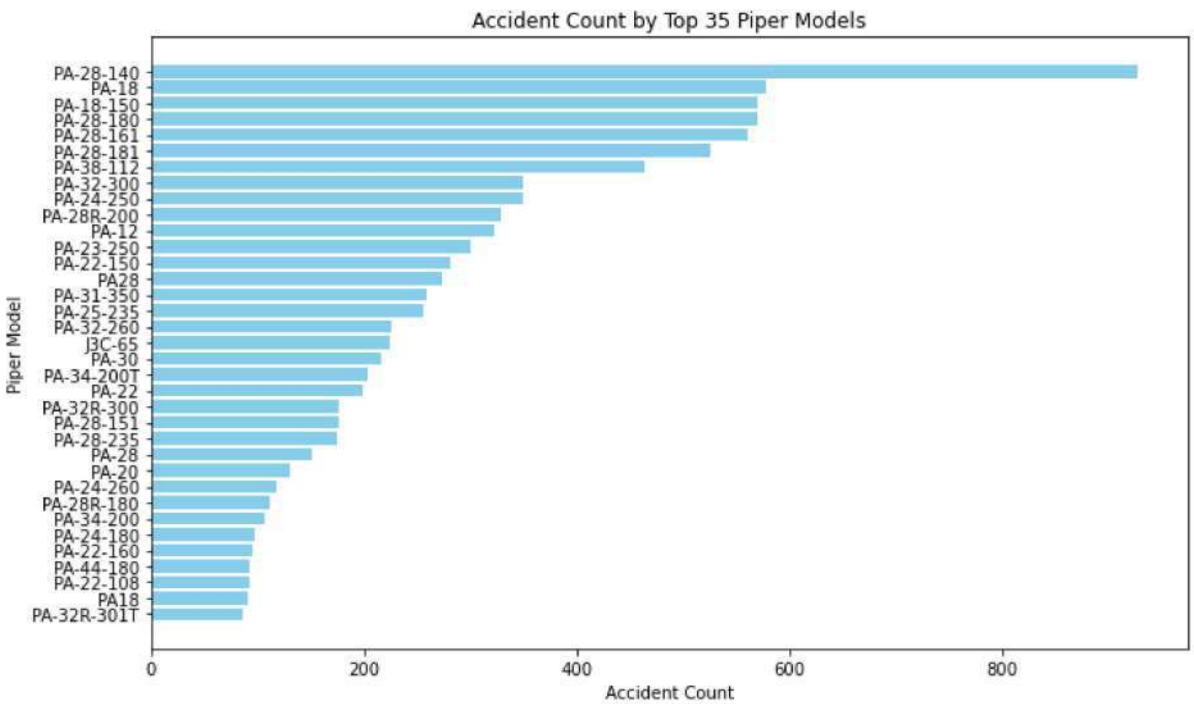
Out[55]:
```
Cessna 152          2368
Cessna 172          1741
Cessna 172N         1157
Piper PA-28-140      928
Cessna 150           821
Cessna 172M          790
Cessna 172P          684
Cessna 182           653
Cessna 180           620
Cessna 150M          582
Piper PA-18          578
Piper PA-18-150      571
Piper PA-28-180      571
Piper PA-28-161      560
Piper PA-28-181      525
Bell 206B            498
Piper PA-38-112      464
Cessna 150L          454
Beech A36            415
Cessna 140           401
Cessna 170B          389
Grumman G-164A       365
Cessna 172S          361
Cessna 182P          350
Piper PA-32-300      350
Piper PA-24-250      349
Piper PA-28R-200     329
Piper PA-12          323
Cessna A188B         316
Piper PA-23-250      300
Cessna 177           294
Cessna A185F         283
Piper PA-22-150      281
Mooney M20J          280
Piper PA28           274
Name: make_model, dtype: int64
```

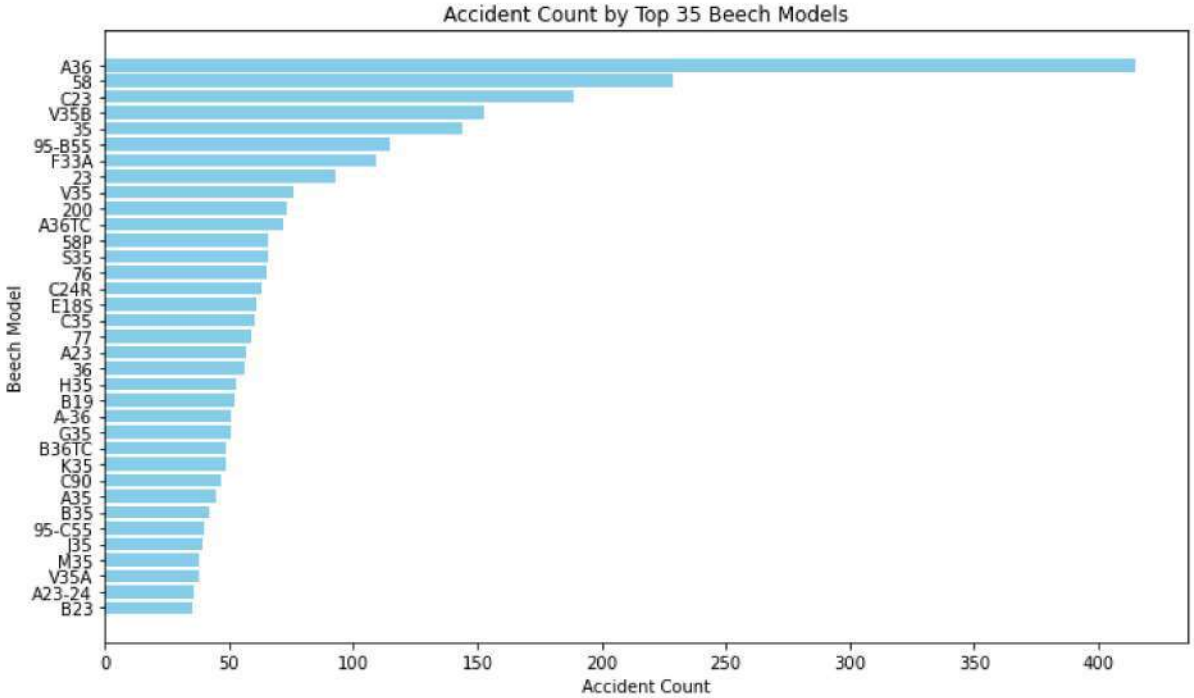
# Horizontal Bar graphs showing the accident count of 10 Common aircraft make grouped by model

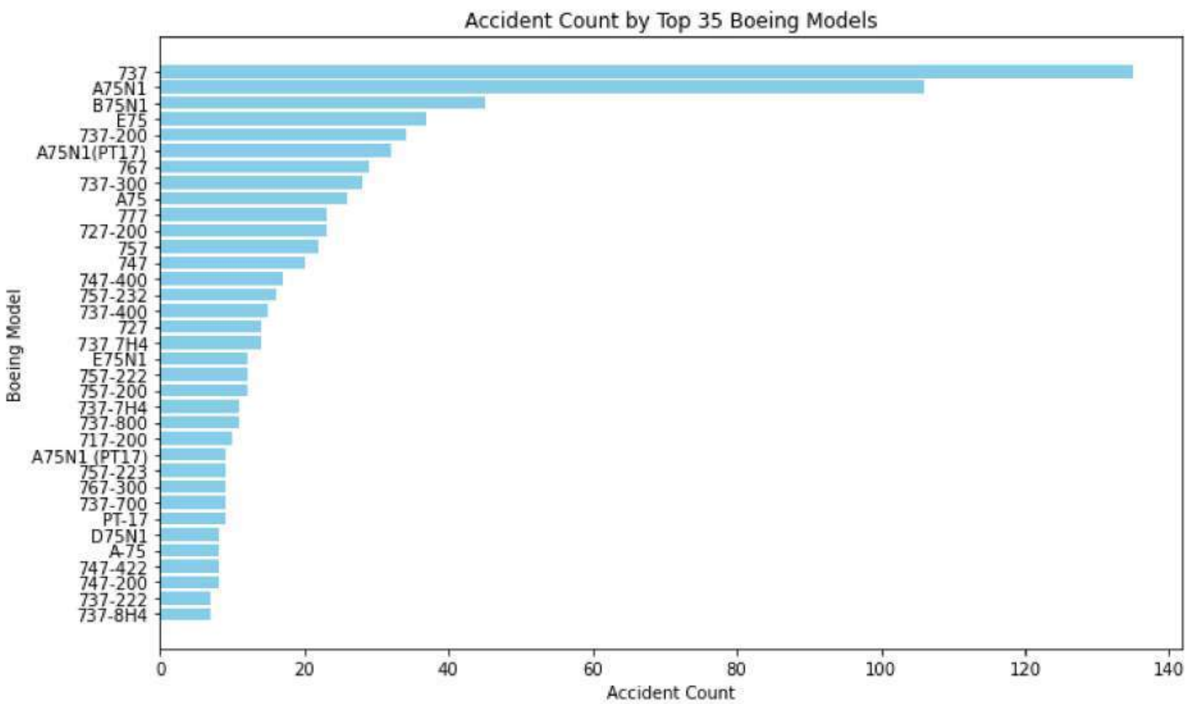
In [56]:

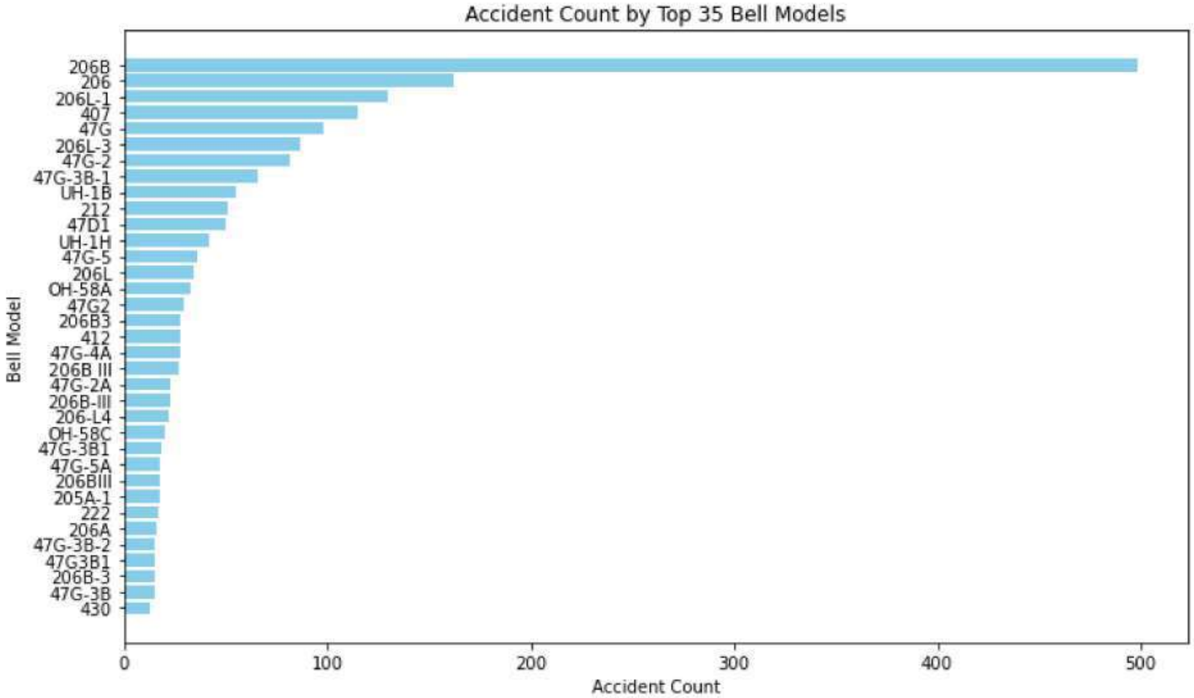
```
def plot_top_models_by_make(dataframe, make_name, top_n=35):
    """
    Plots a horizontal bar chart of the top models by accident count for a giv
en make.

    Parameters:
        dataframe (pd.DataFrame): The DataFrame containing the aviation data.
        make_name (str): The name of the make to filter (e.g., 'Cessna').
        top_n (int): The number of top models to display in the chart.
    """
    # Filter rows for the specified make and group by model
    model_counts = (dataframe[dataframe['make'] == make_name]
                    .groupby(['model'])
                    .size()
                    .reset_index(name='accident_count'))

    # Sort by accident count in descending order and get the top 35 models
    top_models = model_counts.sort_values(by='accident_count', ascending=Fals
e).head(top_n)

    # Extract data for plotting
    models = top_models['model']
    accident_counts = top_models['accident_count']

    # Create the bar chart
    plt.figure(figsize=(10, 6))
    plt.barh(models, accident_counts, color='skyblue')

    # Add titles and labels
    plt.title(f' Accident Count by Top {top_n} {make_name} Models ')
    plt.xlabel('Accident Count')
    plt.ylabel(f'{make_name} Model')

    # Invert y-axis to show the model with the highest count at the top
    plt.gca().invert_yaxis()

    # Display the chart
    plt.tight_layout()
    plt.show()

# Popular makes
makes = ['Cessna', 'Piper', 'Beech', 'Boeing', 'Bell', 'Gulfstream', 'Airbus']

for make in makes:
    plot_top_models_by_make(avi_df_clean, make_name=make, top_n=35)
```

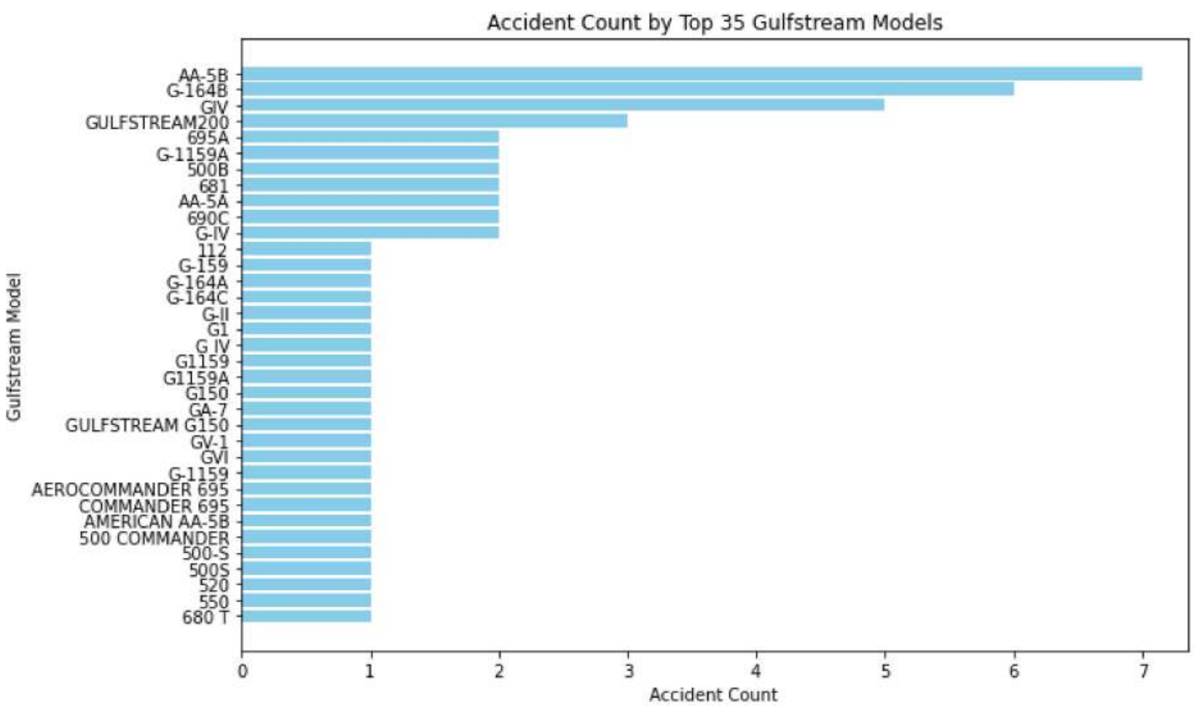

Accident Count by Top 35 Cessna Models
152
172
172N
150
172M
172P
182
180
150M
150L
140
170B
172S
182P
A188B
177
A185F
185
182Q
150F
210
T210N
150H
150G
172RG
177RG
150J
172K
206
182A
P210N
T210M
208B
172H
T210L
Cessna Model
0
500
1000
1500
2000
Accident Count

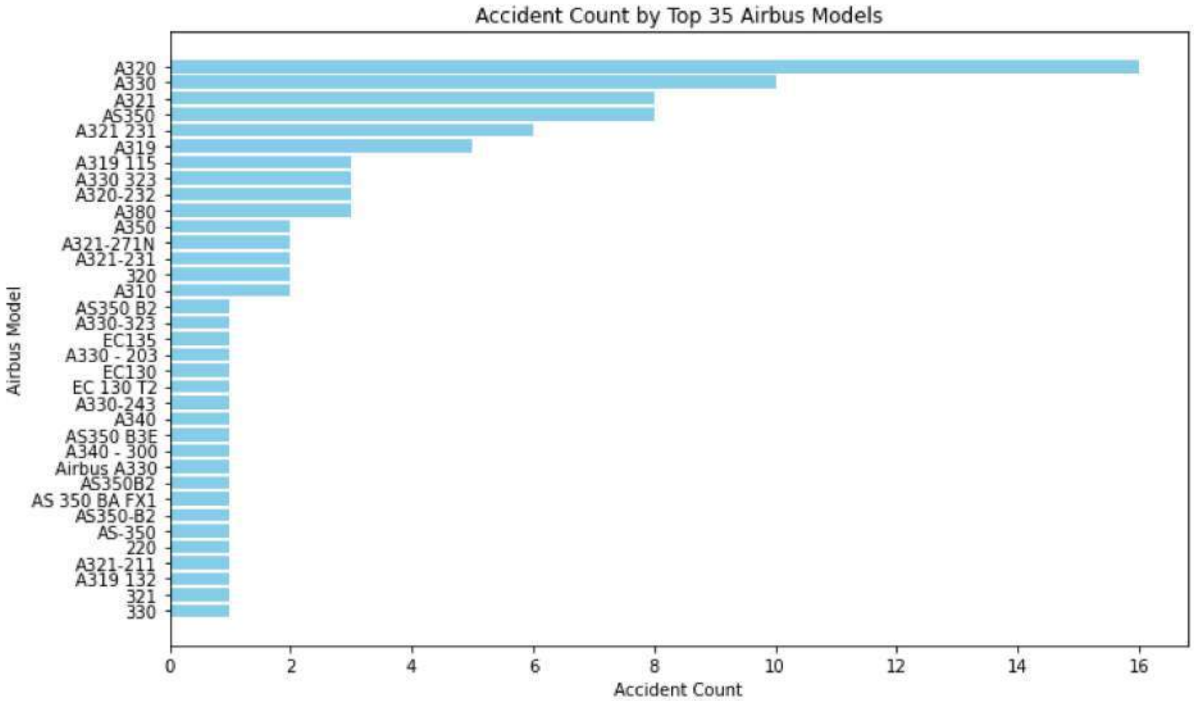

Accident Count by Top 35 Piper Models
PA-28-140
PA-18
PA-18-150
PA-28-180
PA-28-161
PA-28-181
PA-38-112
PA-32-300
PA-24-250
PA-28R-200
PA-12
PA-23-250
PA-22-150
PA28
PA-31-350
PA-25-235
PA-32-260
J3C-65
PA-30
PA-34-200T
PA-22
PA-32R-300
PA-28-151
PA-28-235
PA-28
PA-20
PA-24-260
PA-28R-180
PA-34-200
PA-24-180
PA-22-160
PA-44-180
PA-22-108
PA18
PA-32R-301T
Piper Model
0
200
400
600
800
Accident Count


Accident Count by Top 35 Beech Models
Beech Model
Accident Count
A36
58
C23
V35B
35
95-B55
F33A
23
V35
200
A36TC
58P
S35
76
C24R
E18S
C35
77
A23
36
H35
B19
A-36
G35
B36TC
K35
C90
A35
B35
95-C55
J35
M35
V35A
A23-24
B23
0
50
100
150
200
250
300
350
400


Accident Count by Top 35 Boeing Models
Boeing Model
Accident Count
737
A75N1
B75N1
E75
737-200
A75N1(PT17)
767
737-300
A75
777
727-200
757
747
747-400
757-232
737-400
727
737 7H4
E75N1
757-222
757-200
737-7H4
737-800
717-200
A75N1 (PT17)
757-223
767-300
737-700
PT-17
D75N1
A-75
747-422
747-200
737-222
737-8H4
0
20
40
60
80
100
120
140


Accident Count by Top 35 Bell Models
Bell Model
206B
206
206L-1
407
47G
206L-3
47G-2
47G-3B-1
UH-1B
212
47D1
UH-1H
47G-5
206L
OH-58A
47G2
206B3
412
47G-4A
206B III
47G-2A
206B-III
206-L4
OH-58C
47G-3B1
47G-5A
206BIII
205A-1
222
206A
47G-3B-2
47G3B1
206B-3
47G-3B
430
0
100
200
300
400
500
Accident Count


Accident Count by Top 35 Gulfstream Models
Gulfstream Model
AA-5B
G-164B
GIV
GULFSTREAM200
695A
G-1159A
500B
681
AA-5A
690C
G-IV
112
G-159
G-164A
G-164C
G-II
G1
G IV
G1159
G1159A
G150
GA-7
GULFSTREAM G150
GV-1
GVI
G-1159
AEROCOMMANDER 695
COMMANDER 695
AMERICAN AA-5B
500 COMMANDER
500-S
500S
520
550
680 T
0
1
2
3
4
5
6
7
Accident Count



Accident Count by Top 35 Airbus Models
Airbus Model
A320
A330
A321
AS350
A321 231
A319
A319 115
A330 323
A320-232
A380
A350
A321-271N
A321-231
320
A310
AS350 B2
A330-323
EC135
A330 - 203
EC130
EC 130 T2
A330-243
A340
AS350 B3E
A340 - 300
Airbus A330
AS350B2
AS 350 BA FX1
AS350-B2
AS-350
220
A321-211
A319 132
321
330
0
2
4
6
8
10
12
14
16
Accident Count

## Recommendations for Popular Aircraft Models Based on Accident Counts

### Models to Avoid (Higher Accident Counts)

To reduce risk, the following aircraft models with higher accident counts should be avoided:

- **Cessna Models**:
  - 152, 172, 172N, 150, 172M, 172P, 182, 180, 150M
- **Piper Models**:
  - PA-28-140, PA-18, PA-18-150, PA-28-180, PA-28-161, PA-28-181, PA-38-112
- **Beech Models**:
  - A36, 58, C23, V35B, 35, 95-B55, F33A, 23
- **Boeing Models**:
  - 737, A75N1, B75N1, E75, 737-200, A75N1(PT17)
- **Bell Models**:
  - 206B, 206, 206L-1, 407, 47G, 206L-3

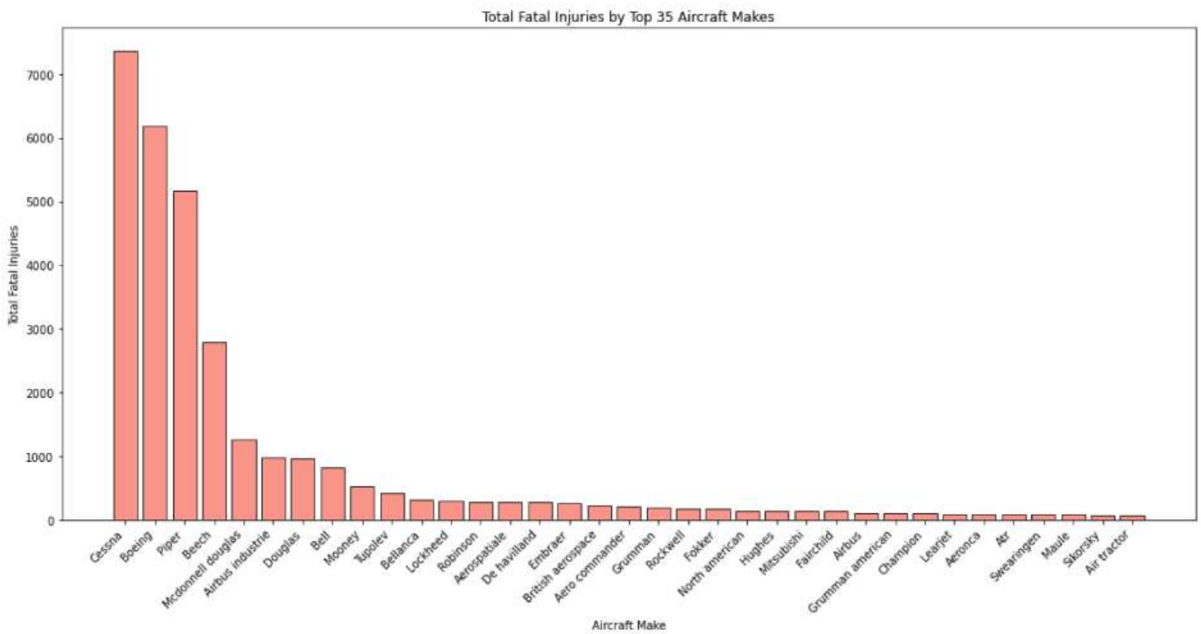
### Models to Consider (Lower Accident Counts)

Aircraft models with lower accident counts are recommended for safer operations.

- **Cessna Models**:
  - T210L, 208B, P210N, 182A
- **Piper Models**:
  - PA-22-108, PA-32R-301T
- **Beech Models**:
  - A23-24 , 95-C55
- **Boeing Models**:
  - 747-422
- **Bell Models**:
  - 430, 222
- **Gulfstream Models**:
  - 550, 680T, 500S
- **Airbus Models**:
  - 330, 321, A319-132, A321-211, AS-350

# Bar chart showing Total Fatal Injuries by top 35 Aircraft Makes

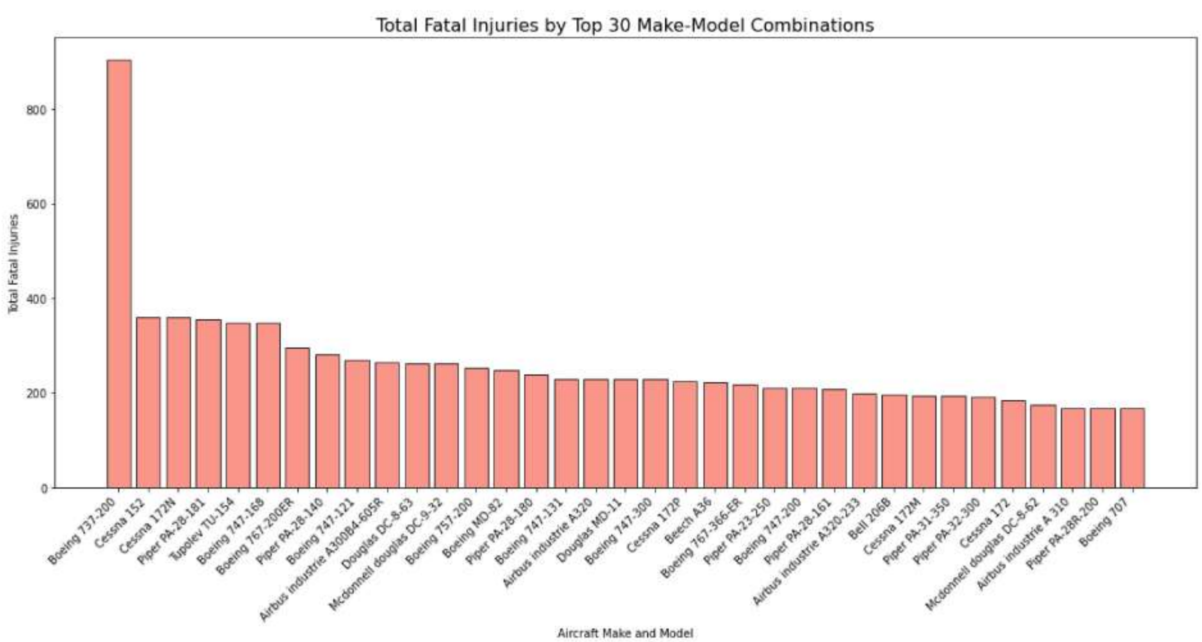
In [57]:
```
# Group by 'make' and sum the total fatal injuries
make_fatal_injuries = (avi_df_clean.groupby('make')['totalfatal_injuries']
                       .sum()
                       .sort_values(ascending=False)
                       .reset_index())

# Extract the top 20 makes with the highest total fatal injuries
top_35_makes_fatal = make_fatal_injuries.head(35)

# Plotting the bar chart
plt.figure(figsize=(15, 8))
plt.bar(top_35_makes_fatal['make'], top_35_makes_fatal['totalfatal_injuries'],
        color='salmon', edgecolor='black', alpha=0.8)

# Customizing the plot
plt.title('Total Fatal Injuries by Top 35 Aircraft Makes')
plt.xlabel('Aircraft Make')
plt.ylabel('Total Fatal Injuries')
plt.xticks(rotation=45,fontsize=10, ha='right')
plt.tight_layout()

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



# Bar chart showing Total Fatal Injuries by Top 35 Make-Model Combinations

In [58]:

```
# Group by 'make_model' and sum the total fatal injuries
make_model_fatal_injuries = (avi_df_clean.groupby('make_model')['totalfatal_in
juries']
                             .sum()
                             .sort_values(ascending=False)
                             .reset_index())

# Extract the top 35 make-model combinations with the highest total fatal inju
ries
top_35_make_model_fatal = make_model_fatal_injuries.head(35)

# Plotting the bar chart
plt.figure(figsize=(15, 8))
plt.bar(top_35_make_model_fatal['make_model'], top_35_make_model_fatal['totalf
atal_injuries'],
        color='salmon', edgecolor='black', alpha=0.8)

# Customizing the plot
plt.title('Total Fatal Injuries by Top 30 Make-Model Combinations', fontsize=1
6)
plt.xlabel('Aircraft Make and Model')
plt.ylabel('Total Fatal Injuries')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

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



## Aircraft Models to Avoid by Make since the have a high fatality count

#### Boeing

- 737-200, 747-168, 747-121, 747-131, 747-200, 747-300, 757-200, 767-200ER, 767-366-ER, 707

#### Cessna

- 152, 172, 172N, 172M, 172P

#### Piper

- PA-28-140, PA-28-161, PA-28-180, PA-28-181, PA-28R-200, PA-31-350, PA-32-300, PA-23-250

#### Airbus Industrie

- A30084-605R, A320, A320-233,A310

#### Tupolev

- TU-154

#### Douglas

- DC-8-63, DC-8-62, MD-11

#### McDonnell Douglas

- DC-9-32, MD-82

#### Beech

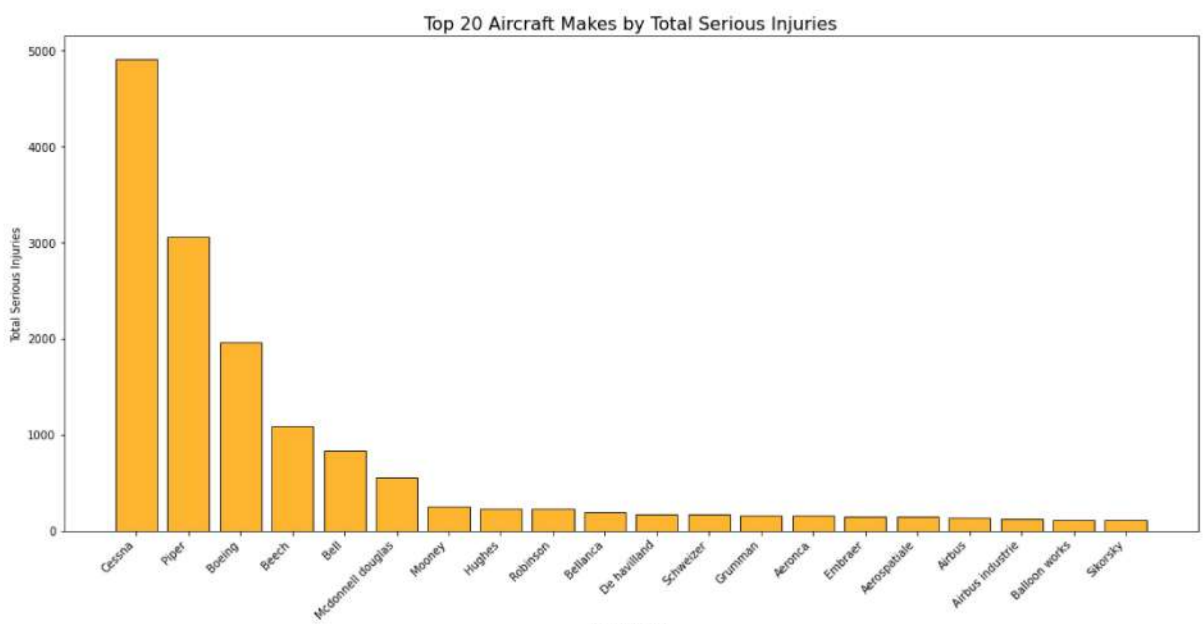
- A36

#### Bell

- 206B

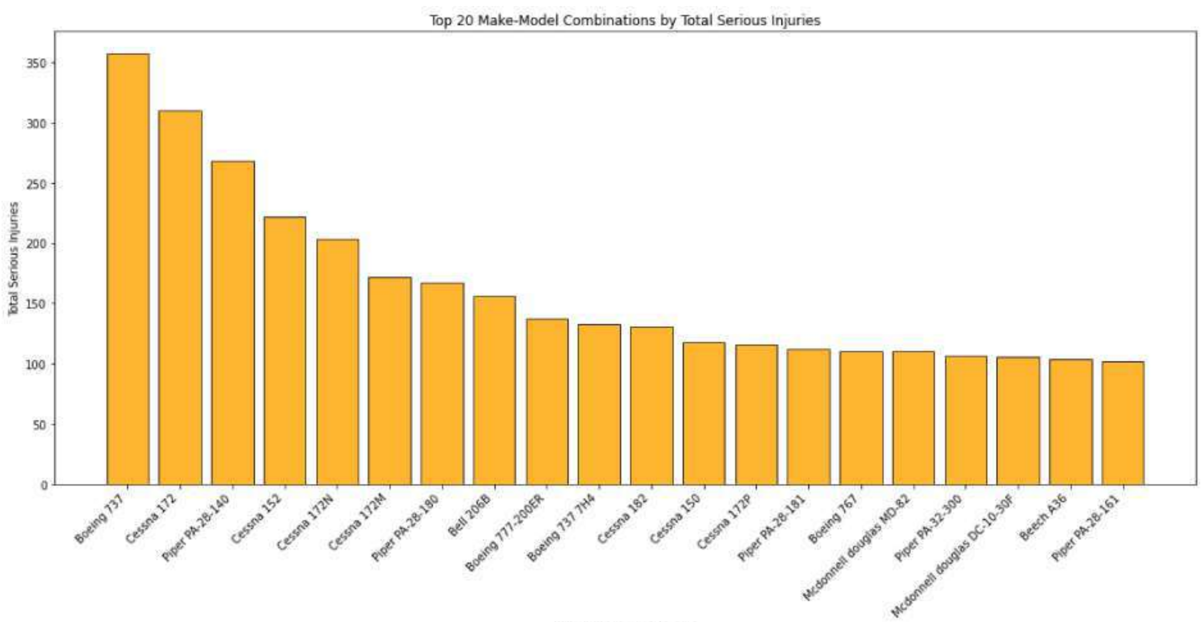
## Top 20 total serious injuries by make

In [59]:
```
# Group by 'make' and sum the total serious injuries
make_serious_injuries = (avi_df_clean.groupby('make')['total_serious_injurie
s']
                         .sum()
                         .sort_values(ascending=False)
                         .reset_index())

# Extract the top 20 makes with the highest total serious injuries
top_20_makes_serious = make_serious_injuries.head(20)

# Plotting the bar chart
plt.figure(figsize=(15, 8))
plt.bar(top_20_makes_serious['make'], top_20_makes_serious['total_serious_inju
ries'],
        color='orange', edgecolor='black', alpha=0.8)

# Customizing the plot
plt.title('Top 20 Aircraft Makes by Total Serious Injuries', fontsize=16)
plt.xlabel('Aircraft Make')
plt.ylabel('Total Serious Injuries')
plt.xticks(rotation=45, fontsize=10, ha='right')
plt.tight_layout()

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



## Top 20 total serious injuries by make and model

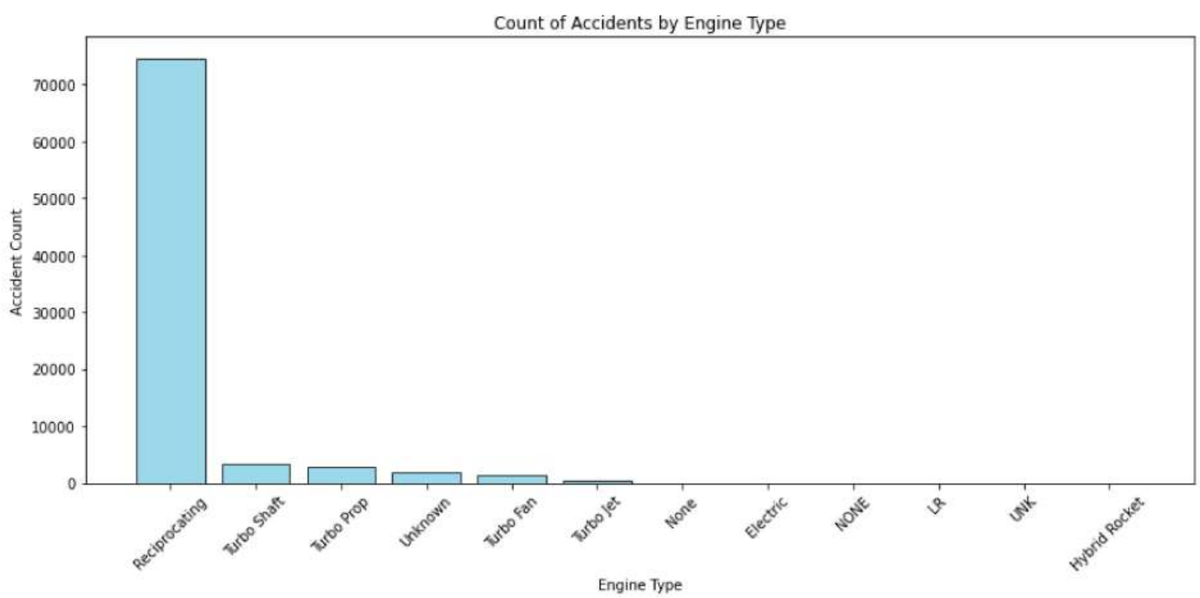
In [60]:
```
# Group by 'make_model' and sum the total serious injuries
make_model_serious_injuries = (avi_df_clean.groupby('make_model')['total_serious_injuries']
                               .sum()
                               .sort_values(ascending=False)
                               .reset_index())

# Extract the top 20 make-model combinations with the highest total serious injuries
top_20_make_model_serious = make_model_serious_injuries.head(20)

# Plotting the bar chart
plt.figure(figsize=(15, 8))
plt.bar(top_20_make_model_serious['make_model'], top_20_make_model_serious['total_serious_injuries'],
        color='orange', edgecolor='black', alpha=0.8)

# Customizing the plot
plt.title('Top 20 Make-Model Combinations by Total Serious Injuries')
plt.xlabel('Aircraft Make and Model')
plt.ylabel('Total Serious Injuries')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

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




**The above aircraft makes and models should be avoided at all cost, they still appear under aircraft Models and makes to Avoid by Total Fatal injuries.**

## Bar chart showing Count of accidents by engine type

In [61]:
```
# Group by 'engine_type' and count the number of accidents
engine_type_counts = avi_df_clean['engine_type'].value_counts()

# Sorting the values in descending order
engine_type_counts = engine_type_counts.sort_values(ascending=False)

# Plotting the bar chart
plt.figure(figsize=(12, 6))
plt.bar(engine_type_counts.index, engine_type_counts.values, color='skyblue',
edgecolor='black', alpha=0.8)

# Adding titles and labels
plt.title('Count of Accidents by Engine Type')
plt.xlabel('Engine Type')
plt.ylabel('Accident Count')
plt.xticks(rotation=45)

# Adjust layout for better visibility
plt.tight_layout()

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



## Insights from the Bar Chart: Engine Types and Accident Counts

**Key Finding:**
Aircraft equipped with **Reciprocating engines** have the **highest count of accidents** compared to other engine types.

---

## Focus on Major Engine Types in Commercial Aviation

The chart highlights accident counts across the **four major engine types** commonly found in commercial airplanes:

1. **Reciprocating Engines**
    - Highest accident count, signaling the need for deeper analysis and improved safety measures.
2. **Turbo Shaft Engines**
    - Typically found in helicopters; exhibits a moderate level of accident occurrences.
3. **Turbo Prop Engines**
    - Used in smaller regional aircraft; shows lower accident counts compared to reciprocating engines.
4. **Turbo Fan Engines**
    - Found in large commercial jets; demonstrates a relatively low accident rate, reflecting advanced safety standards.

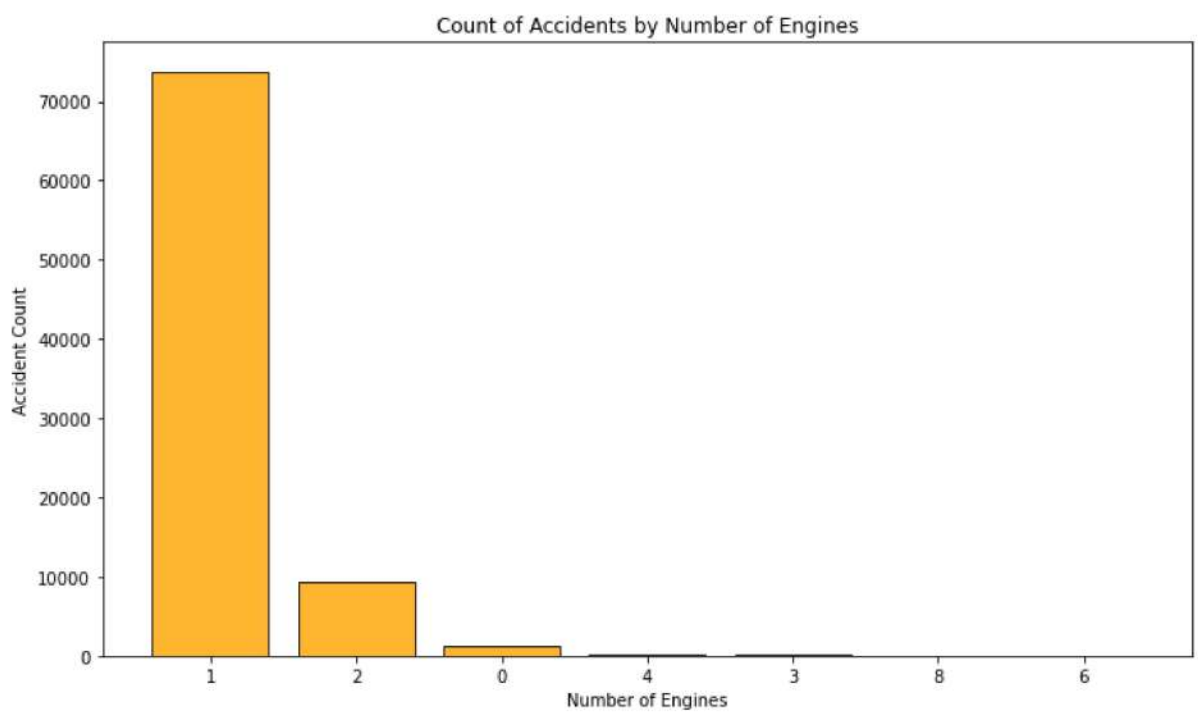
---

## Takeaway:

This data emphasizes the need for targeted safety measures and maintenance practices, particularly for **reciprocating engine aircraft**, to reduce accident rates.

## Bar chart showing the count of accidents grouped by number of engines:

In [62]:
```
import matplotlib.pyplot as plt

# Group by 'number_of_engines' and count the number of accidents
engine_count_accidents = avi_df_clean['number_of_engines'].value_counts()

# Sorting the values in descending order
engine_count_accidents = engine_count_accidents.sort_values(ascending=False)

# Plotting the bar chart
plt.figure(figsize=(10, 6))
plt.bar(engine_count_accidents.index.astype(str), engine_count_accidents.value
s, color='orange', edgecolor='black', alpha=0.8)

# Adding titles and labels
plt.title('Count of Accidents by Number of Engines')
plt.xlabel('Number of Engines')
plt.ylabel('Accident Count')
plt.xticks(rotation=0)

# Adjust layout for better visibility
plt.tight_layout()

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

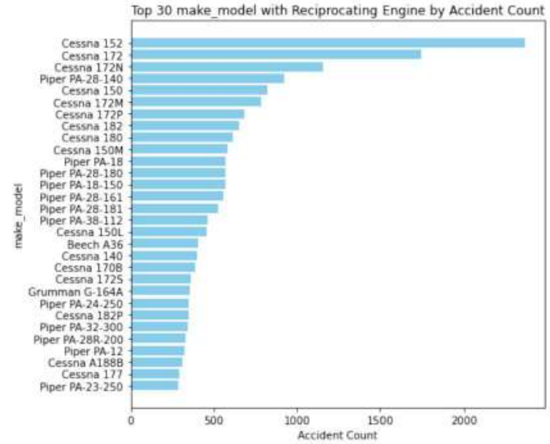


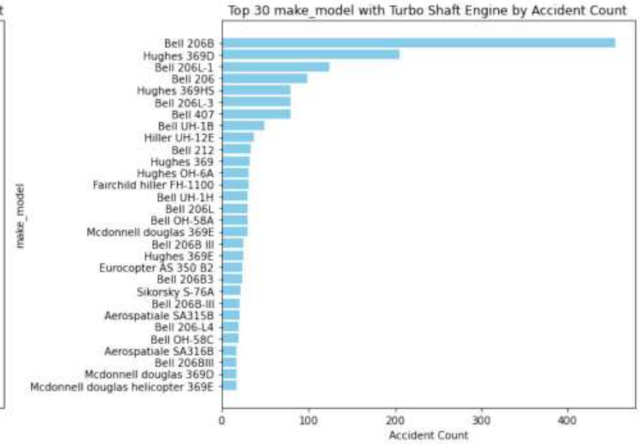

**The above bar chart shows that most aircrafts with only one engine have the highest number of accidents.**

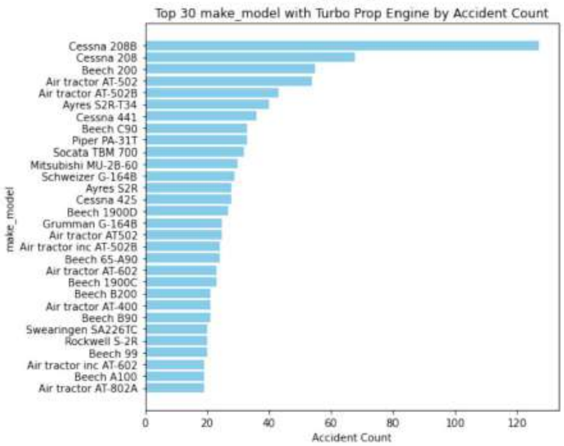
# Bar charts showing the accident counts by make & model engine type

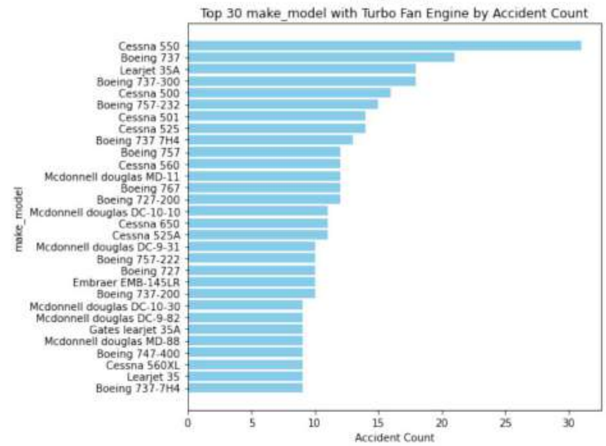
In [63]:

```
# List of engine types I am interested in
engine_types = ['Reciprocating', 'Turbo Shaft', 'Turbo Prop', 'Turbo Fan']

# Create a figure with multiple subplots (one for each engine type)
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Loop through each engine type and plot the top 30 make_model combinations
for i, engine_type in enumerate(engine_types):
    # Filter rows based on the engine type
    engine_df = avi_df_clean[avi_df_clean['engine_type'] == engine_type]

    # Group by make_model and count the number of accidents
    make_model_engine_count = engine_df.groupby('make_model').size().reset_ind
ex(name='accident_count')

    # Sort by accident count in descending order and get the top 20 make_model
combinations
    top_30_engine_make_model = make_model_engine_count.sort_values(by='acciden
t_count', ascending=False).head(30)

    # Plotting the bar chart for the current engine type
    axes[i].barh(top_30_engine_make_model['make_model'], top_30_engine_make_mo
del['accident_count'], color='skyblue')
    axes[i].set_title(f'Top 30 make_model with {engine_type} Engine by Acciden
t Count')
    axes[i].set_xlabel('Accident Count')
    axes[i].set_ylabel('make_model')
    axes[i].invert_yaxis()  # Invert y-axis to show the make_model with the hi
ghest count at the top

# Adjust layout
plt.tight_layout()

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


Top 30 make_model with Reciprocating Engine by Accident Count
Cessna 152
Cessna 172
Cessna 172N
Piper PA-28-140
Cessna 150
Cessna 172M
Cessna 172P
Cessna 182
Cessna 180
Cessna 150M
Piper PA-18
Piper PA-28-180
Piper PA-18-150
Piper PA-28-161
Piper PA-28-181
Piper PA-38-112
Cessna 150L
Beech A36
Cessna 140
Cessna 170B
Cessna 172S
Grumman G-164A
Piper PA-24-250
Cessna 182P
Piper PA-32-300
Piper PA-28R-200
Piper PA-12
Cessna A188B
Cessna 177
Piper PA-23-250
make_model
0
500
1000
1500
2000
Accident Count


Top 30 make_model with Turbo Shaft Engine by Accident Count
Bell 206B
Hughes 369D
Bell 206L-1
Bell 206
Hughes 369HS
Bell 206L-3
Bell 407
Bell UH-1B
Hiller UH-12E
Bell 212
Hughes 369
Hughes OH-6A
Fairchild hiller FH-1100
Bell UH-1H
Bell 206L
Bell OH-58A
Mcdonnell douglas 369E
Bell 206B III
Hughes 369E
Eurocopter AS 350 B2
Bell 206B3
Sikorsky S-76A
Bell 206B-III
Aerospatiale SA315B
Bell 206-L4
Bell OH-58C
Aerospatiale SA316B
Bell 206BIII
Mcdonnell douglas 369D
Mcdonnell douglas helicopter 369E
make_model
0
100
200
300
400
Accident Count


Top 30 make_model with Turbo Prop Engine by Accident Count
Cessna 208B
Cessna 208
Beech 200
Air tractor AT-502
Air tractor AT-502B
Ayres S2R-T34
Cessna 441
Beech C90
Piper PA-31T
Socata TBM 700
Mitsubishi MU-2B-60
Schweizer G-164B
Ayres S2R
Cessna 425
Beech 1900D
Grumman G-164B
Air tractor AT502
Air tractor inc AT-502B
Beech 65-A90
Air tractor AT-602
Beech 1900C
Beech B200
Air tractor AT-400
Beech B90
Swearingen SA226TC
Rockwell S-2R
Beech 99
Air tractor inc AT-602
Beech A100
Air tractor AT-802A
make_model
0
20
40
60
80
100
120
Accident Count


Top 30 make_model with Turbo Fan Engine by Accident Count
Cessna 550
Boeing 737
Learjet 35A
Boeing 737-300
Cessna 500
Boeing 757-232
Cessna 501
Cessna 525
Boeing 737 7H4
Boeing 757
Cessna 560
Mcdonnell douglas MD-11
Boeing 767
Boeing 727-200
Mcdonnell douglas DC-10-10
Cessna 650
Cessna 525A
Mcdonnell douglas DC-9-31
Boeing 757-222
Boeing 727
Embraer EMB-145LR
Boeing 737-200
Mcdonnell douglas DC-10-30
Mcdonnell douglas DC-9-82
Gates learjet 35A
Mcdonnell douglas MD-88
Boeing 747-400
Cessna 560XL
Learjet 35
Boeing 737-7H4
make_model
0
5
10
15
20
25
30
Accident Count

# Recommended Aircraft by Engine Type Based on Lowest Accident Counts

Based on the bar charts, the following aircraft makes and models are recommended for purchase, as they have the least number of accidents for their respective engine types:

---

## Reciprocating Engine

- Piper PA-23-250
- Cessna 177
- Piper PA-32-300

## Turbo Shaft Engine

- McDonnell Douglas Helicopter 369E
- Bell 206BIII
- Sikorsky S-76A

## Turbo Prop Engine

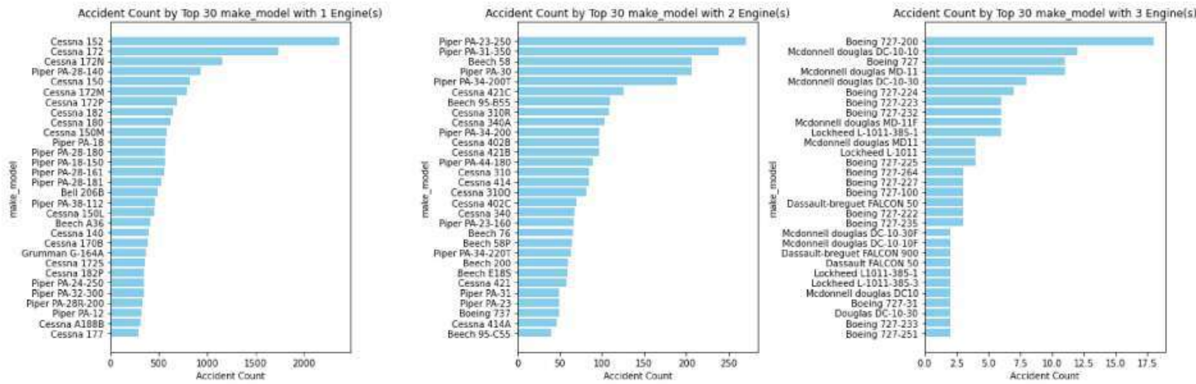
- Beech 99
- Cessna 425

## Turbo Fan Engine

- Cessna 560XL
- Boeing 737-7H4
- Boeing 747-400
- McDonnell Douglas DC-9-82
- McDonnell Douglas DC-10-30
- Boeing 757-222
- Cessna 525A
- Boeing 767

## Bar charts showing the accident counts by make & model number of engines

In [64]:
```
# List of engine counts I am interested in
engine_counts = [1, 2, 3]

# Create a figure with multiple subplots (one for each engine count)
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Loop through each engine count and plot the top 20 make_model combinations
for i, engine_count in enumerate(engine_counts):
    # Filter rows based on the number of engines
    engine_df = avi_df_clean[avi_df_clean['number_of_engines'] == engine_coun
t]

    # Group by make_model and count the number of accidents
    make_model_engine_count = engine_df.groupby('make_model').size().reset_ind
ex(name='accident_count')

    # Sort by accident count in descending order and get the top 30 make_model
combinations
    top_30_engine_make_model = make_model_engine_count.sort_values(by='acciden
t_count', ascending=False).head(30)

    # Plotting the bar chart for the current engine count
    axes[i].barh(top_30_engine_make_model['make_model'], top_30_engine_make_mo
del['accident_count'], color='skyblue')
    axes[i].set_title(f'Accident Count by Top 30 make_model with {engine_coun
t} Engine(s)')
    axes[i].set_xlabel('Accident Count')
    axes[i].set_ylabel('make_model')
    axes[i].invert_yaxis()  # Inverting the y-axis to show the make_model with
the highest count at the top

# Adjust layout
plt.tight_layout()

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




# Recommended Aircraft by Engine Number Based on Lowest Accident Counts

Based on the bar charts, the following aircraft makes and models are recommended for purchase, as they have the **least number of accidents** for their respective engine configurations:

## Single Engine Aircraft

- Piper PA-24-250
- Cessna 177
- Piper PA-32-300

## Twin Engine Aircraft

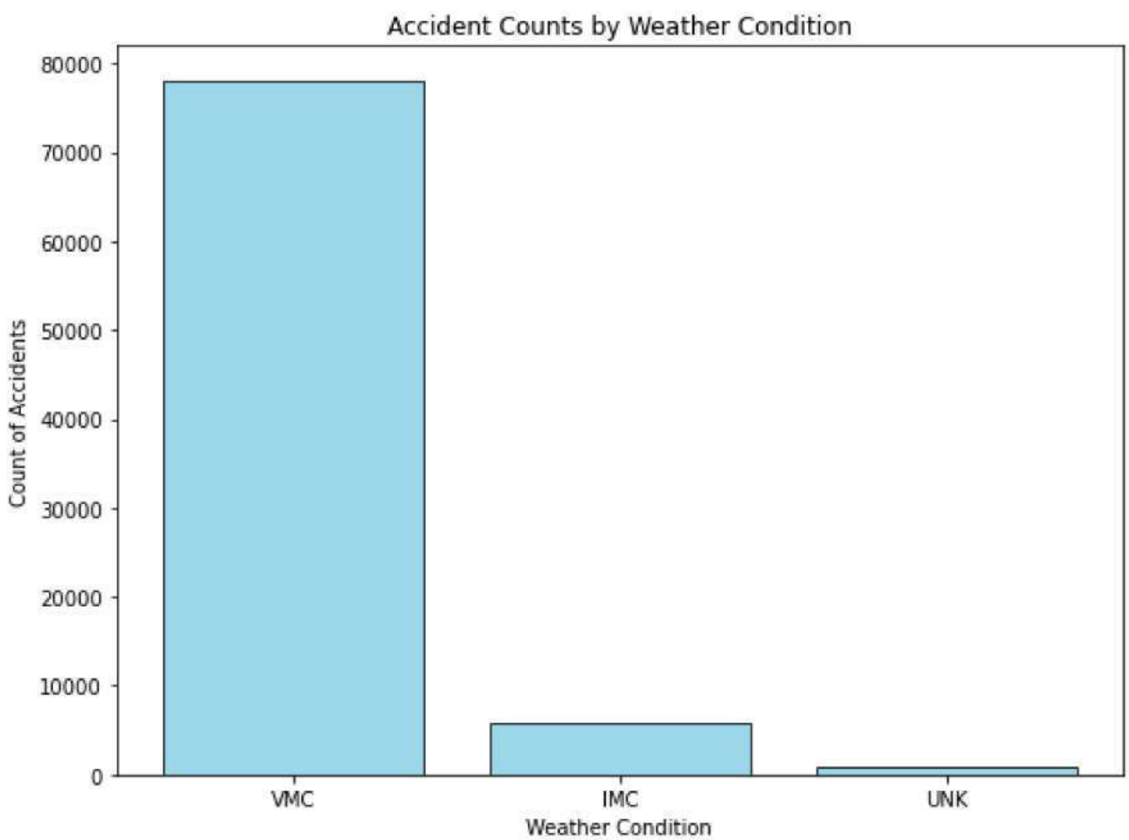
- Cessna 414A
- Piper PA-31

## Triple Engine Aircraft

- Boeing 727-251
- Douglas DC-10-30

## Bar charts showing accident counts based on two weather conditions

In [65]:
```
# Count occurrences of each weather condition
weather_counts = avi_df_clean['weather_condition'].value_counts()

# Plotting the bar chart
plt.figure(figsize=(8, 6))
plt.bar(weather_counts.index, weather_counts.values, color='skyblue', edgecolo
r='black', alpha=0.8)

# Adding titles and labels
plt.title('Accident Counts by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Count of Accidents')

# Displaying the chart
plt.tight_layout()
plt.show()
```




The graph above highlights that most occurrences involve Visual Meteorological Conditions (VMC), with significantly fewer cases for Instrument Meteorological Conditions (IMC).

# Conclusions

## Recommendations

Based on the analysis conducted, the following recommendations are proposed for the business:

1. **Focus on Low-Risk Aircraft**: Prioritize purchasing aircraft models and makes with lower accident counts and minimal severe outcomes, such as fatalities or serious injuries, which aligns with the goal of reducing operational risks. They are the following:

## Models to Consider (Lower Accident Counts)

Aircraft models with lower accident counts are recommended for safer operations.

- **Cessna Models**:
  - T210L, 208B, P210N, 182A
- **Piper Models**:
  - PA-22-108, PA-32R-301T
- **Beech Models**:
  - A23-24 , 95-C55
- **Boeing Models**:
  - 747-422
- **Bell Models**:
  - 430, 222
- **Gulfstream Models**:
  - 550, 680T, 500S
- **Airbus Models**:
  - 330, 321, A319-132, A321-211, AS-350

1. **Engine Type and Configuration**: Invest in aircraft with reliable engine configurations, such as those with turbo-prop & turbo-fan engines, as these were associated with consistent safety records in the analysis. They are the following:

---

# Based on engine type

#### Reciprocating Engine

- Piper PA-23-250
- Cessna 177
- Piper PA-32-300

#### Turbo Shaft Engine

- McDonnell Douglas Helicopter 369E
- Bell 206BIII
- Sikorsky S-76A

**Turbo Prop Engine**

- Beech 99
- Cessna 425

**Turbo Fan Engine**

- Cessna 560XL
- Boeing 737-7H4
- Boeing 747-400
- McDonnell Douglas DC-9-82
- McDonnell Douglas DC-10-30
- Boeing 757-222
- Cessna 525A
- Boeing 767

# Based on number of Engine(s)

**Single Engine Aircraft**

- Piper PA-24-250
- Cessna 177
- Piper PA-32-300

**Twin Engine Aircraft**

- Cessna 414A
- Piper PA-31

**Triple Engine Aircraft**

- Boeing 727-251
- Douglas DC-10-30

1. **Operational Planning**: Develop operational strategies that favor Instrument Meteorological Conditions (IMC), as these conditions have significantly fewer accidents compared to Visual Meteorological Conditions (VMC).

## Limitations

While the analysis provides actionable insights, there are some limitations to consider:

- Missing or inconsistent data, such as unknown weather conditions or incomplete records, could impact the accuracy of the findings.
- External factors, such as pilot experience, maintenance records, or geographical considerations, aircraft year of manufacture were not recorded.

## Next Steps

To further improve this project and provide more robust recommendations, I suggest that one should:

1. Include additional variables like pilot experience, aircraft age & year of manufacture, and maintenance history to better understand their impact on safety.
2. Use a more recent and comprehensive dataset to capture trends in modern aviation safety.
3. Use machine learning models to predict the likelihood of accidents based on aircraft features and operational conditions.