Phase 1 Project Submission

- Student name: Connor Anastasio
- Student pace: self paced (part time)
- Scheduled project review date/time: 05/08/2024 @ 11:00am
- Instructor name: Morgan Jones
- Blog post URL: https://dev.to/connoranastasio/a-history-of-computing-what-led-up-to-ai-4ond

Aircraft Analysis: Low-risk Aviation Investments

Overview

This project focuses on finding low-risk Aviation aircraft. Investment companies can use this information to determine which aircraft makes and models would be the safest (ie, most profitable) to invest in.

The main approach here will be cleaning our dataset to remove irrelevant information, then manipulate it to uncover relational trends that we can use to gain insight into what factors contribute to accidents.

Business Problem

Descriptive analysis of this dataset shows that certain makes, models, and types of aircraft appear to be significantly less likely to be involved in an accident than others. Choosing aircraft with one or two engines or those from overall safer companies such as Cessna, Piper, and Beech will allow investors to face significantly reduced risk.

Data Understanding

We will be using the publicly available Aviation Accident Data from the United States National Transport Safety Board. It contains very detailed information about each occurence, including flight information, weather, purpose of flight, and time of day. We will be focusing on eliminating the 'human element' and less predicatable information from the dataset wherever possible. This approach will enable investors to make informed decisions about what they are investing in. We cannot predict the weather next month or if a flight will need to be made at night or in low visibilty, so this information will be less helpful when considering long-term profitability.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#load our dataset
#Columns 6, 7, and 28 have mixed data types so we will treat them as strings
columns_to_object = [6, 7, 28]
df = pd.read_csv('data/Aviation_Data.csv', dtype={col: 'str' for col in colu

#set max columns to none to see information on each column
pd.set_option('display.max_columns', None)
df
```

Out[2]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Locati
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOO CREEK,
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPOR (
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, '
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, (
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, (
...	...	...	...	...	
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapol M
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, N
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, /
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, U
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, (

90348 rows × 31 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      90348 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            50132 non-null  object 
 9   Airport.Name            52704 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     87507 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             81793 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           82505 non-null  object 
 30  Publication.Date        73659 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

Our dataset contains an impressive amount of individual entries and information on flights dating back to 1948.

As we are primarily concerned with the objective safety of each make and model, we can begin data cleaning and investigation as to which aspects will not be useful for this.

Data Preparation

Data Cleaning

In [4]:
```
#normalize column names for ease of use
df.columns = df.columns.str.lower().str.replace('.', '_')
df['make'].replace('Air Tractor', 'Air Tractor Inc', inplace=True)
```

At a glance, certain columns will not be useful for our analysis and we can drop them now. Others will need to be looked at further before we make our judgment.

In [5]:
```
#drop obvious unnecessary columns
df.drop(columns = ['latitude', 'longitude', 'event_id'], inplace=True)
```

In [6]:
```
#Examine Broad Phase of Flight
print(df['broad_phase_of_flight'].value_counts())
print('broad phase null counts:', df['broad_phase_of_flight'].isnull().sum()
```

```
broad_phase_of_flight
Landing        15428
Takeoff        12493
Cruise         10269
Maneuvering     8144
Approach        6546
Climb           2034
Taxi            1958
Descent         1887
Go-around       1353
Standing         945
Unknown          548
Other            119
Name: count, dtype: int64
broad phase null counts: 28624
```

In [7]:
```
df.drop(columns = ['broad_phase_of_flight'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 27 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   investigation_type      90348 non-null  object 
 1   accident_number         88889 non-null  object 
 2   event_date              88889 non-null  object 
 3   location                88837 non-null  object 
 4   country                 88663 non-null  object 
 5   airport_code            50132 non-null  object 
 6   airport_name            52704 non-null  object 
 7   injury_severity         87889 non-null  object 
 8   aircraft_damage         85695 non-null  object 
 9   aircraft_category       32287 non-null  object 
 10  registration_number     87507 non-null  object 
 11  make                    88826 non-null  object 
 12  model                   88797 non-null  object 
 13  amateur_built           88787 non-null  object 
 14  number_of_engines       82805 non-null  float64
 15  engine_type             81793 non-null  object 
 16  far_description         32023 non-null  object 
 17  schedule                12582 non-null  object 
 18  purpose_of_flight       82697 non-null  object 
 19  air_carrier             16648 non-null  object 
 20  total_fatal_injuries    77488 non-null  float64
 21  total_serious_injuries  76379 non-null  float64
 22  total_minor_injuries    76956 non-null  float64
 23  total_uninjured         82977 non-null  float64
 24  weather_condition       84397 non-null  object 
 25  report_status           82505 non-null  object 
 26  publication_date        73659 non-null  object 
dtypes: float64(5), object(22)
memory usage: 18.6+ MB
```

Broad Phase of Flight could be interesting to consider further but unfortunately it is missing far too many values to be useful in our analysis.

Additionally, investing in a non-professionally made aircraft is undoubtedly a risky and unpredicatable investment. We should only consider accident data for professionally made aircraft. We will have to also drop the rows where it is unknown as we cannot confirm it was not amateur built:

In [8]:
```
#Remove All Non-Accident Investigation Type Rows, and all amateur built airc
df = df[df['investigation_type'] == 'Accident']
df = df[df['amateur_built'] == 'No']
df.drop(columns = ['investigation_type', 'accident_number', 'airport_code', 
                   'amateur_built', 'far_description', 'schedule', 'purpose_
```

Data Engineering

It will be helpful to create a few of our own columns for analysis.

In [9]:
```
df['total_injuries'] = df['total_fatal_injuries'] + df['total_serious_injuri
df['total_non_fatal_injuries'] = df['total_serious_injuries'] + df['total_mi
df
```

Out[9]:

	event_date	location	country	injury_severity	aircraft_damage	mak
0	1948-10-24	MOOSE CREEK, ID	United States	Fatal(2)	Destroyed	Stinsc
1	1962-07-19	BRIDGEPORT, CA	United States	Fatal(4)	Destroyed	Pip
2	1974-08-30	Saltville, VA	United States	Fatal(3)	Destroyed	Cessr
3	1977-06-19	EUREKA, CA	United States	Fatal(2)	Destroyed	Rockwe
4	1979-08-02	Canton, OH	United States	Fatal(1)	Destroyed	Cessr
...	...	...	...	...	...	
90343	2022-12-26	Annapolis, MD	United States	Minor	NaN	PIPE
90344	2022-12-26	Hampton, NH	United States	NaN	NaN	BELLANC
90345	2022-12-26	Payson, AZ	United States	Non-Fatal	Substantial	AMERICA CHAMPIO AIRCRAF
90346	2022-12-26	Morgan, UT	United States	NaN	NaN	CESSN
90347	2022-12-29	Athens, GA	United States	Minor	NaN	PIPE

76520 rows × 17 columns

We need to ensure the date column is properly formatted for all 76,520 remaining rows. We can do this by writing a simple for loop to ensure all of the values can be converted to standard Pandas datetime. If this code does not return an error, we won't have to worry.

In [10]:
```
def check_date_format(date):
    try:
        pd.to_datetime(date)
        return True
    except ValueError:
        return False

# Check if all dates in the column are properly formatted
all_dates_valid = df['event_date'].map(check_date_format).all()

if all_dates_valid:
```

```
    print("All dates are properly formatted.")
else:
    print("Some dates have formatting issues.")
```

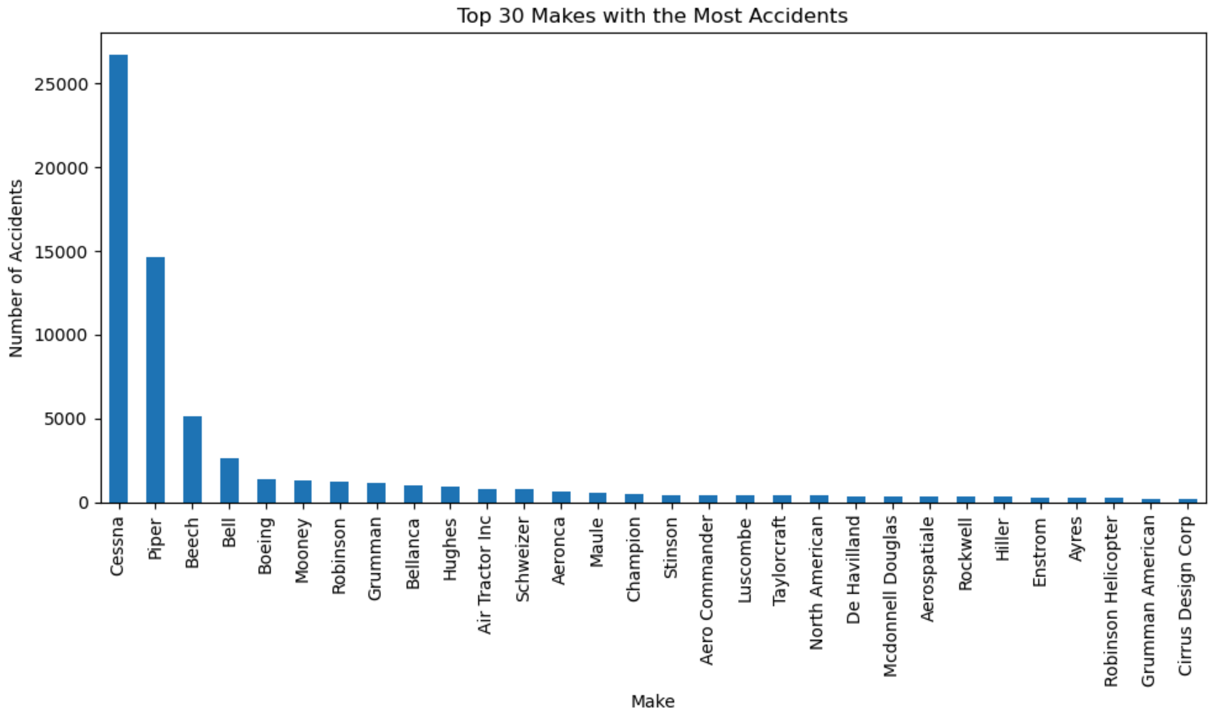
```
All dates are properly formatted.
```

In [11]:
```
#create a new column of make + model
df['make_model'] = df['make'] + '_' + df['model']

#Format 'make' column and describe its properties
df['make'] = df['make'].str.title()
df['make'].describe()
```

Out[11]:
```
count      76500
unique      1880
top       Cessna
freq       26695
Name: make, dtype: object
```

As we can see, there are 1880 different Makes, and many appear only once. This isn't very helpful in looking for data trends, so let's create a histogram with the top 30 most frequently occurring makes in our dataset:

In [12]:
```
#Create a variable to store the top 30 most frequently occurring makes:
top_30_makes = df['make'].value_counts().nlargest(30)

# Plotting the histogram for top 30 makes
plt.figure(figsize=(10, 6))
top_30_makes.plot(kind='bar')
plt.title('Top 30 Makes with the Most Accidents')
plt.xlabel('Make')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Cessna has the most accidents by a wide margin, but we will need additional and deeper analysis in order to make assessments on this.

In [13]:
```
#analyze frequency
df['year'] = pd.to_datetime(df['event_date']).dt.year
accidents_since_2000 = df[df['year'] >= 2000]
accidents_since_2000
```

Out[13]:

	event_date	location	country	injury_severity	aircraft_damage	make
47675	2000-01-01	HOMESTEAD, FL	United States	Non-Fatal	Substantial	Cessna
47676	2000-01-01	MONTEAGLE, TN	United States	Fatal(2)	Destroyed	Bellanca
47677	2000-01-02	VICTORVILLE, CA	United States	Non-Fatal	Substantial	Cessna
47678	2000-01-02	DOS PALOS, CA	United States	Non-Fatal	Substantial	Cessna
47679	2000-01-02	CORNING, AR	United States	Non-Fatal	Substantial	Piper
...	...	...	...	...	...	..
90343	2022-12-26	Annapolis, MD	United States	Minor	NaN	Piper
90344	2022-12-26	Hampton, NH	United States	NaN	NaN	Bellanca
90345	2022-12-26	Payson, AZ	United States	Non-Fatal	Substantial	American Champion Aircraft
90346	2022-12-26	Morgan, UT	United States	NaN	NaN	Cessna
90347	2022-12-29	Athens, GA	United States	Minor	NaN	Piper

34210 rows × 19 columns

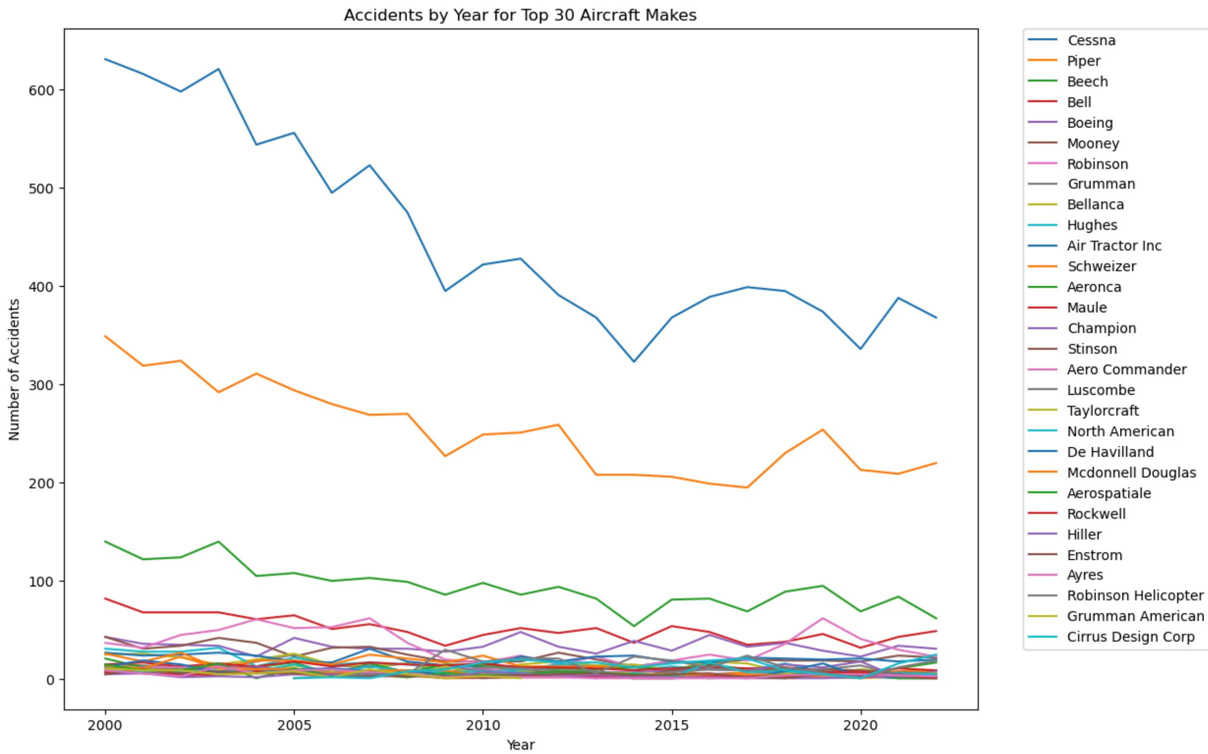
We will focus on the 30 most common aircraft makes for our analysis; these are the companies with enough instances to be statistically relevant.

While older planes are still in service, flight safety and security has improved drastically in our post-9/11 world. For this reason it is worth looking into whether there has been a noticeable decrease in accidents since:

In [14]:

```
top_30_aircraft = df['make'].value_counts().nlargest(30).index.tolist()

# Filtering DataFrame for 30 most prevalent aircraft makes
df_top_30 = df[df['make'].isin(top_30_aircraft)]

# Grouping data by make and year, and counting occurrences
df_top_30_since_2000 = df[(df['make'].isin(top_30_aircraft)) & (df['year'] >
grouped = df_top_30_since_2000.groupby(['make', 'year']).size().reset_index(

# Creating a line chart for each aircraft make
plt.figure(figsize=(12, 9))

for make in top_30_aircraft:
```

```
    make_data = grouped[grouped['make'] == make]
    plt.plot(make_data['year'], make_data['accidents'], label=make)

plt.title('Accidents by Year for Top 30 Aircraft Makes')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```



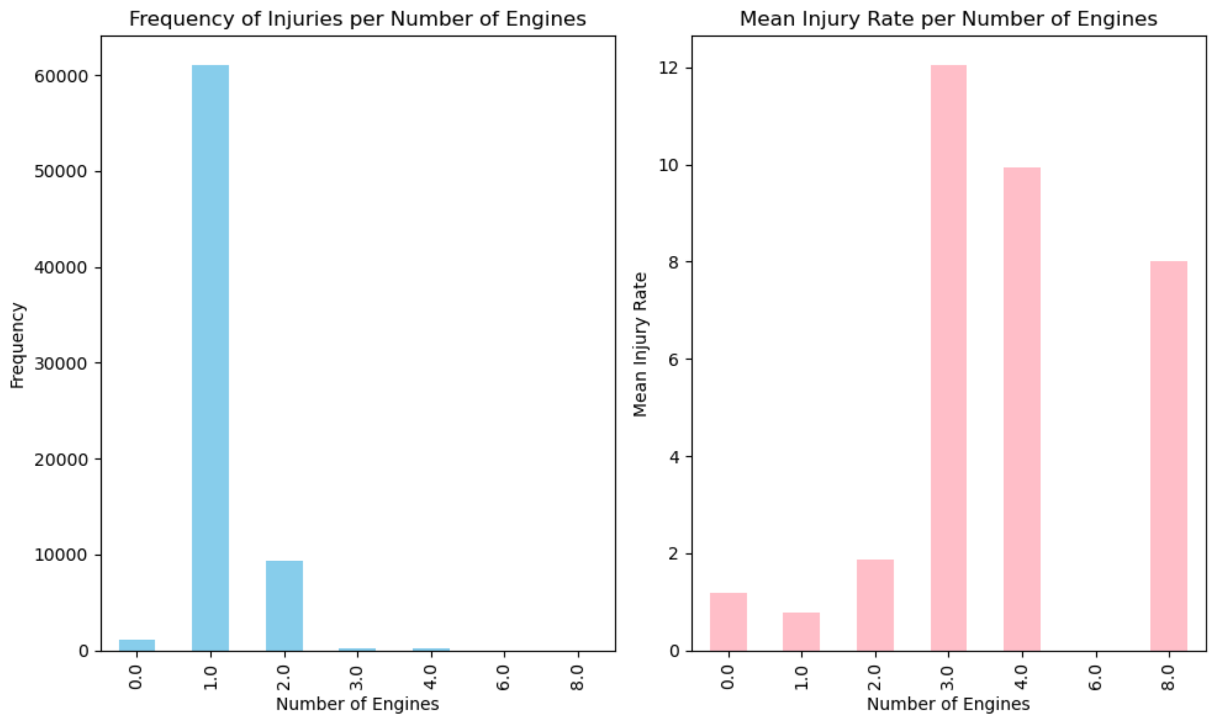
While this provides us with interesting information, it is not the full picture of what is happening. The dataset does not include the amount of planes in service for each company by year, so we don't know the percentage of each company's models that are involved in accidents. Still, it is noteworthy that every aircraft maker is on a downard trend in number of accidents. This implies it is a safer and potentially more profitable to invest in essentially any aircraft company than it was even 15 years ago.

Although we do not have the ability to create an "in-use" aircraft ratio by company, we thankfully have plenty of other useful information in our dataset, and approaches we can take.

The relationship between the number of engines and the total injuries will be instrumental in our analysis. To analyze this, we will create two bar plots:

In [15]:
```
# Calculate frequency of instances for each number of engines
instance_counts = df['number_of_engines'].value_counts().sort_index()

# Calculate mean injury rate for each number of engines
injury_rates = df.groupby('number_of_engines')['total_injuries'].mean().sort
```

```
plt.figure(figsize=(10, 6))

# Plot the bar plots side-by-side
plt.subplot(1, 2, 1)
instance_counts.plot(kind='bar', color='skyblue')
plt.title('Frequency of Injuries per Number of Engines')
plt.xlabel('Number of Engines')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
injury_rates.plot(kind='bar', color='pink')
plt.title('Mean Injury Rate per Number of Engines')
plt.xlabel('Number of Engines')
plt.ylabel('Mean Injury Rate')

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



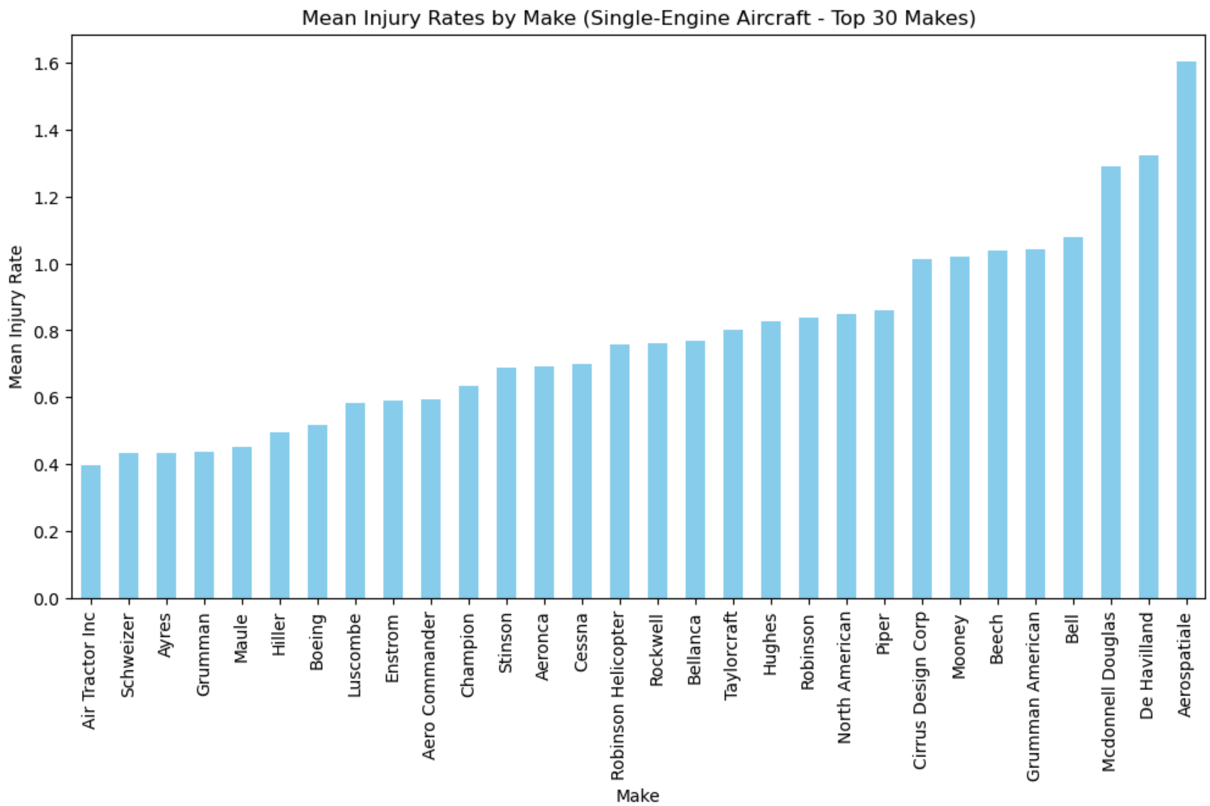
The left subplot displays a bar plot showing the frequency of instances for each number of engines, and the right shows a bar plot representing the mean injury rates for the corresponding number of engines.

Using these plots, it is easy to see that injury rates tend to increase as the number of engines does. In other words, aircraft with 1 or 2 engines appear to be the safest.

Notably, single engine aircraft have the second lowest mean injury rate while making up nearly all of the instances in our dataset. We will focus our analysis on these, and determine the safest make/model with one engine:

In [16]:
```
# Filtering df_top_30 for entries where the number of engines is one
df_top_30_single_engine = df_top_30[df_top_30['number_of_engines'] == 1]
```

```
# Grouping data by 'make' for single-engine aircraft (top 30 makes) and calc
make_injury_rates_single_engine = df_top_30_single_engine.groupby('make')['t

# Plotting bar plot
make_injury_rates_single_engine.plot(kind='bar', figsize=(12, 6), color='sky
plt.title('Mean Injury Rates by Make (Single-Engine Aircraft - Top 30 Makes)
plt.xlabel('Make')
plt.ylabel('Mean Injury Rate')
plt.xticks(rotation=90)
plt.show()
```

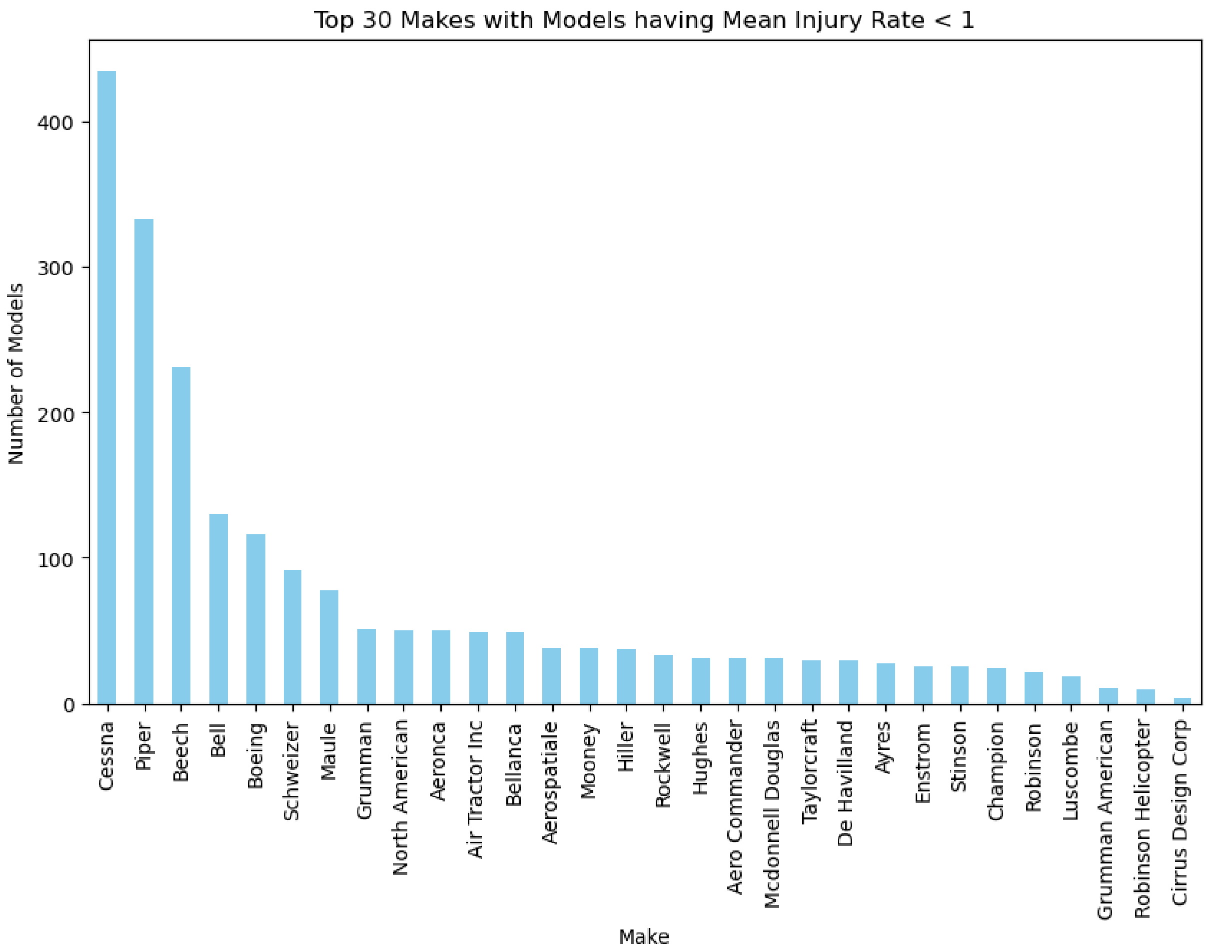


There are 22 makes with an injury rate below 1.0. This is excellent, as further restricting to these will still give us plenty of options to invest in.

In order to see if there is a single company with the overall best safety rating, we will examine the models in a similar fashion (using the top 30 most common again). In order to do this, we will find each model with an average injury rate less than 1, add each by make, and then plot them:

In [17]:
```
# Create a variable with the top 30 most common makes
top_30_makes = df['make'].value_counts().nlargest(30).index.tolist()

# Filter the DataFrame for the top 30 makes
df_top_30_makes = df[df['make'].isin(top_30_makes)]

# Calculate the mean injury rate per model for the top 30 makes
mean_injury_rate_per_model_top_30 = df_top_30_makes.groupby(['make', 'model'

# Filter models with a mean injury rate < 1 for the top 30 makes
```

```
models_with_mean_injury_lt_1_top_30 = mean_injury_rate_per_model_top_30[mear

# Count the number of models with mean injury rate < 1 for each make among t
models_lt_1_count_per_make_top_30 = models_with_mean_injury_lt_1_top_30.grou

# Plotting the top 30 makes with the count of models having mean injury rate
plt.figure(figsize=(10, 6))
models_lt_1_count_per_make_top_30.plot(kind='bar', color='skyblue')
plt.title('Top 30 Makes with Models having Mean Injury Rate < 1')
plt.xlabel('Make')
plt.ylabel('Number of Models')
plt.xticks(rotation=90)
plt.show()
```



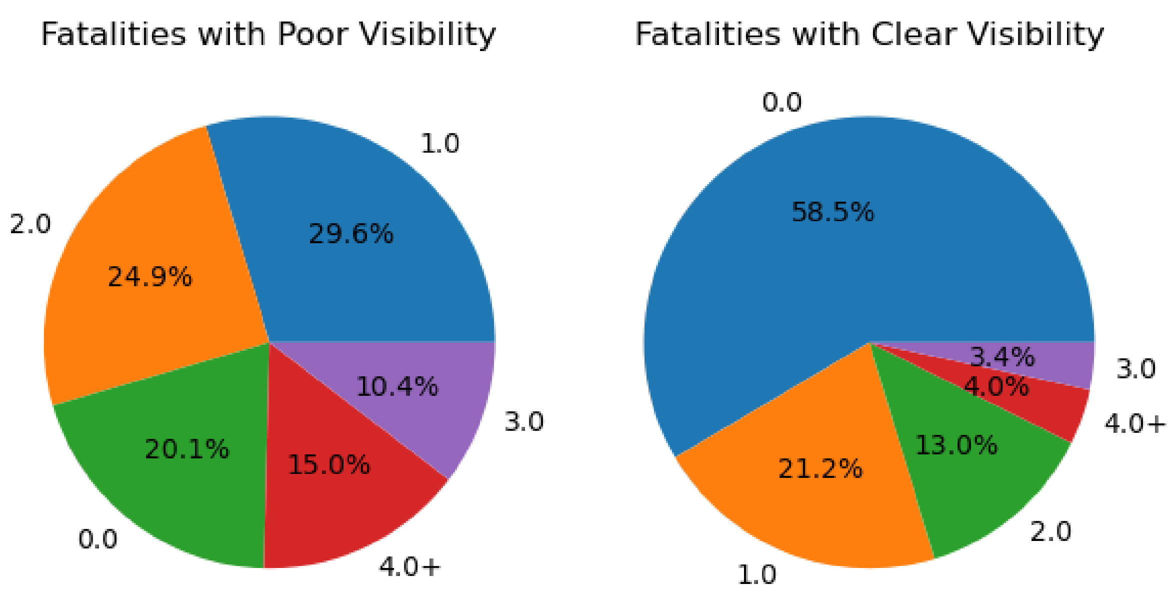
Cessna has the most models with the the lowest injury rates. Hooray!

Weather Influence

It will be useful to see if there is a noticeable relationship between poor visibility and accident statistics.

In [18]:
```
# Filter data for 'IMC' and 'VMC' while dropping null values in 'total_injur
imc_data = df[(df['weather_condition'] == 'IMC') & (df['total_injuries'].not
vmc_data = df[(df['weather_condition'] == 'VMC') & (df['total_injuries'].not
```

```
# Group counts of 4.0 injuries or more as '4+'
imc_data.loc[:, 'total_injuries_grouped'] = imc_data['total_injuries'].apply
vmc_data.loc[:, 'total_injuries_grouped'] = vmc_data['total_injuries'].apply

# Calculate value counts for 'IMC' and 'VMC' with grouping
imc_counts = imc_data['total_injuries_grouped'].value_counts()
vmc_counts = vmc_data['total_injuries_grouped'].value_counts()

# Convert counts to dictionary for pie chart plotting
imc_counts_dict = imc_counts.to_dict()
vmc_counts_dict = vmc_counts.to_dict()

# Replace count of '4+' if it doesn't exist in 'IMC' or 'VMC' counts
if '4.0+' not in imc_counts_dict:
    imc_counts_dict['4.0+'] = 0
if '4.0+' not in vmc_counts_dict:
    vmc_counts_dict['4.0+'] = 0

# Plot pie chart for 'IMC'
plt.subplot(1, 2, 1)
plt.pie(imc_counts_dict.values(), labels=imc_counts_dict.keys(), autopct='%1
plt.title('Fatalities with Poor Visibility')

# Plot pie chart for 'VMC'
plt.subplot(1, 2, 2)
plt.pie(vmc_counts_dict.values(), labels=vmc_counts_dict.keys(), autopct='%1
plt.title('Fatalities with Clear Visibility')

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



It is apparent from these pie charts that aircraft flying in poor visibilty conditions (requiring pilots to rely solely on instruments) leads to a significant increase in casualties. It is highly advised to avoid investing in planes that fly routes that have poor visibilty.

Conclusion:

- In terms of overall safety ratings for investing, single engine planes should make up the majority of investments.
- Investing in a single well-known company may be a safe approach to start with. Cessna is a great choice, as our dataset has provided ample information for model choices.
- Avoiding routes with poor visibility and providing further training pilots to deal with adverse conditions would likely aid in reducing accidents and increase profit.

Future:

- Compare cost of the safest aircraft models to determine which can bring the largest return on investment
- Gathering data on Aircraft sales and in-use data for individual aircraft would allow us to make more informative individual model recommendations