In [1]:

```
from matplotlib import pyplot as plt
%matplotlib inline
import pandas as pd
import numpy as np
import csv
import seaborn as sns
```

Aviation Risk Analysis

Overview

This project analyzes data form the National Transportation Safety Board. Descriptive analysis of the data reveals that Cessna and Boeing were the safest airplane makes represented in the data set. Our client can use this analysis when deciding which aircrafts to purchase.

Business Problem

A company is interested in purchasing and operating airplanes for commercial and private enterprises, but they do not know anything about the potential risks of aircraft.

Our goal was to determine which aircraft are the lowest risk for the company through analysis of the aeroplane's fatality and injury rates in event of a crash.

Data

https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses

The data is from the National Transportation Safety Board. It includes aviation accident data from 1962 to 2023 about civil aviation accidents and incidents in the United States and international waters.

Results

Our top three reccomendations were:

- Boeing 757251 (**0% fatality rate** and a **1.3% injury rate**)
- Cessna 152 (**9.6% fatality rate** and a **15.4% injury rate**)
- Cessna 172-N (**11.3% fatality rate** and a **19.7% injury rate**)

Data Understanding

In [2]:

```
#import and store data
#'mac_roman' was nessisary to import the data - worked without issues on mac and windows
df = pd.read_csv('data/AviationData.csv', encoding='mac_roman', low_memory=False)
#variables
#made this a variable so that we can easily incorporate older data if the client would li
ke
```

```
filter_year = 2001
```

In [3]:

```
df.shape
```

Out[3]:

```
(88889, 31)
```

In [4]:

```
df.head()
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.C
0	20001218X45444	Accident	SEA87LA080	10/24/1948	MOOSE CREEK, ID	United States	NaN	NaN	N
1	20001218X45447	Accident	LAX94LA336	7/19/1962	BRIDGEPORT, CA	United States	NaN	NaN	N
2	20061025X01555	Accident	NYC07LA005	8/30/1974	Saltville, VA	United States	36.922223	-81.878056	N
3	20001218X45448	Accident	LAX96LA321	6/19/1977	EUREKA, CA	United States	NaN	NaN	N
4	20041105X01764	Accident	CHI79FA064	8/2/1979	Canton, OH	United States	NaN	NaN	N

5 rows × 31 columns

In [5]:

```
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            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        75118 non-null   object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

Data Cleaning

Initial Cleaning

We first filtered our dataframe to only include Airplanes, as that is our primary buisness concern

In [6]:

```
df = df.loc[df['Aircraft.Category']=='Airplane']
```

Then we removed all ameteur built aircraft as we are looking to purchase aircraft from a manufacturer

In [7]:

```
df = df.loc[df['Amateur.Built']=='No']
```

Our next step is to filter our date ranges.

We used a 'filter_year' variable to be able to quickly modify the initial year we look at. We have set this variable to '2001' so that we are looking at data post 9/11, but it can be easily modified if our client wishes to look at older data.

In [8]:

```
df['Year'] = df['Event.Date'].str[-4:]
df['Year'] = df['Year'].astype(int)
```

In [9]:

```
df = df.loc[df['Year'] > filter_year]
```

We filled in unknown makes and models with 'unknown'

In [10]:

```
df['Make'] = df['Make'].fillna(value = "Unknown")
df['Model'] = df['Model'].fillna(value = "Unknown")
```

Then we dropped rows with missing data.

In [11]:

```
df.dropna()
df.describe()
```

Out[11]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year
count	18693.000000	18544.000000	18560.000000	18846.000000	20547.000000	21081.000000
mean	1.175360	0.659513	0.313524	0.207949	7.229912	2013.572696
std	0.423166	6.135952	2.343600	0.843311	33.761075	5.151531
min	0.000000	0.000000	0.000000	0.000000	0.000000	2002.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000	2009.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000	2014.000000

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year
75%	1.000000	0.000000	0.000000	0.000000	2.000000	2018.000000
max	8.000000	295.000000	161.000000	50.000000	576.000000	2022.000000

Grouping Incidents by Purpose

We wanted to group the aircraft into Private, Public, Goverment, and Unknown, so we first needed a clear look at the 'Purpose of Flight' Column.

In [12]:

```
df['Purpose.of.flight'].value_counts()
```

Out[12]:

```
Purpose.of.flight
Personal                    11647
Instructional                2737
Aerial Application            925
Business                      488
Positioning                   345
Unknown                       232
Aerial Observation            160
Other Work Use                150
Skydiving                     122
Flight Test                   120
Ferry                         101
Executive/corporate           100
Banner Tow                     89
Public Aircraft - Federal      52
Air Race show                  48
Glider Tow                     35
Public Aircraft                34
Public Aircraft - State        24
Firefighting                   17
Public Aircraft - Local        12
ASHO                            5
Air Race/show                   4
Air Drop                        3
PUBS                            3
External Load                   1
Name: count, dtype: int64
```

In [13]:

```
def categorize_flight(flight_type):
    if flight_type in ['Banner Tow',' Business' , 'Executive/corporate','Ferry', 'Other
Work Use', 'Positioning']:
        return 'Commercial'
    elif flight_type in ['Air Race show', 'Air Race/show', 'Glider Tow', 'Instructional'
, 'Personal', 'Skydiving']:
        return 'Private'
    elif flight_type in ['Aerial Observation', 'Air Drop', 'Firefighting', 'Public Aircr
aft', 'Public Aircraft - Federal', 'Public Aircraft - Local', 'Public Aircraft - State']
:
        return 'Government'
    else:
        return 'Unknown'
```

We sorted the flights into our target values and mapped that onto the new column 'flight_category'

In [14]:

```
df['flight_category'] = df['Purpose.of.flight'].map(categorize_flight)
```

Cleaning names and sorting injuries

We also catergorized injury severity into fewer columns

We also categorized injury severity into fewer columns.

In [15]:

```
def categorize_injury(injury_type):
    level = str(injury_type)
    if 'Non-Fatal' in level:
        return 'Non-Fatal'
    elif 'Fatal' in level:
        return 'Fatal'
    elif 'Minor' == level:
        return 'Minor'
    elif 'Serious' == level:
        return 'Serious'
    else:
        return 'Unavailable'

df['Injury_category'] = df['Injury.Severity'].map(categorize_injury)
```

We noticed that the 'Make' column often had the same names with slight variations. Some had different capitalization and abbreviations i.e. Airbus, Airbus Corp., Airbus Corporation, and others included special characters that needed to be cleaned.

In [16]:

```
#remove all special characters
df = df.replace(r'[^0-9a-zA-Z ] ', '', regex=True).replace("'", '')
```

In [17]:

```
def categorize_make(make):
    if type(make)!= str:
        return make
    make = make.upper()
    if '.' in make:
        make =make.replace('.', ' ')
    if ',' in make:
        make =make.replace(',', ' ')
    if 'COMPANY' in make:
        make =make.replace('COMPANY', ' ')
    if 'LTD' in make:
        make =make.replace('LTD', ' ')
    if 'CORPORATION' in make:
        make =make.replace('CORPORATION', ' ')
    if 'CORP' in make:
        make =make.replace('CORP', ' ')
    if 'AIRCRAFT' in make:
        make =make.replace('AIRCRAFT', ' ')
    if 'DESIGN' in make:
        make =make.replace('DESIGN', ' ')
    if 'INDUSTRIES' in make:
        make =make.replace('INDUSTRIES', ' ')
    if 'AEROSPACE' in make:
        make =make.replace('AEROSPACE', ' ')
    if ' CO' in make:
        make =make.replace('CO', ' ')
    if ' INC' in make:
        make =make.replace('INC', ' ')
    make = make.strip()
    return make
```

In [18]:

```
df['Makes_Standardized'] = df['Make'].map(categorize_make)
```

We also needed to clean the models of whitespace and select characters.

In [19]:

```
def clean_models(model):
```

```
    if type(model)!= str:
        return model
    model = model.upper()
    if '.' in model:
        model =model.replace('.', ' ')
    if ',' in model:
        model =model.replace(',', ' ')
    if '-' in model:
        model =model.replace('-', ' ')
    if ' ' in model:
        model =model.replace(' ', '')
    model = model.strip()
    return model
```

In [20]:

```
#clean the models of whitespace
df['Model'] = df['Model'].map(clean_models)
```

Feature Engineering

We needed two new columns. One of a ratio of fatalities to plane occupancy. Another of total injured to plane occupancy.

The plane plane occupancy is the sum of Total.Fatal.Injuries, Total.Serious.Injuries, Total.Minor.Injuries, and Total.Uninjured.

If those values were empty or missing, they were filled with '0'.

In [21]:

```
#Replacing empty cells with 0
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].replace(np.nan, 0)
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].replace(np.nan, 0)
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].replace(np.nan, 0)
df['Total.Uninjured'] = df['Total.Uninjured'].replace(np.nan, 0)
```

In [22]:

```
#create a plane occupancy column as well as a total injured column
df['Reported Occupancy'] = df['Total.Fatal.Injuries'] + df['Total.Serious.Injuries'] + d
f['Total.Minor.Injuries'] + df['Total.Uninjured']

df['Total Injured'] = df['Total.Minor.Injuries'] + df['Total.Serious.Injuries']

df['Fatality Ratio'] = (df['Total.Fatal.Injuries'] / df['Reported Occupancy']) * 100

df['Injury Ratio'] = (df['Total Injured'] / df['Reported Occupancy']) * 100
```

Analysis

Our analysis began by examining the aeroplanes makes most likely to be involved in accidents for Commercial and Private Flights.

Overall Accidents

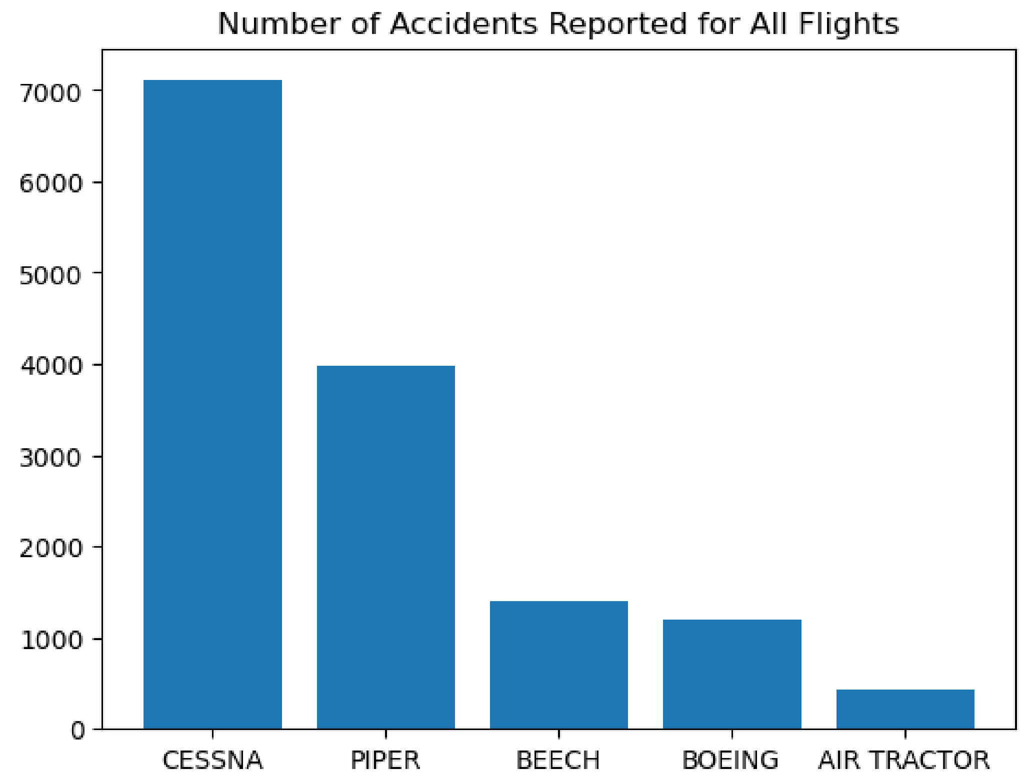
In [23]:

```
fig, ax = plt.subplots()

accidents_per_make = df['Makes_Standardized'].value_counts()[:5]

x = accidents_per_make.index
y = accidents_per_make.values
```

```
ax.bar(x, y)
ax.set_title("Number of Accidents Reported for All Flights")
```

Out[23]:

```
Text(0.5, 1.0, 'Number of Accidents Reported for All Flights')
```



With the most popular models established, we could examine their injury and fatality ratios.

In [24]:

```
popular_models = ['CESSNA', 'PIPER', 'BEECH', 'AIR TRACTOR', 'BOEING']

most_popular_dataframe = df.loc[df['Makes_Standardized'].isin(popular_models)]

most_popular_dataframe.groupby('Makes_Standardized').mean(numeric_only=True)
```

Out[24]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	
Makes_Standardized						
AIR TRACTOR	1.000000	0.208817	0.146172	0.097448	0.573086	2014.11
BEECH	1.376198	0.694385	0.230988	0.192608	1.267235	2013.64
BOEING	2.031250	2.051667	1.000000	0.320000	69.315833	2014.75
CESSNA	1.083893	0.341123	0.225961	0.162467	1.266507	2013.20
PIPER	1.126980	0.393977	0.199247	0.165872	1.146550	2013.52

From this, we determined that Boeing had the overall lowest fatality and injury rate.

Accidents in Commercial Airlines

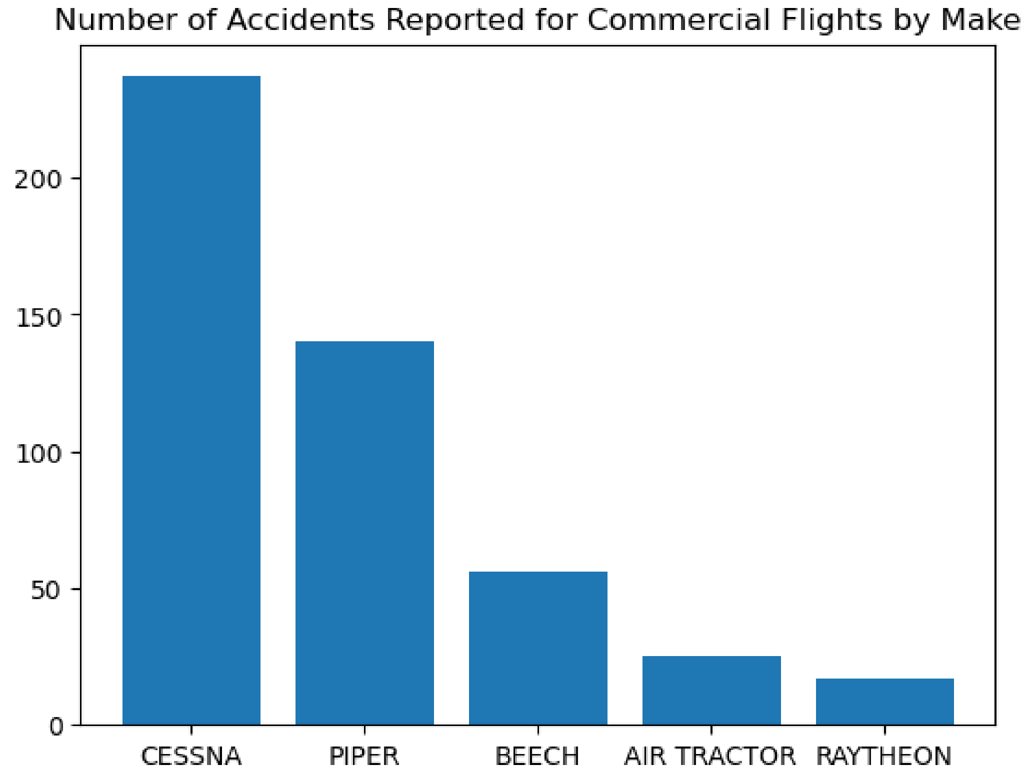
In [25]:

```
fig, ax = plt.subplots()
```

```python
commercial_flights_df = df[(df['flight_category'] == "Commercial")]

accidents_per_model_commercial = commercial_flights_df['Makes_Standardized'].value_counts
()[:5]

x = accidents_per_model_commercial.index
y = accidents_per_model_commercial.values

ax.bar(x, y)
ax.set_title("Number of Accidents Reported for Commercial Flights by Make")
```

Out[25]:

```
Text(0.5, 1.0, 'Number of Accidents Reported for Commercial Flights by Make')
```



In [26]:

```python
popular_commercial_models = ['CESSNA', 'PIPER', 'BEECH', 'AIR TRACTOR', 'RAYTHEON']

most_popular_commercial_dataframe = df.loc[df['Makes_Standardized'].isin(popular_commerci
al_models)]


most_popular_commercial_dataframe.groupby('Makes_Standardized').mean(numeric_only=True)
```

Out[26]:

Makes_Standardized	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	
AIR TRACTOR	1.000000	0.208817	0.146172	0.097448	0.573086	2014.11
BEECH	1.376198	0.694385	0.230988	0.192608	1.267235	2013.64
CESSNA	1.083893	0.341123	0.225961	0.162467	1.266507	2013.20
PIPER	1.126980	0.393977	0.199247	0.165872	1.146550	2013.52
RAYTHEON	1.540541	0.987952	0.277108	0.192771	2.108434	2013.63

Through this analysis, we are able to see that Cessna had the lowest fatality rate of the commercial makes.

Accidents in Private Airlines

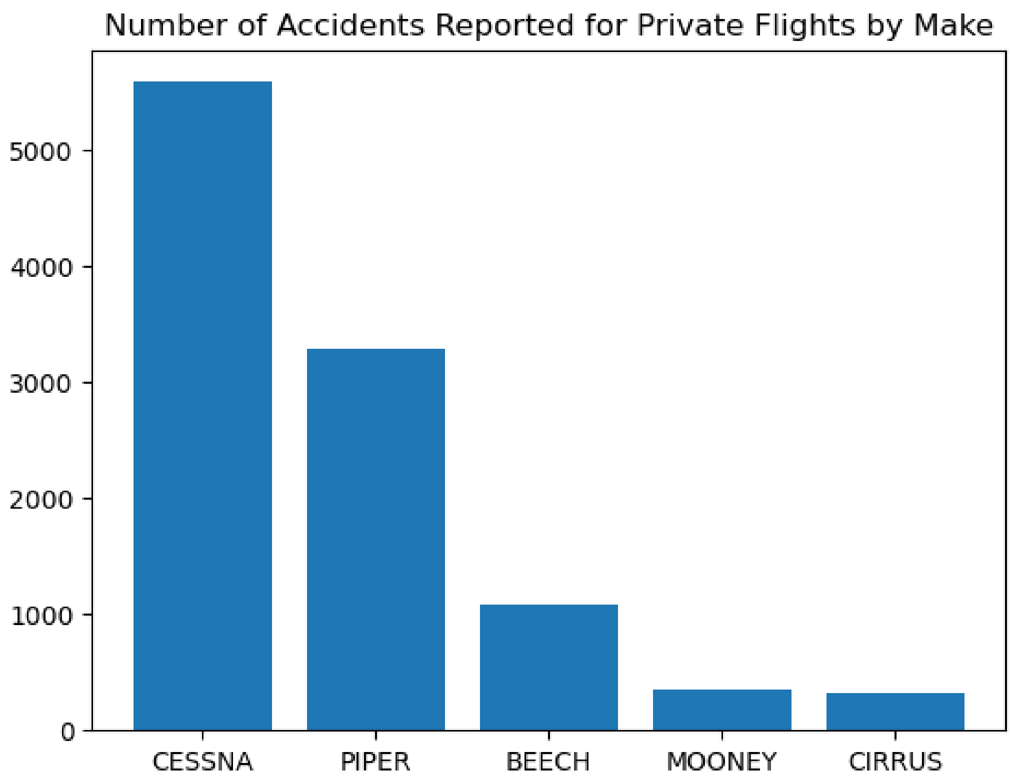
Accidents in Private Airlines

In [27]:

```
#accidents per year
fig, ax = plt.subplots()

personal_flights_df = df[(df['flight_category'] == "Private")]

accidents_per_model_personal = personal_flights_df['Makes_Standardized'].value_counts()[:
5]

x = accidents_per_model_personal.index
y = accidents_per_model_personal.values

ax.bar(x, y)
ax.set_title("Number of Accidents Reported for Private Flights by Make")
```

Out[27]:

```
Text(0.5, 1.0, 'Number of Accidents Reported for Private Flights by Make')
```



In [28]:

```
popular_private_models = ['CESSNA', 'PIPER', 'BEECH', 'MOONEY', 'CIRRUS', ]

most_popular_private_dataframe = df.loc[df['Makes_Standardized'].isin(popular_private_mod
els)]

most_popular_private_dataframe.groupby('Makes_Standardized').mean(numeric_only=True)
```

Out[28]:

Makes_Standardized	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	
BEECH	1.376198	0.694385	0.230988	0.192608	1.267235	2013.64
CESSNA	1.083893	0.341123	0.225961	0.162467	1.266507	2013.20
CIRRUS	1.000000	0.593516	0.264339	0.154613	1.029925	2014.51
MOONEY	1.000000	0.446970	0.194444	0.257576	0.861111	2013.32
PIPER	1.126980	0.393977	0.199247	0.165872	1.146550	2013.52

Cessna also had the lowest fatality rate of the private makes.

Examining Top-Performing Brands

This informed our choices as we decided to provide the client a recomendation for **one Boeing and two Cessna airplanes.** While Boeing crashes were not in the top of the commercial or private flights, we determined that was due to Boeing frequently appearing in both categories. As our client is looking to move into both feilds, a Boeing plane may be prefered.

Boeing

In [29]:

```
boeing_df =  df.loc[df['Makes_Standardized'] == 'BOEING']
boeing_df.describe()
```

Out[29]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year	Repor Occupa
count	608.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000
mean	2.031250	2.051667	1.000000	0.320000	69.315833	2014.753333	72.687
std	0.757615	17.922146	8.303159	1.983485	96.154628	4.567991	97.197
min	1.000000	0.000000	0.000000	0.000000	0.000000	2002.000000	0.000
25%	2.000000	0.000000	0.000000	0.000000	0.000000	2011.000000	0.000
50%	2.000000	0.000000	0.000000	0.000000	2.000000	2015.000000	3.000
75%	2.000000	0.000000	0.000000	0.000000	141.000000	2018.000000	145.000
max	4.000000	295.000000	161.000000	50.000000	501.000000	2022.000000	501.000

In [30]:

```
boeing_df.groupby(pd.Grouper(key='Model')).mean(numeric_only=True)[:10]
```

Out[30]:

Model	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year	O
747	4.0	0.090909	1.090909	0.090909	35.393939	2013.818182	3
737400	NaN	0.000000	1.000000	0.000000	56.000000	2010.571429	5
A75N1	1.0	0.000000	0.368421	0.000000	1.578947	2009.736842	
A75N1(PT17)	1.0	0.054054	0.189189	0.189189	1.378378	2015.027027	
757251	2.0	0.000000	0.500000	1.000000	146.000000	2007.000000	14
737	2.0	3.335821	0.843284	0.206468	50.512438	2016.532338	5
DC1030	NaN	0.000000	0.000000	1.000000	2.000000	2004.000000	
777	2.0	0.000000	0.080000	0.293333	113.653333	2015.960000	1
777200	2.0	0.000000	0.250000	0.250000	103.000000	2009.500000	1
B777	NaN	0.000000	0.000000	2.000000	201.250000	2007.250000	2

Based on the findings, we reccomend the Boeing 757251 which had the higest reported occupancy with very low

injury and fatality rates.

Cessna

In [31]:

```
cessna_df =  df.loc[df['Makes_Standardized'] == 'CESSNA']
cessna_df.describe()
```

Out[31]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year	Repor Occupa
count	6556.000000	7103.000000	7103.000000	7103.000000	7103.000000	7103.000000	7103.000
mean	1.083893	0.341123	0.225961	0.162467	1.266507	2013.206673	1.996
std	0.277248	0.994101	0.673798	0.581866	2.132822	5.319774	2.176
min	1.000000	0.000000	0.000000	0.000000	0.000000	2002.000000	0.000
25%	1.000000	0.000000	0.000000	0.000000	0.000000	2009.000000	1.000
50%	1.000000	0.000000	0.000000	0.000000	1.000000	2013.000000	2.000
75%	1.000000	0.000000	0.000000	0.000000	2.000000	2018.000000	2.000
max	2.000000	14.000000	11.000000	13.000000	124.000000	2022.000000	124.000

In [32]:

```
cessna_df.groupby(pd.Grouper(key='Model')).mean(numeric_only=True)[:10]
```

Out[32]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year	Repor Occupa
Model							
A188	1.0	0.428571	0.107143	0.107143	0.392857	2015.071429	1.035
172	1.0	0.279948	0.225260	0.139323	1.075521	2016.136719	1.720
182L	1.0	0.700000	0.400000	0.250000	1.150000	2010.750000	2.500
182	1.0	0.464646	0.279461	0.144781	1.101010	2015.700337	1.989
U206	1.0	0.782609	0.413043	0.304348	1.065217	2016.565217	2.565
172N	1.0	0.272000	0.220000	0.168000	1.120000	2010.592000	1.780
152	1.0	0.149682	0.136943	0.089172	0.974522	2012.257962	1.350
188	1.0	0.210526	0.105263	0.157895	0.473684	2015.368421	0.947
340A	2.0	1.080000	0.160000	0.120000	0.800000	2013.560000	2.160
T310R	2.0	1.000000	0.153846	0.230769	0.923077	2010.461538	2.307

Based on this data, we reccomend the Cessna 152 as it had the lowest fatality and injury ratio.

Our secondary reccomendation was the 172N, who performed the next best in the Cessna population.

Conclusions

Our top three reccomendations were:

- **For overall safety - Boeing 757251 (0% fatality rate and a 1.3% injury rate)**

- For commercial application - Cessna 152 (**9.6% fatality rate** and a **15.4% injury rate**)
- For private application - Cessna 172-N (**11.3% fatality rate** and a **19.7% injury rate**)

Next Steps

Further analysis could yield additional insights to further improve our aircraft recomendations.

- **Acquire more robust aircraft flight data.** Quantifiying risk for aircraft makes/models could be more accurate if we incorporated data outside of only aircraft incidents.
- **Financial Analysis.** Aircraft models, even variants of the same model, have very different operating costs. A financial analysis on aircrafts could identify the models with the lowest operating costs for our client.