Business understanding

Aircraft vary widely in their safety records depending on manufacturer, model, maintenance history, usage, and geography.While modern aircraft are statistically very safe, historical data shows that certain models and types have accidents rates often linked to operational environment, mechanical complexities or outdated systems. a good safety record of an airline is critical towards its operational market, survival, reputation, prestige and most importantly passengers confidence towards its service offered.

The company being new to aircraft enterprise, lacks experience in aircraft risk assessment. its therefore important to determine which aircraft has lowest risks for the organization to makes decisions on which aircraft to purchase.

Business problem

The organization is expanding into the aviation industry and seeks to identify aircrafts with low-risks in order to proceed with purchase and starts its operations. the goal is to ensure safety and purchasing a reliable aircraft for commercial and private use.

Objectives and business questions

1) To identify the safest aircrafts make/models

2) To evaluate aircraft safety Risks based on purpose of flight

3) To analyze state operational risks

Metric of success

The success of the business depends on evidence-based recommendations on low risk aircraft before initial purchase. insights must be actionable, understandable to non-technical leadership and can be used in procurement process.

.Data understanding

The data to be used in this analysis is from the National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

Loading dataset and getting the information

In [186…

```
# importing necessary libraries to be used
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

%matplotlib inline
```

In [187…

```
# reading and loading the dataset
aviation_data = pd.read_csv("data\AviationData.csv", encoding = "latin1")
aviation_data.head()
```

```
C:\Users\User\anaconda3\envs\learn-env\lib\site-packages\IPython\core\interactiveshell.py:3145: DtypeWarning: Columns (6,
7,28) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[187…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	.
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	NaN	.
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	NaN	.
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9222	-81.8781	NaN	NaN	.
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	NaN	.
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	NaN	.

5 rows × 31 columns

In [188…

```
aviation_data.tail()
```

Out[188…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	NaN
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	NaN
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	PAYSON
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	NaN
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	NaN

5 rows × 31 columns

In [189…

```
aviation_data.sample(5, random_state= 4)
```

Out[189…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
38221	20001207X04511	Accident	LAX95FA321	1995-09-02	PHOENIX, AZ	United States	NaN	NaN	DVT	PHOENIX DEER VALLEY MUN
33359	20001211X12901	Accident	DEN93LA088	1993-07-27	MEEKER, CO	United States	NaN	NaN	NaN	NaN
65184	20080820X01268	Accident	CHI08WA258	2008-08-17	Covington, United Kingdom	United Kingdom	NaN	NaN	NaN	NaN
82316	20180915X11112	Accident	WPR18LA262	2018-09-15	St. Johns, AZ	United States	343049N	1092213W	SJN	St Johns Industria Airpor

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
87501	20220307104735	Accident	ANC22LA022	2022-03-05	Newhalen, AK	United States	593854N	0154597W	NaN	NaN

5 rows × 31 columns

In [190…

```
# checking rows and columns shape
aviation_data.shape
```

Out[190…

```
(88889, 31)
```

The above dataset has a total of 88889 records and 31 features

In [191…

```
# column names
aviation_data.columns
```

Out[191…

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

In [192…

```
# checking data info, dtypes and non null counts
aviation_data.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
```

From the above information, its clearly notable that the dataset has a total of 5 columns with numerical data types and 26 columns consisting of categorical data.some columns such as Event.Date have data type object instead of date time, most of the variables have null values only a few variables have no null values that indicates that data needs to cleaned before analysis.

In [193…

```python
# checking for statistical info
aviation_data.describe().T
```

Out[193…

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

In [194…

```
# categorical info
aviation_data.describe(include= "O").head(10)
```

Out[194…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Na
count	88889	88889	88889	88889	88837	88663	34382	34373	50249	52
unique	87951	2	88863	14782	27758	219	25592	27156	10375	24
top	20001212X19172	Accident	DCA22WA214	2000-07-08	ANCHORAGE, AK	United States	332739N	0112457W	NONE	Pri
freq	3	85015	2	25	434	82248	19	24	1488	

4 rows × 26 columns

In [ ]:

Data preparation

Data cleaning

In [195…

```
# creating a copy of a data frame to be used in cleaning
aviation_cleaned_data = aviation_data.copy(deep= True)
```

In [196…

```
# checking the Country column
aviation_cleaned_data['Country'].value_counts(dropna= False)
```

Out[196…

```
United States                       82248
Brazil                                374
Canada                                359
Mexico                                358
United Kingdom                        344
                                    ...  
Malampa                                 1
Libya                                   1
Saint Vincent and the Grenadines        1
Yemen                                   1
BLOCK 651A                              1
Name: Country, Length: 220, dtype: int64
```

Observation from the above information, we see most of the data comes from the United States. so we will filter out the other countries to remain with united states to be used in this analysis

In [197…

```
# filtering dataset to include country USA
us_aviation_accidents_data= aviation_cleaned_data[aviation_cleaned_data['Country']== "United States"]
us_aviation_accidents_data
```

Out[197…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Nar
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	N
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	N
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9222	-81.8781	NaN	N
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	N
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	N
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	N
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	PAYS(
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	N
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	N

82248 rows × 31 columns

In [198…

```
us_aviation_accidents_data = us_aviation_accidents_data.copy()
```

In [199…

```
# checking for stat summary of numerical columns before filling in NaN values
us_aviation_accidents_data.describe().T
```

Out[199…

	count	mean	std	min	25%	50%	75%	max
Number.of.Engines	80373.0	1.135481	0.427286	0.0	1.0	1.0	1.0	8.0
Total.Fatal.Injuries	71594.0	0.421683	2.433647	0.0	0.0	0.0	0.0	265.0
Total.Serious.Injuries	70873.0	0.257178	1.144189	0.0	0.0	0.0	0.0	137.0
Total.Minor.Injuries	71519.0	0.332974	1.306604	0.0	0.0	0.0	0.0	125.0
Total.Uninjured	77243.0	4.302448	23.794728	0.0	0.0	1.0	2.0	699.0

In [200…

```
# filling missing numerical injury columns with 0
injury_columns= ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']
us_aviation_accidents_data[injury_columns] = us_aviation_accidents_data[injury_columns].fillna(0)
```

In [201…

```
# checking null values
us_aviation_accidents_data[injury_columns].isna().sum()
```

Out[201…

```
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
dtype: int64
```

In [202…

```
 # convert injury columns to numeric
for col in injury_columns:
    us_aviation_accidents_data[col]= pd.to_numeric(us_aviation_accidents_data[col], errors= "coerce")
```

Cleaning categorical columns

In [203…

```
# clean categorical columns
categorical_columns= ['Aircraft.Category', 'Make', 'Model', 'Purpose.of.flight', 'Injury.Severity',
                      'Aircraft.damage', 'Weather.Condition']

for col in us_aviation_accidents_data.columns:
    if us_aviation_accidents_data[col].dtype == "O":
        us_aviation_accidents_data[col] = us_aviation_accidents_data[col].str.strip().str.lower()
```

In [204…

```
# checking categorical columns
us_aviation_accidents_data[categorical_columns].isna().sum()
```

Out[204…

```
Aircraft.Category    54094
Make                    21
Model                   38
Purpose.of.flight     2429
Injury.Severity        108
Aircraft.damage       1979
Weather.Condition      645
dtype: int64
```

In [205…

```
# filling in Aircraft.Category
us_aviation_accidents_data['Aircraft.Category'].value_counts(dropna= False)
```

Out[205…

```
NaN                  54094
airplane             24229
helicopter            2723
glider                 503
balloon                229
gyrocraft              172
weight-shift           161
powered parachute       90
ultralight              25
wsft                     9
unknown                  4
blimp                    4
powered-lift             3
ultr                     1
rocket                   1
Name: Aircraft.Category, dtype: int64
```

The above Column has 54094 misiing values and its notable that some values are abbreviation of others for instance wsft is used instead of weight-shift and ultr inplace of ultralight

In [206…

```
# creating a dic of category abbreviations
category_abb = {'wsft': 'weight-shift',
               'ultr': 'ultralight',
                'unk': 'unknown'}
```

In [207…

```
# replacing abbreviations
us_aviation_accidents_data['Aircraft.Category'] = us_aviation_accidents_data['Aircraft.Category'].replace(category_abb)
```

In [208…

```
# checking
us_aviation_accidents_data['Aircraft.Category'].value_counts()
```

Out[208…

```
airplane             24229
helicopter            2723
glider                 503
balloon                229
gyrocraft              172
weight-shift           170
powered parachute       90
ultralight              26
unknown                  4
blimp                    4
powered-lift             3
rocket                   1
Name: Aircraft.Category, dtype: int64
```

In [209…

```
# filling in null values with unknown
us_aviation_accidents_data['Aircraft.Category']= us_aviation_accidents_data['Aircraft.Category'].fillna('unknown')
```

In [210…

```
# checking
us_aviation_accidents_data['Aircraft.Category'].isna().sum()
```

Out[210…

```
0
```

In [211…

```
# checking null values
# filling in make column

us_aviation_accidents_data['Make'].isna().sum()
```

Out[211…

```
21
```

In [212…

```
# filling in missing values with unknown
us_aviation_accidents_data['Make']= us_aviation_accidents_data['Make'].fillna("unknown")
```

In [213…

```
# checking
us_aviation_accidents_data['Make'].isna().sum()
```

Out[213…

```
0
```

In [214…

```
# cleaning model column
us_aviation_accidents_data['Model'].isna().sum()
```

Out[214…

```
38
```

In [215…

```
# checking value counts
us_aviation_accidents_data[['Model']].value_counts()
```

Out[215…

```
Model
152                    2323
172                    1637
172n                   1136
pa-28-140               910
150                     790
                       ...
lg2h                      1
libelle                   1
liberty xl                1
lighthizer special        1
&gcbc                     1
Length: 10786, dtype: int64
```

In [216…

```
# filling in null values with unkown
us_aviation_accidents_data['Model']= us_aviation_accidents_data['Model'].fillna("unknown")
```

In [217…

```
# cleaning purpose of flight column
us_aviation_accidents_data['Purpose.of.flight'].isna().sum()
```

Out[217…

```
2429
```

In [218…

```
# checking for value counts
us_aviation_accidents_data['Purpose.of.flight'].value_counts().head()
```

Out[218…

```
personal              48544
instructional         10429
unknown                5739
aerial application     4627
business               3843
Name: Purpose.of.flight, dtype: int64
```

In [219…

```
# filling in missing values with unknown
us_aviation_accidents_data['Purpose.of.flight']= us_aviation_accidents_data['Purpose.of.flight'].fillna('unknown')
```

In [220…

```
# checking effectiveness
# us_aviation_accidents_data.info()
```

In [221…

```
# cleaning Injury.Severity
us_aviation_accidents_data['Injury.Severity'].isna().sum()
```

Out[221… 108

In [222…
```
us_aviation_accidents_data['Injury.Severity']= us_aviation_accidents_data['Injury.Severity'].fillna('unknown')
```

In [223…
```
# cleaning location column
us_aviation_accidents_data['Location'].value_counts(dropna= True)
```

Out[223…
```
anchorage, ak       548
miami, fl           275
houston, tx         271
albuquerque, nm     265
chicago, il         256
                   ...
bennet, ne            1
las palmas, pr        1
lawrencevile, ga      1
servia, in            1
mt pocono, pa         1
Name: Location, Length: 17588, dtype: int64
```

In [224…
```
# extracting states from location
us_aviation_accidents_data['State']= aviation_cleaned_data['Location'].str.split(",").str[-1].str.strip()
```

In [225…
```
# checking
us_aviation_accidents_data['State'].isna().sum()
```

Out[225… 11

In [226…
```
# filling missing values with unknown
us_aviation_accidents_data['State']= us_aviation_accidents_data['State'].fillna("unknown")
```

In [227…
```
# checking columns
# us_aviation_accidents_data.isna().sum()
```

In [228…
```
# cleaning weather column
us_aviation_accidents_data['Weather.Condition'].value_counts()
```

Out[228…
```
vmc    75317
imc     5618
unk      668
Name: Weather.Condition, dtype: int64
```

In [229…
```
# filling in missing values with unknown
us_aviation_accidents_data['Weather.Condition']= us_aviation_accidents_data['Weather.Condition'].fillna('unk')
```

Cleaning location column

In [230…
```
# filling in missing values
us_aviation_accidents_data['Location']= us_aviation_accidents_data['Location'].fillna("unkown")
```

In [231…
```
# checking for missing values
us_aviation_accidents_data['Location'].isna().sum()
```

Out[231…
```
0
```

In [232…
```
#
us_aviation_accidents_data= us_aviation_accidents_data.dropna(subset= ['Location'])
```

In [233…
```
us_aviation_accidents_data['Location'].isna().sum()
```

Out[233…
```
0
```

Data analysis

In [234…
```
# plotting the top 5 aircraft make
top_make_counts= list(us_aviation_accidents_data['Make'].value_counts().head().index)
make_counts = list(us_aviation_accidents_data['Make'].value_counts().head())
```

In [235…
```
top_make_counts
```

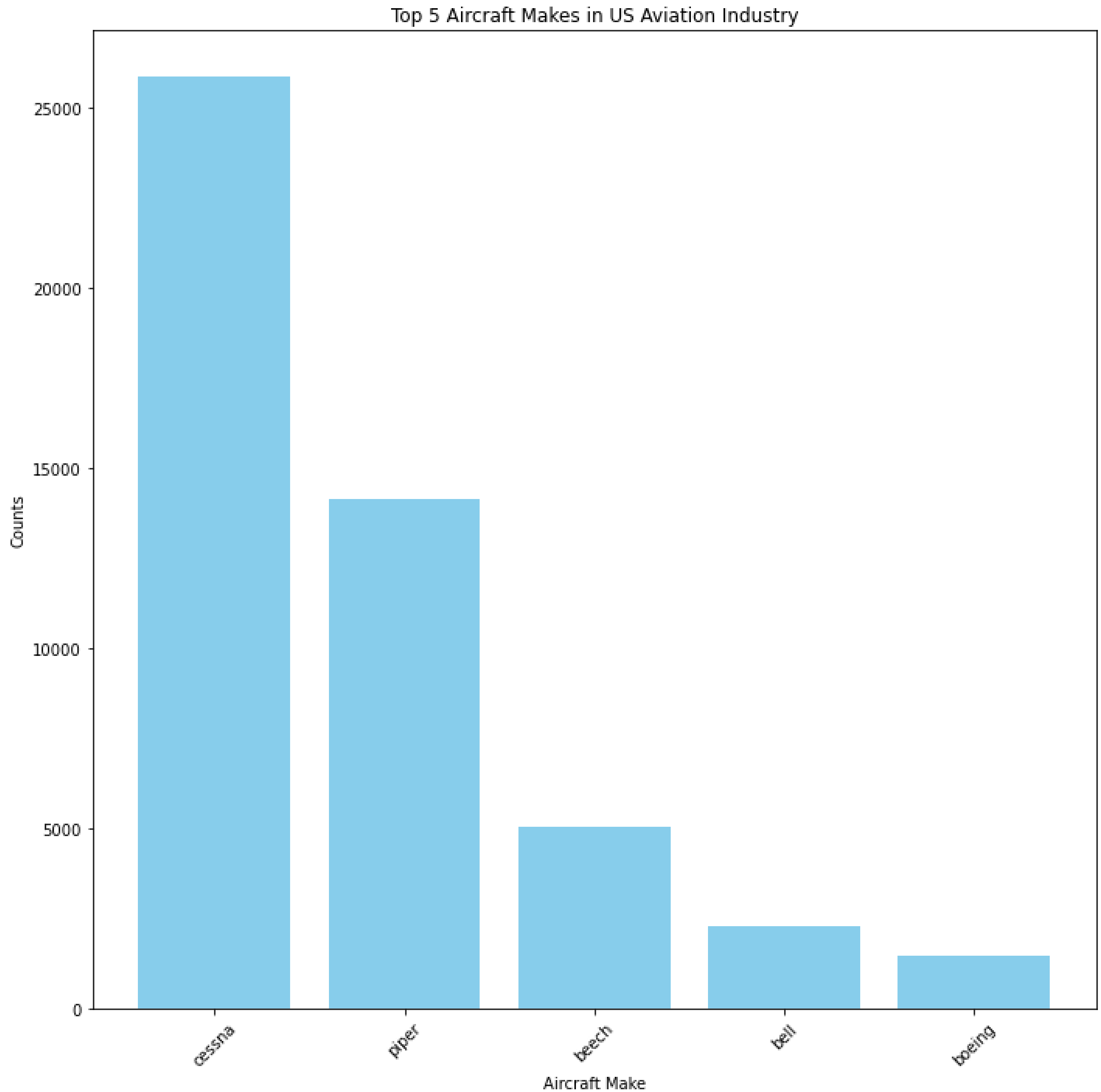
Out[235…
```
['cessna', 'piper', 'beech', 'bell', 'boeing']
```

In [236…
```
make_counts
```

Out[236…
```
[25853, 14168, 5059, 2285, 1485]
```

In [237…
```
# plotting top_make counts
fig, ax = plt.subplots(figsize=(10, 10))
ax.bar(top_make_counts, make_counts, color='skyblue')
ax.set_title('Top 5 Aircraft Makes in US Aviation Industry')
ax.set_xlabel('Aircraft Make')
ax.set_ylabel('Counts')
plt.xticks(rotation=45)  # Rotate labels if they overlap
```

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


Top 5 Aircraft Makes in US Aviation Industry
25000
20000
15000
10000
5000
0
Counts
cessna
piper
beech
bell
boeing
Aircraft Make

Cessna accounts for the largest number of accident records, followed by Piper and Beech. These three manufacturers are well-known for producing light general aviation aircraft, which are widely used for private flying, pilot training, and small-scale commercial operations. Their high representation likely reflects their large operational footprint in civilian aviation rather than disproportionately high accident rates.

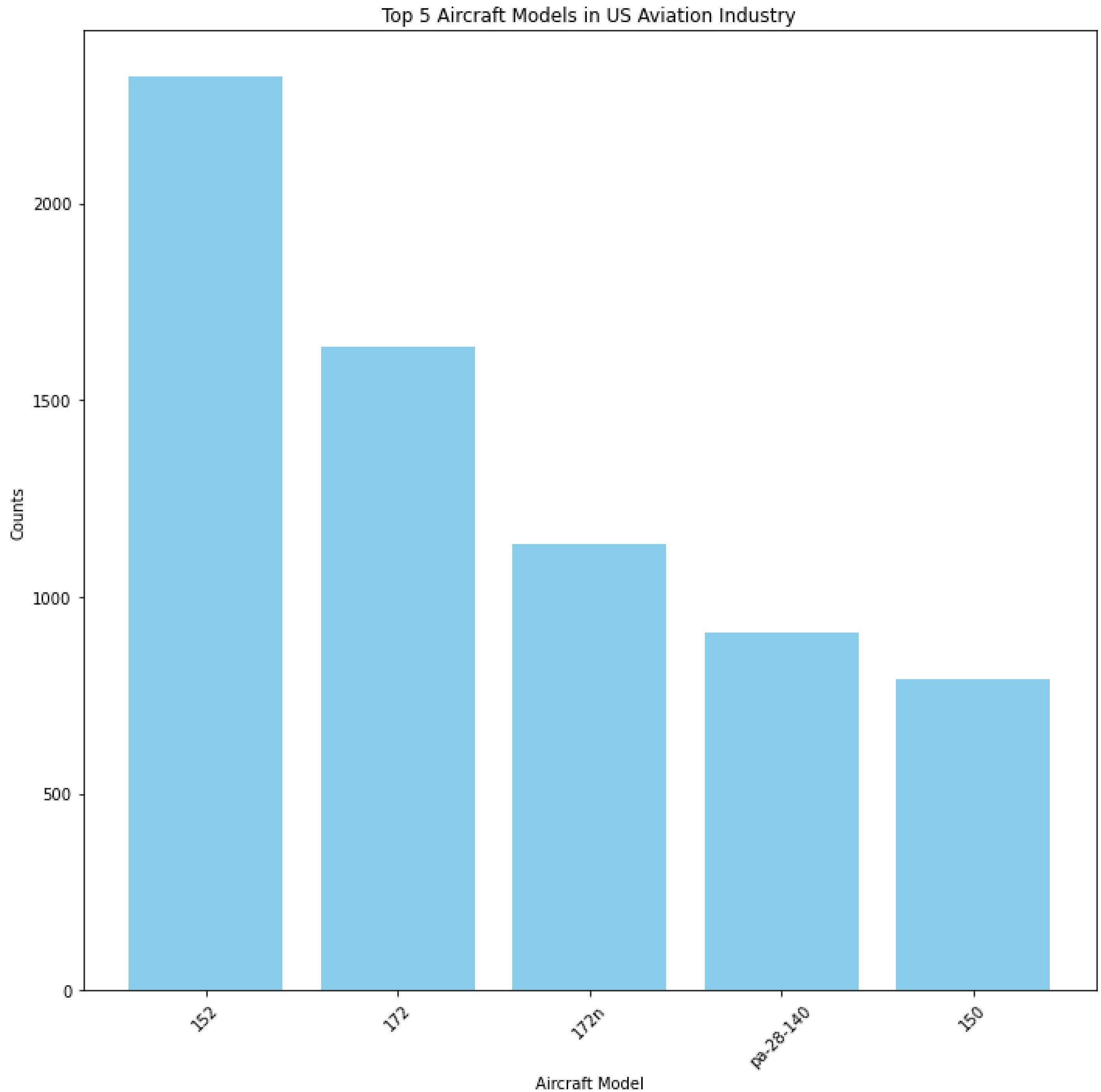
Bell and Boeing, primarily known for helicopters and large commercial airliners respectively, appear less frequently. This distribution aligns with their more specialized roles and smaller share in general aviation activity compared to the light aircraft segment.

In [238…

```
# Top Model counts
top_model_counts= list(us_aviation_accidents_data['Model'].value_counts().head().index)
model_counts = list(us_aviation_accidents_data['Model'].value_counts().head())
```

In [239…

```
# plotting the top 5 model
fig, ax = plt.subplots(figsize=(10, 10))
ax.bar(top_model_counts, model_counts, color='skyblue')
ax.set_title('Top 5 Aircraft Models in US Aviation Industry')
ax.set_xlabel('Aircraft Model')
ax.set_ylabel('Counts')
plt.xticks(rotation=45)  # Rotate labels if they overlap
plt.tight_layout()
plt.show()
```


Top 5 Aircraft Models in US Aviation Industry
Counts
2000
1500
1000
500
0
152
172
172n
pa-28-140
150
Aircraft Model

Observation

Cessna 152, 172, 172n and 150 are among the most commonly used models in aviation. cessna 152 has highest number of accidents records in this dataset, followed by cessna 172, 172n and 150. pa-28-140(PA-28-140) is model from Piper make.

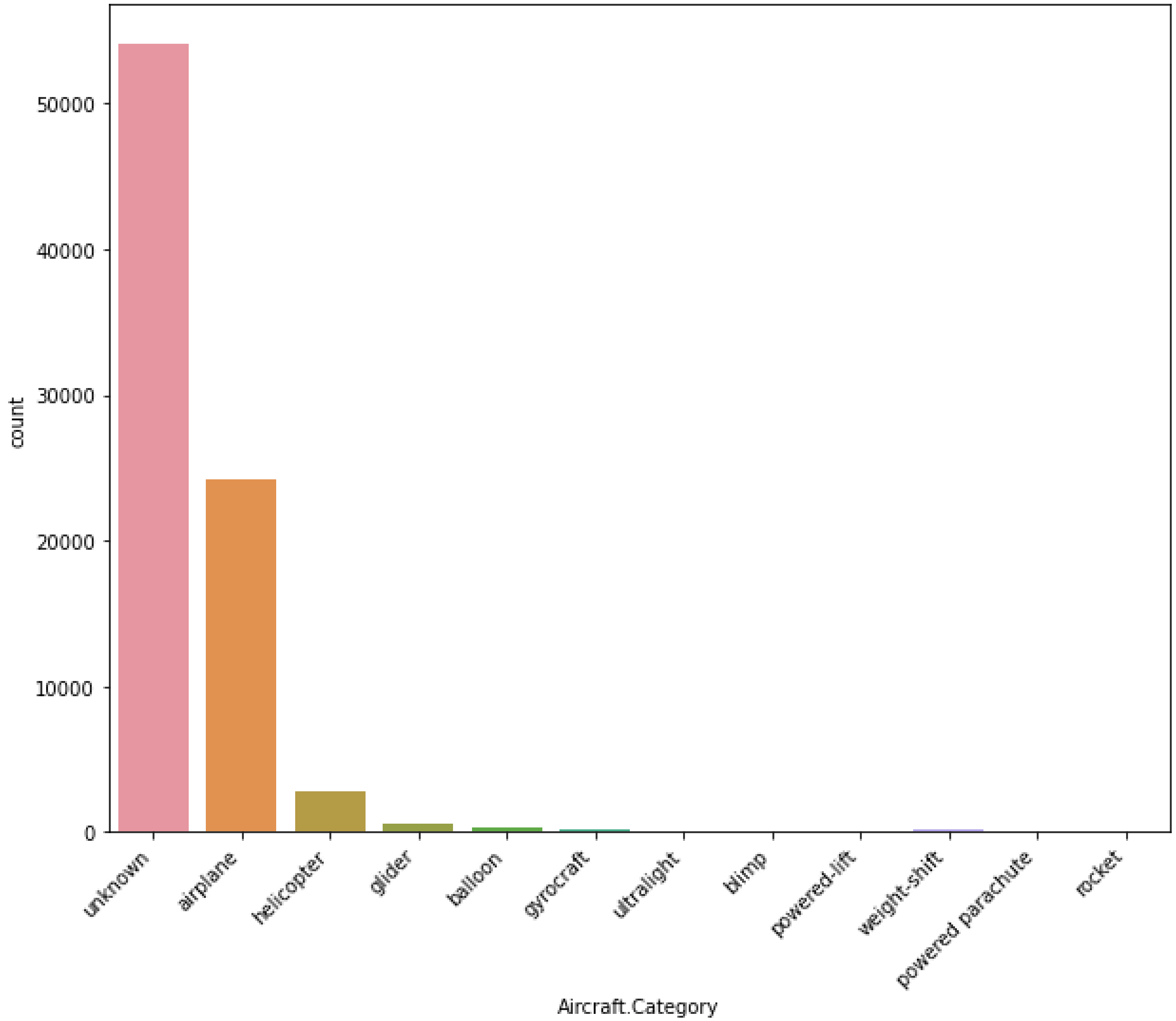
In [240…

```
# plotting count plot of aircraft category
plt.figure(figsize=(10, 8))
sns.countplot(us_aviation_accidents_data['Aircraft.Category'])

plt.xticks(rotation=45, ha="right")

plt.show()
```

```
C:\Users\User\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argume
nts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



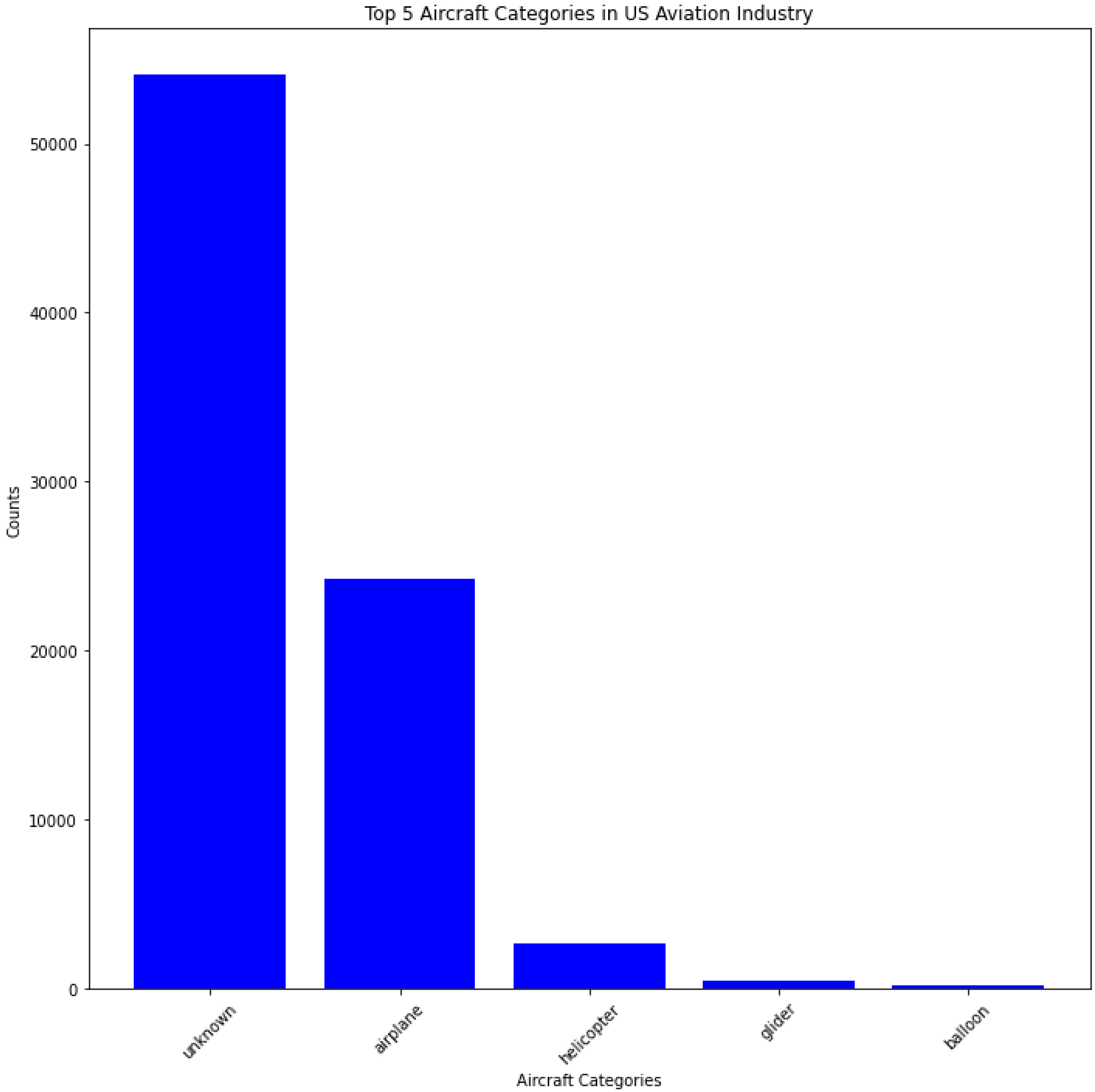
In [241…

```
# creating a list of top 5 aircraft category and counts
top_aircraft_category = list(us_aviation_accidents_data['Aircraft.Category'].value_counts().head().index)
top_aircraft_counts = list(us_aviation_accidents_data['Aircraft.Category'].value_counts().head())
```

In [242…

```
# plotting top aircraft categories used in Aviation
fig, ax = plt.subplots(figsize=(10, 10))
ax.bar(top_aircraft_category, top_aircraft_counts, color='blue')
ax.set_title('Top 5 Aircraft Categories in US Aviation Industry')
ax.set_xlabel('Aircraft Categories')
ax.set_ylabel('Counts')
```

```
plt.xticks(rotation=45)  # Rotate labels if they overlap
plt.tight_layout()
plt.show()
```


Top 5 Aircraft Categories in US Aviation Industry
Counts
Aircraft Categories
0
10000
20000
30000
40000
50000
unknown
airplane
helicopter
glider
balloon

Observation

from the data above, The "Unknown" category accounts for the highest number of entries, suggesting a significant proportion of records either lacked complete information or were not properly categorized during reporting. While this limits definitive conclusions, it underscores the importance of accurate data entry in aviation safety analysis.

Among the identifiable categories, airplanes dominate, which aligns with their widespread use in both commercial and general aviation. Helicopters, gliders, and balloons appear less frequently, likely reflecting their lower operational volume.

In [243…

```
# purpose of flight counts in aviation accidents data
purpose_counts= aviation_cleaned_data['Purpose.of.flight'].value_counts().head()
```

In [244…

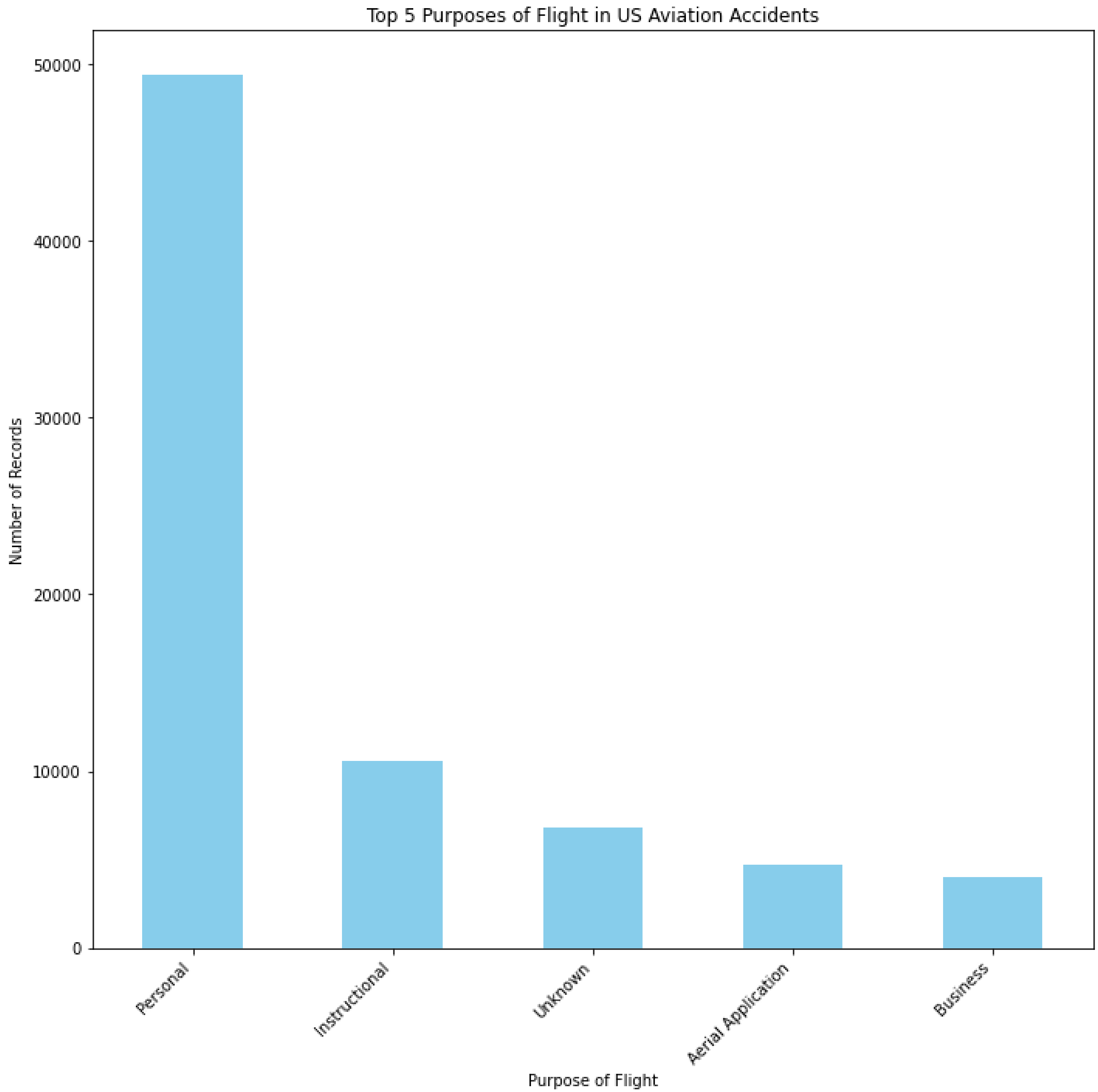
```
purpose_counts
```

Out[244…

```
Personal              49448
Instructional         10601
Unknown                6802
Aerial Application     4712
Business               4018
Name: Purpose.of.flight, dtype: int64
```

In [245…

```
# plotting a count plot of purpose of flight variable
plt.figure(figsize=(10, 10))
purpose_counts.plot(kind='bar', color='skyblue')
plt.title('Top 5 Purposes of Flight in US Aviation Accidents')
plt.xlabel('Purpose of Flight')
plt.ylabel('Number of Records')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```


Top 5 Purposes of Flight in US Aviation Accidents
Number of Records
50000
40000
30000
20000
10000
0
Personal
Instructional
Unknown
Aerial Application
Business
Purpose of Flight

observation

Personal category dominates in accidents records, accounting to nearly five times the second category. instructional is flight training, represents the second largest group, this in most cases used in training new pilot students. unknown suggests that flight purposes is not known while during data collection, which should be improved. Flight for Aerial Application and business purposes have less accidents records.

In [246…
```
# correlation between accidents and aircraft category
accident_by_category= (us_aviation_accidents_data['Aircraft.Category'].value_counts().sort_values(ascending= False))
```

Analyzing Risks

In [247…
```
# calculating total incidents and fatalities per make model
model_stats= us_aviation_accidents_data.groupby(["Make", "Model"]).agg({'Event.Id': 'count',
                                                     'Total.Fatal.Injuries':'sum',
                                                     'Total.Serious.Injuries': 'sum',
                                                     'Total.Minor.Injuries': 'sum',
                                                     'Total.Uninjured': 'sum',
                                                     })
```

In [248…
```
# calculating total people involved
model_stats["Total People"]= (model_stats['Total.Fatal.Injuries'] +
                             model_stats['Total.Serious.Injuries'] +
                             model_stats['Total.Minor.Injuries'] +
                             model_stats['Total.Uninjured'] )
```

In [ ]:
```

```

In [249…
```
# calculating safety metrics (risk factor)
model_stats['Fatality Rate']= model_stats['Total.Fatal.Injuries'] / model_stats['Total People']
```

In [250…
```
model_stats['Injury Rates']= (model_stats['Total.Fatal.Injuries'] + model_stats['Total.Serious.Injuries'] +
                              model_stats['Total.Minor.Injuries'])/ model_stats["Total People"]
```

In [251…
```
model_stats
```

Out[251…

Make	Model	Event.Id	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Total People	Fatality Rate	Injury Rates
1200	**g103**	1	0.0	1.0	0.0	0.0	1.0	0.0	1.0
177mf llc	**pitts model 12**	1	0.0	2.0	0.0	0.0	2.0	0.0	1.0
1977 colfer-chan	**steen skybolt**	1	0.0	0.0	1.0	0.0	1.0	0.0	1.0
1st ftr gp	**focke-wulf 190**	1	1.0	0.0	0.0	0.0	1.0	1.0	1.0
2000 mccoy	**genesis**	1	1.0	0.0	0.0	0.0	1.0	1.0	1.0
...	**...**	...	...	...	...	...	...	...	...
zubair s khan	**raven**	1	1.0	0.0	0.0	0.0	1.0	1.0	1.0
zuber thomas p	**zuber super drifter**	1	0.0	0.0	0.0	1.0	1.0	0.0	0.0
zukowski	**eaa biplane**	1	0.0	0.0	0.0	1.0	1.0	0.0	0.0
zwart	**kit fox vixen**	1	0.0	0.0	0.0	2.0	2.0	0.0	0.0
zwicker murray r	**glastar**	1	0.0	0.0	0.0	2.0	2.0	0.0	0.0

17094 rows × 8 columns

In [252…

```
# filtering model with minimum number of incidents
# model_stats.sort_values(by='Fatality Rate', ascending= False).index
```

In [253…

```
# resetting columns
plot_df= model_stats.sort_values(by='Fatality Rate', ascending= False).reset_index()
# creating a combined label for plotting
plot_df['make_model'] = plot_df['Make'] + ' ' + plot_df['Model']
```

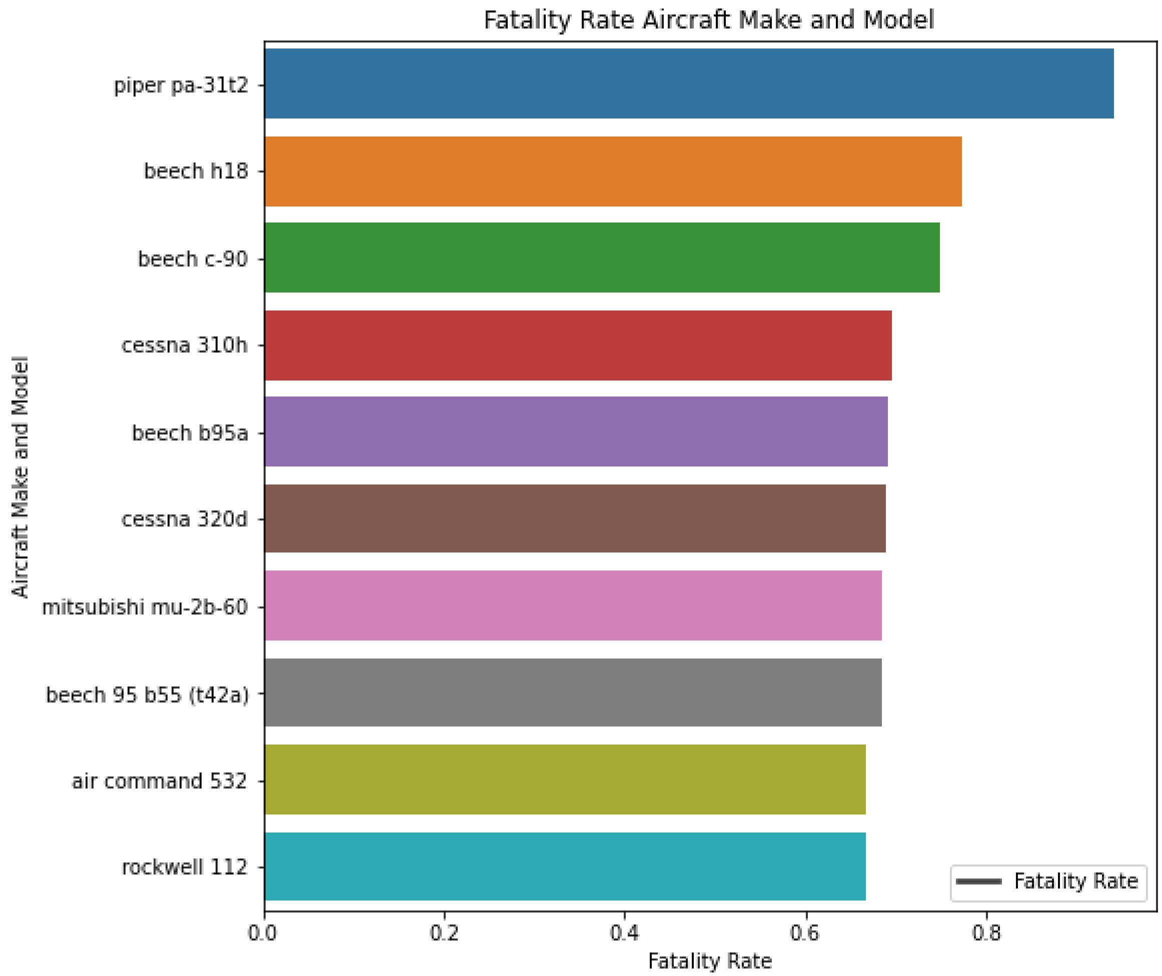
In [254…
```
top_make_model_stat_sorted= plot_df[plot_df['Event.Id'] > 5].head(10)
```

In [255…
```
top_make_model_stat_sorted
```

Out[255…

	Make	Model	Event.Id	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Total People	Fatality Rate	Injury Rates	mak
2738	piper	pa-31t2	6	16.0	0.0	0.0	1.0	17.0	0.941176	0.941176	
2768	beech	h18	8	17.0	0.0	2.0	3.0	22.0	0.772727	0.863636	b
2771	beech	c-90	7	12.0	0.0	0.0	4.0	16.0	0.750000	0.750000	be
2805	cessna	310h	13	23.0	3.0	2.0	5.0	33.0	0.696970	0.848485	ces
2807	beech	b95a	6	9.0	0.0	4.0	0.0	13.0	0.692308	1.000000	be
2808	cessna	320d	16	31.0	1.0	1.0	12.0	45.0	0.688889	0.733333	ces
2810	mitsubishi	mu-2b-60	31	61.0	8.0	2.0	18.0	89.0	0.685393	0.797753	n n
2811	beech	95 b55 (t42a)	7	13.0	0.0	0.0	6.0	19.0	0.684211	0.684211	b
2847	air command	532	6	4.0	1.0	0.0	1.0	6.0	0.666667	0.833333	air c
2922	rockwell	112	8	8.0	1.0	1.0	2.0	12.0	0.666667	0.833333	rocl

In [256…
```
# plotting
plt.figure(figsize=(8, 8))
sns.barplot(x= 'Fatality Rate',
           y= 'make_model',
           data= top_make_model_stat_sorted)
plt.title("Fatality Rate Aircraft Make and Model")
plt.xlabel('Fatality Rate')
plt.ylabel('Aircraft Make and Model')
plt.legend(labels=['Fatality Rate'], loc='lower right')
palete= "set2"
plt.show()
```



Observation

The above bar plot shows the top 10 aircraft make/model sorted using fatality rate according to the number of incident or accident occurence. and from that we can deduce that Piper PA-31T2 make model has the highest fatality rate, followed by BEECH H18 make model with 80% fatality rates, models like CESSNA 310H, BEECH B95A, CESSNA 320D, MITSUBISHI MU-2B-60 have moderate fatality and make model Air Command 532 and Rockell 112 Have the lowest fatality risks.

In [257…

```
# grouping by purpose of fight
purpose_stats = us_aviation_accidents_data.groupby("Purpose.of.flight").agg({'Event.Id': 'count',
                                                    'Total.Fatal.Injuries':'sum',
                                                    'Total.Serious.Injuries': 'sum',
                                                    'Total.Minor.Injuries': 'sum',
```

```
                                              'Total.Uninjured': 'sum',
                                              })
```

In [258…

```
purpose_stats
```

Out[258…

Purpose.of.flight	Event.Id	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
aerial application	4627	493.0	585.0	781.0	2925.0
aerial observation	707	291.0	287.0	296.0	845.0
air drop	8	1.0	4.0	0.0	10.0
air race show	82	27.0	14.0	18.0	63.0
air race/show	57	30.0	21.0	10.0	38.0
asho	6	14.0	1.0	0.0	1.0
banner tow	101	19.0	31.0	10.0	52.0
business	3843	2006.0	824.0	1052.0	6280.0
executive/corporate	509	384.0	119.0	177.0	1523.0
external load	112	33.0	25.0	33.0	62.0
ferry	729	176.0	102.0	197.0	580.0
firefighting	29	14.0	5.0	6.0	20.0
flight test	391	109.0	88.0	80.0	458.0
glider tow	52	15.0	11.0	7.0	32.0
instructional	10429	1693.0	1512.0	2031.0	12480.0
other work use	1192	374.0	376.0	526.0	1604.0
personal	48544	17453.0	10421.0	12730.0	51259.0
positioning	1566	574.0	221.0	286.0	2046.0
publ	1	0.0	0.0	0.0	2.0
public aircraft	685	309.0	182.0	237.0	1586.0
public aircraft - federal	98	33.0	20.0	27.0	245.0

Purpose.of.flight	Event.Id	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
public aircraft - local	74	13.0	49.0	19.0	96.0
public aircraft - state	63	23.0	21.0	26.0	65.0
pubs	4	0.0	0.0	2.0	5.0
skydiving	171	220.0	83.0	48.0	509.0
unknown	8168	5886.0	3225.0	5215.0	249548.0

In [259…

```
# calculating total people
purpose_stats["Total People"]= (purpose_stats['Total.Fatal.Injuries'] +
                             purpose_stats['Total.Serious.Injuries'] +
                             purpose_stats['Total.Minor.Injuries'] +
                             purpose_stats['Total.Uninjured'] )
```

In [260…

```
# calculating fatality rate
purpose_stats['Fatality Rate']= purpose_stats['Total.Fatal.Injuries'] / purpose_stats['Total People']
```

In [261…

```
# injury rate
purpose_stats['Injury Rates']= (purpose_stats['Total.Fatal.Injuries'] + purpose_stats['Total.Serious.Injuries'] +
                              purpose_stats['Total.Minor.Injuries'])/ purpose_stats["Total People"]
```

In [262…

```
purpose_stats.sort_values(by= 'Fatality Rate')
```

Out[262…

Purpose.of.flight	Event.Id	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Total People	Fatality Rate	Injury Rates
pubs	4	0.0	0.0	2.0	5.0	7.0	0.000000	0.285714
publ	1	0.0	0.0	0.0	2.0	2.0	0.000000	0.000000
unknown	8168	5886.0	3225.0	5215.0	249548.0	263874.0	0.022306	0.054291
air drop	8	1.0	4.0	0.0	10.0	15.0	0.066667	0.333333
public aircraft - local	74	13.0	49.0	19.0	96.0	177.0	0.073446	0.457627
instructional	10429	1693.0	1512.0	2031.0	12480.0	17716.0	0.095563	0.295552

Purpose.of.flight	Event.Id	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Total People	Fatality Rate	Injury Rates
public aircraft - federal	98	33.0	20.0	27.0	245.0	325.0	0.101538	0.246154
aerial application	4627	493.0	585.0	781.0	2925.0	4784.0	0.103052	0.388587
other work use	1192	374.0	376.0	526.0	1604.0	2880.0	0.129861	0.443056
public aircraft	685	309.0	182.0	237.0	1586.0	2314.0	0.133535	0.314607
flight test	391	109.0	88.0	80.0	458.0	735.0	0.148299	0.376871
ferry	729	176.0	102.0	197.0	580.0	1055.0	0.166825	0.450237
aerial observation	707	291.0	287.0	296.0	845.0	1719.0	0.169284	0.508435
banner tow	101	19.0	31.0	10.0	52.0	112.0	0.169643	0.535714
public aircraft - state	63	23.0	21.0	26.0	65.0	135.0	0.170370	0.518519
executive/corporate	509	384.0	119.0	177.0	1523.0	2203.0	0.174308	0.308670
positioning	1566	574.0	221.0	286.0	2046.0	3127.0	0.183563	0.345699
personal	48544	17453.0	10421.0	12730.0	51259.0	91863.0	0.189989	0.442006
business	3843	2006.0	824.0	1052.0	6280.0	10162.0	0.197402	0.382011
external load	112	33.0	25.0	33.0	62.0	153.0	0.215686	0.594771
air race show	82	27.0	14.0	18.0	63.0	122.0	0.221311	0.483607
glider tow	52	15.0	11.0	7.0	32.0	65.0	0.230769	0.507692
skydiving	171	220.0	83.0	48.0	509.0	860.0	0.255814	0.408140
air race/show	57	30.0	21.0	10.0	38.0	99.0	0.303030	0.616162
firefighting	29	14.0	5.0	6.0	20.0	45.0	0.311111	0.555556
asho	6	14.0	1.0	0.0	1.0	16.0	0.875000	0.937500

In [263...

```
purpose_stats= purpose_stats.sort_values(by='Fatality Rate', ascending= False).reset_index()
```

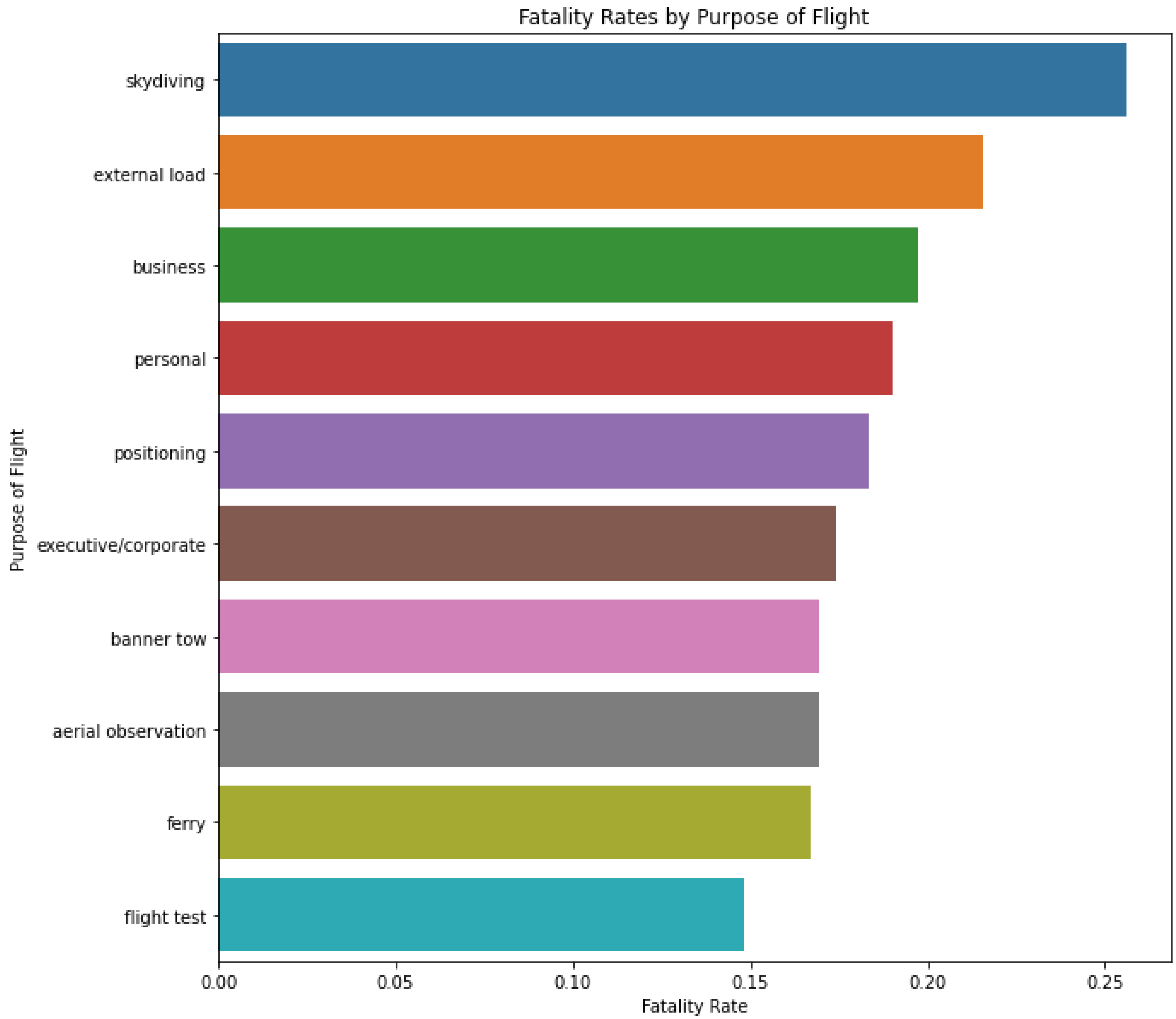
In [264...

```
top_purpose_stats= purpose_stats[purpose_stats["Event.Id"] > 100].head(10)
```

In [265…

```
# plotting
plt.figure(figsize=(10, 10))
sns.barplot(x= 'Fatality Rate',
           y= 'Purpose.of.flight',
           data= top_purpose_stats)
plt.title('Fatality Rates by Purpose of Flight')
plt.xlabel('Fatality Rate')
plt.ylabel('Purpose of Flight')
plt.show()

plt.show()
```



Observation

After selecting the activities by event counts of more than 100, purpose of flight with the highest fatality rate is Air race/show with fatality rate of about 30%, business and aerial observation purpose follows with fatality rate of about 25%.

The activities with moderate risks are public aircraft purpose and personal of less than 25%

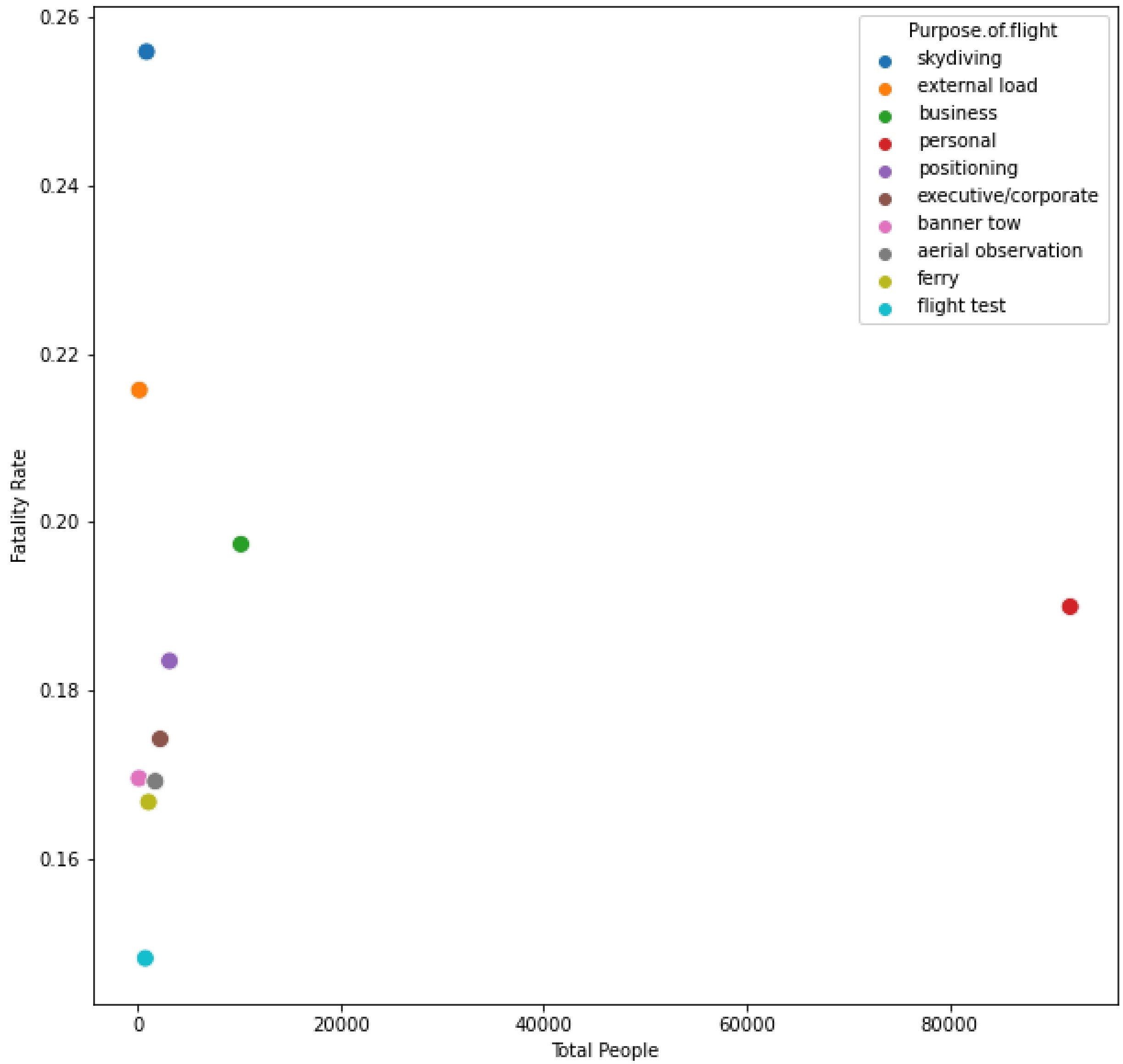
the flight purpose with lowest fatality rate is public aircraft-state with fatality rate of 17%

In [266…

```
# scatter plot showing total number of people, fatality rate per flight purpose
plt.figure(figsize=(10, 10))
sns.scatterplot(data= top_purpose_stats, x='Total People', y='Fatality Rate', hue= "Purpose.of.flight", alpha=1.0, s= 100
```

Out[266…

```
<AxesSubplot:xlabel='Total People', ylabel='Fatality Rate'>
```



In [267…

```
# analyzing risk by state
states_stats= us_aviation_accidents_data.groupby("State").agg({'Event.Id': 'count',
                                                    'Total.Fatal.Injuries':'sum',
                                                    'Total.Serious.Injuries': 'sum',
                                                    'Total.Minor.Injuries': 'sum',
                                                    'Total.Uninjured': 'sum',
                                                    }).rename(columns={'Event.Id': 'Incident Count'})
```

In [268…

```
# calculating total people
states_stats["Total People"]= (states_stats['Total.Fatal.Injuries'] +
                             states_stats['Total.Serious.Injuries'] +
                             states_stats['Total.Minor.Injuries'] +
                             states_stats['Total.Uninjured'] )
```

In [269…

```
# calculating risk factor per state
states_stats['Fatality Rate']= states_stats['Total.Fatal.Injuries'] / states_stats['Total People']
```

In [270…

```
states_stats= states_stats= states_stats.sort_values(by='Fatality Rate', ascending= False).reset_index()
```

In [271…

```
top_states = states_stats[states_stats['Incident Count'] >= 5].head(15)
```

In [272…

```
top_states
```

Out[272…

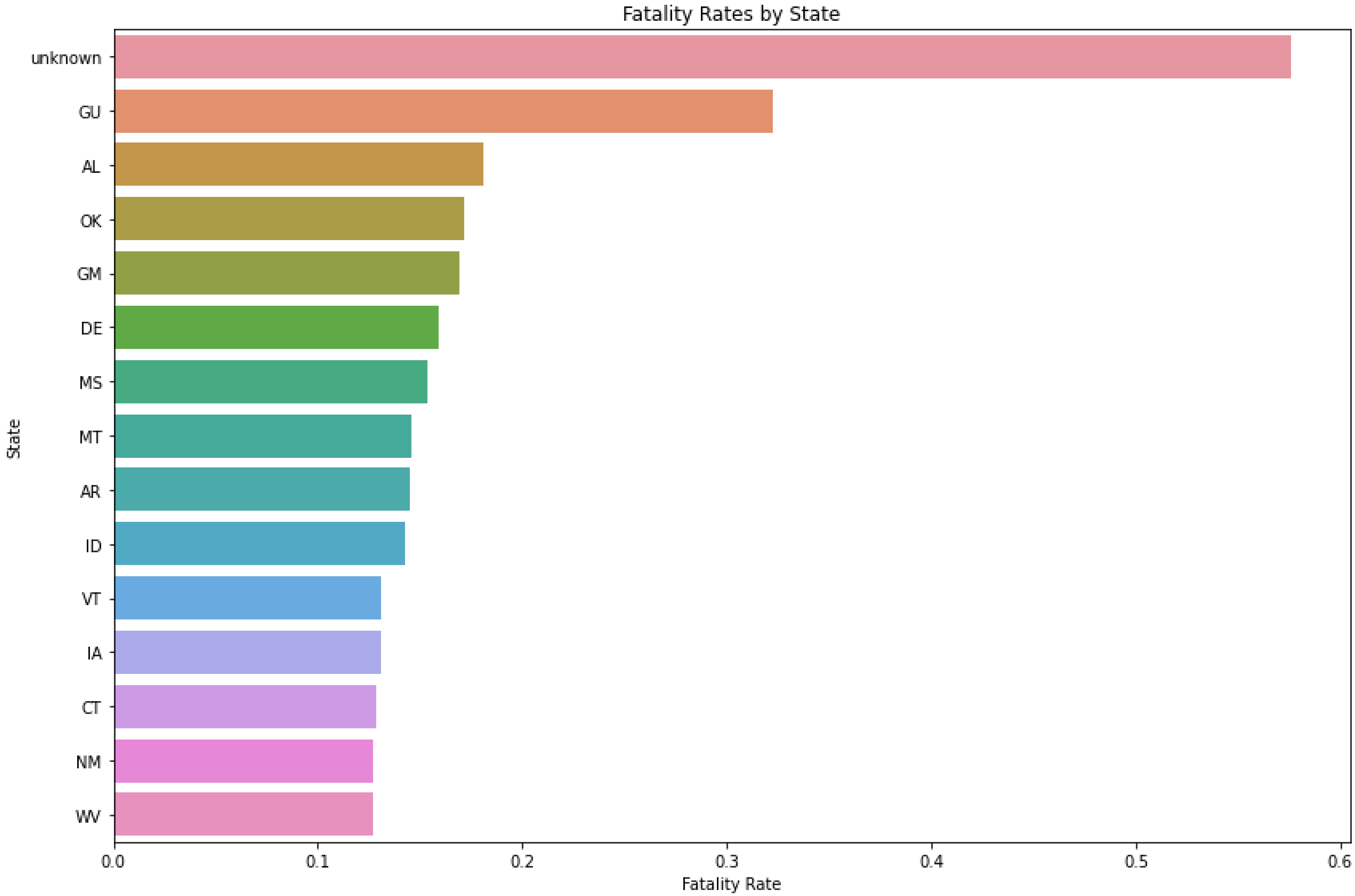
	State	Incident Count	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Total People	Fatality Rate
1	unknown	11	19.0	1.0	0.0	13.0	33.0	0.575758
2	GU	8	233.0	26.0	7.0	456.0	722.0	0.322715
3	AL	1153	475.0	239.0	259.0	1657.0	2630.0	0.180608
4	OK	1240	494.0	300.0	337.0	1748.0	2879.0	0.171587
5	GM	44	21.0	12.0	22.0	69.0	124.0	0.169355
6	DE	114	43.0	22.0	35.0	170.0	270.0	0.159259
7	MS	813	248.0	148.0	213.0	1005.0	1614.0	0.153656
8	MT	1050	363.0	186.0	215.0	1729.0	2493.0	0.145608
9	AR	1519	470.0	324.0	398.0	2043.0	3235.0	0.145286
10	ID	1436	468.0	289.0	299.0	2232.0	3288.0	0.142336

	State	Incident Count	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Total People	Fatality Rate
11	VT	241	89.0	64.0	66.0	461.0	680.0	0.130882
12	IA	819	340.0	217.0	333.0	1711.0	2601.0	0.130719
13	CT	502	172.0	105.0	140.0	919.0	1336.0	0.128743
14	NM	1358	485.0	366.0	424.0	2538.0	3813.0	0.127196
15	WV	394	190.0	144.0	129.0	1031.0	1494.0	0.127175

In [273…

```
# plotting
plt.figure(figsize=(12, 8))
sns.barplot(x= "Fatality Rate", y= 'State', data= top_states)

plt.title('Fatality Rates by State')
plt.xlabel('Fatality Rate')
plt.ylabel('State')
plt.tight_layout()
palette= "Reds r"
```



The bar at the top indicates the state with the highest fatality rate, and in this case those states are unknown probably because the states records were not recorded during incident or accident occurrence.

The top state with higher fatality rate then the rest is Guam, it is a USA state territory and its leading in fatality rates.

The middle states like Arizona have moderate fatality rates as compared to the ones at the top. Connecticut, New Mexico, West Virginia have less than 15% fatality rate which makes it ideal for aviation operations.

In [274…
```
us_code_data= pd.read_csv("data/USState_Codes.csv", encoding= "latin1")
```

In [275…
```
us_aviation_accidents_data = us_aviation_accidents_data.merge(us_code_data, how='left',
                                                              left_on='State',       # column in aviation data
                                                              right_on='Abbreviation'  # column in state code data
)
```

In [276…
```
us_aviation_accidents_data
```

Out[276…

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name
0	20001218x45444	accident	sea87la080	1948-10-24	moose creek, id	united states	NaN	NaN	NaN	NaN
1	20001218x45447	accident	lax94la336	1962-07-19	bridgeport, ca	united states	NaN	NaN	NaN	NaN
2	20061025x01555	accident	nyc07la005	1974-08-30	saltville, va	united states	NaN	NaN	NaN	NaN
3	20001218x45448	accident	lax96la321	1977-06-19	eureka, ca	united states	NaN	NaN	NaN	NaN
4	20041105x01764	accident	chi79fa064	1979-08-02	canton, oh	united states	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
82243	20221227106491	accident	era23la093	2022-12-26	annapolis, md	united states	NaN	NaN	NaN	NaN
82244	20221227106494	accident	era23la095	2022-12-26	hampton, nh	united states	NaN	NaN	NaN	NaN
82245	20221227106497	accident	wpr23la075	2022-12-26	payson, az	united states	341525n	1112021w	pan	payson
82246	20221227106498	accident	wpr23la076	2022-12-26	morgan, ut	united states	NaN	NaN	NaN	NaN
82247	20221230106513	accident	era23la097	2022-12-29	athens, ga	united states	NaN	NaN	NaN	NaN

82248 rows × 34 columns

In [277…

```
# saving csv file
us_aviation_accidents_data.to_csv("us_aviation_accidents_data", index= False)
```

Conclusion

Based on the above analysis: 1) make/model safe are Air Command 532 and Rockell 112 and models like CESSNA 310H, BEECH B95A, CESSNA 320D, MITSUBISHI MU-2B-60 have moderate fatality rates. Air Command 532 and Rockell 112 are safe aircrafts models for private and commercial business.

2) Upon Evaluation of risks associated with the purpose of flight, air race/show emerged as the activity with highest fatality rate, this activity should be therefore be avoided when the company ventures into the new aviation business

3) On states operational risks, states with high fatality rates are unknown, this requires improvement on data collection when the incidents occur to inform in decision making