Aircraft Safety Risk Analysis

This analysis aims to assess the safety of different aircraft makes based on historical aviation accident data. The goal is to identify which aircraft manufacturers have the highest and lowest risk profiles to guide safer investments and operational choices. Using a structured and analytical approach, we processed and enriched the dataset to derive meaningful safety metrics.

Process

Data Cleaning & Preparation

- `Loaded the historical aircraft accident dataset from a CSV file.`
- `Filtered out irrelevant or incomplete records.`
- `Handled missing values and ensured key numerical fields (e.g. fatalities, i`
- `Combined with a secondary dataset containing calculated scores for deeper a`

Feature Engineering

- `Introduced new columns for:`
- `Survival Rate = Total Non-Injured / Total Aboard`
- `Injury Rate = (Total Serious Injuries + Total Minor Injuries) / Total Aboar`
- `Risk Score = Combined metric based on fatalities, injuries, and survival ra`

Visualization and Insights

- `Created visualizations using seaborn and Tableau:`
- `Bar charts of aircraft makes by risk score.`
- `Correlations between survival and injury rates.`
- `Map visualizations of crash data.`
- `Time series breakdown by year of occurrence.`

Decision Making

- `Identified least and most risky aircraft makes.`
- `Flagged high-risk vs low-risk aircraft based on risk scoring.`

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


#Open the CSV data file from Kaggle
ntsb_data = pd.read_csv("NTSB_database.csv")
ntsb_data.head(5)
```

	Event Id	Investigation Type	Country	Aircraft Damage	Aircraft Category	Make	Model	Am
0	20001218X45444	Accident	United States	Destroyed	fixed wing single engine	stinson	108-3	
1	20001218X45447	Accident	United States	Destroyed	weight-shift-control	piper	pa24-180	
2	20061025X01555	Accident	United States	Destroyed	fixed wing single engine	cessna	172m	
3	20001218X45448	Accident	United States	Destroyed	weight-shift-control	rockwell	112	
4	20041105X01764	Accident	United States	Destroyed	fixed wing multi engine	cessna	501	

```
ntsb_data.shape
```

```
(87951, 45)
```

```
ntsb_data.head()
```

	Event Id	Investigation Type	Country	Aircraft Damage	Aircraft Category	Make	Model	Am
0	20001218X45444	Accident	United States	Destroyed	fixed wing single engine	stinson	108-3	
1	20001218X45447	Accident	United States	Destroyed	weight-shift-control	piper	pa24-180	
2	20061025X01555	Accident	United States	Destroyed	fixed wing single engine	cessna	172m	
3	20001218X45448	Accident	United States	Destroyed	weight-shift-control	rockwell	112	
4	20041105X01764	Accident	United States	Destroyed	fixed wing multi engine	cessna	501	

```
ntsb_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87951 entries, 0 to 87950
Data columns (total 45 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Event Id                    87951 non-null  object
 1   Investigation Type          87951 non-null  object
 2   Country                     87951 non-null  object
 3   Aircraft Damage             87951 non-null  object
 4   Aircraft Category           87951 non-null  object
 5   Make                        87951 non-null  object
 6   Model                       87951 non-null  object
 7   Amateur Built               87951 non-null  object
 8   Number Of Engines           87951 non-null  int64
 9   Engine Type                 87951 non-null  object
 10  Far Description             87951 non-null  object
 11  Schedule                    87951 non-null  object
 12  Purpose Of Flight           87951 non-null  object
 13  Total Fatal Injuries        87951 non-null  int64
 14  Total Serious Injuries      87951 non-null  int64
 15  Total Minor Injuries        87951 non-null  int64
 16  Total Uninjured             87951 non-null  int64
 17  Weather Condition           87951 non-null  object
 18  Broad Phase Of Flight       87951 non-null  object
 19  Analysis                    87951 non-null  object
 20  City                        87939 non-null  object
 21  Longitude                   87951 non-null  float64
 22  Latitude                    87951 non-null  float64
 23  Address                     87951 non-null  object
 24  geometry                    87951 non-null  object
 25  Place                       87951 non-null  object
 26  Number Of Seats             87951 non-null  int64
 27  Type Aircraft               87951 non-null  int64
```

```
 28  Type Engine                    87951 non-null  int64
 29  Total Person                   87951 non-null  int64
 30  Far Description Factorized     87951 non-null  int64
 31  Schedule Factorized            87951 non-null  int64
 32  Purpose Of Flight Factorized   87951 non-null  int64
 33  Make Factorized                87951 non-null  int64
 34  Model Factorized               87951 non-null  int64
 35  Event Year                     87951 non-null  int64
 36  Publication Year               87951 non-null  int64
 37  Event Month                    87951 non-null  int64
 38  Publication Month              87951 non-null  int64
 39  Event Day                      87951 non-null  int64
 40  Publication Day                87951 non-null  float64
 41  Date Difference                87951 non-null  int64
 42  Publication Month Name         87951 non-null  object
 43  Event Month Name               87951 non-null  object
 44  Season                         87951 non-null  object
dtypes: float64(3), int64(20), object(22)
memory usage: 30.2+ MB
```

```
ntsb_data.describe(include="all")
```

	Event Id	Investigation Type	Country	Aircraft Damage	Aircraft Category	Make	Model
count	87951	87951	87951	87951	87951	87951	87951
unique	87951	2	219	14	21	7552	11563
top	20001218X45444	Accident	United States	Substantial	fixed wing single engine	cessna	152
freq	1	84190	81355	66154	30655	26839	2313
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN

11 rows × 45 columns

```
#Check for duplicates
ntsb_data.duplicated()
```

```
0        False
1        False
2        False
```

```
3        False
4        False
         ...
87946    False
87947    False
87948    False
87949    False
87950    False
Length: 87951, dtype: bool
```

```
# Confirm Date column types
ntsb_data[['Event Year', 'Event Month', 'Event Year']].dtypes
```

```
Event Year     int64
Event Month    int64
Event Year     int64
dtype: object
```

```
#Create an event date column for later use
ntsb_data['Event Date'] = pd.to_datetime(
    ntsb_data[['Event Year', 'Event Month', 'Event Day']].rename(
        columns={'Event Year': 'year', 'Event Month': 'month', 'Event Day': 'day'
    ),
    errors='coerce' #Create NaN types in case of non-matching data
)
```

```
#Check the new date field
ntsb_data.head()
```

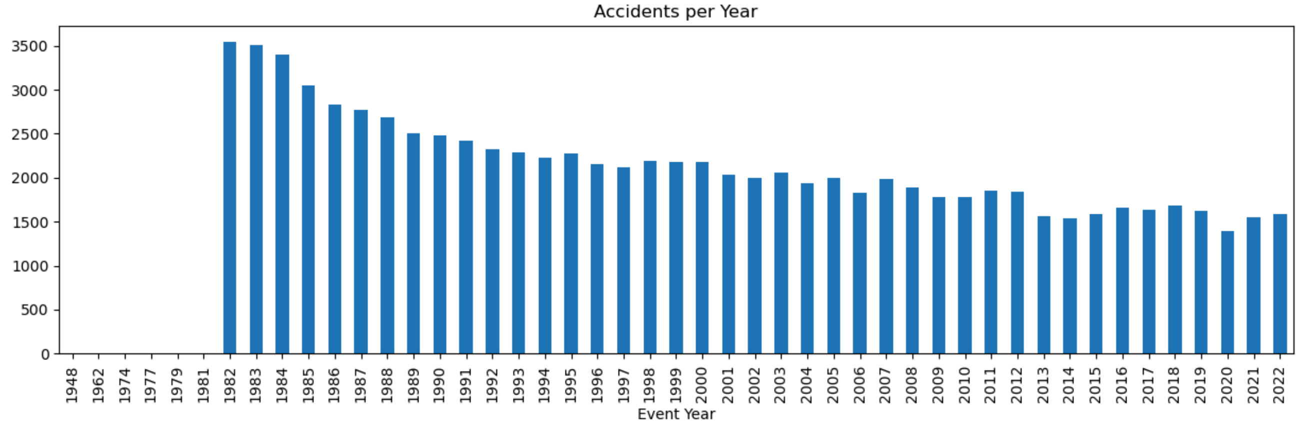
	Event Id	Investigation Type	Country	Aircraft Damage	Aircraft Category	Make	Model	Am
0	20001218X45444	Accident	United States	Destroyed	fixed wing single engine	stinson	108-3	
1	20001218X45447	Accident	United States	Destroyed	weight-shift-control	piper	pa24-180	
2	20061025X01555	Accident	United States	Destroyed	fixed wing single engine	cessna	172m	
3	20001218X45448	Accident	United States	Destroyed	weight-shift-control	rockwell	112	
4	20041105X01764	Accident	United States	Destroyed	fixed wing multi engine	cessna	501	

```
ntsb_data['Event Date'].info()
#Confirm date has no Nan values
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 87951 entries, 0 to 87950
Series name: Event Date
Non-Null Count  Dtype
--------------  -----
87951 non-null  datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 687.2 KB
```

```
#Check if the dates have missing values or oddly placed i.e not in chronolical or
ntsb_data['Event Year'].value_counts().sort_index().plot(kind='bar', figsize=(15,
```

```
<Axes: title={'center': 'Accidents per Year'}, xlabel='Event Year'>
```




```
#Check accidents between 1948 to 1981 and decide whether to keep them, flag them
early_accidents = ntsb_data[(ntsb_data['Event Year'] >= 1948) & (ntsb_data['Event
early_accidents.head(10)
```

	Event Id	Investigation Type	Country	Aircraft Damage	Aircraft Category	Make	Model
0	20001218X45444	Accident	United States	Destroyed	fixed wing single engine	stinson	108-3
1	20001218X45447	Accident	United States	Destroyed	weight-shift-control	piper	pa24-180
2	20061025X01555	Accident	United States	Destroyed	fixed wing single engine	cessna	172m
3	20001218X45448	Accident	United States	Destroyed	weight-shift-control	rockwell	112
4	20041105X01764	Accident	United States	Destroyed	fixed wing multi engine	cessna	501
5	20170710X52551	Accident	United States	Substantial	airplane	mcdonnell douglas	dc9
6	20001218X45446	Accident	United States	Destroyed	fixed wing single engine	cessna	180

7 rows × 46 columns

```
early_accidents['Make'].value_counts()
```

```
Make
cessna              3
stinson             1
piper               1
rockwell            1
mcdonnell douglas   1
Name: count, dtype: int64
```

```
early_accidents['Type Aircraft'].value_counts()
```

```
Type Aircraft
4     3
7     2
5     1
12    1
Name: count, dtype: int64
```

```
#Confirm whether the aircraft types are still in use
modern_accidents = ntsb_data[(ntsb_data['Event Year'] > 1981)] #Create later acci

#Create unique values
```

```
early_make = set(early_accidents['Make'].dropna().unique())
modern_make = set(modern_accidents['Make'].dropna().unique())

#Check if the makes in the early group were discontinued
common_makes = early_make & modern_make
common_makes
```

```
{'cessna', 'mcdonnell douglas', 'piper', 'rockwell', 'stinson'}
```

We'll keep the early ones in the analysis for no since they're still in use

```
#Checking which aircraft has the most accidents. Note that this is before removin
ntsb_data['Type Aircraft'].describe()
```

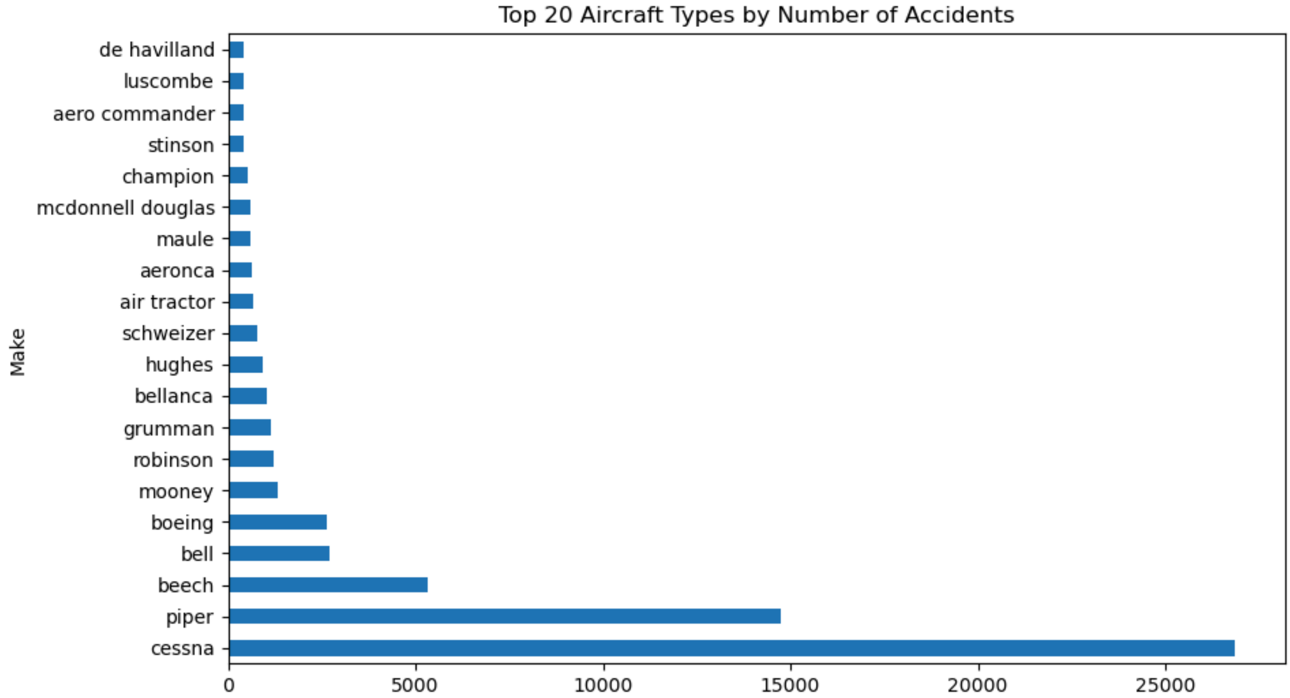
```
count    86653.000000
mean         7.443643
std          3.552684
min          1.000000
25%          4.000000
50%          7.000000
75%         12.000000
max         21.000000
Name: Type Aircraft, dtype: float64
```

```
ntsb_data['Make'].describe()
```

```
count      87951
unique      7552
top       cessna
freq       26839
Name: Make, dtype: object
```

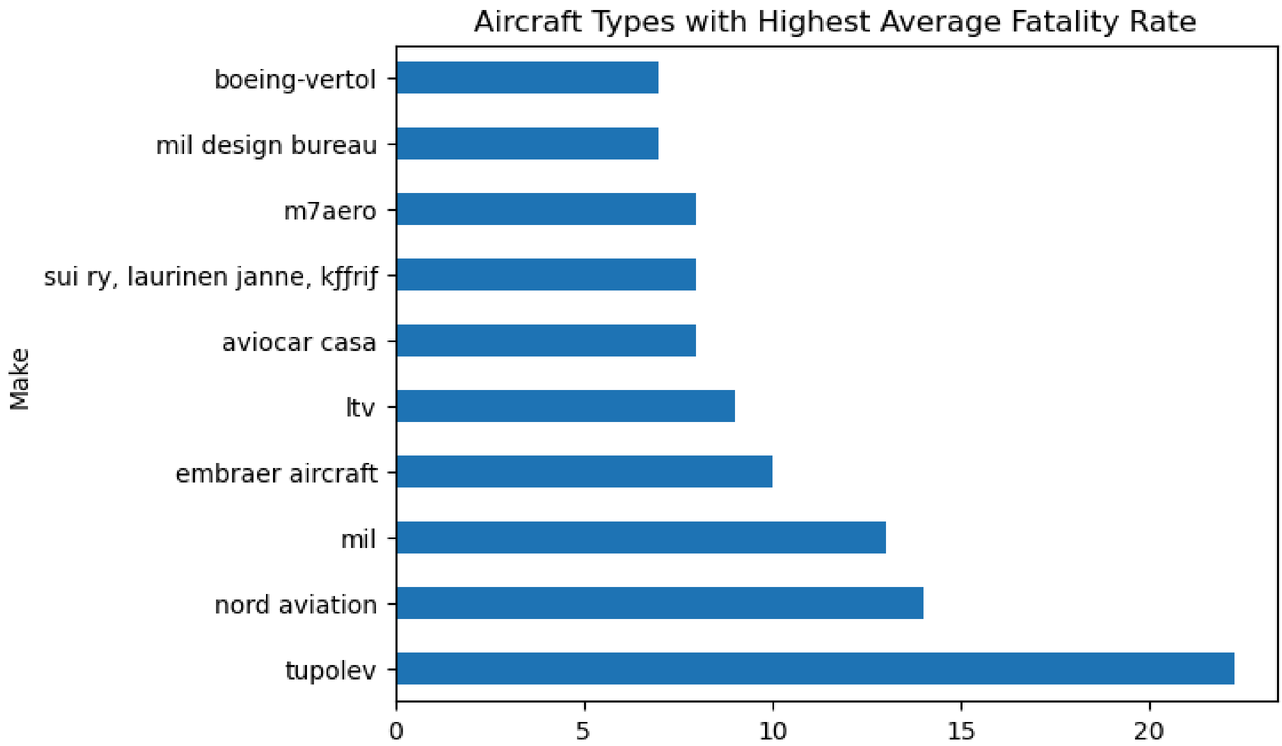
```
aircraft_make_counts = ntsb_data['Make'].value_counts().head(20)
aircraft_make_counts.plot(kind='barh', figsize=(10,6), title='Top 20 Aircraft Typ
```

```
<Axes: title={'center': 'Top 20 Aircraft Types by Number of Accidents'}, ylabel='Make'>
```




```
#Check fatality per aircraft make
fatality_by_make = ntsb_data.groupby('Make')['Total Fatal Injuries'].mean().sort_
fatality_by_make.head(10).plot(kind='barh', title='Aircraft Types with Highest Av
```

```
<Axes: title={'center': 'Aircraft Types with Highest Average Fatality Rate'},
ylabel='Make'>
```




```
ntsb_data.head(3)
```

	Event Id	Investigation Type	Country	Aircraft Damage	Aircraft Category	Make	Model	Ama B
0	20001218X45444	Accident	United States	Destroyed	fixed wing single engine	stinson	108-3	
1	20001218X45447	Accident	United States	Destroyed	weight-shift-control	piper	pa24-180	
2	20061025X01555	Accident	United States	Destroyed	fixed wing single engine	cessna	172m	

```
ntsb_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87951 entries, 0 to 87950
Data columns (total 46 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Event Id                   87951 non-null  object
 1   Investigation Type         87951 non-null  object
 2   Country                    87951 non-null  object
 3   Aircraft Damage            87951 non-null  object
 4   Aircraft Category          87951 non-null  object
```

```
 5   Make                           87951 non-null  object
 6   Model                          87951 non-null  object
 7   Amateur Built                  87951 non-null  object
 8   Number Of Engines              87951 non-null  int64
 9   Engine Type                    87951 non-null  object
 10  Far Description                87951 non-null  object
 11  Schedule                       87951 non-null  object
 12  Purpose Of Flight              87951 non-null  object
 13  Total Fatal Injuries           87951 non-null  int64
 14  Total Serious Injuries         87951 non-null  int64
 15  Total Minor Injuries           87951 non-null  int64
 16  Total Uninjured                87951 non-null  int64
 17  Weather Condition              87951 non-null  object
 18  Broad Phase Of Flight          87951 non-null  object
 19  Analysis                       87951 non-null  object
 20  City                           87939 non-null  object
 21  Longitude                      87951 non-null  float64
 22  Latitude                       87951 non-null  float64
 23  Address                        87951 non-null  object
 24  geometry                       87951 non-null  object
 25  Place                          87951 non-null  object
 26  Number Of Seats                87951 non-null  int64
 27  Type Aircraft                  87951 non-null  int64
 28  Type Engine                    87951 non-null  int64
 29  Total Person                   87951 non-null  int64
 30  Far Description Factorized     87951 non-null  int64
 31  Schedule Factorized            87951 non-null  int64
 32  Purpose Of Flight Factorized   87951 non-null  int64
 33  Make Factorized                87951 non-null  int64
 34  Model Factorized               87951 non-null  int64
 35  Event Year                     87951 non-null  int64
 36  Publication Year               87951 non-null  int64
 37  Event Month                    87951 non-null  int64
 38  Publication Month              87951 non-null  int64
 39  Event Day                      87951 non-null  int64
 40  Publication Day                87951 non-null  float64
 41  Date Difference                87951 non-null  int64
 42  Publication Month Name         87951 non-null  object
 43  Event Month Name               87951 non-null  object
 44  Season                         87951 non-null  object
 45  Event Date                     87951 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(20), object(22)
memory usage: 30.9+ MB
```

We need to check the correlation of accidents and the weather conditions

```
ntsb_data['Weather Condition'].unique()
```

```
array(['UNK', 'IMC', 'VMC'], dtype=object)
```

The weather conditions are split into three with the following meanings: UNK - Unknown weather conditions IMC - Instrument Meteorological Condition (Poor weather conditions i.e Pilot has to use instruments for navigation) VMC - Visual Meteorological Conditions (Good weather conditions i.e Pilot can use normal visual references)

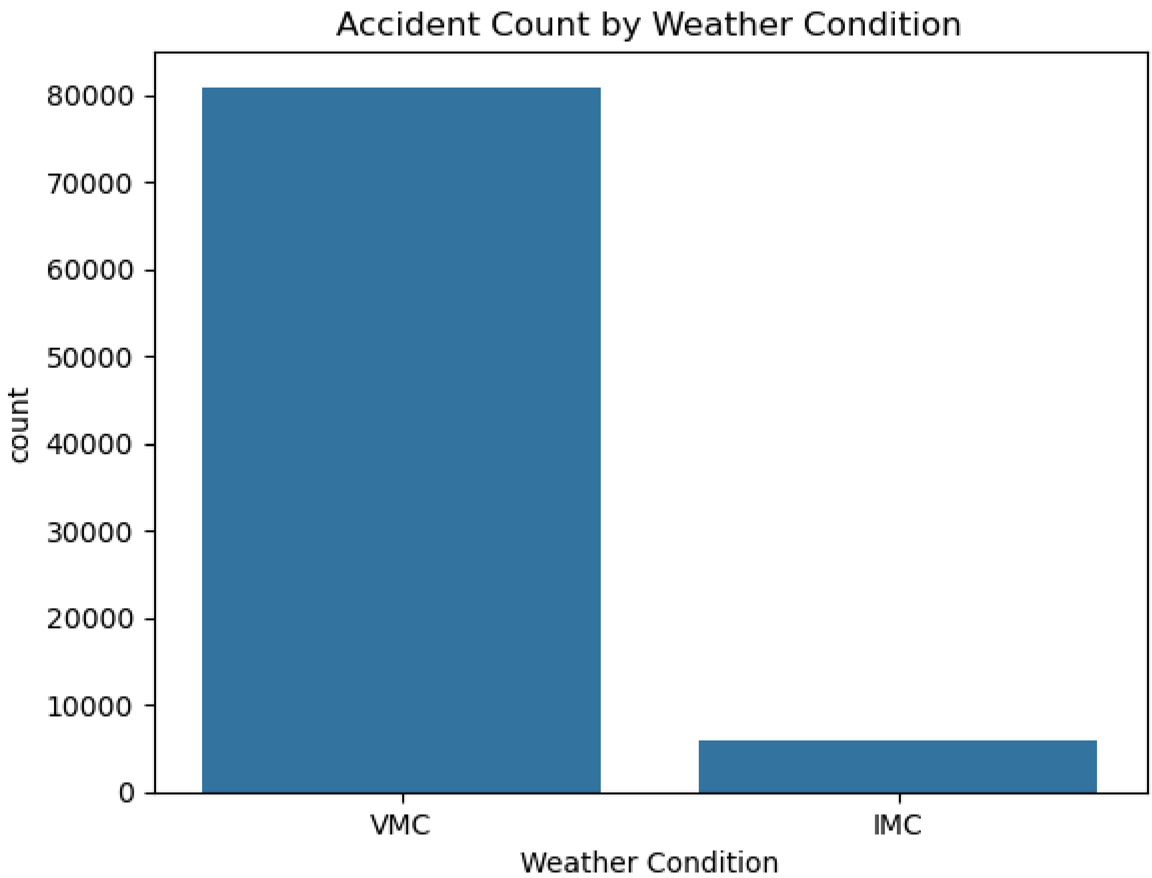
```
# Remove rows with unknown weather conditions
data_weather = ntsb_data[ntsb_data['Weather Condition'].isin(['IMC', 'VMC'])]
data_weather['Weather Condition'].unique()
```

```
array(['IMC', 'VMC'], dtype=object)
```

```
# Count total accidents under each weather condition
accidents_by_weather = data_weather['Weather Condition'].value_counts()
accidents_by_weather
```

```
Weather Condition
VMC    80890
IMC     5949
Name: count, dtype: int64
```

```
#Show the relationship between weather condition and accident count
sns.countplot(data=data_weather, x='Weather Condition', order=['VMC', 'IMC'])
plt.title('Accident Count by Weather Condition')
plt.show()
```



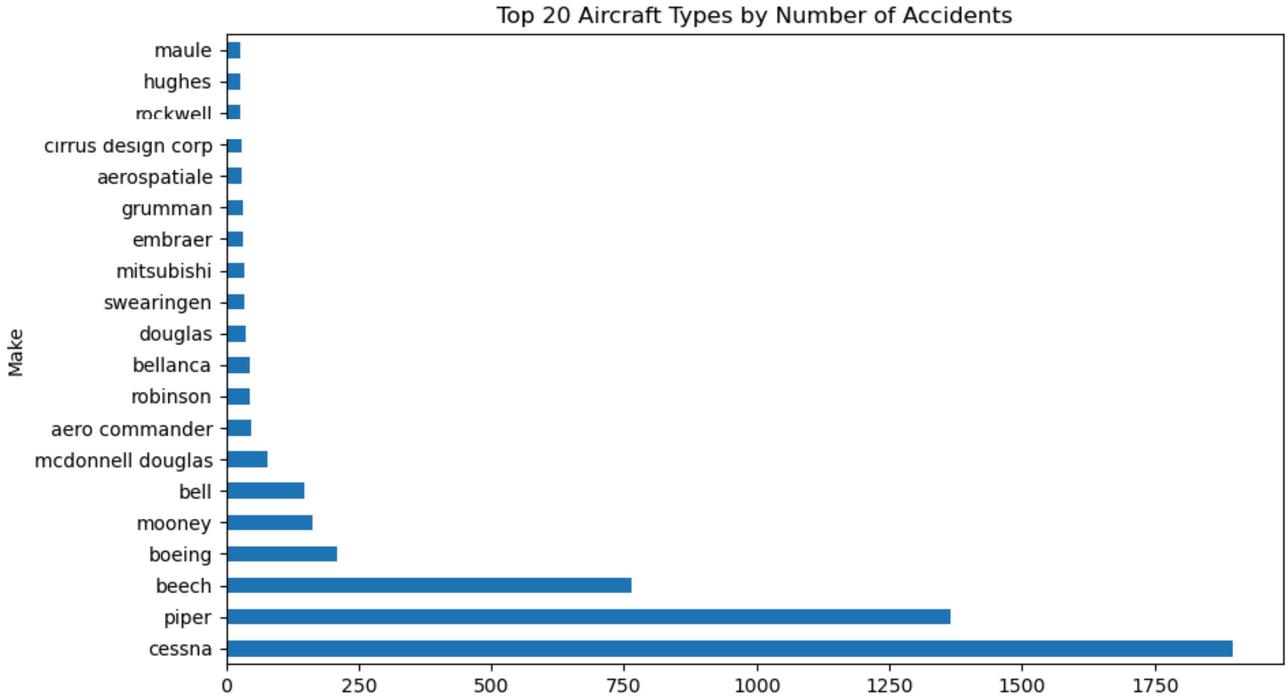
Check which make is mostly affected by bad weather conditions

```
# Barplot of IMC Accidents by Aircraft Make
poor_weather = ntsb_data[ntsb_data['Weather Condition'] == 'IMC']
```

```
imc_make_counts = poor_weather['Make'].value_counts().head(20)
```

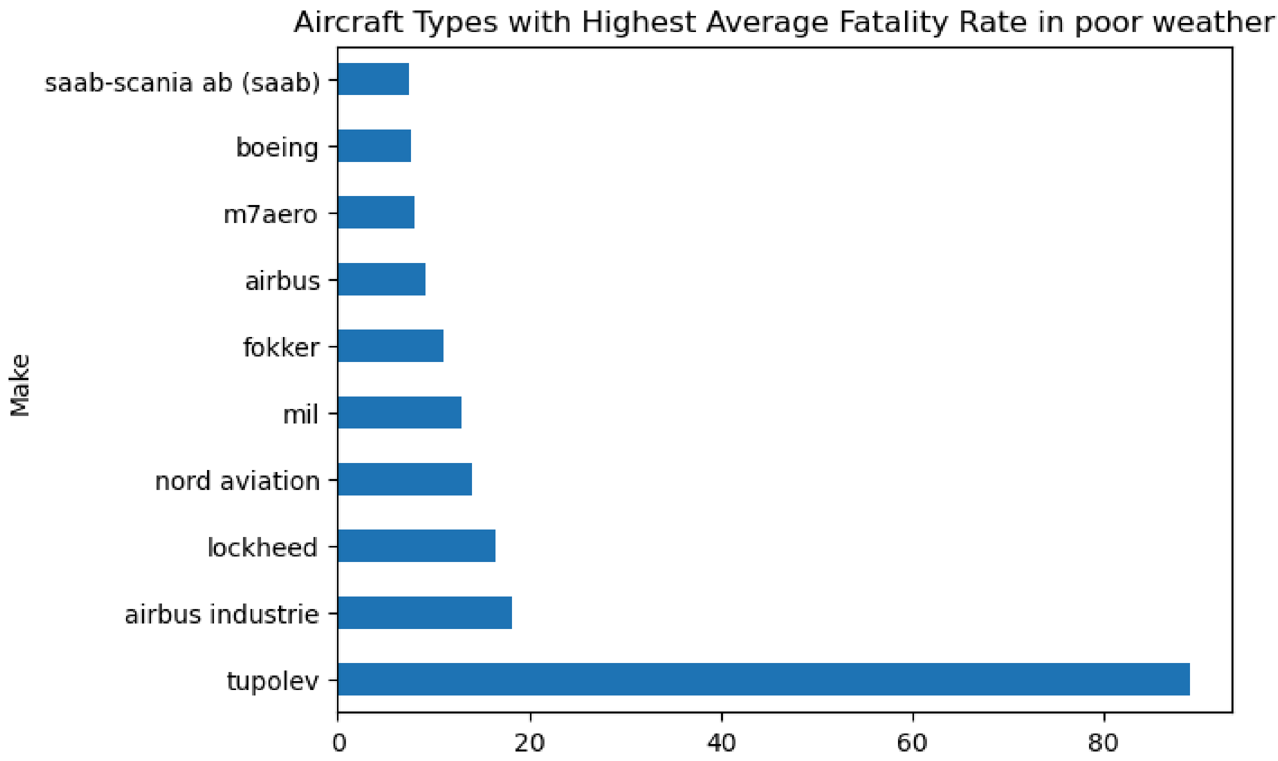
```
imc_make_counts.plot(kind='barh', figsize=(10,6), title='Top 20 Aircraft Types by
```

```
<Axes: title={'center': 'Top 20 Aircraft Types by Number of Accidents'},
ylabel='Make'>
```




```
# Barplot of Fatalities per Aircraft make in poor weather
fatality_by_make = poor_weather.groupby('Make')['Total Fatal Injuries'].mean().so
fatality_by_make.head(10).plot(kind='barh', title='Aircraft Types with Highest Av
```

```
<Axes: title={'center': 'Aircraft Types with Highest Average Fatality Rate in poor weather'}, ylabel='Make'>
```

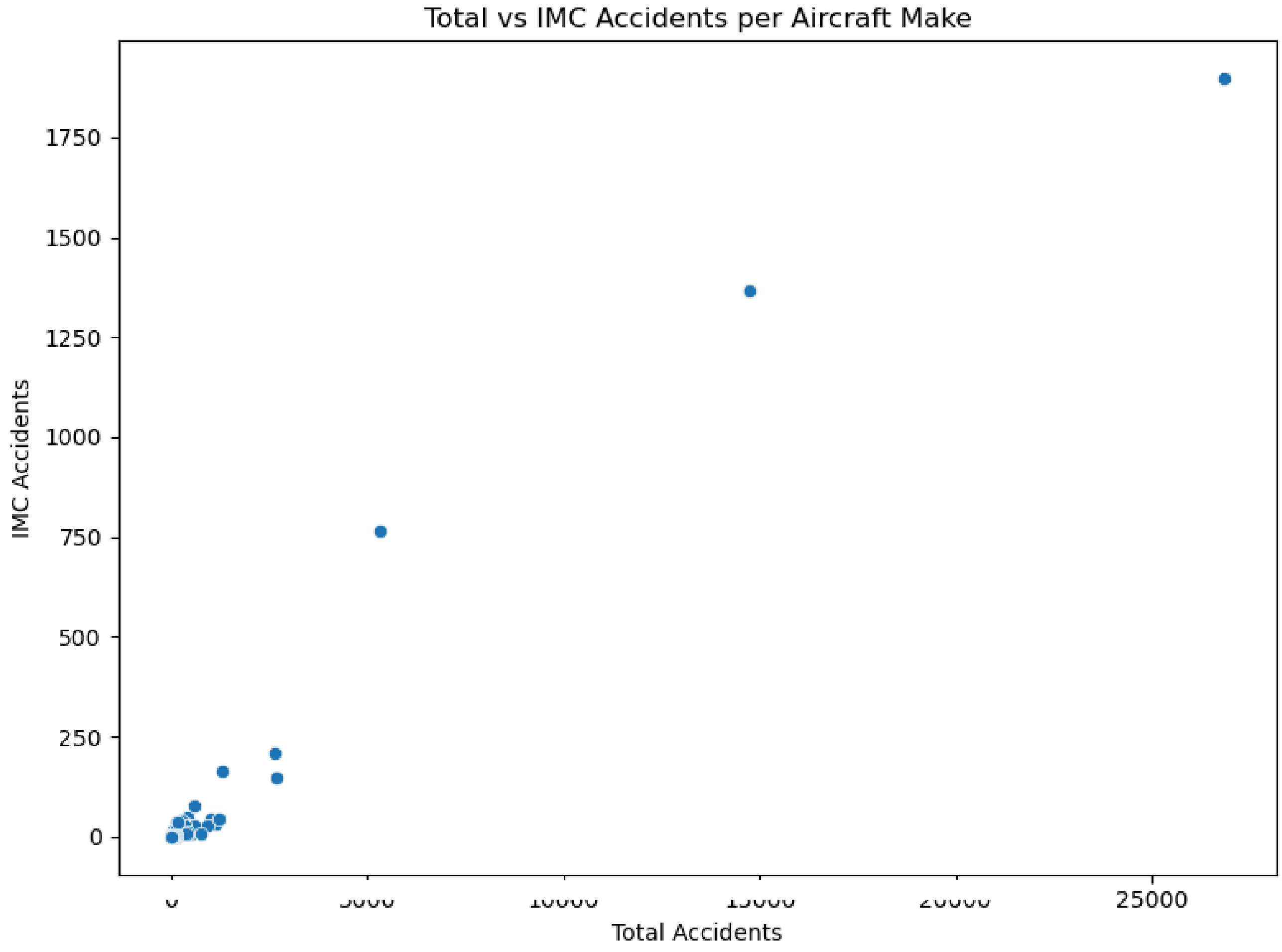


check if there's a direct correlation between total accidents per aircraft make and accidents in poor weather (IMC) per make,

```
# Total accidents per make
total_accidents = ntsb_data['Make'].value_counts()

# IMC accidents per make (excluding UNK and VMC)
imc_accidents = poor_weather['Make'].value_counts()

# Combine both into one DataFrame
accident_comparison = pd.DataFrame({
    'Total_Accidents': total_accidents,
    'IMC_Accidents': imc_accidents
}).fillna(0)

#correlation calculation
correlation = accident_comparison['Total_Accidents'].corr(accident_comparison['IM
print(f"Correlation between total accidents and IMC accidents per make: {correlat

plt.figure(figsize=(8,6))
sns.scatterplot(
    data=accident_comparison,
    x='Total_Accidents',
    y='IMC_Accidents'
)
plt.title("Total vs IMC Accidents per Aircraft Make")
plt.xlabel("Total Accidents")
plt.ylabel("IMC Accidents")
```

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

```
Correlation between total accidents and IMC accidents per make: 0.98
```



Conclusion

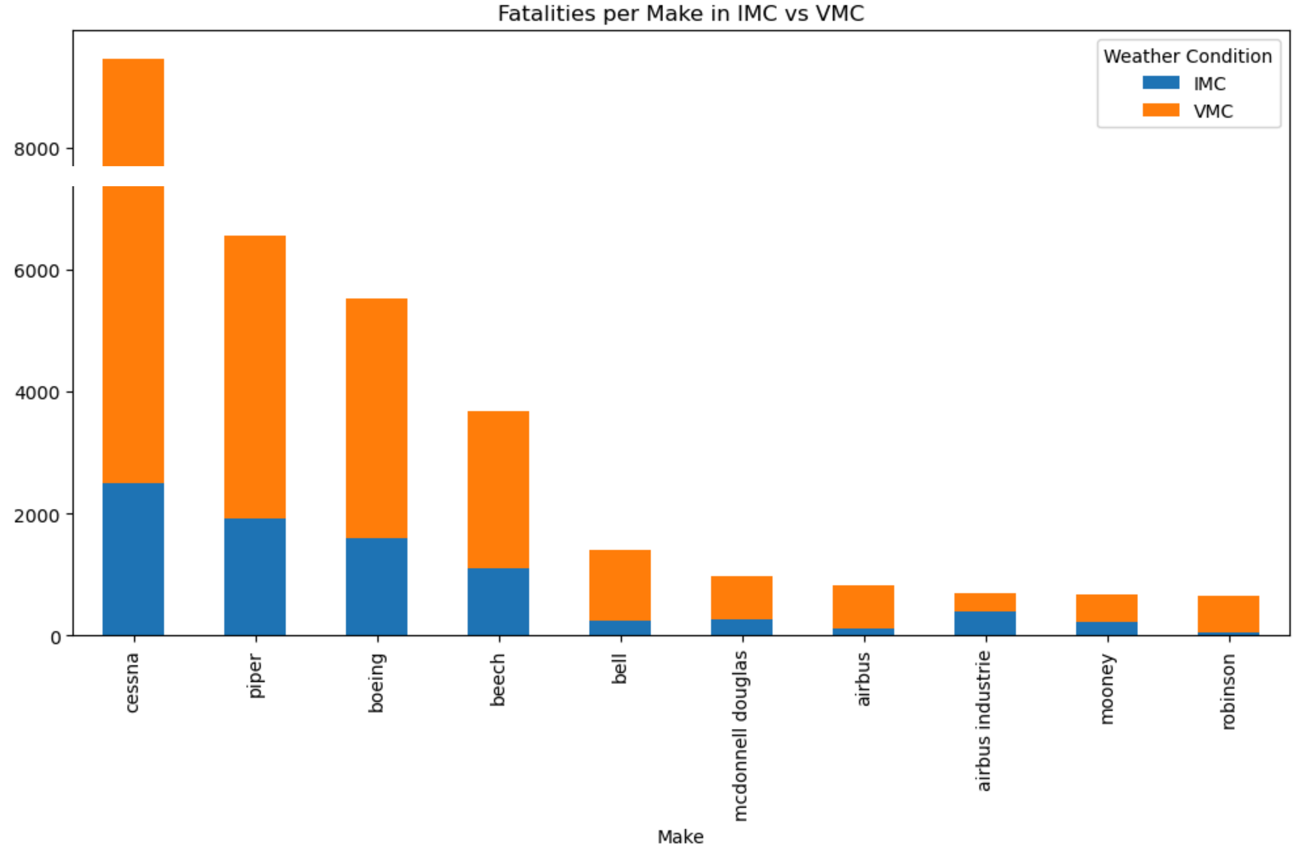
Aircraft makes with more total accidents tend to also have more accidents in IMC.The distribution of weather-related accidents scales proportionally with overall accident frequency across makes.This suggests no particular make is disproportionately more vulnerable to IMC relative to its overall accident count.

Check the Fatalities per make in IMC vs all fatalities per make

```
# Group by make and weather condition
fatalities_by_make_weather = data_weather.groupby(['Make', 'Weather Condition'])[

# Add a total column
fatalities_by_make_weather['Total'] = fatalities_by_make_weather.sum(axis=1)
```

```
# Sort by total fatalities
fatalities_by_make_weather = fatalities_by_make_weather.sort_values('Total', asce

# Optional: plot top 10 makes
fatalities_by_make_weather.head(10)[['IMC', 'VMC']].plot(kind='bar', stacked=True
```

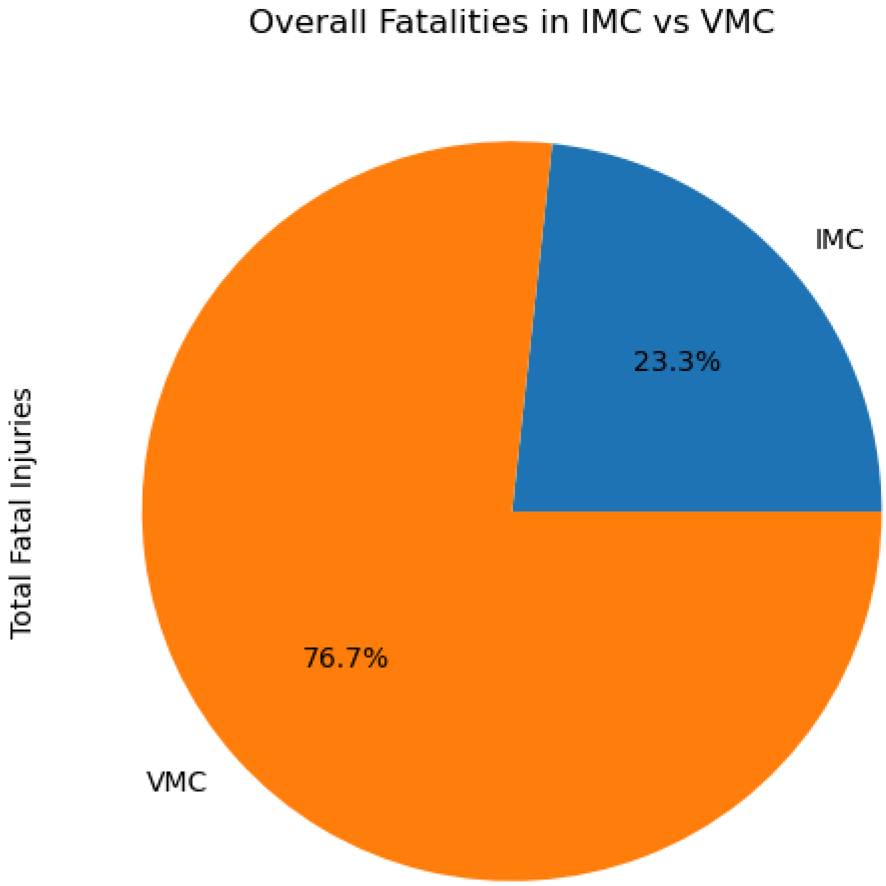
```
<Axes: title={'center': 'Fatalities per Make in IMC vs VMC'}, xlabel='Make'>
```



Airbus industries seems to have the most fatalities by bad weather conditions and Airbus has the least fatalities in poor weather conditions

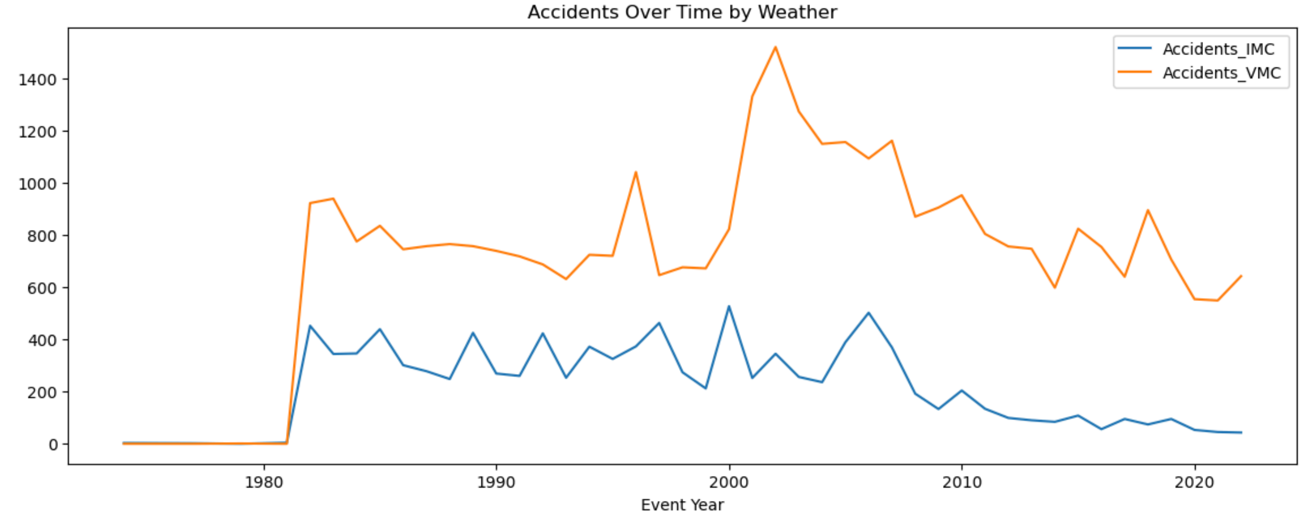
```
# Total fatalities by weather
overall_fatalities_by_weather = data_weather.groupby('Weather Condition')['Total
print(overall_fatalities_by_weather)

# Pie chart or bar plot
overall_fatalities_by_weather.plot(kind='pie', autopct='%1.1f%%', title='Overall
```

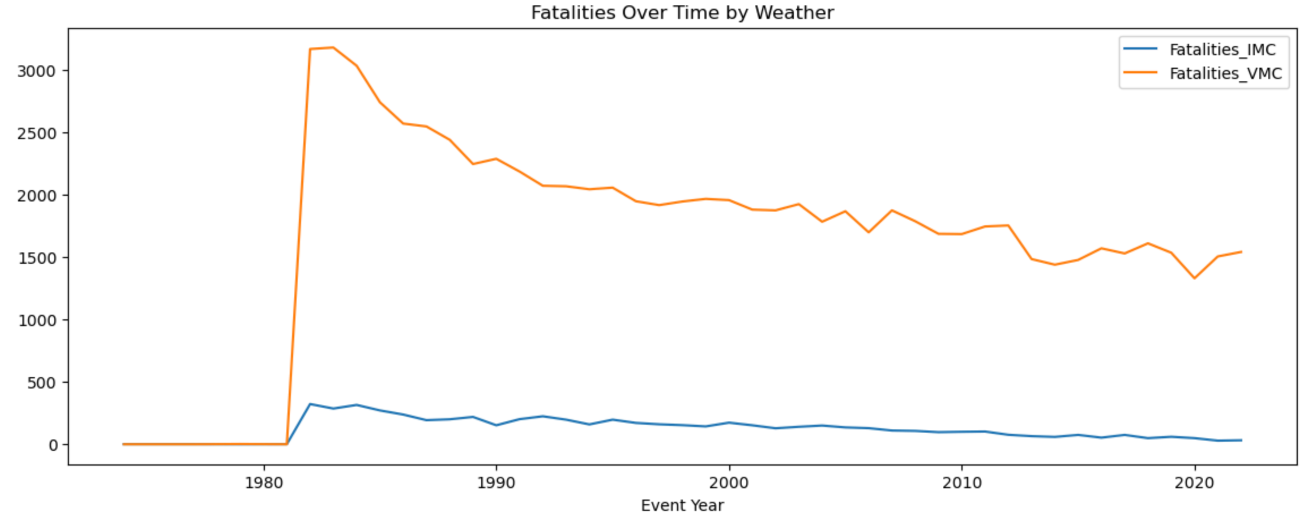
```
Weather Condition
IMC    10447
VMC    34451
Name: Total Fatal Injuries, dtype: int64
<Axes: title={'center': 'Overall Fatalities in IMC vs VMC'}, ylabel='Total
Fatal Injuries'>
```




```
# Group by year and weather condition
trend = data_weather.groupby(['Event Year', 'Weather Condition']).agg({
    'Total Fatal Injuries': 'sum',
    'Event Id': 'count'
}).unstack().fillna(0)
#Confirm agg function later
# Rename for clarity
trend.columns = ['Accidents_IMC', 'Accidents_VMC', 'Fatalities_IMC', 'Fatalities_

# Plot trends
trend[['Accidents_IMC', 'Accidents_VMC']].plot(figsize=(14, 5), title='Accidents
trend[['Fatalities_IMC', 'Fatalities_VMC']].plot(figsize=(14, 5), title='Fataliti
```

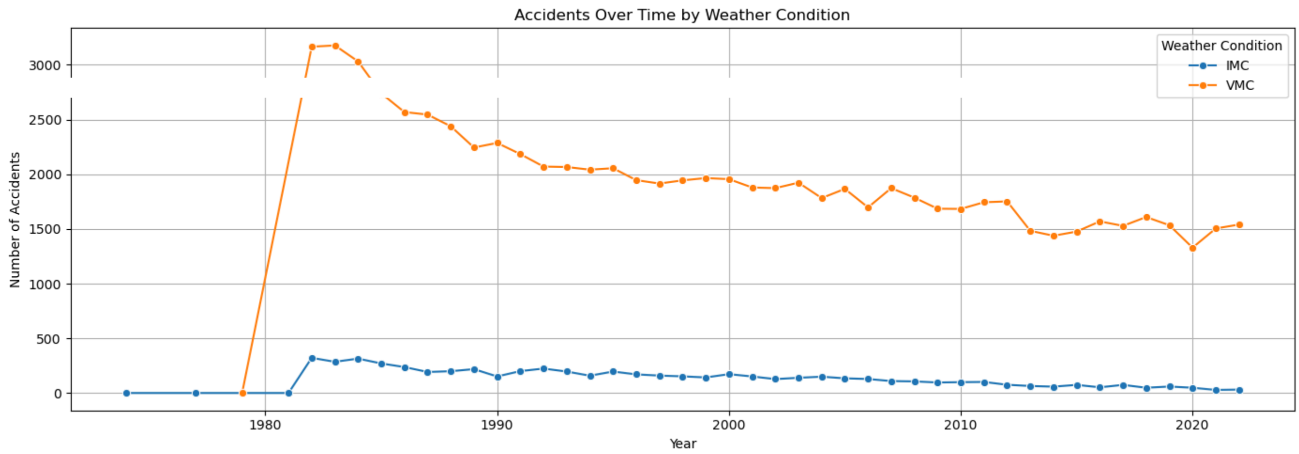
```
<Axes: title={'center': 'Fatalities Over Time by Weather'}, xlabel='Event Year'>
```



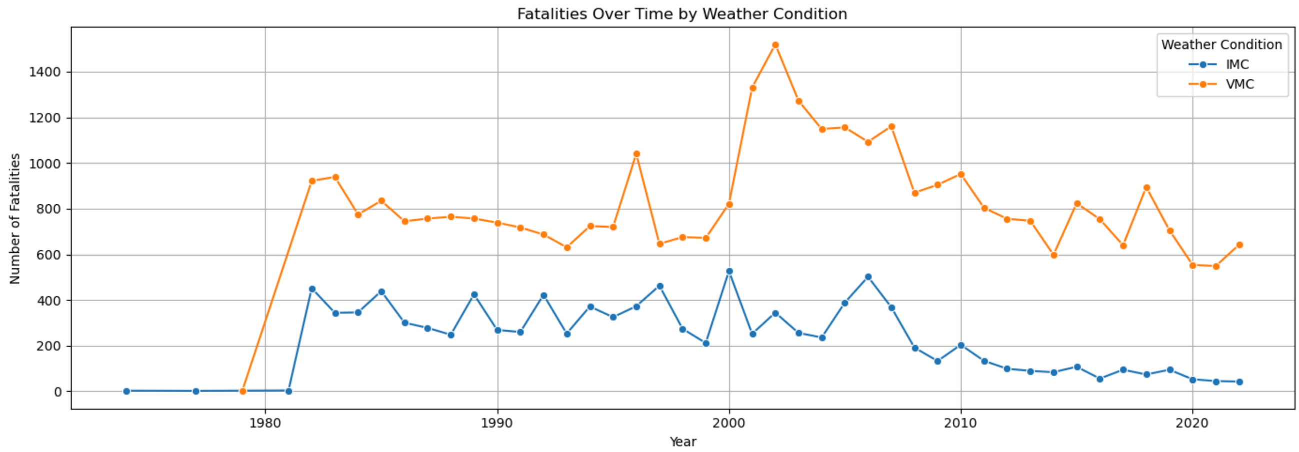



```
#Using Seaborn

# Group by Year and Weather
```

```
accident_trend = data_weather.groupby(['Event Year', 'Weather Condition']).agg(
    Accidents=('Event Id', 'count'),
    Fatalities=('Total Fatal Injuries', 'sum')
).reset_index()

plt.figure(figsize=(14, 5))
sns.lineplot(data=accident_trend, x='Event Year', y='Accidents', hue='Weather Con
plt.title('Accidents Over Time by Weather Condition')
plt.ylabel('Number of Accidents')
plt.xlabel('Year')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(14, 5))
sns.lineplot(data=accident_trend, x='Event Year', y='Fatalities', hue='Weather Co
plt.title('Fatalities Over Time by Weather Condition')
plt.ylabel('Number of Fatalities')
plt.xlabel('Year')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Conclusion: Weather is not a major discriminator in risk analysis per make

Since: 1. High fatalities in VMC (Visual Meteorological Conditions) suggest that good weather does not guarantee safety — possibly pointing to pilot error, mechanical failure, or other factors. 2. Only one aircraft make has significantly higher fatalities in IMC (Instrument Meteorological Conditions), and only one has lower — these are outliers, not a trend. 3. The correlation (≈0.98) between total accidents and accidents in poor weather per make indicates that weather affects all makes almost similarly.

What this tells us: • Weather is not a key differentiator between makes. • Including it in the final model won't improve discrimination between aircraft risk levels.

Check purpose of the flights to decide whether we need to keep instructional flights

```
ntsb_data['Purpose Of Flight'].unique()
```

```
array(['Personal', 'Unknown', 'Business', 'Instructional', 'Ferry',
       'Executive/corporate', 'Aerial Observation', 'Aerial Application',
       'Public Aircraft', 'Skydiving', 'Other Work Use', 'Positioning',
       'Flight Test', 'Air Race/show', 'Air Drop',
       'Public Aircraft - Federal', 'Glider Tow',
       'Public Aircraft - Local', 'External Load',
```

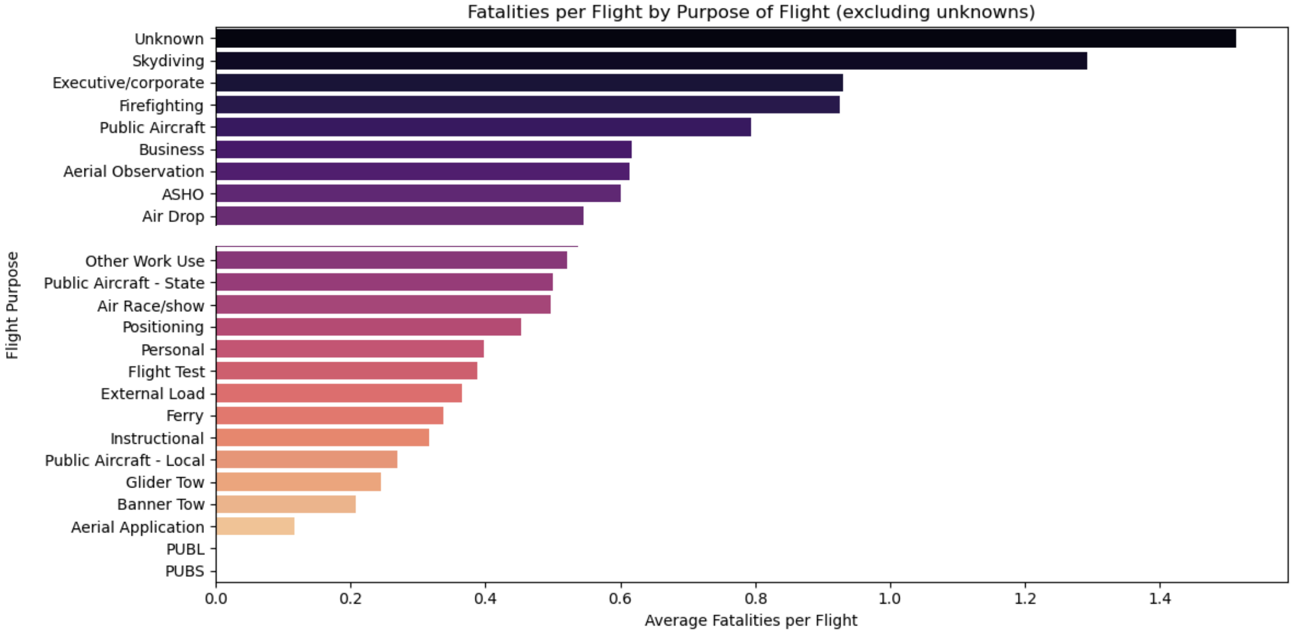
```
            'Public Aircraft - State', 'Banner Tow', 'Firefighting',
            'Air Race show', 'PUBS', 'ASHO', 'PUBL'], dtype=object)
```

```
#Clean the above to match i.e add Air Race/show with Air Race show
ntsb_data['Purpose Of Flight'] = ntsb_data['Purpose Of Flight'].replace({
    'Air Race show': 'Air Race/show'
})


#Check distribution of every category
ntsb_data['Purpose Of Flight'].value_counts(normalize=True) * 100
```

```
Purpose Of Flight
Personal                      55.799252
Instructional                 13.521165
Unknown                       12.367114
Aerial Application             5.341611
Business                       4.521836
Positioning                    1.855579
Other Work Use                 1.421246
Public Aircraft                1.196121
Ferry                          0.953940
Aerial Observation             0.894816
Executive/corporate            0.617389
Flight Test                    0.468443
Skydiving                      0.205796
Air Race/show                  0.173961
External Load                  0.139851
Public Aircraft - Federal      0.120522
Banner Tow                     0.114837
Public Aircraft - Local        0.084138
Public Aircraft - State        0.072768
Glider Tow                     0.060261
Firefighting                   0.045480
Air Drop                       0.012507
ASHO                           0.005685
PUBS                           0.004548
PUBL                           0.001137
Name: proportion, dtype: float64
```

```
# Total fatalities per purpose
fatalities_by_purpose = ntsb_data.groupby('Purpose Of Flight')['Total Fatal Injur

# Optional: normalize by count of flights per purpose
counts_by_purpose = ntsb_data['Purpose Of Flight'].value_counts()

# Fatality rate per flight
fatality_rate_by_purpose = (fatalities_by_purpose / counts_by_purpose).sort_value


# Convert to DataFrame for plotting
fatality_rate_df = fatality_rate_by_purpose.reset_index()
fatality_rate_df.columns = ['Purpose of Flight', 'Fatalities per Flight']

plt.figure(figsize=(12,6))
sns.barplot(data=fatality_rate_df, x='Fatalities per Flight', y='Purpose of Fligh
plt.title('Fatalities per Flight by Purpose of Flight (excluding unknowns)')
```

```
plt.xlabel('Average Fatalities per Flight')
plt.ylabel('Flight Purpose')
plt.tight_layout()
plt.show()
```

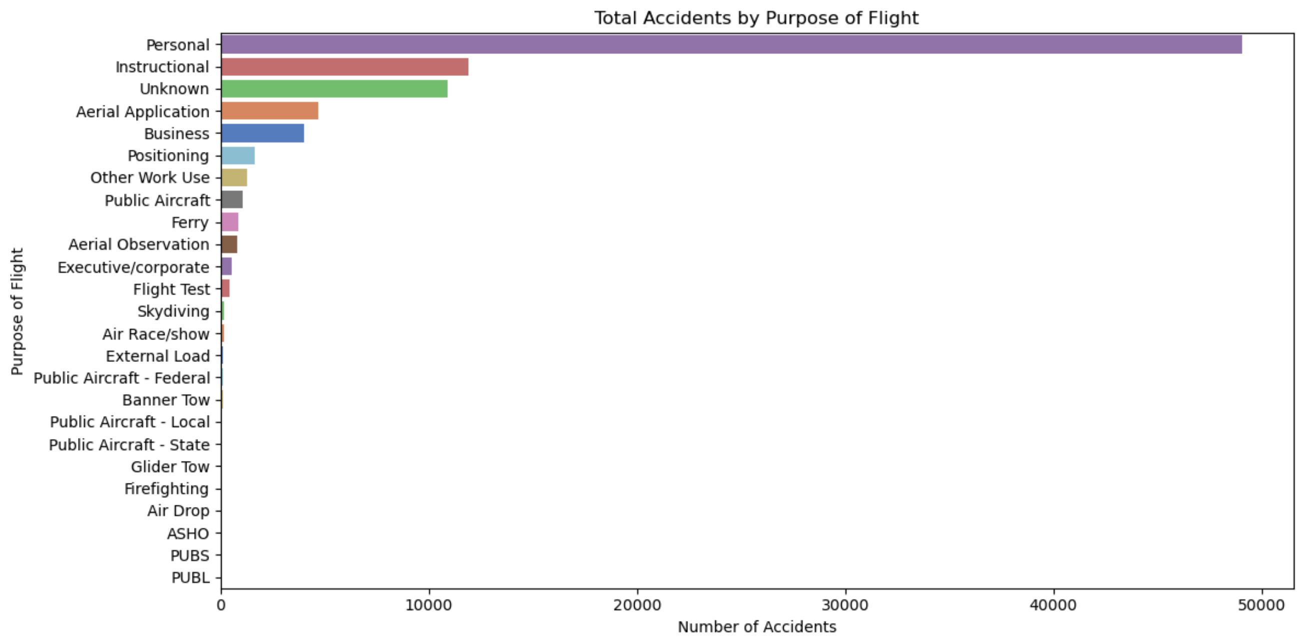



```
#Accidents per purpose of Flight
accidents_per_purpose = ntsb_data['Purpose Of Flight'].value_counts().sort_values
print(accidents_per_purpose)
```

```
Purpose Of Flight
Personal                  49076
Instructional             11892
Unknown                   10877
Aerial Application         4698
Business                   3977
Positioning                1632
Other Work Use             1250
Public Aircraft            1052
Ferry                       839
Aerial Observation          787
Executive/corporate         543
Flight Test                 412
Skydiving                   181
```

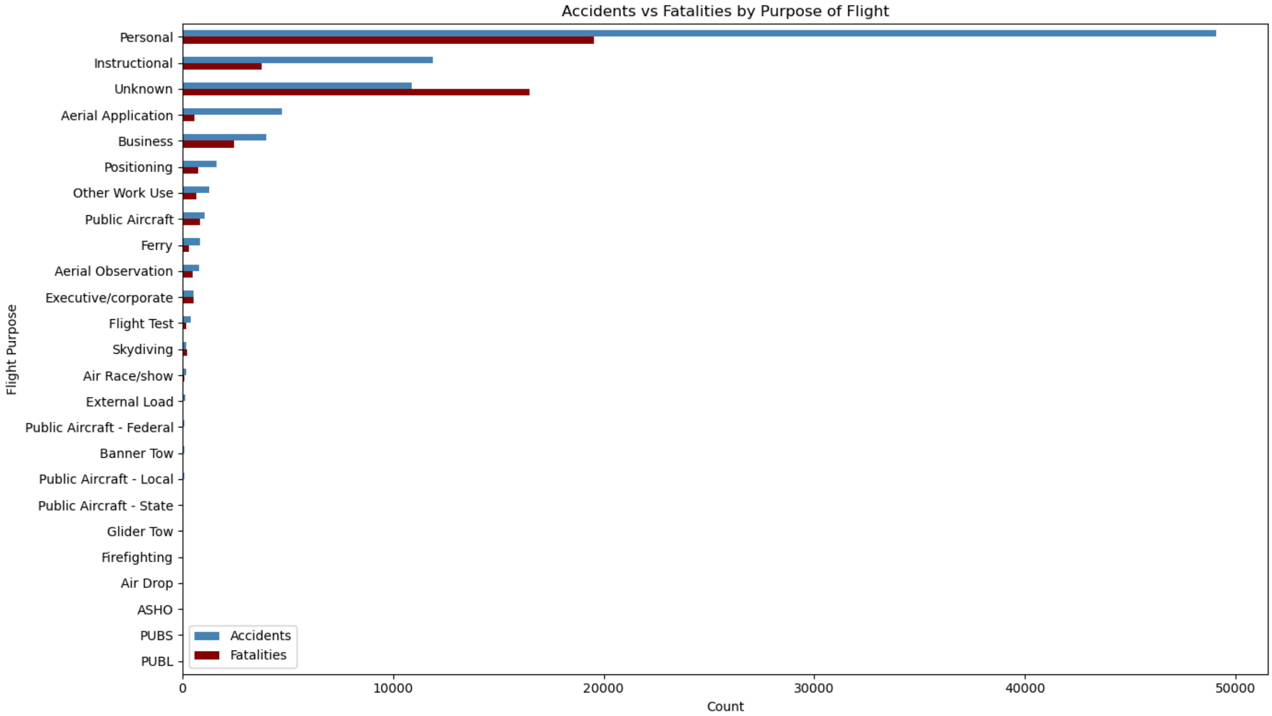
```
Air Race/show              153
External Load              123
Public Aircraft - Federal  106
Banner Tow                 101
Public Aircraft - Local     74
Public Aircraft - State     64
Glider Tow                  53
Firefighting                40
Air Drop                    11
ASHO                         5
PUBS                         4
PUBL                         1
Name: count, dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x=accidents_per_purpose.values, y=accidents_per_purpose.index, hue=ac
plt.title("Total Accidents by Purpose of Flight")
plt.xlabel("Number of Accidents")
plt.ylabel("Purpose of Flight")
plt.tight_layout()
plt.show()
```

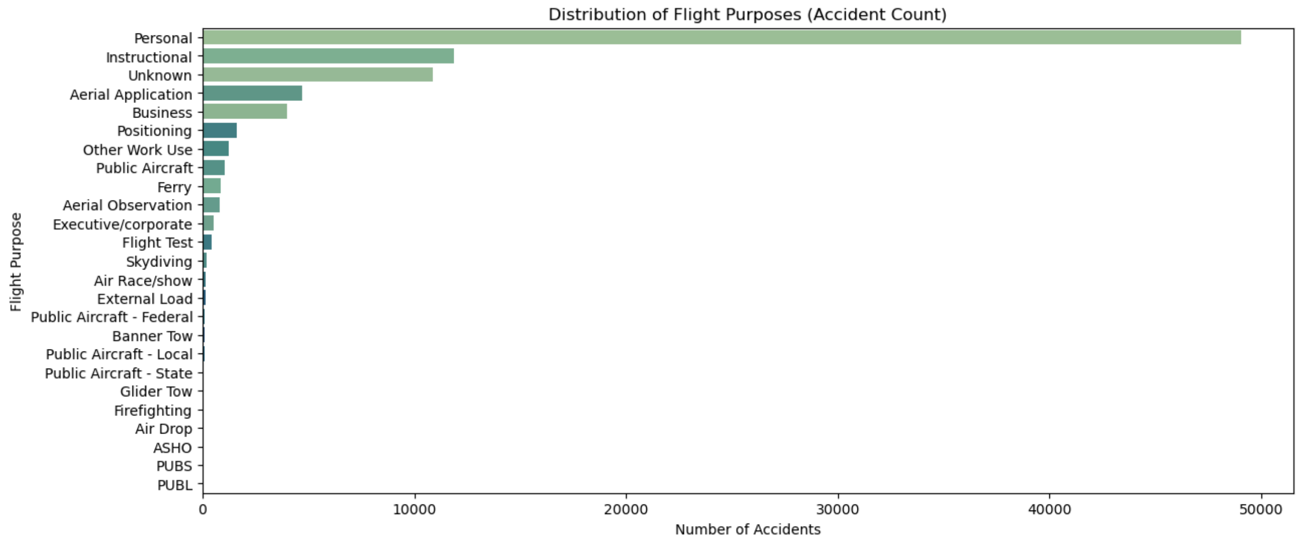


```
# Total accidents per purpose
accidents = ntsb_data['Purpose Of Flight'].value_counts()

# Total fatalities per purpose
fatalities = ntsb_data.groupby('Purpose Of Flight')['Total Fatal Injuries'].sum()


comparison = pd.DataFrame({
    'Accidents': accidents,
    'Fatalities': fatalities
}).fillna(0).astype(int).sort_values('Accidents', ascending=False)

print(comparison.head(10))



plt.figure(figsize=(14, 7))
comparison.plot(kind='barh', figsize=(14, 8), color=['steelblue', 'darkred'])
plt.title("Accidents vs Fatalities by Purpose of Flight")
plt.xlabel("Count")
plt.ylabel("Flight Purpose")
plt.gca().invert_yaxis()  # To show highest at top
plt.tight_layout()
plt.show()
```

```
                      Accidents  Fatalities
Purpose Of Flight
Personal                  49076       19542
Instructional             11892        3765
Unknown                   10877       16467
Aerial Application         4698         552
Business                   3977        2454
Positioning                1632         739
Other Work Use             1250         652
Public Aircraft            1052         835
Ferry                       839         284
Aerial Observation          787         483
<Figure size 1400x700 with 0 Axes>
```



```
plt.figure(figsize=(14, 6))
sns.countplot(data=ntsb_data, y='Purpose Of Flight', order=ntsb_data['Purpose Of
plt.title("Distribution of Flight Purposes (Accident Count)")
plt.xlabel("Number of Accidents")
plt.ylabel("Flight Purpose")
plt.show()
```



Conclusion: The flight purpose does not directly affect the number of accidents i.e the more the flights the more the risk of accidents

Calculate risk score

Risk score = (Fatal Injuries + 3/4 Serious Injuries + 1/4 Minor Injuries) / Total people Aboard

```
# Group and calculate totals per make
summary = ntsb_data.groupby('Make').agg({
    'Event Id': 'count',        # Number of accidents
    'Total Fatal Injuries': 'sum',        # Total fatalitie
    'Total Person': 'sum',              # Total people aboar
    'Total Serious Injuries': 'sum',
```

```
    'Total Minor Injuries': 'sum'
}).reset_index()


# Weighted risk score , Create a weighted risk score to define the composite scor
summary['Risk_Score'] = (
    (summary['Total Fatal Injuries'] * 1.0) +
    (summary['Total Serious Injuries'] * 0.75) +
    (summary['Total Minor Injuries'] * 0.25)
) / summary['Total Person']


summary = summary[summary['Event Id'] >= 30]

# Sort by safest (lowest risk score)
safest_makes = summary.sort_values(by='Risk_Score').head(15)
safest_makes
```

	Make	Event Id	Total Fatal Injuries	Total Person	Total Serious Injuries	Total Minor Injuries	Risk_Score
822	boeing	2652	6135	209652	3779	2689	0.045988