Final Project Submission

- Student name: Colleta Kiilu
- Student pace: self paced / **part time** / full time
- Scheduled project review date/time: 08/09/2024
- Instructor name: William Okomba
- Blog post URL: https://github.com/Collz472/Phase1_project.git (https://github.com/Collz472/Phase1_project.git)

Source: GIFDB.com (https://gifdb.com/)

OVERVIEW

As part of the company's strategic growth into new markets, there is increasing interest in joining the aviation industry. The goal is to purchase and operate aircraft for both commercial and private enterprises. However, before making any decisions, the business needs to be aware of the possible dangers associated with various aircraft types.

This analysis focuses on identifying the lowest-risk aircraft models by reviewing data from the National Transportation Safety Board (NTSB) aviation accident database. The key indicators to be evaluated are damage to the aircraft, frequency of accidents/incidents and severity of injuries.

The data covers the period between 24 October 1948 and 29 December 2022.

Ultimately, this analysis will provide the company, under the guidance of the Head of the new Aviation Division, with data-driven insights and recommendations about which aircraft models to invest in for this new business venture.

BUSINESS UNDERSTANDING

Boeing's 2024 commercial Market Outlooks (CMO) predics that air travel demand will outpace economic growth through 2043 with a forecast of nearly 44,0000 new airplane deliveries over the next 20years.

The total number rof passegers is expected to reach around 5 billion in 2024 with a load factor of 82.5% demonstrating high demand fro air travel.

Some of the facors that significantly increase the risk of accidents in aviation as reported by Skybrary include weather conditions, Flight phases, pilot decision making, aircraft design and geomagnetic storms among others.

Findings from this assessment will seek to identify trends over time as well as identify some of the facotors contributing to the accidnets and incidents reported

Business Questions

1. Which aircraft models have the lowest accident rates?
2. Are there specific factors (e.g., weather conditions, flight phases) that significantly increase the risk of accidents for certain aircraft?
3. What are the trends over time, and how do they impact decision-making for future aircraft purchases?

DATA UNDERSTANDING

In [37]:
```
# import data analysis libraries
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [38]:
```
# import data and create df
# df = pd.read_csv("AviationData.csv") # encoding issues
df = pd.read_csv("AviationData.csv", encoding="latin-1")

#checking the first 5 columns
df.head()
```

Out[38]:

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

5 rows × 31 columns

In [39]:
```
#checking the last 5 columns
df.tail()
```

Out[39]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States

5 rows × 31 columns

In [40]:
```
# checking the dataset information
df.info()
```

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

In [41]:
```
#checking the dataset shape
df.shape
```

Out[41]: (88889, 31)

The dataset 88889 records and 31 columns.

In [42]:
```
# Listing the columns in the dataset
df.columns
```

Out[42]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Dat
e',
       '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.Descrip
tion',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Inju
ries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

1. Data Cleaning

a. Duplicates

In [43]:
```
df.duplicated().sum()

# The data set has no duplicates
```

Out[43]:
```
0
```

In [44]:
```
# make df copy to be used in data cleaning

data = df.copy()
```

b. Missing Values

In [45]:
```
# percentage of missing values per column
# sorted in descending order

data.isna().sum().sort_values(ascending=False)/len(data)*100
```

Out[45]:
```
Schedule                 85.845268
Air.carrier              81.271023
FAR.Description          63.974170
Aircraft.Category        63.677170
Longitude                61.330423
Latitude                 61.320298
Airport.Code             43.469946
Airport.Name             40.611324
Broad.phase.of.flight    30.560587
Publication.Date         15.492356
Total.Serious.Injuries   14.073732
Total.Minor.Injuries     13.424608
Total.Fatal.Injuries     12.826109
Engine.Type               7.961615
Report.Status             7.178616
Purpose.of.flight         6.965991
Number.of.Engines         6.844491
Total.Uninjured           6.650992
Weather.Condition         5.053494
Aircraft.damage           3.593246
Registration.Number       1.481623
Injury.Severity           1.124999
Country                   0.254250
Amateur.Built             0.114750
Model                     0.103500
Make                      0.070875
Location                  0.058500
Event.Date                0.000000
Accident.Number           0.000000
Investigation.Type        0.000000
Event.Id                  0.000000
dtype: float64
```

In [46]:
```
# check for unique values in each column

for col in data.columns:
    print({col})
    print(data[col].unique())
    print()
```

```
{'Event.Id'}
['20001218X45444' '20001218X45447' '20061025X01555' ... '2022122710649
7'
 '20221227106498' '20221230106513']

{'Investigation.Type'}
['Accident' 'Incident']

{'Accident.Number'}
['SEA87LA080' 'LAX94LA336' 'NYC07LA005' ... 'WPR23LA075' 'WPR23LA076'
 'ERA23LA097']

{'Event.Date'}
['1948-10-24' '1962-07-19' '1974-08-30' ... '2022-12-22' '2022-12-26'
 '2022-12-29']

{'Location'}
['MOOSE CREEK, ID' 'BRIDGEPORT, CA' 'Saltville, VA' ... 'San Manual, A
Z'
```

i. Drop Unnecessary Columns

In [47]:
```
#Drop irrelevant columns with a high percentage of missing values and may

data.drop(['Schedule', 'Air.carrier', 'Latitude', 'Longitude','FAR.Descrip
           'Accident.Number', 'Airport.Code', 'Airport.Name', 'Publicatior
          axis=1, inplace=True)
```

In [48]:
```
# confirm dropped columns
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      88889 non-null  object 
 2   Event.Date              88889 non-null  object 
 3   Country                 88663 non-null  object 
 4   Injury.Severity         87889 non-null  object 
 5   Aircraft.damage         85695 non-null  object 
 6   Aircraft.Category       32287 non-null  object 
 7   Make                    88826 non-null  object 
 8   Model                   88797 non-null  object 
 9   Amateur.Built           88787 non-null  object 
 10  Number.of.Engines       82805 non-null  float64
 11  Engine.Type             81812 non-null  object 
 12  Purpose.of.flight       82697 non-null  object 
 13  Total.Fatal.Injuries    77488 non-null  float64
 14  Total.Serious.Injuries  76379 non-null  float64
 15  Total.Minor.Injuries    76956 non-null  float64
 16  Total.Uninjured         82977 non-null  float64
 17  Weather.Condition       84397 non-null  object 
 18  Broad.phase.of.flight   61724 non-null  object 
dtypes: float64(5), object(14)
memory usage: 12.9+ MB
```

ii. Check Missing Values Percentage per Column

In [49]:
```
# Check the number of missing values in each column

data.isna().sum().sort_values(ascending=False)
```

Out[49]:
```
Aircraft.Category         56602
Broad.phase.of.flight     27165
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Fatal.Injuries      11401
Engine.Type                7077
Purpose.of.flight          6192
Number.of.Engines          6084
Total.Uninjured            5912
Weather.Condition          4492
Aircraft.damage            3194
Injury.Severity            1000
Country                     226
Amateur.Built               102
Model                        92
Make                         63
Event.Date                    0
Investigation.Type            0
Event.Id                      0
dtype: int64
```

In [50]:
```
print(data['Aircraft.Category'].unique())
```

```
[nan 'Airplane' 'Helicopter' 'Glider' 'Balloon' 'Gyrocraft' 'Ultralight'
 'Unknown' 'Blimp' 'Powered-Lift' 'Weight-Shift' 'Powered Parachute'
 'Rocket' 'WSFT' 'UNK' 'ULTR']
```

- Missing values in the data will be dealt with as per the data in each column.

iii. .fillna()

- The `fillna()` method will be used to fill missing (NaN) values with a specified value.
- In the code below, *Unknown* and *0* were used.

In [51]:

```
# Replace missing values in 'Aircraft.Category' column with 'Unknown'
data['Aircraft.Category'].fillna('Unknown', inplace=True)

# Replace missing values in 'Broad.phase.of.flight' column with 'Unknown'
data['Broad.phase.of.flight'].fillna('Unknown', inplace=True)

# Replace missing values in 'Engine.Type' column with 'Unknown'
data['Engine.Type'].fillna('Unknown', inplace=True)

# Replace missing values in 'Purpose.of.flight' column with 'Unknown'
data['Purpose.of.flight'].fillna('Unknown', inplace=True)

# Replace missing values in 'Number.of.Engines' column with 'Unknown'
# the number of engines is also a critical factor when determing the model
data['Number.of.Engines'].fillna(0, inplace=True)

# Replace missing values in 'Weather.Condition' column with 'UNK'
data['Weather.Condition'].fillna('Unknown', inplace=True)

# Replace missing values in 'Aircraft.damage' column with 'Unspecified'
data['Aircraft.damage'].fillna('Unspecified', inplace=True)

# Replace missing values in 'Country' column with 'Unknown'
data['Country'].fillna('Unknown', inplace=True)

# Replace missing values in 'Amateur.Built' column with 'Unknown'
data['Amateur.Built'].fillna('Unknown', inplace=True)

# Replace missing values in 'Make' column with 'Unknown'
data['Make'].fillna('Unknown', inplace=True)

# Replace missing values in 'Total.Serious.Injuries' column with 0
data['Total.Serious.Injuries'].fillna(0, inplace=True)

# Replace missing values in 'Total.Minor.Injuries' column with 0
data['Total.Minor.Injuries'].fillna(0, inplace=True)



# Replace all missing values(Blanks) in 'Total.Fatal.Injuries' column with
# when compared with data in the 'Injury.severity' column
data['Total.Fatal.Injuries'].fillna(0, inplace=True)

# Replace missing values in 'Injury.Severity' column with 'Unknown'
data['Injury.Severity'].fillna('Unknown', inplace=True)
```

c. Data Manipulation

- The `.replace()` method will substitute specific values with new values depending on the data in the column.
- This is mainly for uniformity and ease of understanding the findings.

In [52]:
```
# Replace all 'Incident' in 'Injury.Severity' column with 'Non-Fatal'
data['Injury.Severity'].replace('Incident',  'Non-Fatal', inplace=True)

# Replace all 'Unavailable' in 'Injury.Severity' column with 'Unknown'
data['Injury.Severity'].replace('Unavailable',  'Unknown', inplace=True)

# Strip all numbericals from 'Injury.Severity' column as they are already
# data['Injury.Severity'].str.replace(r'\(\d+\)', '', regex=True)
data['Injury.Severity'] = data['Injury.Severity'].str.replace(r'\(\d+\)',

# Replace all 'UNK' and 'Unk\ in 'Weather.Condition' column with 'Unknown'
data['Weather.Condition'].replace('UNK',  'Unknown', inplace=True)
data['Weather.Condition'].replace('Unk',  'Unknown', inplace=True)
```

- The `Total.Fatal.Injuries` and `Total.Uninjured` columns required some additonal manipulation because some records had a figure for both fatality and uninjured.
- However, it is impossible to have a fatality and uninjured within the same event since the fatalities suffred 'fatal injuries'.

In [53]:
```
# Replace all blanks in 'Total.Uninjured' column with '0' if 'Total.Fatal.
data.loc[(data['Total.Fatal.Injuries'] >= 1) & (data['Total.Uninjured'].is
data['Total.Uninjured'].fillna(0, inplace=True)
```

- Convert all float64 to int64 to make them whole numbers since human beings are whole.

In [54]:
```
# Convert columns with float64 to int64
data['Total.Fatal.Injuries'] = data['Total.Fatal.Injuries'].astype('int64'
data['Total.Serious.Injuries'] = data['Total.Serious.Injuries'].astype('in
data['Total.Minor.Injuries'] = data['Total.Minor.Injuries'].astype('int64'
data['Total.Uninjured'] = data['Total.Uninjured'].astype('int64')
data['Number.of.Engines'] = data['Number.of.Engines'].astype('int64')
```

- The `Make` column needs some case conversion

In [55]:
```
data['Make']
```

Out[55]:
```
0                            Stinson
1                              Piper
2                             Cessna
3                           Rockwell
4                             Cessna
                    ...
88884                          PIPER
88885                       BELLANCA
88886    AMERICAN CHAMPION AIRCRAFT
88887                        CESSNA
88888                          PIPER
Name: Make, Length: 88889, dtype: object
```

In [56]:
```
# convert case
data['Make'] = data['Make'].str.title()
data['Make'].tail(10)
```

Out[56]:
```
88879                          Airbus
88880                           Piper
88881                          Cessna
88882    Grumman American Avn. Corp.
88883                     Air Tractor
88884                           Piper
88885                        Bellanca
88886     American Champion Aircraft
88887                          Cessna
88888                           Piper
Name: Make, dtype: object
```

- The `Model` column also required some extra cleaning to make the data uniform.

In [57]:
```
column_name = 'Model'
print(data[column_name].unique())
```

```
['108-3' 'PA24-180' '172M' ... 'ROTORWAY EXEC 162-F' 'KITFOX S5'
 'M-8 EAGLE']
```

In [58]:

```
# Data cleaning for the 'model' column which is very critical for analysis
# The goal is to purchase and operate aircraft for both commercial and pri

# Replace all 'wrongly spelt model names' in 'model' column with 'the corr
data['Model'].replace('PA24-180',  'PA-24-180', inplace=True)
data['Model'].replace('PA28-161',  'PA-28-161', inplace=True)
data['Model'].replace('R22 MARINER',  'R-22 MARINER', inplace=True)
data['Model'].replace('S2R', 'S-2R', inplace=True)

# Replace missing values in 'Model' column with 'Unknown'
data['Model'].fillna('Unknown', inplace=True)
```

In [59]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      88889 non-null  object
 2   Event.Date              88889 non-null  object
 3   Country                 88889 non-null  object
 4   Injury.Severity         88889 non-null  object
 5   Aircraft.damage         88889 non-null  object
 6   Aircraft.Category       88889 non-null  object
 7   Make                    88889 non-null  object
 8   Model                   88889 non-null  object
 9   Amateur.Built           88889 non-null  object
 10  Number.of.Engines       88889 non-null  int64 
 11  Engine.Type             88889 non-null  object
 12  Purpose.of.flight       88889 non-null  object
 13  Total.Fatal.Injuries    88889 non-null  int64 
 14  Total.Serious.Injuries  88889 non-null  int64 
 15  Total.Minor.Injuries    88889 non-null  int64 
 16  Total.Uninjured         88889 non-null  int64 
 17  Weather.Condition       88889 non-null  object
 18  Broad.phase.of.flight   88889 non-null  object
dtypes: int64(5), object(14)
memory usage: 12.9+ MB
```

In [60]:
```
# view the data to confirm data cleaning
data.head()
```

Out[60]:

	Event.Id	Investigation.Type	Event.Date	Country	Injury.Severity	Aircraft.damage
0	20001218X45444	Accident	1948-10-24	United States	Fatal	Destroyed
1	20001218X45447	Accident	1962-07-19	United States	Fatal	Destroyed
2	20061025X01555	Accident	1974-08-30	United States	Fatal	Destroyed
3	20001218X45448	Accident	1977-06-19	United States	Fatal	Destroyed
4	20041105X01764	Accident	1979-08-02	United States	Fatal	Destroyed

- All missing values in the data set have been filled.
- All records have been maintained.

c. Convert Event.Date Column to DateTime

- The time-related `Event.Date` column will be used to generate year and month for periodic analysis.
- First, it is converted to `DateTime` format.

In [61]:
```
data['Event.Date'] = pd.to_datetime(data['Event.Date'])
data['Event.Date'].dtype # confirm datatype
```

Out[61]:
```
dtype('<M8[ns]')
```

In [62]:
```
data['Event.Date'].head()
```

Out[62]:
```
0   1948-10-24
1   1962-07-19
2   1974-08-30
3   1977-06-19
4   1979-08-02
Name: Event.Date, dtype: datetime64[ns]
```

- The `Event.Year` and `Event.Month` columns are then created.

In [63]:

```
# extract year and month and create Event.Year and Event.Month columns

data['Event.Year'] = data['Event.Date'].dt.year
data['Event.Month'] = data['Event.Date'].dt.month_name()
data.head()
```

Out[63]:

	Event.Id	Investigation.Type	Event.Date	Country	Injury.Severity	Aircraft.damage
0	20001218X45444	Accident	1948-10-24	United States	Fatal	Destroyed
1	20001218X45447	Accident	1962-07-19	United States	Fatal	Destroyed
2	20061025X01555	Accident	1974-08-30	United States	Fatal	Destroyed
3	20001218X45448	Accident	1977-06-19	United States	Fatal	Destroyed
4	20041105X01764	Accident	1979-08-02	United States	Fatal	Destroyed

5 rows × 21 columns

In [64]:

```
# confirm the unique values
for col in data.columns:
    print({col})
    print(data[col].unique())
    print()
```

```
{'Event.Id'}
['20001218X45444' '20001218X45447' '20061025X01555' ... '2022122710649
7'
 '20221227106498' '20221230106513']

{'Investigation.Type'}
['Accident' 'Incident']

{'Event.Date'}
['1948-10-24T00:00:00.000000000' '1962-07-19T00:00:00.000000000'
 '1974-08-30T00:00:00.000000000' ... '2022-12-22T00:00:00.000000000'
 '2022-12-26T00:00:00.000000000' '2022-12-29T00:00:00.000000000']

{'Country'}
['United States' 'Unknown' 'GULF OF MEXICO' 'Puerto Rico' 'ATLANTIC OC
EAN'
 'HIGH ISLAND' 'Bahamas' 'MISSING' 'Pakistan' 'Angola' 'Germany'
 'Korea, Republic Of' 'Martinique' 'American Samoa' 'PACIFIC OCEAN'
 'Canada' 'Bolivia' 'Mexico' 'Dominica' 'Netherlands Antilles' 'Icelan
d'
```

In [65]:
```
# The data is now clean for Exploratory data analysis
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   Event.Id                88889 non-null  object        
 1   Investigation.Type      88889 non-null  object        
 2   Event.Date              88889 non-null  datetime64[ns]
 3   Country                 88889 non-null  object        
 4   Injury.Severity         88889 non-null  object        
 5   Aircraft.damage         88889 non-null  object        
 6   Aircraft.Category       88889 non-null  object        
 7   Make                    88889 non-null  object        
 8   Model                   88889 non-null  object        
 9   Amateur.Built           88889 non-null  object        
 10  Number.of.Engines       88889 non-null  int64         
 11  Engine.Type             88889 non-null  object        
 12  Purpose.of.flight       88889 non-null  object        
 13  Total.Fatal.Injuries    88889 non-null  int64         
 14  Total.Serious.Injuries  88889 non-null  int64         
 15  Total.Minor.Injuries    88889 non-null  int64         
 16  Total.Uninjured         88889 non-null  int64         
 17  Weather.Condition       88889 non-null  object        
 18  Broad.phase.of.flight   88889 non-null  object        
 19  Event.Year              88889 non-null  int64         
 20  Event.Month             88889 non-null  object        
dtypes: datetime64[ns](1), int64(6), object(14)
memory usage: 14.2+ MB
```

- After data cleaning, the data now has 21 columns and 88889 records/events.
- Converting to CSV will also ensure that the cleaned data is ready for dashboard creation.

In [66]:
```
#save the new dataframe in csv format

data.to_csv('cleaned_aviation_data.csv', index=False)
```

EXPLORATORY DATA ANALYSIS (EDA)

EDA columns of interest

- 'Country'
- 'Injury.Severity',
- 'Aircraft.damage',
- 'Aircraft.Category',,
- 'Make',
- 'Model',
- 'Engine.Type',

- 'Schedule',
- 'Purpose.of.flight',
- 'Total.Fatal.Injuries',
- 'Total.Uninjured',
- 'Weather.Condition',
- 'Broad.phase.of.flight'

The columns will be analysed to answer the questions below:

1. What are the most reliable aircraft models(with lowest risks) in the market, both for commercial and private use
2. Are there specific factors (e.g., weather conditions, flight phases) that significantly increase the risk of accidents for certain aircraft?
3. What is the level of severity for the top 15 prefered models?
4. What is the incidence rate by Country, year, purpose of flight and model?

In [67]:
```
# Now load the clean data set for EDA

aviation = pd.read_csv('cleaned_aviation_data.csv')

aviation['Event.Date'] = pd.to_datetime(aviation['Event.Date'])

aviation.head()
```

Out[67]:

	Event.Id	Investigation.Type	Event.Date	Country	Injury.Severity	Aircraft.damage
0	20001218X45444	Accident	1948-10-24	United States	Fatal	Destroyed
1	20001218X45447	Accident	1962-07-19	United States	Fatal	Destroyed
2	20061025X01555	Accident	1974-08-30	United States	Fatal	Destroyed
3	20001218X45448	Accident	1977-06-19	United States	Fatal	Destroyed
4	20041105X01764	Accident	1979-08-02	United States	Fatal	Destroyed

5 rows × 21 columns

In [68]:
```
aviation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   Event.Id                88889 non-null  object        
 1   Investigation.Type      88889 non-null  object        
 2   Event.Date              88889 non-null  datetime64[ns]
 3   Country                 88889 non-null  object        
 4   Injury.Severity         88889 non-null  object        
 5   Aircraft.damage         88889 non-null  object        
 6   Aircraft.Category       88889 non-null  object        
 7   Make                    88889 non-null  object        
 8   Model                   88889 non-null  object        
 9   Amateur.Built           88889 non-null  object        
 10  Number.of.Engines       88889 non-null  int64         
 11  Engine.Type             88889 non-null  object        
 12  Purpose.of.flight       88889 non-null  object        
 13  Total.Fatal.Injuries    88889 non-null  int64         
 14  Total.Serious.Injuries  88889 non-null  int64         
 15  Total.Minor.Injuries    88889 non-null  int64         
 16  Total.Uninjured         88889 non-null  int64         
 17  Weather.Condition       88889 non-null  object        
 18  Broad.phase.of.flight   88889 non-null  object        
 19  Event.Year              88889 non-null  int64         
 20  Event.Month             88889 non-null  object        
dtypes: datetime64[ns](1), int64(6), object(14)
memory usage: 14.2+ MB
```

In [69]:
```
# summary statistics for numerical columns
aviation.describe()
```

Out[69]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Tota
count	88889.000000	88889.000000	88889.000000	88889.000000	88
mean	1.068107	0.564761	0.240491	0.309127	
std	0.519181	5.126649	1.434614	2.083715	
min	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	380.000000	

1. Univariate Data Analysis

a. Count of Events

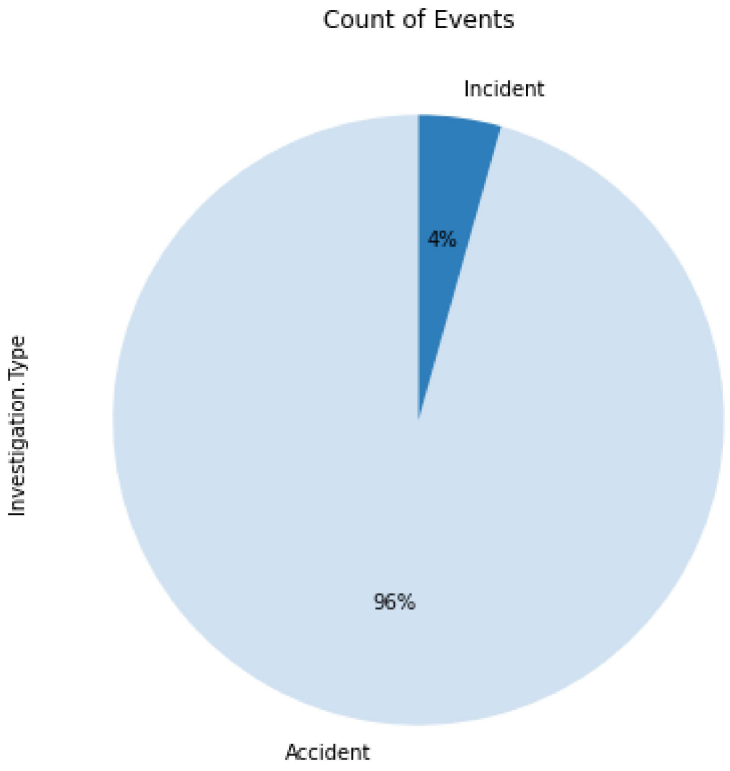
Insights

- The Figure below shows that 96% of all events that happened were accidents while only 4% were incidents.
- An **accident** is defined as “an occurrence associated with the operation of an aircraft which takes place between the time any person boards the aircraft with the intention of flight and all such persons have disembarked, and in which any person suffers death or serious injury, or in which the aircraft receives substantial damage”.
- An **incident** is defined as “an occurrence other than an accident, associated with the operation of an aircraft, which affects or could affect the safety of operations.”

In [70]:

```
# number of occurences for each event type
investigation_type_counts = aviation['Investigation.Type'].value_counts()
print(investigation_type_counts)
print('*'*100)

# pie chart
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(investigation_typ

plt.figure(figsize=(10, 7))
investigation_type_counts.plot(
    kind='pie', autopct='%1.0f%%',
    startangle=90,
    colors=colors
    )
plt.title("Count of Events")
plt.show()
```

```
Accident    85015
Incident     3874
Name: Investigation.Type, dtype: int64
****************************************************************************************************
```



b. Count of events by Make of Aircraft

- This will reveal the aircraft makes that was involved in the most events over the period under analysis.

Insights

- There are 7587 makes in the dataset.
- The top 5 aircraft makes with the highest number of events are; Cessna (27,146), Piper (14,869), Beech (5,372), Boeing (2,745) and Bell (2,722).
- This may mean that these makes are either:
  - a preferred model in the industry and/or;
  - most prone to experiencing aviation incidents and accidents.
- Royse Ralph L, Izatt, Lutes, Kitchens and Cohen had one incident/accident each.

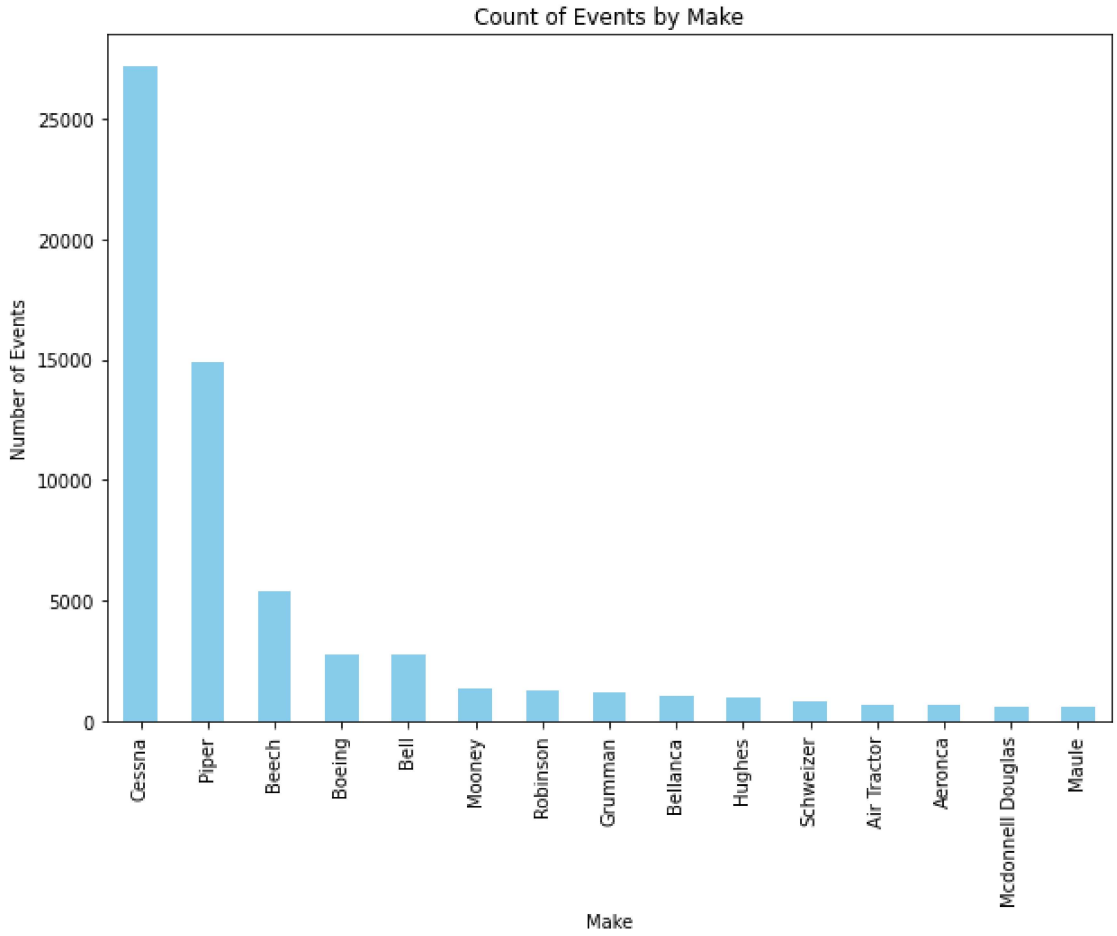
In [71]:

```
# events by make
make_counts = aviation['Make'].value_counts()

print('Top 10:\n', make_counts.head(10))

print()

print('Bottom 10:\n', make_counts.tail(10))
```

```
Top 10:
 Cessna      27149
Piper       14870
Beech        5372
Boeing       2745
Bell         2722
Mooney       1334
Robinson     1230
Grumman      1172
Bellanca     1045
Hughes        932
Name: Make, dtype: int64

Bottom 10:
 Fuller/Findlay    1
Harris-Runyan      1
Mcmurtrie          1
Davis/Crouch       1
Greer              1
Eachon             1
Hastings           1
Spezio             1
Earl F. Folsom     1
Bartle             1
Name: Make, dtype: int64
```

In [73]:

```
# bar chart
plt.figure(figsize=(10, 7))
make_counts.head(15).plot(kind='bar', color='skyblue')
plt.title("Count of Events by Make")
plt.xlabel("Make")
plt.ylabel("Number of Events")
plt.show()
```

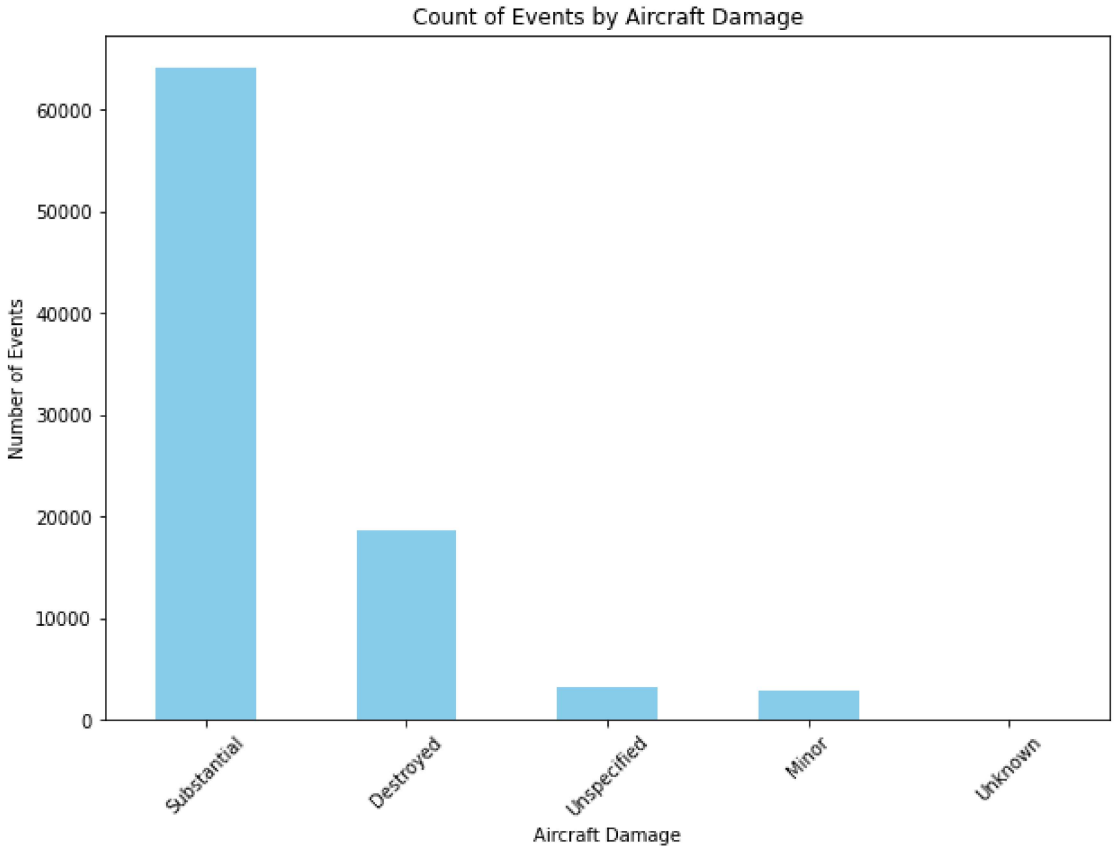


Damage to the Aircraft per Event

In [74]:

```
print(aviation['Aircraft.damage'].value_counts())
print('*'*150)

plt.figure(figsize=(10, 7))
aviation['Aircraft.damage'].value_counts().plot(kind='bar', color='skyblue
plt.title("Count of Events by Aircraft Damage")
plt.xlabel("Aircraft Damage")
plt.ylabel("Number of Events")
plt.xticks(rotation=45)
plt.show()
```

```
Substantial    64148
Destroyed      18623
Unspecified     3194
Minor           2805
Unknown          119
Name: Aircraft.damage, dtype: int64
**************************************************************************
**************************************************************************
******
```



b. Count of events by country

Insights

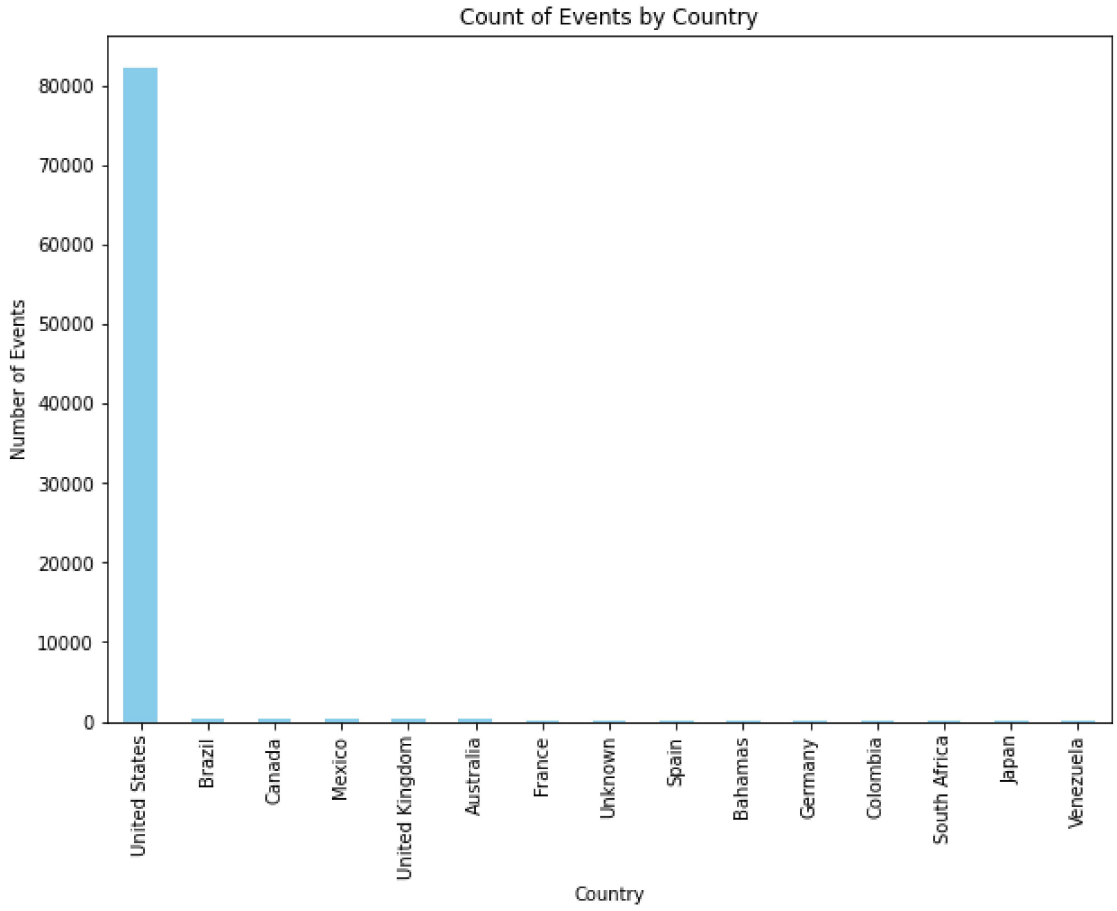
- United States had the highest numbers of events at 82,248 (92.5%) including both accidents and incidents, followed by Brazil with 374 events (0.4%).
- The high concentration of events in the United States (US) is due to the fact that the National Transportation Safety Board (NTSB) mainly deals with accidents within the US, in international waters, or with US aircrafts.

In [75]:
```
# events per country
country_counts = aviation['Country'].value_counts()
print('Top 10: ', country_counts.head(10))
print('*'*150)
print('Bottom 10: ', country_counts.tail(10))
```

```
Top 10:  United States     82248
Brazil              374
Canada              359
Mexico              358
United Kingdom      344
Australia           300
France              236
Unknown             229
Spain               226
Bahamas             216
Name: Country, dtype: int64
**********************************************************************
**********************************************************************
******
Bottom 10:  Gibraltar                1
Ivory Coast              1
Bosnia And Herzegovina   1
Guernsey                 1
Benin                    1
French Guiana            1
Obyan                    1
Wolseley                 1
Scotland                 1
Libya                    1
Name: Country, dtype: int64
```

In [76]:

```
# bar chart
plt.figure(figsize=(10, 7))
aviation["Country"].value_counts().head(15).plot(kind='bar', color='skyblu
plt.title("Count of Events by Country")
plt.xlabel("Country")
plt.ylabel("Number of Events")
plt.show()
```



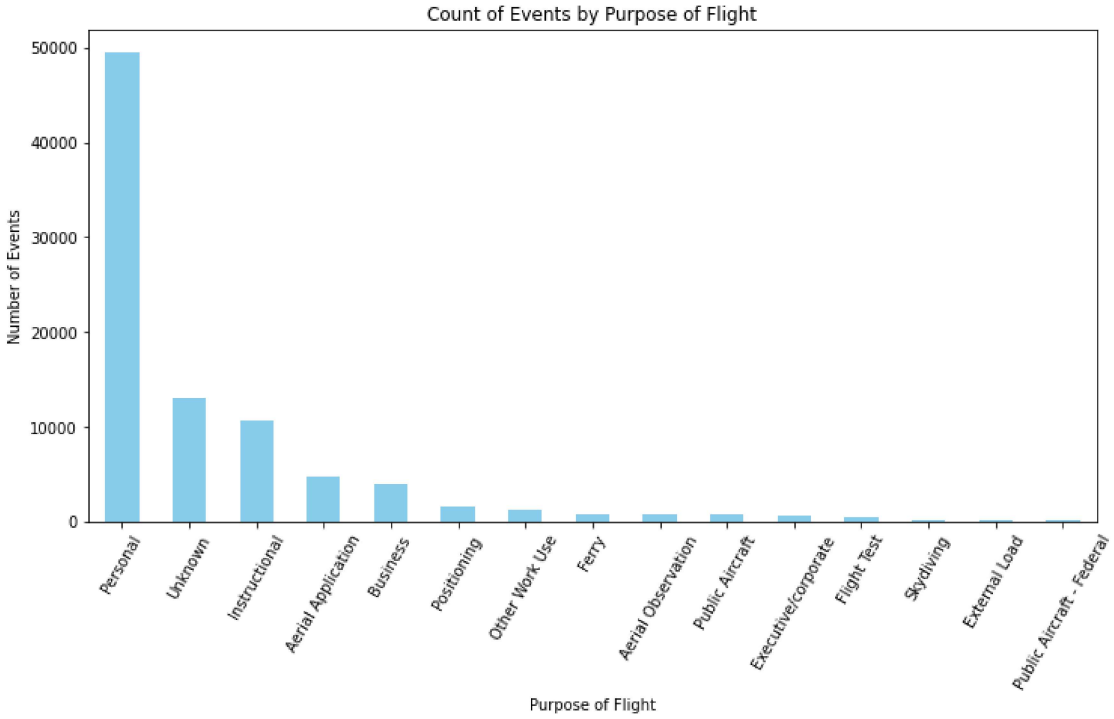
Events by Purpose of Flight

- Most events occured during personal flights.

In [77]:

```
flight_purpose = aviation['Purpose.of.flight'].value_counts()
print(flight_purpose)
print('*'*150)

# bar chart
plt.figure(figsize=(12, 6))
flight_purpose.head(15).plot(kind='bar', color='skyblue')
plt.title("Count of Events by Purpose of Flight")
plt.xlabel("Purpose of Flight")
plt.ylabel("Number of Events")
plt.xticks(rotation=60)
plt.show()
```

```
Personal                    49448
Unknown                     12994
Instructional               10601
Aerial Application           4712
Business                     4018
Positioning                  1646
Other Work Use               1264
Ferry                         812
Aerial Observation            794
Public Aircraft               720
Executive/corporate           553
Flight Test                   405
Skydiving                     182
External Load                 123
Public Aircraft - Federal     105
Banner Tow                    101
Air Race show                  99
Public Aircraft - Local        74
Public Aircraft - State        64
Air Race/show                  59
Glider Tow                     53
Firefighting                   40
Air Drop                       11
ASHO                            6
PUBS                            4
PUBL                            1
Name: Purpose.of.flight, dtype: int64
******************************************************************************************************************************************************
******
```


Count of Events by Purpose of Flight
Number of Events
50000
40000
30000
20000
10000
0
Personal
Unknown
Instructional
Aerial Application
Business
Positioning
Other Work Use
Ferry
Aerial Observation
Public Aircraft
Executive/corporate
Flight Test
Skydiving
External Load
Public Aircraft - Federal
Purpose of Flight

c. Count of Events by Year and Month

In [78]:

```
# events per year
aviation.groupby("Event.Year").size().reset_index(name="Count")
```

Out[78]:

	Event.Year	Count
0	1948	1
1	1962	1
2	1974	1
3	1977	1
4	1979	2
5	1981	1
6	1982	3593
7	1983	3556
8	1984	3457
9	1985	3096
10	1986	2880
11	1987	2828
12	1988	2730
13	1989	2544
14	1990	2518
15	1991	2462
16	1992	2355
17	1993	2313
18	1994	2257
19	1995	2309
20	1996	2187
21	1997	2148
22	1998	2226
23	1999	2209
24	2000	2220
25	2001	2063
26	2002	2020
27	2003	2085
28	2004	1952
29	2005	2031
30	2006	1851
31	2007	2016
32	2008	1893
33	2009	1783
34	2010	1786
35	2011	1850

	Event.Year	Count
36	2012	1835
37	2013	1561
38	2014	1535
39	2015	1582
40	2016	1664
41	2017	1638
42	2018	1681
43	2019	1624
44	2020	1392
45	2021	1545
46	2022	1607

- On further analysis, it was noted that events recorded between 1948 and 1981 are few.
- Records with those years will be dropped.
- There are now 88,882 records/events.

In [79]:

```
# drop events recorded between 1948 and 1981
aviation = aviation[(aviation['Event.Year'] < 1948) | (aviation['Event.Yea
aviation.reset_index(drop=True, inplace=True)
aviation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88882 entries, 0 to 88881
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88882 non-null  object
 1   Investigation.Type      88882 non-null  object
 2   Event.Date              88882 non-null  datetime64[ns]
 3   Country                 88882 non-null  object
 4   Injury.Severity         88882 non-null  object
 5   Aircraft.damage         88882 non-null  object
 6   Aircraft.Category       88882 non-null  object
 7   Make                    88882 non-null  object
 8   Model                   88882 non-null  object
 9   Amateur.Built           88882 non-null  object
 10  Number.of.Engines       88882 non-null  int64
 11  Engine.Type             88882 non-null  object
 12  Purpose.of.flight       88882 non-null  object
 13  Total.Fatal.Injuries    88882 non-null  int64
 14  Total.Serious.Injuries  88882 non-null  int64
 15  Total.Minor.Injuries    88882 non-null  int64
 16  Total.Uninjured         88882 non-null  int64
 17  Weather.Condition       88882 non-null  object
 18  Broad.phase.of.flight   88882 non-null  object
 19  Event.Year              88882 non-null  int64
 20  Event.Month             88882 non-null  object
dtypes: datetime64[ns](1), int64(6), object(14)
memory usage: 14.2+ MB
```

In [80]:

```
# group events by year
events_per_year = aviation.groupby("Event.Year").size().reset_index(name="
events_per_year
```

Out[80]:

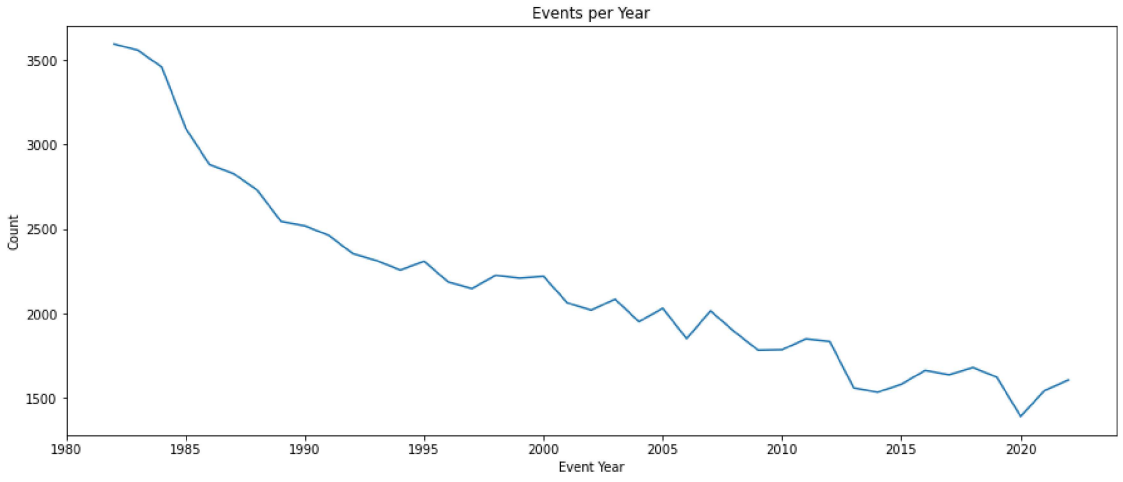
	Event.Year	Count
0	1982	3593
1	1983	3556
2	1984	3457
3	1985	3096
4	1986	2880
5	1987	2828
6	1988	2730
7	1989	2544
8	1990	2518
9	1991	2462
10	1992	2355
11	1993	2313
12	1994	2257
13	1995	2309
14	1996	2187
15	1997	2148
16	1998	2226
17	1999	2209
18	2000	2220
19	2001	2063
20	2002	2020
21	2003	2085
22	2004	1952
23	2005	2031
24	2006	1851
25	2007	2016
26	2008	1893
27	2009	1783
28	2010	1786
29	2011	1850
30	2012	1835
31	2013	1561
32	2014	1535
33	2015	1582
34	2016	1664
35	2017	1638

	Event.Year	Count
36	2018	1681
37	2019	1624
38	2020	1392
39	2021	1545
40	2022	1607

Insights

- The year 1982 recorded the highest number of events while 2020 had the least.
- The declining count of events may point to increased safety measures in the aviation industry.
- The COVID19 pandemic in 2020 may have caused lesser air travel, hence, the low number of accidents and incidents.

In [81]:
```
# line graph
plt.figure(figsize=(15, 6))
plt.plot(events_per_year['Event.Year'], events_per_year['Count'])
plt.xlabel('Event Year')
plt.ylabel('Count')
plt.title('Events per Year')
plt.show()
```



In [82]:
```
# events per month
events_per_month = aviation.groupby('Event.Month').size().reset_index(name
events_per_month
```

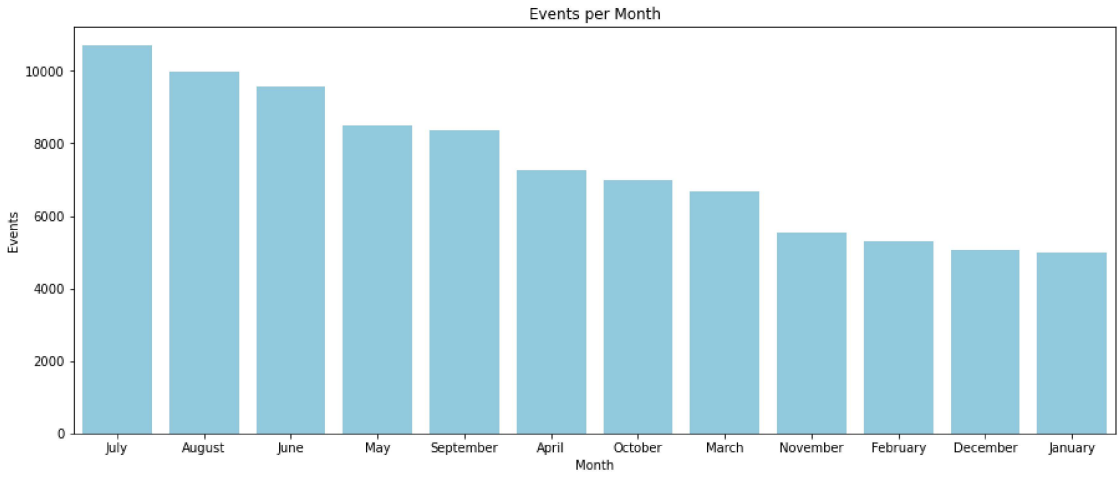
Out[82]:

	Event.Month	Count
0	April	7248
1	August	9983
2	December	5060
3	February	5285
4	January	4985
5	July	10697
6	June	9560
7	March	6686
8	May	8514
9	November	5538
10	October	6981
11	September	8345

Insights

- The months of June, July and August record the highest number of events (accidents and incidents).
- On the other hand, November, December, January and February have the least number of events.
- The trend can be attributed to the fact that June, July and August are summer months in US (country with most events) and people tend to travel more during the season. November, December, January and February are winter months and record less air travel.

In [83]:
```
plt.figure(figsize=(15, 6))
sns.barplot(x="Event.Month", y="Count",
            data=events_per_month.sort_values(by="Count", ascending=False))
plt.title("Events per Month")
plt.xlabel("Month")
plt.ylabel("Events")
plt.show()
```



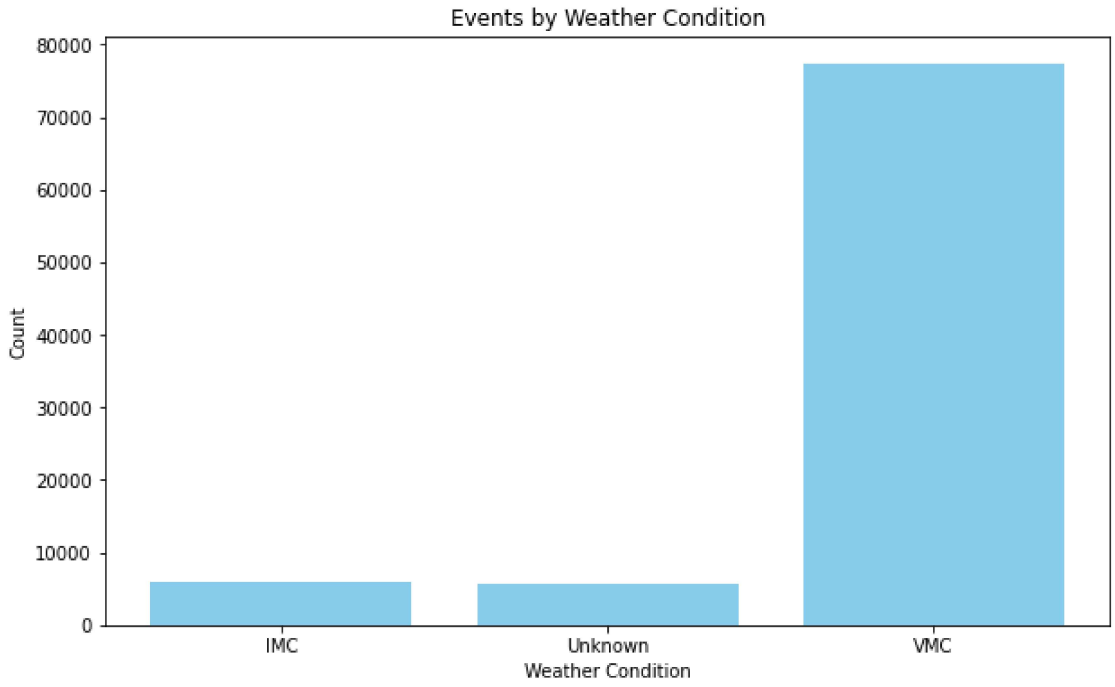
d. Events by Weather Condition

- Weather conditions description:
  - **IMC**: Instrument Meteorological Conditions; bad weather
  - **VMC**: Visual Meteorological Condition; means good weather
- Most events happened during good weather.

In [84]:

```
weather_counts = aviation.groupby('Weather.Condition').size().reset_index(
print(weather_counts)
print('*'*100)

# bar chart
plt.figure(figsize=(10, 6))
plt.bar(weather_counts['Weather.Condition'], weather_counts['Count'], colo
plt.xlabel('Weather Condition')
plt.ylabel('Count')
plt.title('Events by Weather Condition')
plt.show()
```

```
  Weather.Condition  Count
0               IMC   5973
1           Unknown   5608
2               VMC  77301
****************************************************************************
****************************
```



f. Severity of Injuries

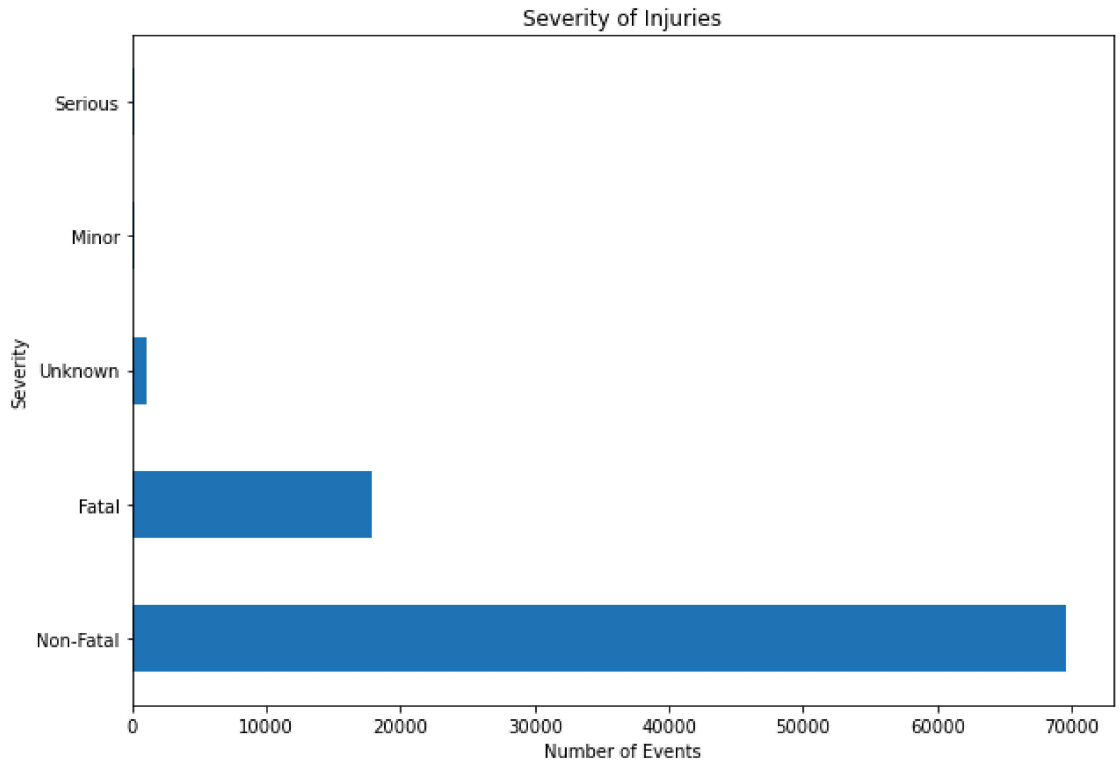
Insights

- **78.3%** of all events that occured were Non-Fatal while only **20.1%** were Fatal.

In [85]:

```
# injury severity count
injury_severity_counts = aviation['Injury.Severity'].value_counts()
print(injury_severity_counts)
print('*'*150)

# bar chart
plt.figure(figsize=(10, 7))
injury_severity_counts.head(15).plot(kind='barh')
plt.title("Severity of Injuries")
plt.xlabel("Number of Events")
plt.ylabel("Severity")
plt.show()
```

```
Non-Fatal    69575
Fatal        17820
Unknown       1096
Minor          218
Serious        173
Name: Injury.Severity, dtype: int64
**********************************************************************
**********************************************************************
******
```



2. Bivariate Analysis

a. Aircraft Make and the Severity of Injuries

In [86]:
```
# injury severity by make
injury_severity_by_make = aviation.groupby(
    ['Make', 'Injury.Severity']).size().reset_index(
        name='Count').sort_values(
            by=['Count'], ascending=[False]
            )
injury_severity_by_make.head(10)
```

Out[86]:

	Make	Injury.Severity	Count
1503	Cessna	Non-Fatal	22329
6005	Piper	Non-Fatal	11586
1501	Cessna	Fatal	4634
785	Beech	Non-Fatal	3766
6003	Piper	Fatal	3200
811	Bell	Non-Fatal	2092
1004	Boeing	Non-Fatal	2073
783	Beech	Fatal	1574
3333	Grumman	Non-Fatal	1040
5418	Mooney	Non-Fatal	949

Insights

- Cessna, having the highest number of events recorded against it, also has the highest number of both fatal and non-fatal injuries.
- Piper follows at second place.

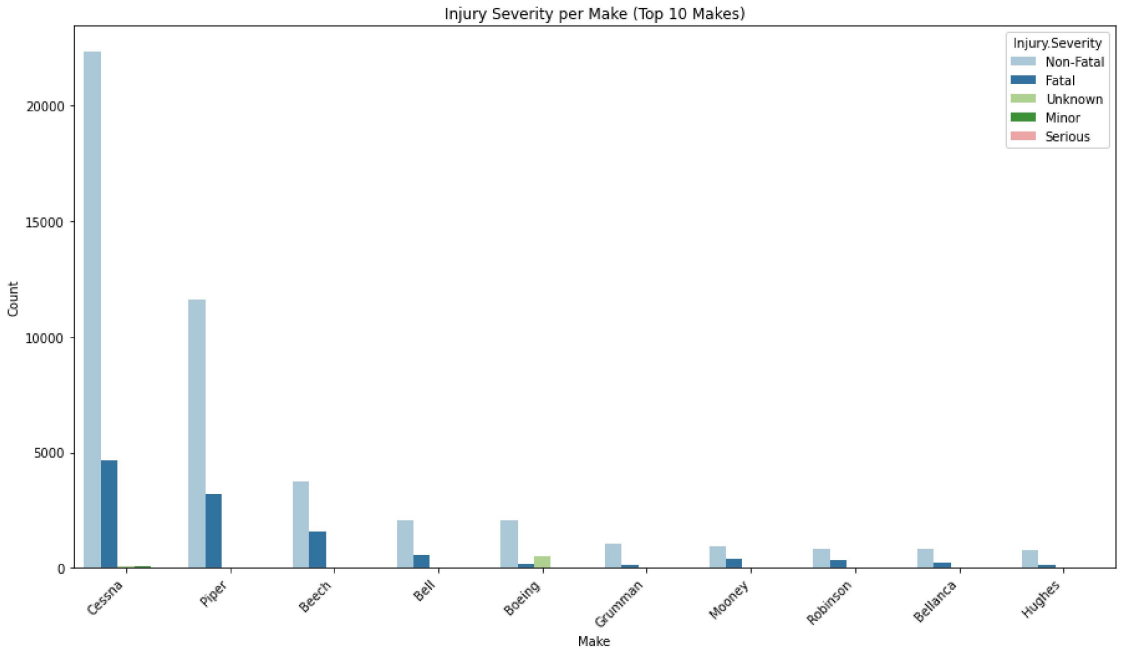
In [87]:
```
# Filter for the top 10 makes with the highest number of events
top_10_makes = make_counts.head(10).index
bottom_10_makes = make_counts.tail(10).index
# make_counts = aviation['Make'].value_counts()
print(top_10_makes)

# only have top 10 makes
filtered_top_10_makes = injury_severity_by_make[injury_severity_by_make['M
```

```
Index(['Cessna', 'Piper', 'Beech', 'Boeing', 'Bell', 'Mooney', 'Robinso
n',
       'Grumman', 'Bellanca', 'Hughes'],
      dtype='object')
```

In [88]:
```
bottom_10_makes
```

Out[88]:
```
Index(['Fuller/Findlay', 'Harris-Runyan', 'Mcmurtrie', 'Davis/Crouch',
'Greer',
       'Eachon', 'Hastings', 'Spezio', 'Earl F. Folsom', 'Bartle'],
      dtype='object')
```

In [89]:
```
# Create a bar plot
plt.figure(figsize=(15, 8))
sns.barplot(x='Make', y='Count', hue='Injury.Severity', data=filtered_top_
plt.title('Injury Severity per Make (Top 10 Makes)')
plt.xlabel('Make')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [90]:

```
# no injuries by make
no_injury_make = aviation.groupby(
    ['Make', 'Total.Uninjured']).size().reset_index(
        name='Count').sort_values(
            by=['Count'], ascending=[False]
            )
no_injury_make.head(10)
```

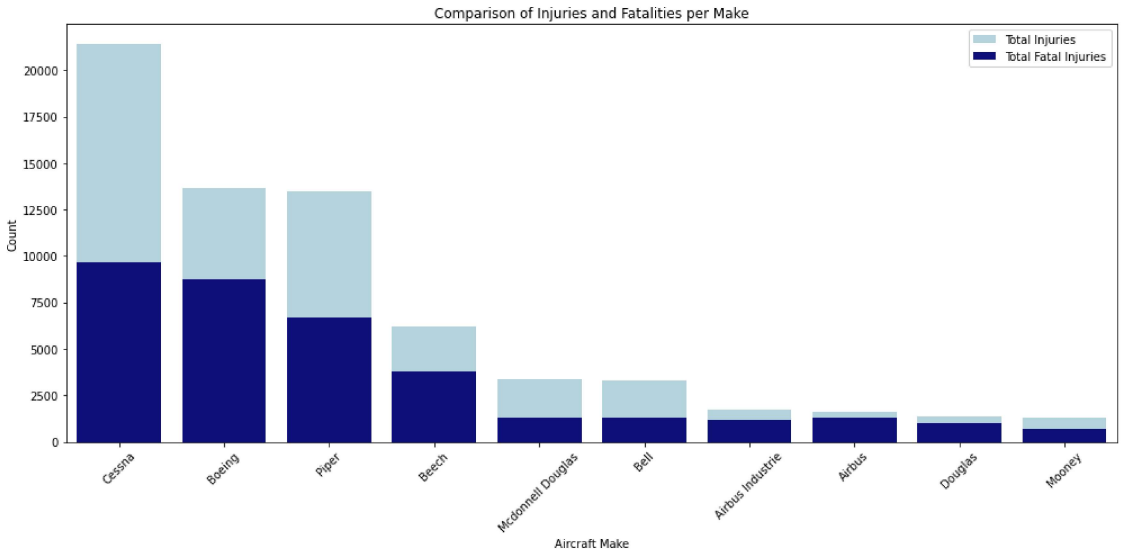
Out[90]:

	Make	Total.Uninjured	Count
2493	Cessna	0	9320
2494	Cessna	1	9010
8127	Piper	0	5888
2495	Cessna	2	5520
8128	Piper	1	4040
8129	Piper	2	3152
1158	Beech	0	2408
2496	Cessna	3	1586
1210	Bell	0	1259
1159	Beech	1	1089

b. Compare Injuries and Fatalities per Make

In [91]:

```
# df with make and injuries,fatalities
make_injuries_fatalities = aviation.groupby('Make').agg({
    'Total.Fatal.Injuries': 'sum',
    'Total.Serious.Injuries': 'sum',
    'Total.Minor.Injuries': 'sum'
}).reset_index()
# make_injuries_fatalities

# sum all injuries and sort descending
make_injuries_fatalities['Total.Injuries'] = make_injuries_fatalities['Tot
make_injuries_fatalities = make_injuries_fatalities.sort_values(by='Total.
# make_injuries_fatalities

# plot the top 10 makes by total injuries and fatalities
plt.figure(figsize=(14, 7))
sns.barplot(x='Make', y='Total.Injuries', data=make_injuries_fatalities.he
sns.barplot(x='Make', y='Total.Fatal.Injuries', data=make_injuries_fatalit
plt.xlabel('Aircraft Make')
plt.ylabel('Count')
plt.title('Comparison of Injuries and Fatalities per Make')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



Damage to Aircraft by Make

Insights

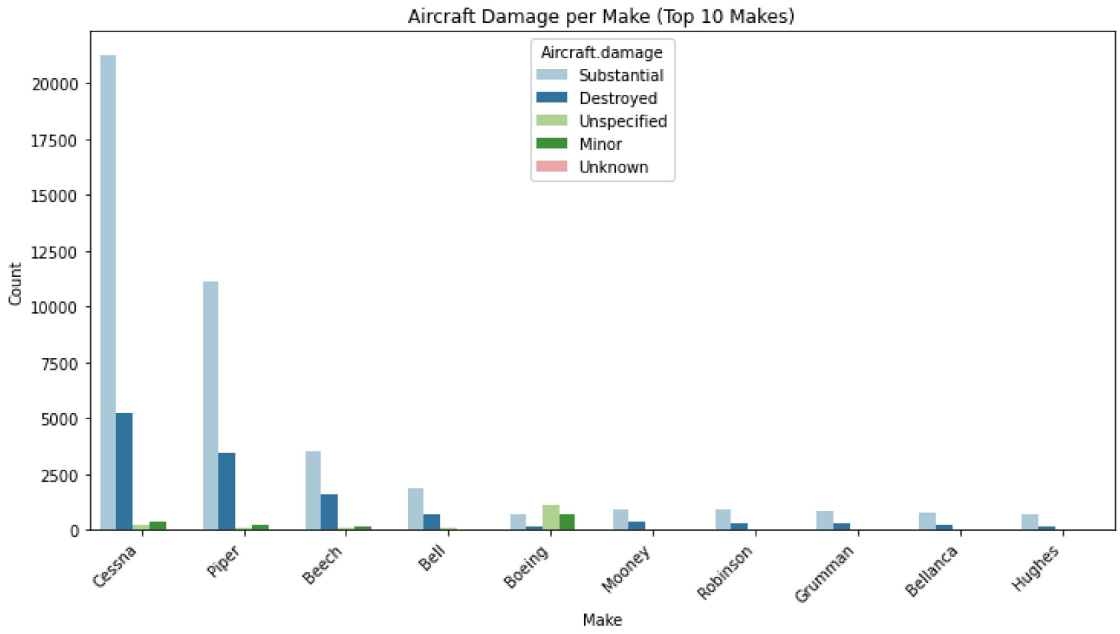
- Most Cessna, Piper, Beech and Bell had the highest Substantial and Destroyed damages.
- Notably, Boeing had higher minor destruction counts than other makes.

In [92]:

```
# group by make and aircraft damage
make_damage = aviation.groupby(['Make', 'Aircraft.damage']).size().reset_i

# get only top 10 makes by event counts
filtered_make_damage = make_damage[make_damage['Make'].isin(top_10_makes)]
print(filtered_make_damage)
print('*'*150)
# top_10_makes = make_counts.head(10).index

# bar chart
plt.figure(figsize=(12, 6))
sns.barplot(x='Make', y='Count', hue='Aircraft.damage', data=filtered_make
plt.title('Aircraft Damage per Make (Top 10 Makes)')
plt.xlabel('Make')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```

	Make	Aircraft.damage	Count
808	Beech	Destroyed	1585
809	Beech	Minor	170
810	Beech	Substantial	3539
811	Beech	Unknown	5
812	Beech	Unspecified	73
838	Bell	Destroyed	708
839	Bell	Minor	47
840	Bell	Substantial	1900
841	Bell	Unknown	5
842	Bell	Unspecified	62
881	Bellanca	Destroyed	233
882	Bellanca	Minor	7
883	Bellanca	Substantial	800
884	Bellanca	Unspecified	5
1037	Boeing	Destroyed	170
1038	Boeing	Minor	711
1039	Boeing	Substantial	739
1040	Boeing	Unknown	11
1041	Boeing	Unspecified	1114
1554	Cessna	Destroyed	5199
1555	Cessna	Minor	387
1556	Cessna	Substantial	21268
1557	Cessna	Unknown	38
1558	Cessna	Unspecified	254
3426	Grumman	Destroyed	299
3427	Grumman	Minor	18
3428	Grumman	Substantial	845
3429	Grumman	Unspecified	10
4015	Hughes	Destroyed	190
4016	Hughes	Minor	10
4017	Hughes	Substantial	718
4018	Hughes	Unspecified	14
5555	Mooney	Destroyed	373
5556	Mooney	Minor	24
5557	Mooney	Substantial	923
5558	Mooney	Unknown	3
5559	Mooney	Unspecified	11
6166	Piper	Destroyed	3427
6167	Piper	Minor	204
6168	Piper	Substantial	11100
6169	Piper	Unknown	15
6170	Piper	Unspecified	123
6737	Robinson	Destroyed	282
6738	Robinson	Minor	8
6739	Robinson	Substantial	916
6740	Robinson	Unknown	4
6741	Robinson	Unspecified	20

****
****
******



b. The Severity of Injuries in Certain Weather Conditions

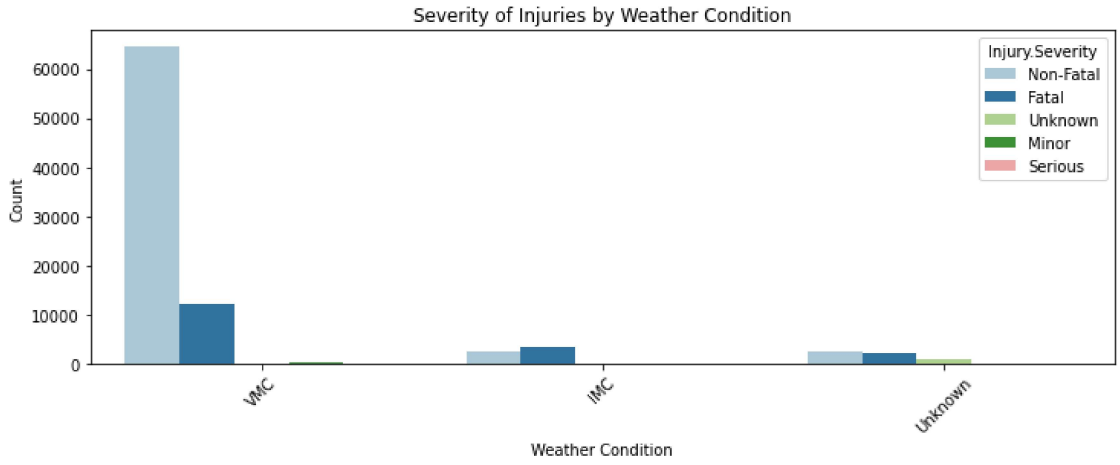
Insights

- Most accidents are expected to happen during bad weather due to poor visibilty.
- IMC generrally means that the weather is bad and therefore, more accidents expected due to poor visibility
- VMC generally means that the visibility is good and clouds are clear, allowing the aircraft to fly by visual reference.
- The figure below shows that most of the accidents that occur during IMC are fatal whereas the majority of accidents that occur in VMC are Non-Fatal.
- The high count of events in VMC matches up with the fact that summer is considered as generally good weather to fly.

In [93]:

```
# Severity of the injuries by weather condition
injuries_weather = aviation.groupby(['Weather.Condition', 'Injury.Severity
print(injuries_weather)
print('*'*150)

plt.figure(figsize=(12, 4))
sns.countplot(x='Weather.Condition', hue='Injury.Severity', data=aviation,
plt.title('Severity of Injuries by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

```
   Weather.Condition Injury.Severity  Count
0                IMC           Fatal   3459
1                IMC           Minor      4
2                IMC       Non-Fatal   2493
3                IMC         Serious      3
4                IMC         Unknown     14
5            Unknown           Fatal   2098
6            Unknown           Minor     45
7            Unknown       Non-Fatal   2432
8            Unknown         Serious     38
9            Unknown         Unknown    995
10               VMC           Fatal  12263
11               VMC           Minor    169
12               VMC       Non-Fatal  64650
13               VMC         Serious    132
14               VMC         Unknown     87
**********************************************************************
**********************************************************************
******
```



b. Severity of Injuries by Broad Phase of Flight

Insights

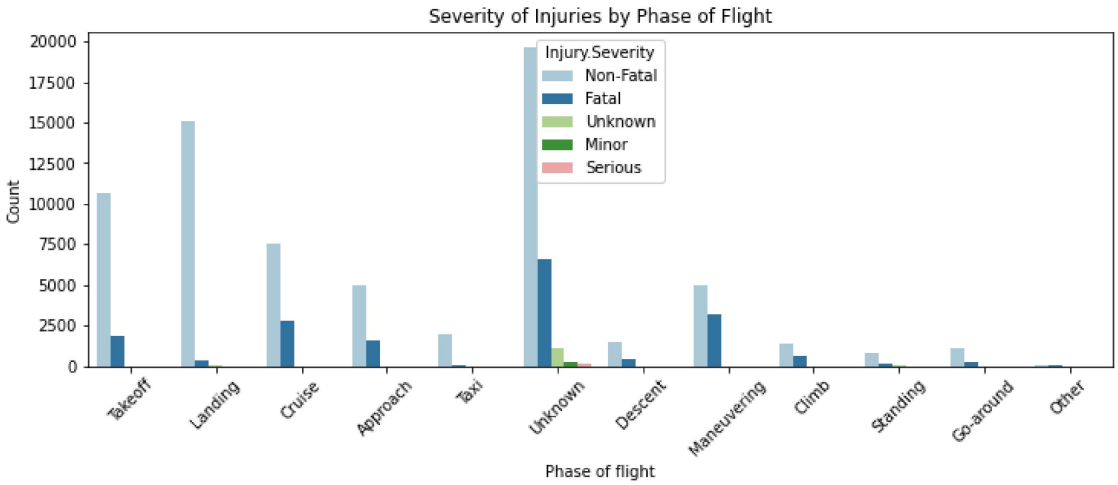
- The figure shows that many events occured during landing and take-off.
- In both cases, non-fatal injuries were more than fatal and other injuries.
- Cruise, approach and maneuvering also show significant fatal and non-fatal events.

N/B: The significant number of events with `Unknown` Phase of Flight (27,713, 31%) may indicate data recording issues that can be improved. All minor, serious and Unknown injury severities were all under Unknown Phase of Flight.

In [94]:

```
phase_flight_severity = aviation.groupby(['Broad.phase.of.flight', 'Injury
print(phase_flight_severity)
print('*'*150)

# bar chart
plt.figure(figsize=(12, 4))
sns.countplot(x='Broad.phase.of.flight', hue='Injury.Severity', data=aviat
plt.title('Severity of Injuries by Phase of Flight')
plt.xlabel('Phase of flight')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

```
   Broad.phase.of.flight Injury.Severity  Count
0               Approach           Fatal   1596
1               Approach       Non-Fatal   4949
2                  Climb           Fatal    618
3                  Climb       Non-Fatal   1415
4                 Cruise           Fatal   2778
5                 Cruise       Non-Fatal   7488
6                Descent           Fatal    457
7                Descent       Non-Fatal   1430
8              Go-around           Fatal    267
9              Go-around       Non-Fatal   1086
10               Landing           Fatal    291
11               Landing       Non-Fatal  15136
12               Landing         Unknown      1
13           Maneuvering           Fatal   3183
14           Maneuvering       Non-Fatal   4961
15                 Other           Fatal     45
16                 Other       Non-Fatal     74
17              Standing           Fatal    101
18              Standing       Non-Fatal    842
19              Standing         Unknown      2
20               Takeoff           Fatal   1824
21               Takeoff       Non-Fatal  10669
22                  Taxi           Fatal     39
23                  Taxi       Non-Fatal   1919
24               Unknown           Fatal   6621
25               Unknown           Minor    218
26               Unknown       Non-Fatal  19606
27               Unknown         Serious    173
28               Unknown         Unknown   1093
******************************************************************************
******************************************************************************
******
```



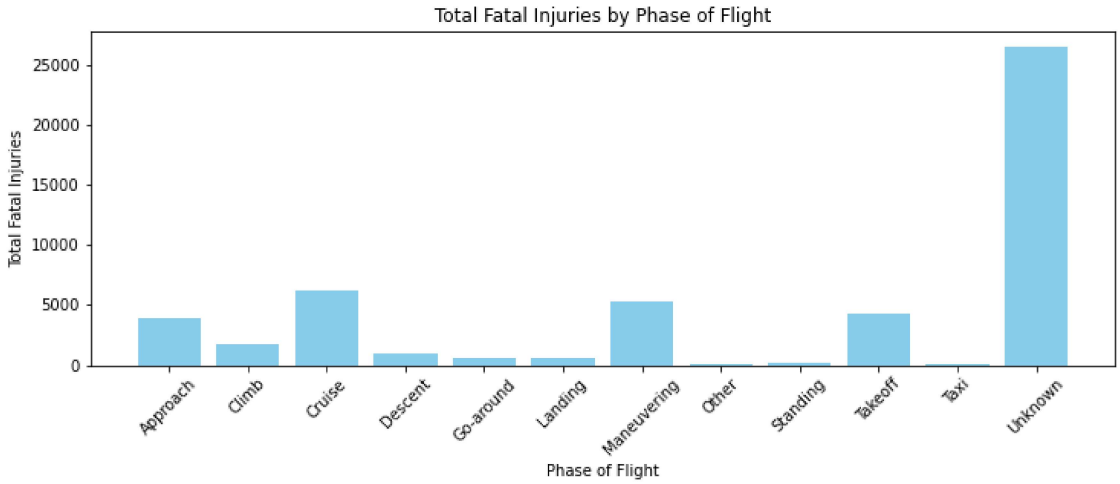
c. Fatalities by Broad Phase of Flight

- Many fatalities occured during Cruise, Maneuvering (turning, climbing, or descending close to the ground), Takeoff and Approach.

In [95]:

```
phase_flight_fatalities = aviation.groupby('Broad.phase.of.flight')['Total
print(phase_flight_fatalities)
print('*'*150)

# bar chart
plt.figure(figsize=(12, 4))
plt.bar(phase_flight_fatalities['Broad.phase.of.flight'], phase_flight_fat
plt.xlabel('Phase of Flight')
plt.ylabel('Total Fatal Injuries')
plt.title('Total Fatal Injuries by Phase of Flight')
plt.xticks(rotation=45)
plt.show()
```

```
   Broad.phase.of.flight  Total.Fatal.Injuries
0               Approach                  3841
1                  Climb                  1762
2                 Cruise                  6166
3                Descent                   913
4              Go-around                   587
5                Landing                   518
6            Maneuvering                  5323
7                  Other                    85
8               Standing                   161
9                Takeoff                  4304
10                  Taxi                   102
11               Unknown                 26423
******************************************************************************
******************************************************************************
******
```



In [ ]:

```
#check relationship between Injury.Severity, Total.Fatal.Injuries and Broa
sns.scatterplot(x='Injury.Severity', y='Total.Fatal.Injuries', hue='Broad.
plt.title("injury severity by phase of flight")
plt.show()
```