Aviation Incident Analysis for Safety Optimization

Project Overview

This project focuses on assessing aviation incident and accident data to identify patterns, trends, and risk factors that contribute to aircraft safety events. By checking into structured records which includes; year, location, aircraft type and model, engine type,aircraft damage, amateur built, flight purpose, weather conditions and injury severity. The analysis aims to establish insights that can enhance aviation safety. The findings are intended to support improvements in policy, pilot training, aircraft design, and operational procedures. The key stakeholders include aviation authorities (e.g., FAA, KCAA), aircraft manufacturers, airlines, flight schools, safety analysts, data scientists, and regulators. The project operates within the aviation and aerospace industry, specifically in the domain of safety analytics and accident investigation, serving departments such as safety oversight, risk management, and data science.

Business Problem

Even with modern improvements in aviation technology, safety incidents still continue to occur across different types of aircraft and in various parts of the world. Stakeholders don't yet have a clear, data-based understanding of which factors like aircraft model, engine type, or location contribute to serious accidents. This gap in insight risks misdirecting safety interventions and limiting their effectiveness. This project aims to identify trends in incident severity over time and geography, find out which aircraft models and engine types are most often involved, assess the effect of weather changes on aviation incidences, explore how different aircraft features relate to accident types and determine the number of aviation incidences originating from amateur built aircrafts. The goal is to offer useful insights that can improve safety rules and pilot training. Success will be measured by the clear identification of high-risk aircraft models or regions, creation of easy-to-understand visuals that show trends, get safety teams to use the findings, and a potential long-term reduction in incident rates.

Data Understanding

The dataset used in this project is sourced from aviation safety reports from the National Transportation Safety Board (NTSB). It contains structured information including the year of each incident, the location (city and country), the type of incident (Fatal, Non-Fatal, Substantial), aircraft type (e.g., airplane, helicopter), specific aircraft models (e.g., Cessna_172, Piper_PA-28), engine types (Reciprocating, Turboprop, Jet), and other identifiers such as registration numbers or codes, models, make. The data spans multiple formats, including categorical variables (like incident type and aircraft model), temporal data (year), and geographic data (location), making it suitable for comprehensive analysis of aviation safety trends. The Data spans from 1962 -2023 with a total of 90,348 entries (31 columns)

In [6]:

```
# initial Preview
import pandas as pd

df = pd.read_csv("c:/Users/User/dsc-phase-1-project-v3/data/aviation_data.csv")
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_7728\3046178936.py:4: DtypeWarning: Columns (6,7,28) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("c:/Users/User/dsc-phase-1-project-v3/data/aviation_data.csv")
```

In [7]:

```
# Set low_memory=False - to tell pandas to read the file
df = pd.read_csv("c:/Users/User/dsc-phase-1-project-v3/data/aviation_data.csv", low_memory=False)
```

In [8]:

```
# treating columns with different data types as strings to avoid errors and ensures consistent data types for ana
lysis and visualization
df = pd.read_csv("c:/Users/User/dsc-phase-1-project-v3/data/aviation_data.csv", dtype={
    'column_name_6': str,
    'column_name_7': str,
    'column_name_28': str
},
low_memory=False
)
```

In [9]:

```
# column index of the dataframe
print(df.columns.tolist())
```

```
['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date', 'Location', 'Country', 'Latitude
', 'Longitude', 'Airport.Code', 'Airport.Name', 'Injury.Severity', 'Aircraft.damage', 'Aircraft.Cate
gory', 'Registration.Number', 'Make', 'Model', 'Amateur.Built', 'Number.of.Engines', 'Engine.Type',
'FAR.Description', 'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries', 'Total.Se
rious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition', 'Broad.phase.of.fli
ght', 'Report.Status', 'Publication.Date']
```

In [10]:

```
# summary of the columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      90348 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50132 non-null  object 
 9   Airport.Name            52704 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87507 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82505 non-null  object 
 30  Publication.Date        73659 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

In [11]:

```
df.head()
```

Out[11]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airp
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	

5 rows × 31 columns

In [12]:

```
# checking duplicated data
df.duplicated().sum()
```

Out[12]:

1390

Data Cleaning/Preparation

The data will be cleaned and standardized by handling missing values, correcting inconsistent text formats, and ensuring uniform data types across key columns. This is aimed at improving data quality, reduce ambiguity and enable reliable analysis of incident patterns and severity.

Cleaning Steps

- Standardizing column names
- Handling Duplicates
- Handling missing values

Standardizing Column Names

In [13]:

```
## Standardizing columns
aviation = pd.read_csv("aviation_data.csv", low_memory=False)
aviation['Weather.Condition'] = aviation['Weather.Condition'].str.title().str.strip()
aviation['Injury.Severity'] = aviation['Injury.Severity'].str.title().str.strip()
```

In [14]:

```
## Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
```

In [17]:

```
## clean location column
df['country'] = df['country'].str.strip().str.upper()
df['location'] = df['location'].str.strip().str.upper()
```

In [ ]:

```
## Convert date column
df['event.date'] = pd.to_datetime(df['event.date'], errors='coerce')
df['year'] = df['event.date'].dt.year
```

Handling Duplicate Data

In [ ]:

```
## Checking for duplicates
duplicates = df[df.duplicated()]
print(len(duplicates))
duplicates.head()
```

1390

Out[ ]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	.
64050	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	.
64052	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	.
64388	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	.
64541	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	.
64552	NaN	25-09-2020	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	.

5 rows × 31 columns

In [ ]:

```
## checking for duplicate rows based on Event.Id subset
duplicates = df[df.duplicated(subset = 'Event.Id')]
print(len(duplicates))
duplicates.tail()
```

2396

Out[ ]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.I
90097	NaN	20-12-2022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
90236	20221112106276	Accident	CEN23MA034	2022-11-12	Dallas, TX	United States	324026N	0965146W	RBD	[Exe(
90255	20221121106336	Accident	WPR23LA041	2022-11-18	Las Vegas, NV	United States	361239N	1151140W	VGT	NORTH VE
90257	20221122106340	Incident	DCA23WA071	2022-11-18	Marrakech,	Morocco	NaN	NaN	NaN	
90273	20221123106354	Accident	WPR23LA045	2022-11-22	San Diego, CA	United States	323414N	1165825W	SDM	Brown Mur A

5 rows × 31 columns

In [ ]:

```
## checking for duplicate rows based on Accident.Number sub set
duplicates = df[df.duplicated(subset = 'Accident.Number')]
print(len(duplicates))
duplicates.tail()
```

1484

Out[ ]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpo
90097	NaN	20-12-2022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
90236	20221112106276	Accident	CEN23MA034	2022-11-12	Dallas, TX	United States	324026N	0965146W	RBD	E
90255	20221121106336	Accident	WPR23LA041	2022-11-18	Las Vegas, NV	United States	361239N	1151140W	VGT	NOR
90257	20221122106340	Incident	DCA23WA071	2022-11-18	Marrakech,	Morocco	NaN	NaN	NaN	
90273	20221123106354	Accident	WPR23LA045	2022-11-22	San Diego, CA	United States	323414N	1165825W	SDM	Bro M

5 rows × 31 columns

In [ ]:

```
## Drop duplicates
df.drop_duplicates(inplace=True)
```

Handling Missing Values

This is targeting a few columns; Aircraft Model, Engine Type and Injury Severity

1. Aircraft Model

Replacing the missing value 'NaN' with the word 'Unknown' . This keeps the data clean and consistent label, keeping the row data hence preventing errors during modelling.

In [ ]:

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

In [ ]:

```
df.fillna({'model': 'Unknown'}, inplace=True)
```

2. Engine Type

Replacing the missing value 'NaN' with the most common value 'Mode'. This fills the missing values with the most common category, keeping the column consistent hence improves model performance

In [ ]:

```
df['engine.type'] = df['engine.type'].fillna(df['engine.type'].mode()[0])
```

3. Injury Severity

Replacing the missing value 'NaN' with the word 'Substantial'. Substantial is a reasonable word to describe the severity of the accident. This prevents data loss, hence consistency in reporting.

In [20]:

```
df['injury.severity'] = df['injury.severity'].fillna('Substantial')
```

In [18]:

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

Data Analysis

To enhance reduction in accident incidents, we seek to find out; how incident numbers have changed over time, which aircraft models are most frequently involved in fatal incidents, whether certain engine types are more prone to severe outcomes, which countries or regions report the highest number of incidents, whether there is a correlation between adverse weather conditions and severity of aviation accidents, whether there is a relationship between aircraft type and incident severity and which aviation accident incidents originate from amateur built aircrafts. To answer these, the project uses visual tools such as time series plots to show trends over the years, bar charts to highlight high-risk aircraft models, heatmaps to explore severity by engine type, geographic maps to visualize incident density across regions, and pie charts to illustrate the distribution of incident types. These visualizations help communicate insights clearly and support data-driven decision-making for aviation safety.

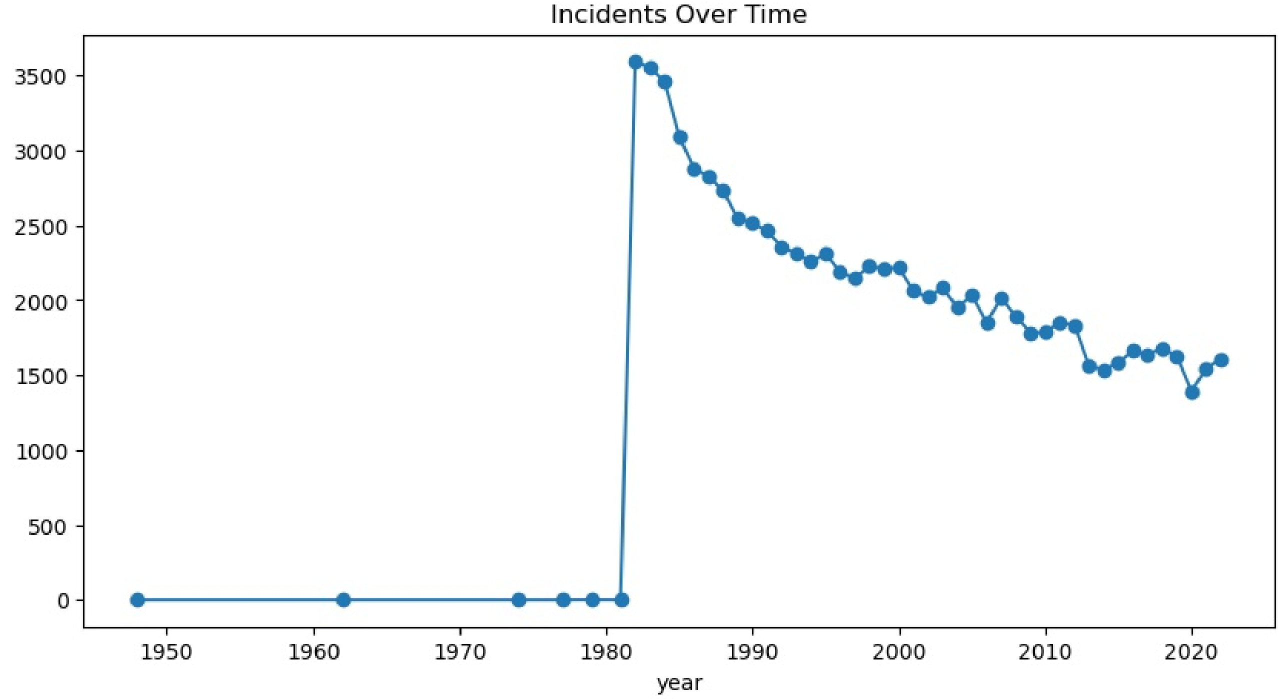
1. Incident Trends Over Time

In [ ]:

```
incident_trend = df.groupby('year').size()

incident_trend.plot(kind='line', marker='o', figsize=(10, 5), title='Incidents Over Time')
```

Out[ ]:

```
<Axes: title={'center': 'Incidents Over Time'}, xlabel='year'>
```



2. Top Aircraft Models in Fatal Incidents

In [ ]:

```
df[df['injury.severity'] == 'fatal']
```

Out[ ]:

	event.id	investigation.type	accident.number	event.date	location	country	latitude	longitude	airport.code	airport.name	...	air.carrie

0 rows × 32 columns

In [ ]:

```
df['injury.severity'].unique()
```

Out[ ]:

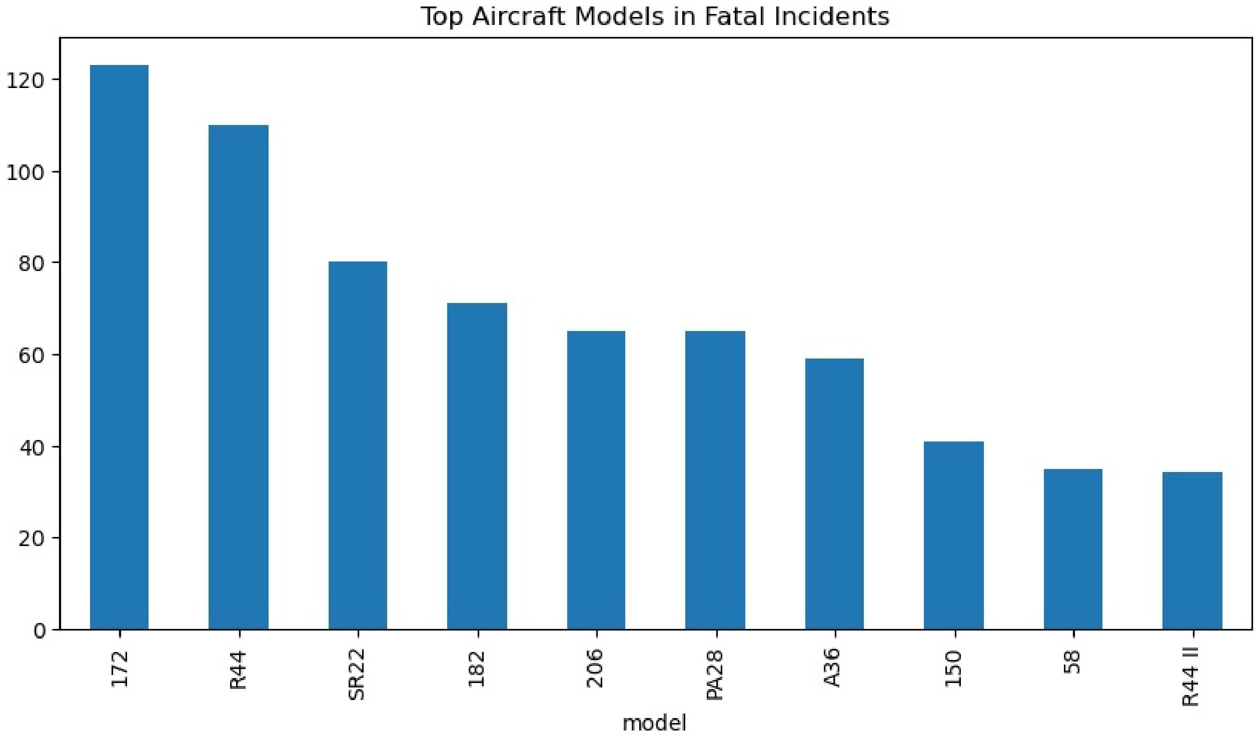
```
array(['Fatal(2)', 'Fatal(4)', 'Fatal(3)', 'Fatal(1)', 'Non-Fatal',
       'Incident', 'Fatal(8)', 'Fatal(78)', 'Fatal(7)', 'Fatal(6)',
       'Fatal(5)', 'Fatal(153)', 'Fatal(12)', 'Fatal(14)', 'Fatal(23)',
       'Fatal(10)', 'Fatal(11)', 'Fatal(9)', 'Fatal(17)', 'Fatal(13)',
       'Fatal(29)', 'Fatal(70)', 'Unavailable', 'Fatal(135)', 'Fatal(31)',
       'Fatal(256)', 'Fatal(25)', 'Fatal(82)', 'Fatal(156)', 'Fatal(28)',
       'Fatal(18)', 'Fatal(43)', 'Fatal(15)', 'Fatal(270)', 'Fatal(144)',
       'Fatal(174)', 'Fatal(111)', 'Fatal(131)', 'Fatal(20)', 'Fatal(73)',
       'Fatal(27)', 'Fatal(34)', 'Fatal(87)', 'Fatal(30)', 'Fatal(16)',
       'Fatal(47)', 'Fatal(56)', 'Fatal(37)', 'Fatal(132)', 'Fatal(68)',
       'Fatal(54)', 'Fatal(52)', 'Fatal(65)', 'Fatal(72)', 'Fatal(160)',
       'Fatal(189)', 'Fatal(123)', 'Fatal(33)', 'Fatal(110)',
       'Fatal(230)', 'Fatal(97)', 'Fatal(349)', 'Fatal(125)', 'Fatal(35)',
       'Fatal(228)', 'Fatal(75)', 'Fatal(104)', 'Fatal(229)', 'Fatal(80)',
       'Fatal(217)', 'Fatal(169)', 'Fatal(88)', 'Fatal(19)', 'Fatal(60)',
       'Fatal(113)', 'Fatal(143)', 'Fatal(83)', 'Fatal(24)', 'Fatal(44)',
       'Fatal(64)', 'Fatal(92)', 'Fatal(118)', 'Fatal(265)', 'Fatal(26)',
       'Fatal(138)', 'Fatal(206)', 'Fatal(71)', 'Fatal(21)', 'Fatal(46)',
       'Fatal(102)', 'Fatal(115)', 'Fatal(141)', 'Fatal(55)',
       'Fatal(121)', 'Fatal(45)', 'Fatal(145)', 'Fatal(117)',
       'Fatal(107)', 'Fatal(124)', 'Fatal(49)', 'Fatal(154)', 'Fatal(96)',
       'Fatal(114)', 'Fatal(199)', 'Fatal(89)', 'Fatal(57)', 'Fatal',
       'Substantial', 'Minor', 'Serious'], dtype=object)
```

In [ ]:

```
fatal_models = df[df['injury.severity'].str.lower() == 'fatal']['model'].value_counts().head(10)
```

In [ ]:

```
if not fatal_models.empty:
    fatal_models.plot(kind='bar', figsize=(10, 5), title='Top Aircraft Models in Fatal Incidents')
else:
    print("No fatal incidents found in the dataset.")
```



3. Engine Type vs Severity

In [ ]:

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site-pac
kages (0.13.2)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\user\.anaconda\conda3\envs\learn-env
\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site
-packages (from seaborn) (2.3.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\user\.anaconda\conda3\envs\learn-
env\lib\site-packages (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\.anaconda\conda3\envs\learn-env\lib
\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\.anaconda\conda3\envs\learn-env\li
b\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\.anaconda\conda3\envs\learn-env\li
b\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\
site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site-p
ackages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\user\.anaconda\conda3\envs\learn-env\lib
\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\.anaconda\conda3\envs\learn-env
\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\user\.anaconda\conda3\envs\lea
rn-env\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (6.4.0)
Requirement already satisfied: zipp>=3.1.0 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site
-packages (from importlib-resources>=3.2.0->matplotlib!=3.6.1,>=3.4->seaborn) (3.21.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\sit
e-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\s
ite-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site-pa
ckages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
```

In [ ]:

```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site-pack
ages (6.3.1)
Requirement already satisfied: narwhals>=1.15.1 in c:\users\user\.anaconda\conda3\envs\learn-env\lib
\site-packages (from plotly) (2.6.0)
Requirement already satisfied: packaging in c:\users\user\.anaconda\conda3\envs\learn-env\lib\site-p
ackages (from plotly) (25.0)
Note: you may need to restart the kernel to use updated packages.
```

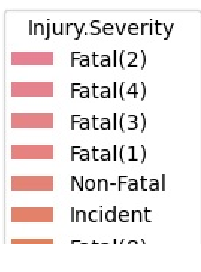
In [ ]:

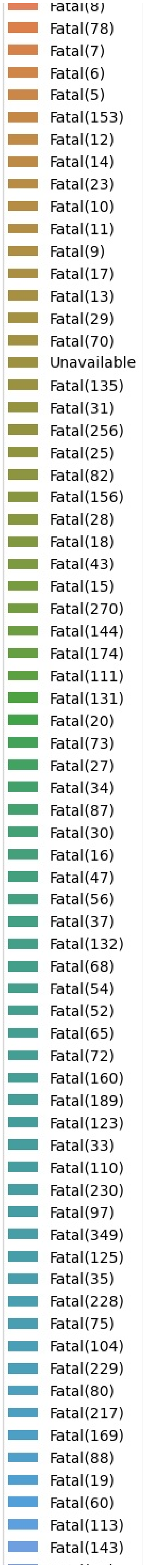
```python
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import pandas as pd


df = pd.read_csv("c:/Users/User/dsc-phase-1-project-v3/data/aviation_data.csv")

plt.figure(figsize=(10, 5))
sns.countplot(data=df, x='Engine.Type', hue='Injury.Severity')
plt.title('Incident Severity by Engine Type')
plt.figure(figsize=(8, 4))  # Width = 8 inches, Height = 4 inches
```

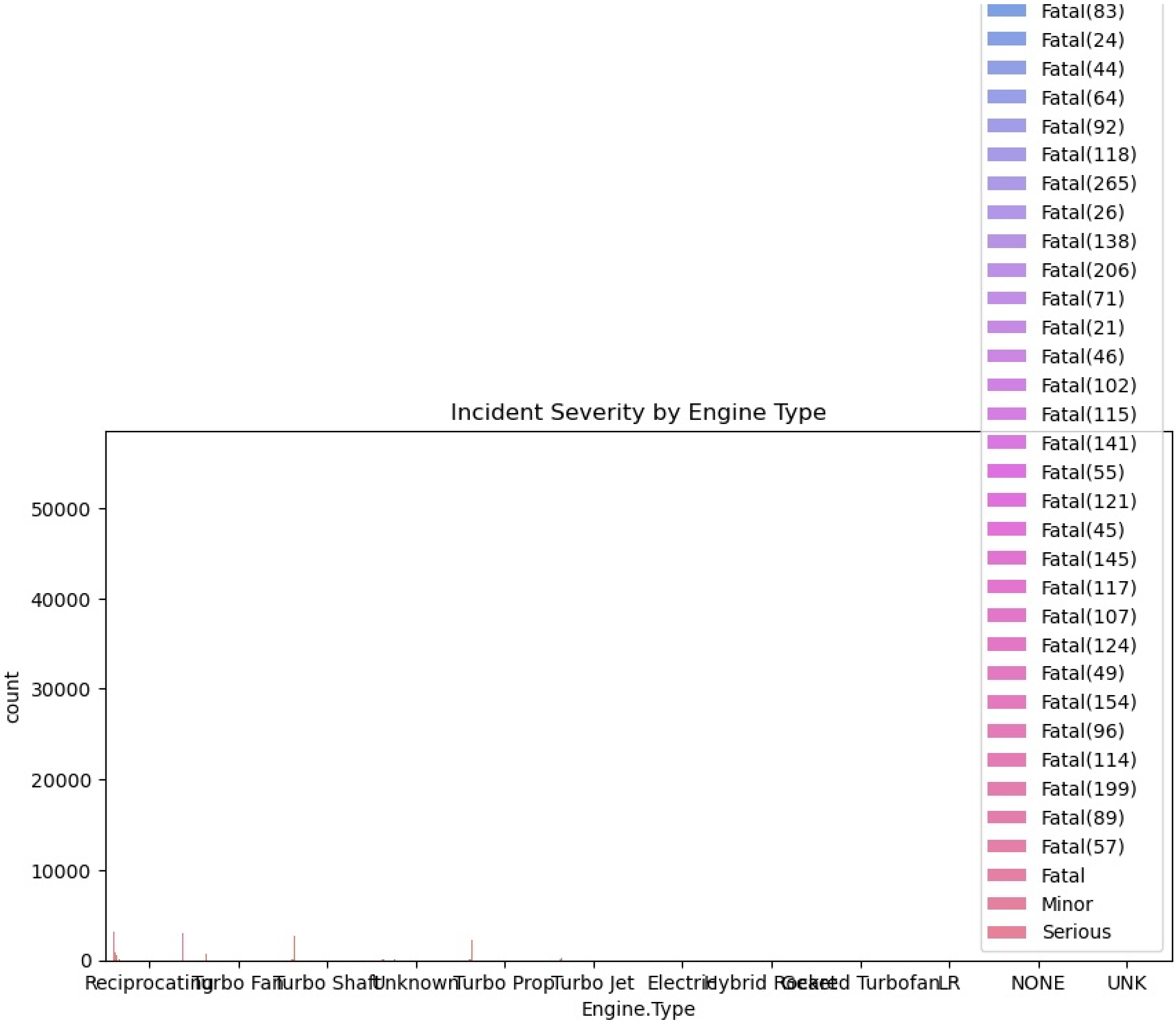
```
C:\Users\User\AppData\Local\Temp\ipykernel_16960\2572090593.py:7: DtypeWarning: Columns (6,7,28) hav
e mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("c:/Users/User/dsc-phase-1-project-v3/data/aviation_data.csv")
```

Out[ ]:

```
<Figure size 800x400 with 0 Axes>
```




Fatal(8)
Fatal(78)
Fatal(7)
Fatal(6)
Fatal(5)
Fatal(153)
Fatal(12)
Fatal(14)
Fatal(23)
Fatal(10)
Fatal(11)
Fatal(9)
Fatal(17)
Fatal(13)
Fatal(29)
Fatal(70)
Unavailable
Fatal(135)
Fatal(31)
Fatal(256)
Fatal(25)
Fatal(82)
Fatal(156)
Fatal(28)
Fatal(18)
Fatal(43)
Fatal(15)
Fatal(270)
Fatal(144)
Fatal(174)
Fatal(111)
Fatal(131)
Fatal(20)
Fatal(73)
Fatal(27)
Fatal(34)
Fatal(87)
Fatal(30)
Fatal(16)
Fatal(47)
Fatal(56)
Fatal(37)
Fatal(132)
Fatal(68)
Fatal(54)
Fatal(52)
Fatal(65)
Fatal(72)
Fatal(160)
Fatal(189)
Fatal(123)
Fatal(33)
Fatal(110)
Fatal(230)
Fatal(97)
Fatal(349)
Fatal(125)
Fatal(35)
Fatal(228)
Fatal(75)
Fatal(104)
Fatal(229)
Fatal(80)
Fatal(217)
Fatal(169)
Fatal(88)
Fatal(19)
Fatal(60)
Fatal(113)
Fatal(143)




```
<Figure size 800x400 with 0 Axes>
```

Region and the fatal incidences

In [ ]:

```
# Count incidents by country
country_counts = df['Country'].value_counts().reset_index()
country_counts.columns = ['Country', 'Incident_Count']
print(country_counts.head())
```

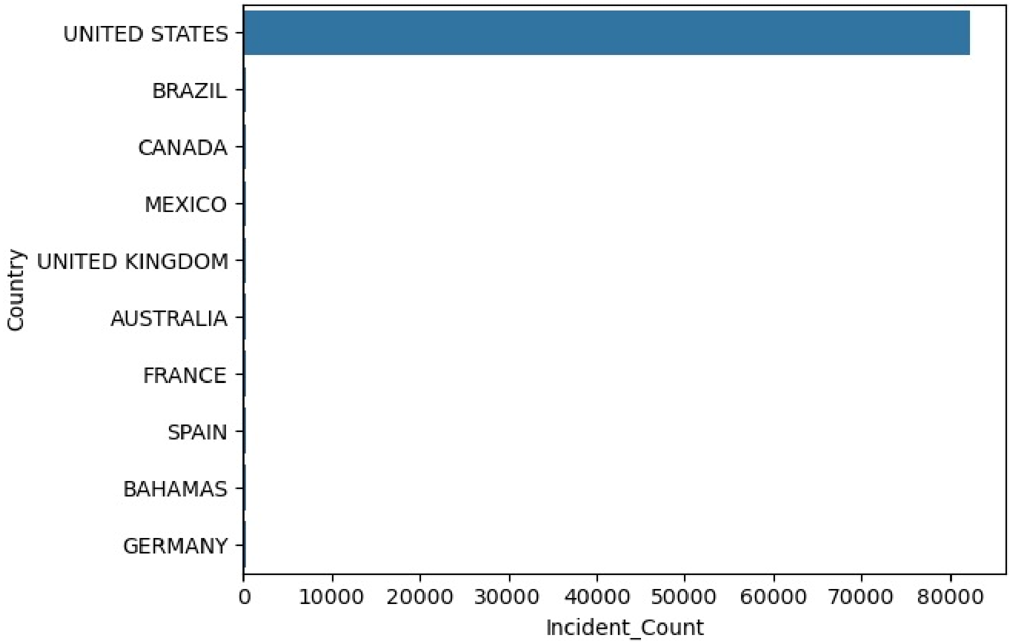
```
          Country  Incident_Count
0   UNITED STATES           82248
1          BRAZIL             374
2          CANADA             359
3          MEXICO             358
4  UNITED KINGDOM             344
```

In [ ]:

```
# Top 10 countries by incident count
top_countries = country_counts.head(10)

sns.barplot(data=top_countries, x='Incident_Count', y='Country')
```

Out[ ]:

```
<Axes: xlabel='Incident_Count', ylabel='Country'>
```



A relationship between aircraft Category and incident severity

In [ ]:

```
# Create aircraft type by combining Make and Model
df['Aircraft_Type'] = df['Make'] + " " + df['Model']
```

In [ ]:

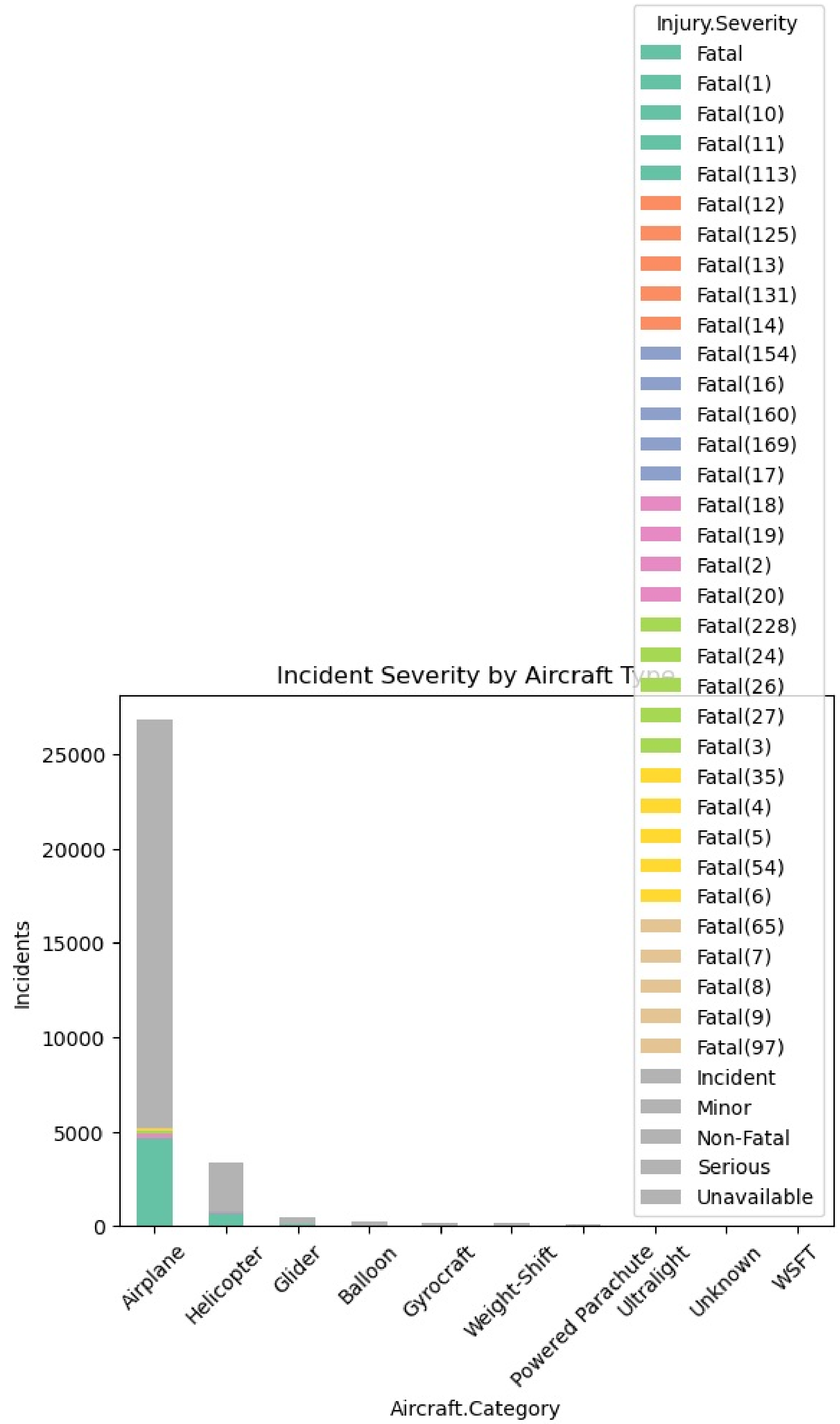
```
# Count incidents by aircraft type and severity
type_severity_counts = df.groupby(['Aircraft.Category', 'Injury.Severity']).size().reset_index(name='Incident_Cou
nt')
```

In [ ]:

```
pivot_df = type_severity_counts.pivot(index='Aircraft.Category', columns='Injury.Severity', values='Incident_Coun
t').fillna(0)

# focus on top 10 aircraft types by total incidents
top_types = pivot_df.sum(axis=1).sort_values(ascending=False).head(10).index
filtered_df = pivot_df.loc[top_types]
```

In [ ]:

```
filtered_df.plot.bar(stacked=True, colormap='Set2')
plt.title("Incident Severity by Aircraft Type")
plt.ylabel("Incidents")
plt.xticks(rotation=45)
plt.show()
```



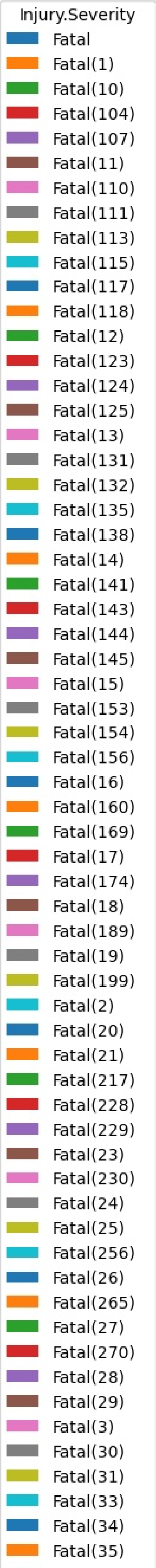
The correlation between weather patterns and incident severity

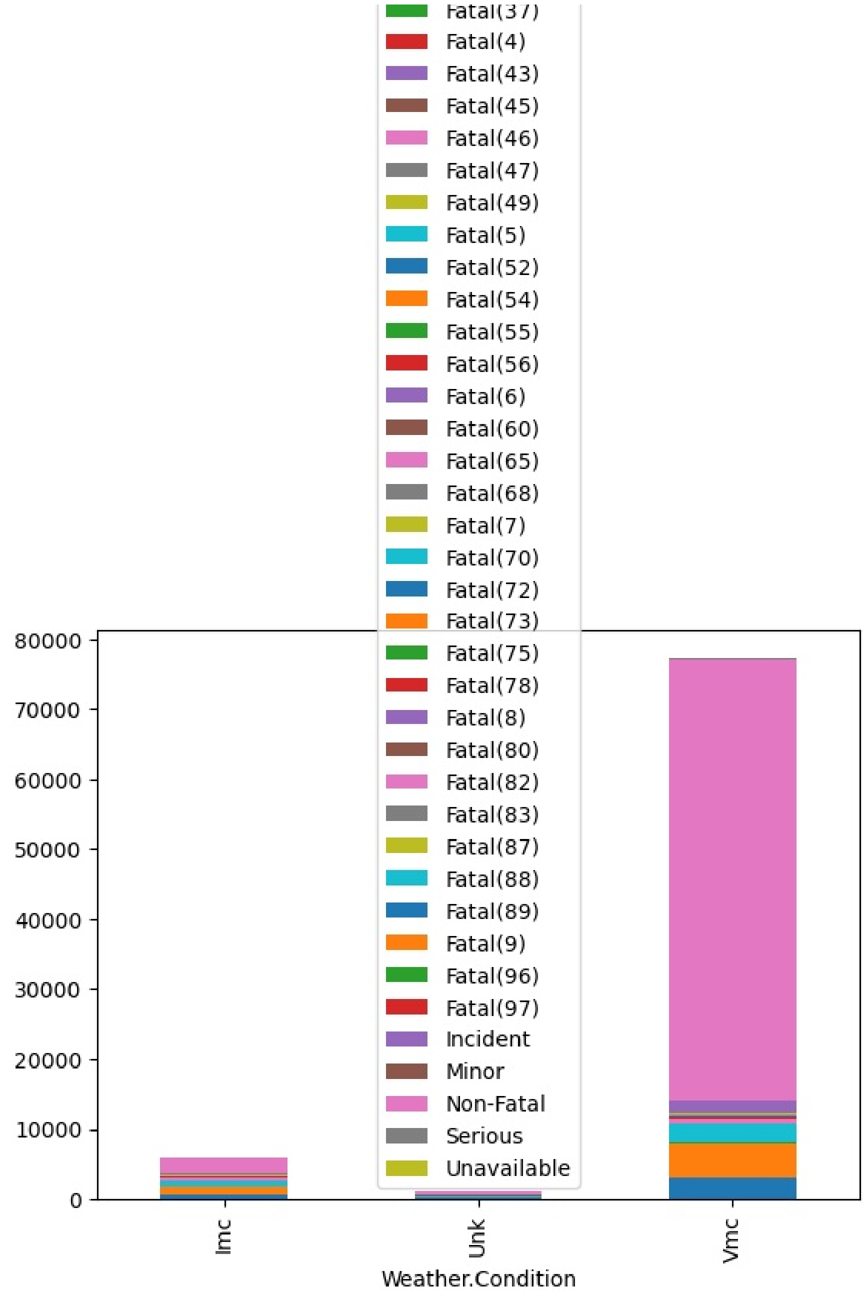
In [ ]:

```
# Groupby
weather_severity = aviation.groupby(['Weather.Condition', 'Injury.Severity']).size().reset_index(name='Incident_C
ount')

pivot_df = weather_severity.pivot(index='Weather.Condition', columns='Injury.Severity', values='Incident_Count').
fillna(0)
```

In [ ]:

```
pivot_df.plot(kind='bar', stacked=True)
plt.show()
```





Aviation incidences in relation to amateur built aircrafts

In [ ]:

```
# Standardize the 'Amateur_Built' column for consistent analysis
aviation['Amateur.Built'] = aviation['Amateur.Built'].str.title().str.strip()
```

In [ ]:

```
# Incidence Counts
build_counts = aviation['Amateur.Built'].value_counts()
print(build_counts)
```

```
Amateur.Built
No     80312
Yes     8475
Name: count, dtype: int64
```

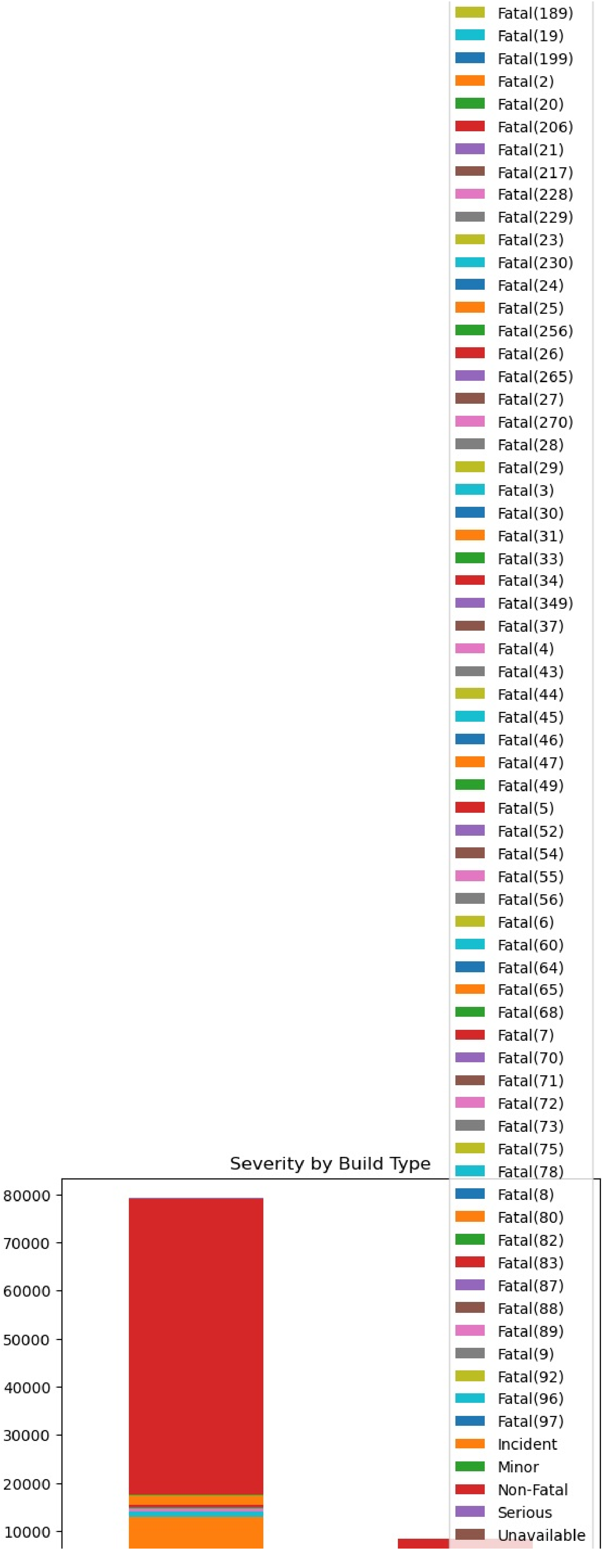
In [ ]:

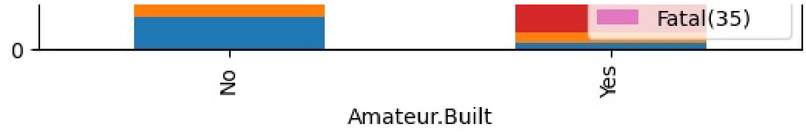
```python
# Group data by build type and injury severity, then count incidents
severity_by_build = aviation.groupby(['Amateur.Built', 'Injury.Severity']).size().unstack().fillna(0)
print(severity_by_build)
```

```
Injury.Severity   Fatal  Fatal(1)  Fatal(10)  Fatal(102)  Fatal(104)  \
Amateur.Built
No               4559.0    4945.0       31.0         2.0         1.0
Yes               703.0    1211.0        0.0         0.0         0.0

Injury.Severity  Fatal(107)  Fatal(11)  Fatal(110)  Fatal(111)  Fatal(113)  \
Amateur.Built
No                      1.0       10.0         1.0         1.0         2.0
Yes                     0.0        0.0         0.0         0.0         0.0

Injury.Severity  ...  Fatal(9)  Fatal(92)  Fatal(96)  Fatal(97)  Incident  \
Amateur.Built    ...
No               ...      18.0        2.0        1.0        2.0    2150.0
Yes              ...       0.0        0.0        0.0        0.0      25.0

Injury.Severity  Minor  Non-Fatal  Serious  Unavailable  Fatal(35)
Amateur.Built
No               196.0    61298.0    153.0         80.0        0.0
Yes               22.0     6044.0     20.0          6.0        1.0

[2 rows x 107 columns]
```

In [ ]:

```python
# Create a stacked bar chart to visualize severity across build types
severity_by_build.plot(kind='bar', stacked=True)
plt.title("Severity by Build Type")
plt.show()
```


Severity by Build Type
80000
70000
60000
50000
40000
30000
20000
10000
Fatal(189)
Fatal(19)
Fatal(199)
Fatal(2)
Fatal(20)
Fatal(206)
Fatal(21)
Fatal(217)
Fatal(228)
Fatal(229)
Fatal(23)
Fatal(230)
Fatal(24)
Fatal(25)
Fatal(256)
Fatal(26)
Fatal(265)
Fatal(27)
Fatal(270)
Fatal(28)
Fatal(29)
Fatal(3)
Fatal(30)
Fatal(31)
Fatal(33)
Fatal(34)
Fatal(349)
Fatal(37)
Fatal(4)
Fatal(43)
Fatal(44)
Fatal(45)
Fatal(46)
Fatal(47)
Fatal(49)
Fatal(5)
Fatal(52)
Fatal(54)
Fatal(55)
Fatal(56)
Fatal(6)
Fatal(60)
Fatal(64)
Fatal(65)
Fatal(68)
Fatal(7)
Fatal(70)
Fatal(71)
Fatal(72)
Fatal(73)
Fatal(75)
Fatal(78)
Fatal(8)
Fatal(80)
Fatal(82)
Fatal(83)
Fatal(87)
Fatal(88)
Fatal(89)
Fatal(9)
Fatal(92)
Fatal(96)
Fatal(97)
Incident
Minor
Non-Fatal
Serious
Unavailable



Conclusion

Time Trends

• Incident frequency shows seasonal variation, with peaks during months of increased flight activity (e.g., summer and holiday seasons).

• Long-term trends suggest a gradual decline in total incidents, possibly due to improved safety protocols and technology—but fatal incidents persist, especially in general aviation.

Aircraft Models

• A small number of high-usage models (e.g., Cessna 172, Piper PA-28) account for a large share of incidents.

• These models are often used in training and private flights, which may correlate with less experienced pilots or less stringent maintenance oversight.

Engine Types

• Single-engine aircraft are disproportionately represented in fatal and serious incidents.

• Turboprop and jet engines show fewer incidents per flight hour, suggesting better performance under stress and more robust safety systems.

Aircraft Categories

• Airplanes dominate the dataset, but helicopters and experimental aircraft show higher severity rates when incidents occur.

• Amateur-built aircraft have elevated risk profiles, often linked to mechanical failure or pilot error.

These insights can guide targeted safety audits, training programs, and engine modernization efforts.

Recommendations

Time-Based Safety Interventions

• Increase seasonal safety campaigns during high-traffic period, targeting private and recreational pilots.

• Use historical incident data to forecast risky periods and allocate inspection resources accordingly.

Model-Specific Oversight

• Conduct targeted audits and maintenance reviews for frequently involved models like the Cessna 172.

• Encourage manufacturers to analyze incident data and improve design or training materials for high-risk models.

Engine-Type Risk Mitigation

• Mandate additional training for pilots operating single-engine aircraft, especially in adverse weather conditions.

• Promote engine redundancy upgrades or enhanced emergency protocols for older single-engine fleets.

Aircraft Category Focus

• Require stricter certification and inspection for amateur-built and experimental aircraft.

• Develop category-specific safety dashboards for regulators to monitor trends and intervene early.