Aviation Accident Database & Synopses, up to 2023 Data Analysis

Project Overview

This project analyzes the `Aviation Accident Database & Synopses` `dataset`, which contains records of civil aviation accidents and selected incidents from 1962 to 2023.

The goal is to extract meaningful insights through data cleaning, transformation, and visualization to support business decisions regarding which airplane types to consider for commercial and private use.

By identifying accident trends, evaluating safety records, and understanding the contributing factors to aviation incidents, this analysis aims to guide a new aviation division in selecting the safest and most suitable aircraft models for operation.

Project Objectives

This project seeks to answer key business questions by:

- Identifying trends in airplane accidents and incidents.
- Highlighting low-risk airplane with the fewest accidents and fatalities.
- Analyzing contributing factors such as weather, mechanical failure or human error.
- Comparing safety records and identifying the best airplanes for commercial and private enterprises.
- Providing actionable recommendations to support airplane purchase decisions.

Data Understanding

The `dataset` for this analysis is a from `Kaggle` [Aviation Accident Database & Synopses, up to 2023](#) which covers civil aviation accidents and selected incidents from 1962 to 2023, in the United States and international waters.

It includes detailed information on:

- Event Accident\Incident date and location
- Severity of injuries and damage
- Weather conditions and flight phase
- Investigation outcomes

The data is stored in an CSV file (`AviationData.csv`) and requires cleaning and preprocessing before analysis

Business Understanding

The core business question is: `Which type of operating airplanes should be purchased for safe and reliable Commercial and Private operations?` To answer this, the analysis will explore the following sub-questions:

1. What are the specifications of aircrafts and filter airplane in the dataset?
2. How many accidents or incidents has each airplane been involved in and Top 10 safest airplane?
3. What were the causes of the accidents or the incidents and the level of damage sustained on the airplane?
4. Are the said safest airplanes useful for commercial and private operations?

Data Preparation

Requirements

- Load and preview the data Understand the structure and contents.
- Handle missing values Identify and treat nulls appropriately.
- Convert date fields Standardize time-related features.
- Aggregate and clean text data Normalize categories for consistency and easier analysis.

In [108]:

```
# Importing data using the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [109]:

```
# Loading and previewing the data
df = pd.read_csv("AviationData.csv", encoding ="cp1252", low_memory=False)
df
```

Out[109]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airp
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

88889 rows × 31 columns

In [110]:

```
# Checking the first five columns
df.head()
```

Out[110]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.C
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	N
				1962-07-	BRIDGEPORT,	United			

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Co
1	20001218X45447	Accident	LAX94LA336	19	CA	States	NaN	NaN	N
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	N
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	N
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	N

5 rows × 31 columns

In [111]:

```
# Checking the last 5 columns
df.tail()
```

Out[111]:

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

5 rows × 31 columns

In [112]:

```
# Checking the shape and dimensionality of the dataset.
df.shape
```

Out[112]:

```
(88889, 31)
```

- **The dataset contains 88889 records(rows) and 31 features(columns).**

In [113]:

```
#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            50132 non-null  object
 9   Airport.Name            52704 non-null  object
 10  Injury.Severity         87889 non-null  object
```

```
 10  Injury.Severity         87889 non-null  object
 11  Aircraft.damage         85695 non-null  object
 12  Aircraft.Category       32287 non-null  object
 13  Registration.Number     87507 non-null  object
 14  Make                    88826 non-null  object
 15  Model                   88797 non-null  object
 16  Amateur.Built           88787 non-null  object
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object
 19  FAR.Description         32023 non-null  object
 20  Schedule                12582 non-null  object
 21  Purpose.of.flight       82697 non-null  object
 22  Air.carrier             16648 non-null  object
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object
 28  Broad.phase.of.flight   61724 non-null  object
 29  Report.Status           82505 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [114]:

```
# Checking for the dataset information 2
df.info(verbose = False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Columns: 31 entries, Event.Id to Publication.Date
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

- **The columns in the dataset contain both string represented as object and decimal numbers as float. That is 5 numerical data and 26 categorical data.**
- **The data also contain dates `Publication.Date` identified as object.**
- **There are several columns with missing values. Records should be `88889` which is not the case for most columns.**

In [115]:

```
# Checking for statistical summary to get a better understanding of the dataset
df.describe()
```

Out[115]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.000000
mean	1.146585	0.647855	0.279881	0.357061	5.325440
std	0.446510	5.485960	1.544084	2.235625	27.913634
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	0.000000	2.000000
max	8.000000	349.000000	161.000000	380.000000	699.000000

In [116]:

```
# Checking for summaries in the categorical data.
df.describe(include = 'object')
```

Out[116]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airpo
count	88889	88889	88889	88889	88837	88663	34382	34373	
unique	87951	2	88863	14782	27758	219	25589	27154	
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	United States	332739N	0112457W	
freq	3	85015	2	25	434	82248	19	24	

4 rows × 26 columns

In [117]:

```
# Checking for column names
df.columns
```

Out[117]:

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

The columns names are partly clean a few needs cleaning. That is;

- **They have no special characters**
- **No white spaces**
- **Names are descriptive and meaningful**
- **They contain dots(.) which is fine. However,**
- **The title casing should be standardized**
- **Dates should be converted to Datetime**

In [118]:

```
# Checking for duplicates
df.duplicated().sum()
```

Out[118]:

```
0
```

- **The dataset has no duplicates**

In [119]:

```
df.isna().sum()
```

Out[119]:

```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
Airport.Code            38757
Airport.Name            36185
Injury.Severity          1000
Aircraft.damage          3194
```

```
Aircraft.Category         56602
Registration.Number        1382
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7096
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6384
Publication.Date          13771
dtype: int64
```

- **The are missing values in most columns in this dataset.**

In [120]:

```
# Creating a data frame copy for use in data cleaning
df1 = df.copy(deep = True)
df1
```

Out[120]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airp
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

88889 rows × 31 columns

In [121]:

```
# rechecking the column names
df1.columns
```

Out[121]:

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

In [122]:

```
# Standardizing the column name cases
df1.columns= df1.columns.str.title()
df1.columns
```

Out[122]:

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

Question 1 : What are the specifications of aircrafts and filter airplane in the dataset?

- **To identify will specification of the aircraft types, I will use the following columns;** `Make`, `Model`, `Number.of.Engines`, `Engine.Type`, `Registration.Number`, `Aircraft.Category`, `Amateu` **and** `Air.Carrier`
- **I will filter by** `Aircraft.Category` **for** `Airplane`**.**

Data Preparation

In [123]:

```
# Checking for unique values in each categorical column.
columns1= ["Make","Model","Engine.Type","Registration.Number","Aircraft.Category","Amateu
r.Built","Far.Description", "Air.Carrier","Number.Of.Engines"]
unique_values ={col: df1[col].unique() for col in columns1}
for col, values in unique_values.items():
    print(f"\n{col}:\n{values}\n")
```

```
Make:
['Stinson' 'Piper' 'Cessna' ... 'JAMES R DERNOVSEK' 'ORLICAN S R O'
 'ROYSE RALPH L']


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


Engine.Type:
['Reciprocating' nan 'Turbo Fan' 'Turbo Shaft' 'Unknown' 'Turbo Prop'
 'Turbo Jet' 'Electric' 'Hybrid Rocket' 'Geared Turbofan' 'LR' 'NONE'
 'UNK']


Registration.Number:
```

```
Registration.Number:
['NC6404' 'N5069P' 'N5142R' ... 'N749PJ' 'N210CU' 'N9026P']


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


Amateur.Built:
['No' 'Yes' nan]


Far.Description:
[nan 'Part 129: Foreign' 'Part 91: General Aviation'
 'Part 135: Air Taxi & Commuter' 'Part 125: 20+ Pax,6000+ lbs'
 'Part 121: Air Carrier' 'Part 137: Agricultural'
 'Part 133: Rotorcraft Ext. Load' 'Unknown' 'Part 91F: Special Flt Ops.'
 'Non-U.S., Non-Commercial' 'Public Aircraft' 'Non-U.S., Commercial'
 'Public Use' 'Armed Forces' 'Part 91 Subpart K: Fractional' '091' 'NUSC'
 '135' 'NUSN' '121' '137' '129' '133' '091K' 'UNK' 'PUBU' 'ARMF' '103'
 '125' '437' '107']


Air.Carrier:
[nan 'Air Canada' 'Rocky Mountain Helicopters, In' ...
 'SKY WEST AVIATION INC TRUSTEE' 'GERBER RICHARD E' 'MC CESSNA 210N LLC']


Number.Of.Engines:
[ 1. nan  2.  0.  3.  4.  8.  6.]
```

In [124]:

```
# Checking missing values
df1[columns1].isna().sum()
```

Out[124]:

```
Make                      63
Model                     92
Engine.Type             7096
Registration.Number     1382
Aircraft.Category      56602
Amateur.Built            102
Far.Description        56866
Air.Carrier            72241
Number.Of.Engines       6084
dtype: int64
```

Data Cleaning

In [125]:

```
# cleaning each column by removing any white spaces, updating the cases, characters
df1["Make"]= df1["Make"].str.strip().str.title().str.replace(" ", ".")
df1["Make"].unique()
```

Out[125]:

```
array(['Stinson', 'Piper', 'Cessna', ..., 'James.R.Dernovsek',
       'Orlican.S.R.O', 'Royse.Ralph.L'], dtype=object)
```

In [126]:

```
# dealing with null values by dropping them since null values represent only 0.7% of reco
rds, hence not significant
df1 = df1.dropna(subset=["Make"])
df1["Make"].isna().sum()
```

Out[126]:

```
0
```

In [127]:

```
# cleaning Model same case with make with few missing null at 0.10%
df1= df1.dropna(subset=["Model"])
df1["Model"].isna().sum()
```

Out[127]:

```
0
```

In [128]:

```
# Engine types
df1["Engine.Type"].value_counts()
```

Out[128]:

```
Engine.Type
Reciprocating      69496
Turbo Shaft         3609
Turbo Prop          3390
Turbo Fan           2478
Unknown             2048
Turbo Jet            703
Geared Turbofan       12
Electric              10
LR                     2
NONE                   2
Hybrid Rocket          1
UNK                    1
Name: count, dtype: int64
```

In [129]:

```
# standardizing casing and removing white spaces
df1["Engine.Type"]= df1["Engine.Type"].str.strip().str.title()
df1["Engine.Type"].value_counts()
```

Out[129]:

```
Engine.Type
Reciprocating      69496
Turbo Shaft         3609
Turbo Prop          3390
Turbo Fan           2478
Unknown             2048
Turbo Jet            703
Geared Turbofan       12
Electric              10
Lr                     2
None                   2
Hybrid Rocket          1
Unk                    1
Name: count, dtype: int64
```

In [130]:

```
# replacing none,Unk with unknown and Lr with Long Range(Domain Knowledge)
df1["Engine.Type"]= df1["Engine.Type"].replace({"None":"Unknown",
                                                "Unk":"Unknown",
                                                "Lr":"Long Range"})
df1["Engine.Type"].isna().sum()
```

Out[130]:

```
7025
```

In [131]:

```
# fillna missing values with unknown since engine type cannot be assumed, and its truthfu
l
df1["Engine.Type"].fillna("Unknown", inplace=True)
df1["Engine.Type"].isna().sum()
```

Out[131]:

```
0
```

In [132]:

```
# Registration Number replacing NONE and UNK with UNKNOWN since they represent the same t
hing
df1["Registration.Number"]= df1["Registration.Number"].replace({
    "NONE": "UNKNOWN",
    "UNK": "UNKNOWN"})
```

In [133]:

```
# fillna with UNKNOWN to avoid making assumptions
df1["Registration.Number"].fillna("UNKNOWN", inplace=True)
df1["Registration.Number"].isna().sum()
```

Out[133]:

```
0
```

In [134]:

```
#Aircraft Category replacing WSFT and ULTR for Weight-Shift and Ultralight respectively(d
omain knowledge)
# UNK for Unknown and standardizing casing
df1["Aircraft.Category"]= df1["Aircraft.Category"].str.title()
df1["Aircraft.Category"].replace({"Unk":"Unknown",
                                  "Wsft":"Weight-Shift",
                                  "Ultr":"Ultralight"}, inplace=True)
df1["Aircraft.Category"].value_counts()
```

Out[134]:

```
Aircraft.Category
Airplane            27580
Helicopter           3435
Glider                508
Balloon               231
Gyrocraft             173
Weight-Shift          170
Powered Parachute      91
Ultralight             31
Unknown                16
Powered-Lift            5
Blimp                   4
Rocket                  1
Name: count, dtype: int64
```

In [135]:

```
# fillna unknown for missing values to avoid assumptions
df1["Aircraft.Category"].fillna("Unknown", inplace=True)
df1["Aircraft.Category"].isna().sum()
```

Out[135]:

```
0
```

In [136]:

```
# Amateur Built fillna with unknown although nulls represent only 0.11% of the records.
# I prefer to make it unknown without making any assumptions
df1["Amateur.Built"].fillna("Unknown", inplace=True)
df1["Amateur.Built"].isna().sum()
```

Out[136]:

0

In [137]:

```
#Far Desription, contain numericals, i will map, to correct format and then replace them
df1["Far.Description"].unique()
```

Out[137]:

```
array([nan, 'Part 129: Foreign', 'Part 91: General Aviation',
       'Part 135: Air Taxi & Commuter', 'Part 125: 20+ Pax,6000+ lbs',
       'Part 121: Air Carrier', 'Part 137: Agricultural',
       'Part 133: Rotorcraft Ext. Load', 'Unknown',
       'Part 91F: Special Flt Ops.', 'Non-U.S., Non-Commercial',
       'Public Aircraft', 'Non-U.S., Commercial', 'Public Use',
       'Armed Forces', 'Part 91 Subpart K: Fractional', '091', 'NUSC',
       '135', 'NUSN', '121', '137', '129', '133', '091K', 'UNK', 'PUBU',
       'ARMF', '103', '125', '437', '107'], dtype=object)
```

In [138]:

```
#mapping to correct format
F_map = {"Part 91: General Aviation": "Part 91: General Aviation",
    "091": "Part 91: General Aviation",
    "091K": "Part 91: Fractional",
    "Part 91 Subpart K: Fractional": "Part 91: Fractional",

    "Part 121: Air Carrier": "Part 121: Air Carrier",
    "121": "Part 121: Air Carrier",

    "Part 135: Air Taxi & Commuter": "Part 135: Air Taxi",
    "135": "Part 135: Air Taxi",

    "Part 129: Foreign": "Part 129: Foreign",
    "129": "Part 129: Foreign",

    "Part 137: Agricultural": "Part 137: Agricultural",
    "137": "Part 137: Agricultural",

    "Part 125: 20+ Pax,6000+ lbs": "Part 125: Large Aircraft",
    "125": "Part 125: Large Aircraft",

    "Part 133: Rotorcraft Ext. Load": "Part 133: Rotorcraft",
    "133": "Part 133: Rotorcraft",

    "Part 91F: Special Flt Ops.": "Part 91F: Special Flight Ops",

    "Non-U.S., Non-Commercial": "Foreign: Non-Commercial",
    "Non-U.S., Commercial": "Foreign: Commercial",

    "Public Aircraft": "Public Use",
    "Public Use": "Public Use",
    "PUBU": "Public Use",

    "Armed Forces": "Military",
    "ARMF": "Military",
    "NUSC": "Military",
    "NUSN": "Military",

    "103": "Other",
    "107": "Other",
    "437": "Other",

    "UNK": "Unknown",
    "Unknown": "Unknown"}

df1["Far.Description"]= df1["Far.Description"].replace(F_map)
df1["Far.Description"].nunique()
```

Out[138]:

15

In [139]:

```
# filling the missing values with unknown
df1["Far.Description"].fillna("Unknown", inplace=True)
df1["Far.Description"].value_counts()
```

Out[139]:

```
Far.Description
Unknown                          57225
Part 91: General Aviation        24682
Military                          2568
Part 137: Agricultural            1445
Part 135: Air Taxi                1043
Part 121: Air Carrier              839
Part 129: Foreign                  342
Public Use                         274
Part 133: Rotorcraft               139
Foreign: Non-Commercial             96
Foreign: Commercial                 91
Part 91: Fractional                 15
Part 125: Large Aircraft            10
Other                                7
Part 91F: Special Flight Ops         1
Name: count, dtype: int64
```

In [140]:

```
# Air Carrier, standardizing cases, checking for value counts
df1["Air.Carrier"] = df1["Air.Carrier"].str.title()
df1["Air.Carrier"].value_counts().head(20)
```

Out[140]:

```
Air.Carrier
Pilot                         258
American Airlines              89
United Airlines                89
Delta Air Lines                53
Delta Air Lines Inc            44
Southwest Airlines Co          44
American Airlines Inc          36
On File                        33
Continental Airlines           27
Ryanair                        27
Private Individual             27
American Airlines, Inc.        25
Usair                          24
Southwest Airlines             23
United Air Lines Inc           23
Continental Airlines, Inc.     21
Air Methods Corp               20
Air Canada                     20
Unknown                        17
Civil Air Patrol Inc           17
Name: count, dtype: int64
```

In [141]:

```
# mapping the air carrier to be able to replace the repetitions and check counts
C_map ={"American Airlines Inc": "American Airlines",
    "American Airlines, Inc.": "American Airlines",
    "Delta Air Lines Inc": "Delta Air Lines",
    "United Air Lines Inc": "United Airlines",
    "Southwest Airlines Co": "Southwest Airlines",
    "Continental Airlines, Inc.": "Continental Airlines",
    "Pilot": "Private Individual",
    "On File": "Unknown",
    "Unknown": "Unknown"}
df1["Air.Carrier"].replace(C_map, inplace=True)
df1["Air.Carrier"].value_counts()
```

Out[141]:

```
Air.Carrier
Private Individual    285
American Airlines     150
United Airlines       112
Delta Air Lines        97
Southwest Airlines     67
                     ...
Fabbri Nancy W          1
Nfss Inc                1
Williams Evan H         1
Dell Aero Inc           1
Mc Cessna 210N Llc      1
Name: count, Length: 13171, dtype: int64
```

In [142]:

```
# filling missing values with unknown, to avoid assumptions
df1["Air.Carrier"].fillna("Unknown", inplace= True)
df1["Air.Carrier"].value_counts()
```

Out[142]:

```
Air.Carrier
Unknown               72218
Private Individual      285
American Airlines       150
United Airlines         112
Delta Air Lines          97
                      ...
Mng Airlines              1
Fabbri Nancy W            1
Nfss Inc                  1
Williams Evan H           1
Mc Cessna 210N Llc        1
Name: count, Length: 13171, dtype: int64
```

In [143]:

```
#  checking counts for Number of Engines
df1["Number.Of.Engines"].value_counts()
```

Out[143]:

```
Number.Of.Engines
1.0    69538
2.0    11072
0.0     1226
3.0      483
4.0      431
8.0        3
6.0        1
Name: count, dtype: int64
```

In [144]:

```
# converting the column to numeric and fillna with unknown
df1["Number.Of.Engines"]= pd.to_numeric(df1["Number.Of.Engines"],errors="coerce")
df1["Number.Of.Engines"] = df1["Number.Of.Engines"].apply(lambda x: "Unknown" if pd.isna
(x) else str(int(x)))
df1["Number.Of.Engines"].isna().sum()
```

Out[144]:

```
0
```

Data Analysis

In [145]:

```
# Filtering Airplane from Aircraft.Category and rechecking counts
```

```
df1["Aircraft.Category"].value_counts()
```

Out[145]:

```
Aircraft.Category
Unknown              56548
Airplane             27580
Helicopter            3435
Glider                 508
Balloon                231
Gyrocraft              173
Weight-Shift           170
Powered Parachute       91
Ultralight              31
Powered-Lift             5
Blimp                    4
Rocket                   1
Name: count, dtype: int64
```

In [146]:

```
#Filtering Airplane and confirming shape
Airplanes_df1 = df1[df1["Aircraft.Category"] == "Airplane"]
Airplanes_df1.shape[0]
```

Out[146]:

```
27580
```

In [147]:

```
# creating a simple version of Aircraft Type to enhance readability and plotting in the n
ext questions
df1["Aircraft.Simple"] = (df1["Make"] + df1["Model"])
df1["Aircraft.Simple"].value_counts()
```

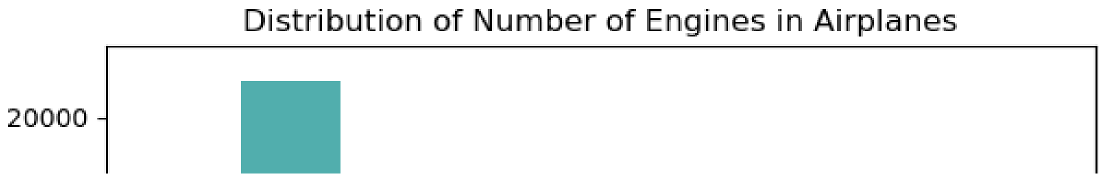
Out[147]:

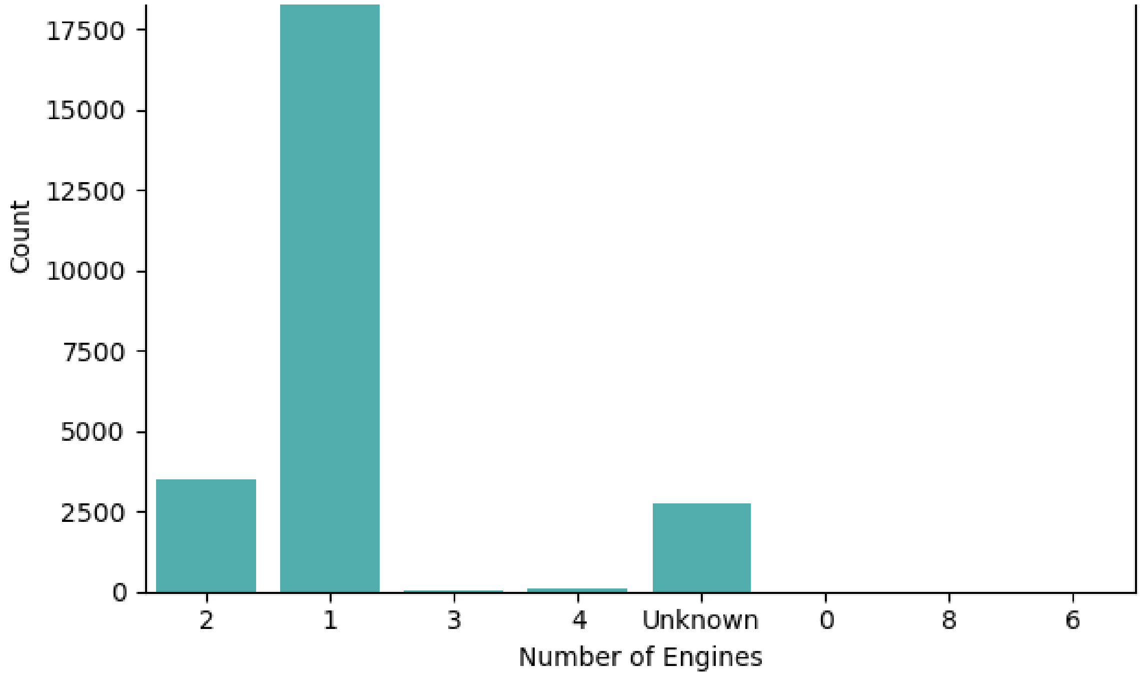
```
Aircraft.Simple
Cessna152                               2366
Cessna172                               1753
Cessna172N                              1163
PiperPA-28-140                           932
Cessna150                                829
                                        ...
BauerVANS RV-4                             1
VelocityVELOCITY ELITE RG                  1
IverslieKIT FOX                            1
Consolidated-VulteePBY-5A(28-5ACF)         1
Royse.Ralph.LGLASAIR                       1
Name: count, Length: 18465, dtype: int64
```

Univariate Analysis : Distribution of Number of Engines in Airplanes

In [148]:

```
# Distribution of Number of Engines in Airplanes using bar chart
Airplanes_df1 = df1[df1["Aircraft.Category"] == "Airplane"]

sns.countplot(data=Airplanes_df1, x="Number.Of.Engines", color='#42bdbc')
plt.title("Distribution of Number of Engines in Airplanes")
plt.xlabel("Number of Engines")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

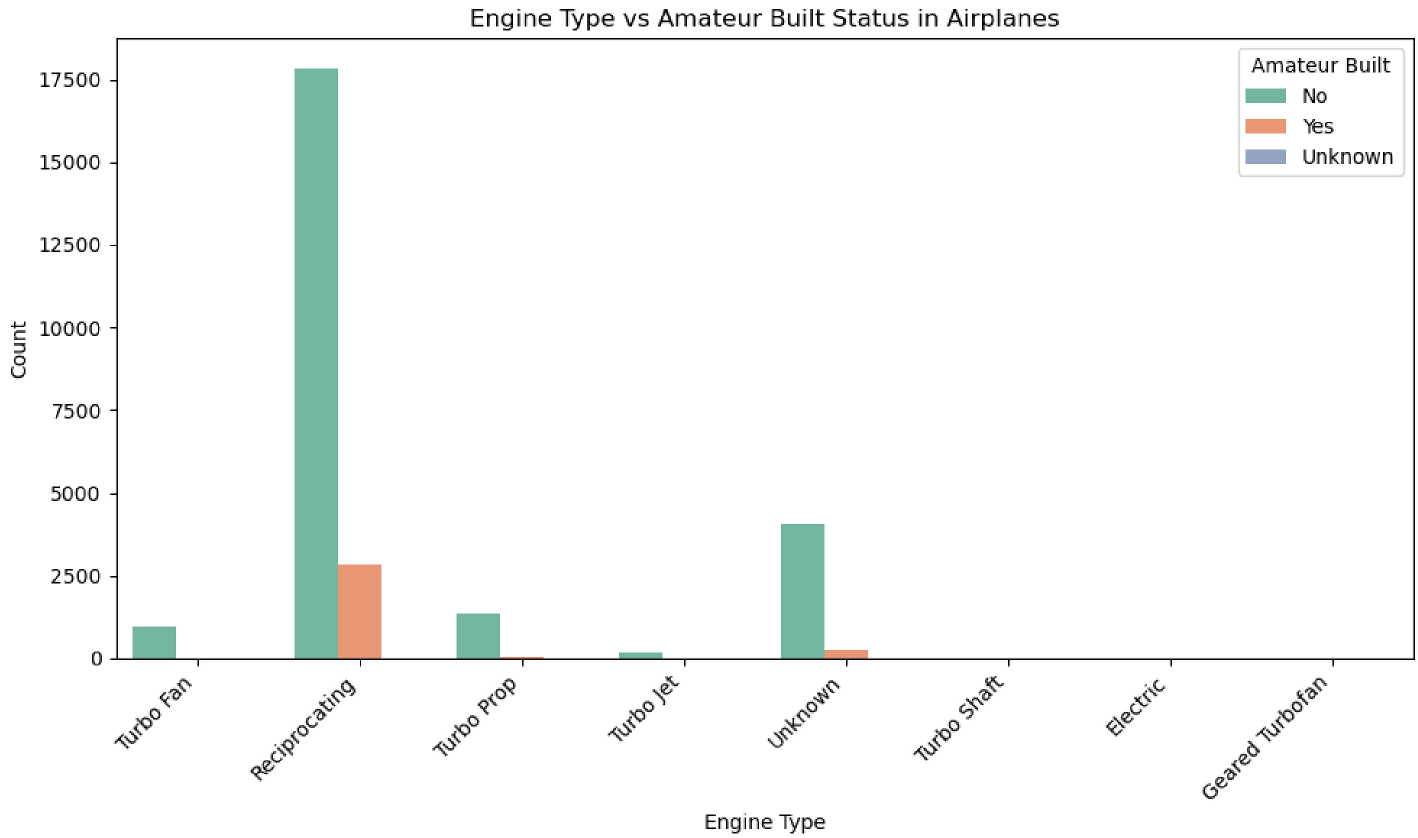




Bivariate Analysis: Engine Type vs Amateur Built in Airplanes

In [149]:

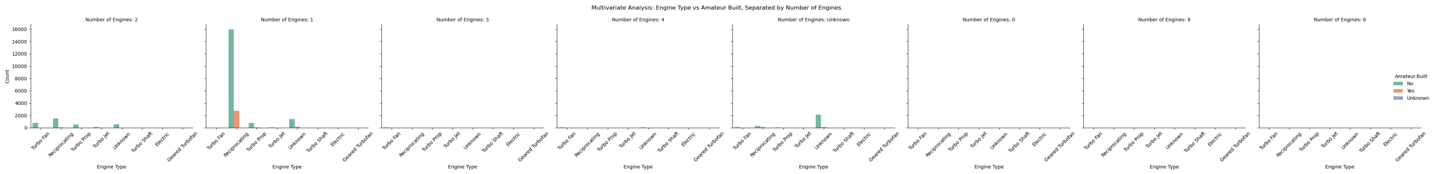
```
plt.figure(figsize=(10, 6))
sns.countplot(data=Airplanes_df1, x="Engine.Type", hue="Amateur.Built", palette="Set2")
plt.title("Engine Type vs Amateur Built Status in Airplanes")
plt.xlabel("Engine Type")
plt.ylabel("Count")
plt.xticks(rotation=45, ha="right")
plt.legend(title="Amateur Built")
plt.tight_layout()
plt.show()
```



Multivariate Analysis : Engine Type vs Amateur Built, grouped by Number of Engines

In [150]:

```
plot_df1 = Airplanes_df1[["Engine.Type", "Amateur.Built", "Number.Of.Engines"]].copy()
plot_df1["Number.Of.Engines"] = plot_df1["Number.Of.Engines"].astype(str)
grouped = sns.catplot(data=plot_df1,x="Engine.Type", hue="Amateur.Built",col="Number.Of.E
ngines",kind="count",palette="Set2")
grouped.set_titles("Number of Engines: {col_name}")
grouped.set_axis_labels("Engine Type", "Count")
grouped.set_xticklabels(rotation=45)
grouped.fig.subplots_adjust(top=0.85)
grouped.fig.suptitle("Multivariate Analysis: Engine Type vs Amateur Built, Separated by N
umber of Engines", fontsize=12)
plt.tight_layout()
plt.show()
```



Key Insights.

Analysis on Aircraft Specifications with focus on Airplanes indicate that

- **Most Airplanes have 1 engine.**
- **Reciprocating engine types are the most common among airplanes.**
- **Most Airplanes are not Amateur Built**

Note

- `Far.Description` **and** `Air.Category` **will be extracted later when answering purpose of flight.**

Dropping columns not relevant for my business question. The columns are `Latitude`**,** `Longitude`**,** `Airport.Code`**,** `Airport.Name`**,** `Schedule`**.**

In [151]:

```
# dropping irrelevant columns
df1.drop(["Latitude", "Longitude", "Airport.Code", "Airport.Name", "Schedule"], axis=1,
inplace=True)
```

In [152]:

```
# confirming if dropped
df1.columns
```

Out[152]:

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Injury.Severity', 'Aircraft.Damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.Of.Engines', 'Engine.Type', 'Far.Description',
       'Purpose.Of.Flight', 'Air.Carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.Phase.Of.Flight', 'Report.Status',
       'Publication.Date', 'Aircraft.Simple'],
      dtype='object')
```

Question 2: How many accidents or incidents has each airplane been involved in and Top 10 safest airplane?

- **Columns to incorporate** `Event.Id`**,** `Investigation.Type`**,**`Accident.Number`**,**`Injury.Severity`**,** `Location`**,** `Country`**,**`Total.Fatal.Injuries`**,** `Total.Serious.Injuries`**,** `Total.Minor.Injuries`**,** `Total.Uninjured`**,**`Event.Date`**,**`Publication.Date` **to determine accidents and incidents each airplane** type was involved in, and its related risk rates.

Data Preparation

In [153]:

```
# checking for the unique values for the columns incorporated
columns2= ["Event.Id","Investigation.Type","Accident.Number","Injury.Severity","Location"
,"Country","Total.Fatal.Injuries",
           "Total.Serious.Injuries","Total.Minor.Injuries","Total.Uninjured","Event.Date
","Publication.Date"]
unique_values ={col: df1[col].unique() for col in columns2}
for col, values in unique_values.items():
    print(f"\n{col}:\n{values}\n")
```

```
Event.Id:
['20001218X45444' '20001218X45447' '20061025X01555' ... '20221227106497'
 '20221227106498' '20221230106513']


Investigation.Type:
['Accident' 'Incident']


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


Injury.Severity:
['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' nan 'Minor' 'Serious']


Location:
['MOOSE CREEK, ID' 'BRIDGEPORT, CA' 'Saltville, VA' ... 'San Manual, AZ'
 'Auburn Hills, MI' 'Brasnorte, ']


Country:
['United States' nan 'GULF OF MEXICO' 'Puerto Rico' 'ATLANTIC OCEAN'
 'HIGH ISLAND' 'Bahamas' 'MISSING' 'Pakistan' 'Angola' 'Germany'
 'Korea, Republic Of' 'Martinique' 'American Samoa' 'PACIFIC OCEAN'
 'Canada' 'Bolivia' 'Mexico' 'Dominica' 'Netherlands Antilles' 'Iceland'
 'Greece' 'Guam' 'Australia' 'CARIBBEAN SEA' 'West Indies' 'Japan'
 'Philippines' 'Venezuela' 'Bermuda' 'San Juan Islands' 'Colombia'
 'El Salvador' 'United Kingdom' 'British Virgin Islands' 'Netherlands'
 'Costa Rica' 'Mozambique' 'Jamaica' 'Panama' 'Guyana' 'Norway'
 'Hong Kong' 'Portugal' 'Malaysia' 'Turks And Caicos Islands'
 'Northern Mariana Islands' 'Dominican Republic' 'Suriname' 'Honduras'
 'Congo' 'Belize' 'Guatemala' 'Anguilla' 'France'
 'St Vincent And The Grenadines' 'Haiti' 'Montserrat' 'Papua New Guinea'
```

```
 'Cayman Islands' 'Sweden' 'Taiwan' 'Senegal' 'Barbados' 'BLOCK 651A'
 'Brazil' 'Mauritius' 'Argentina' 'Kenya' 'Ecuador' 'Aruba' 'Saudi Arabia'
 'Cuba' 'Italy' 'French Guiana' 'Denmark' 'Sudan' 'Spain'
 'Federated States Of Micronesia' 'St Lucia' 'Switzerland'
 'Central African Republic' 'Algeria' 'Turkey' 'Nicaragua'
 'Marshall Islands' 'Trinidad And Tobago' 'Poland' 'Austria' 'Malta'
 'Cameroon' 'Solomon Islands' 'Zambia' 'Peru' 'Croatia' 'Fiji'
 'South Africa' 'India' 'Ethiopia' 'Ireland' 'Chile' 'Antigua And Barbuda'
 'Uganda' 'China' 'Cambodia' 'Paraguay' 'Thailand' 'Belgium' 'Gambia'
 'Uruguay' 'Tanzania' 'Mali' 'Indonesia' 'Bahrain' 'Kazakhstan' 'Egypt'
 'Russia' 'Cyprus' "Cote D'ivoire" 'Nigeria' 'Greenland' 'Vietnam'
 'New Zealand' 'Singapore' 'Ghana' 'Gabon' 'Nepal' 'Slovakia' 'Finland'
 'Liberia' 'Romania' 'Maldives' 'Antarctica' 'Zimbabwe' 'Botswana'
 'Isle of Man' 'Latvia' 'Niger' 'French Polynesia' 'Guadeloupe'
 'Ivory Coast' 'Tunisia' 'Eritrea' 'Gibraltar' 'Namibia' 'Czech Republic'
 'Benin' 'Bosnia And Herzegovina' 'Israel' 'Estonia' 'St Kitts And Nevis'
 'Sierra Leone' 'Corsica' 'Scotland' 'Reunion' 'United Arab Emirates'
 'Afghanistan' 'Ukraine' 'Hungary' 'Bangladesh' 'Morocco' 'Iraq' 'Jordan'
 'Qatar' 'Madagascar' 'Malawi' 'Unknown' 'Central Africa' 'South Sudan'
 'Saint Barthelemy' 'Micronesia' 'South Korea' 'Kyrgyzstan'
 'Turks And Caicos' 'Eswatini' 'Tokelau' 'Sint Maarten' 'Macao'
 'Seychelles' 'Rwanda' 'Palau' 'Luxembourg' 'Lebanon'
 'Bosnia and Herzegovina' 'Libya' 'Saint Vincent and the Grenadines' 'UN'
 'Iran' 'Lithuania' 'Malampa' 'Antigua and Barbuda' 'AY' 'Chad' 'Cayenne'
 'New Caledonia' 'Yemen' 'Slovenia' 'Nauru' 'Niue' 'Bulgaria'
 'Republic of North Macedonia' 'Virgin Islands' 'Somalia' 'Guinea'
 'Pacific Ocean' 'Obyan' 'Mauritania' 'Albania' 'Wolseley'
 'Wallis and Futuna' 'Saint Pierre and Miquelon' 'Georgia' "Côte d'Ivoire"
 'South Korean' 'Serbia' 'MU' 'Guernsey' 'Great Britain'
 'Turks and Caicos Islands']


Total.Fatal.Injuries:
[  2.   4.   3.   1.  nan   0.   8.  78.   7.   6.   5. 153.  12.  14.
  23.  10.  11.   9.  17.  13.  29.  70. 135.  31. 256.  25.  82. 156.
  28.  18.  43.  15. 270. 144. 174. 111. 131.  20.  73.  27.  34.  87.
  30.  16.  47.  56.  37. 132.  68.  54.  52.  65.  72. 160. 189. 123.
  33. 110. 230.  97. 349. 125.  35. 228.  75. 104. 229.  80. 217. 169.
  88.  19.  60. 113. 143.  83.  24.  44.  64.  92. 118. 265.  26. 138.
 206.  71.  21.  46. 102. 115. 141.  55. 121.  45. 145. 117. 107. 124.
  49. 154.  96. 114. 199.  89.  57. 152.  90. 103. 158. 157.  42.  77.
 127.  50. 239. 295.  58. 162. 150. 224.  62.  66. 112. 188.  41. 176.]


Total.Serious.Injuries:
[  0.  nan   2.   1.   6.   4.   5.  10.   3.   8.   9.   7.  15.  17.
  28.  26.  47.  14.  81.  13. 106.  60.  16.  21.  50.  44.  18.  12.
  45.  39.  43.  11.  25.  59.  23.  55.  63.  88.  41.  34.  53.  33.
  67.  35.  20. 137.  19.  27. 125. 161.  22.]


Total.Minor.Injuries:
[  0.  nan   1.   3.   2.   4.  24.   6.   5.  25.  17.  19.  33.  14.
   8.  13.  15.   7.   9.  16.  20.  11.  12.  10.  38.  42.  29.  62.
  28.  31.  39.  32.  18.  27.  57.  50.  23. 125.  45.  26.  36.  69.
  21.  96.  30.  22.  58. 171.  65.  71. 200.  68.  47. 380.  35.  43.
  84.  40.]


Total.Uninjured:
[  0.  nan  44.   2.   1.   3.   6.   4. 149.  12. 182. 154.   5.  10.
   7. 119.  36.  51.  16.  83.   9.  68.  30.  20.  18.   8. 108.  11.
 152.  21.  48.  56. 113. 129. 109.  29.  13.  84.  74. 142. 102. 393.
 128. 112.  17.  65.  67. 136.  23. 116.  22.  57.  58.  73. 203.  31.
 201. 412. 159.  39. 186. 588.  82.  95. 146. 190. 245. 172.  52.  25.
  59. 131. 151. 180. 150.  86.  19. 133. 240.  15. 145. 125. 440.  77.
 122. 205. 289. 110.  79.  66.  87.  78.  49. 104. 250.  33. 138. 100.
  53. 158. 127. 160. 260.  47.  38. 165. 495.  81.  41.  14.  72.  98.
 263. 188. 239.  27. 105. 111. 212. 157.  46. 121.  75.  71.  45.  91.
  99.  85.  96.  50.  93. 276. 365. 371. 200. 103. 189.  37. 107.  61.
  26. 271. 130.  89. 439. 132. 219.  43. 238. 195. 118. 175.  32. 507.
 421.  90. 225. 269. 169. 236. 224. 134. 106. 331. 140.  94. 192. 161.
```

```
 270.  69. 436. 213. 233. 115.  42. 167. 137. 114. 148. 222.  92. 375.
  76. 171. 173. 246. 234. 123. 220. 202. 408. 279. 363. 135. 528. 334.
 178. 147. 126.  62.  70.  97. 228. 226.  64. 290. 206. 297. 349. 208.
 144.  54.  24. 258. 304. 274. 286.  55. 199. 221.  80. 272. 211. 262.
 441. 194. 309. 185. 261. 241. 383. 177. 259. 244. 254. 156.  40.  34.
 247. 176.  63.  28. 218. 282. 320. 204. 124. 215. 298. 120. 280. 179.
 315. 461. 153.  60. 308.  88. 361. 277. 191. 235. 187. 101. 162.  35.
 197. 193. 164. 370. 387. 163. 139. 267. 357. 339. 288. 231. 300. 255.
 306. 443. 385. 248. 459. 141. 414. 229. 166. 209. 184. 168. 170. 198.
 299. 573. 223. 265. 322. 196. 117. 253. 399. 360. 252. 217. 155. 183.
 227. 249. 329. 340. 699. 325. 287. 143. 243. 230. 386. 181. 257. 283.
 404. 319. 450. 356. 216. 174. 558. 214. 448. 324. 338. 273. 232. 401.
 312. 368. 501. 237. 307. 296. 291. 403. 314. 285. 311. 293. 352. 332.
 384. 275. 210. 268. 326. 454. 278. 576. 380. 394. 362. 397. 359. 264.
 333. 367. 302. 348. 351. 358. 295. 321. 521. 301. 294. 378. 207. 406.
 251. 455.]


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


Publication.Date:
[nan '19-09-1996' '26-02-2007' ... '22-12-2022' '23-12-2022' '29-12-2022']
```

In [154]:

```
# Checking for missing values in the columns
df1[columns2].isna().sum()
```

Out[154]:

```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Injury.Severity           979
Location                   52
Country                   225
Total.Fatal.Injuries    11386
Total.Serious.Injuries  12490
Total.Minor.Injuries    11914
Total.Uninjured          5897
Event.Date                  0
Publication.Date        13765
dtype: int64
```

Data Cleaning

In [155]:

```
# Filling null values in Publication Date with by assumption "1900-01-01"
df1["Publication.Date"].fillna(pd.Timestamp("1900-01-01"), inplace=True)
df1["Publication.Date"].isna().sum()
```

Out[155]:

```
0
```

In [156]:

```
# filling null values in Location and Country with Unknown
df1["Location"].fillna("Unknown", inplace=True)
df1["Country"].fillna("Unknown", inplace=True)
df1["Country"].isna().sum()
df1["Location"].isna().sum()
```

Out[156]:

```
0
```

In [157]:

```
# cleaning and standardizing Injury severity cases using 'isinstance(x, str) and'parse fl
oat values and also fillna with unknown
df1["Injury.Severity"] = ["Incident" if isinstance(x, str) and "incident" in x.lower()
    else "Fatal" if isinstance(x, str) and "fatal" in x.lower()
    else "Non-Fatal" if isinstance(x, str) and "non-fatal" in x.lower()
    else "Unknown"
    for x in df1["Injury.Severity"]]
df1["Injury.Severity"].value_counts().head(20)
```

Out[157]:

```
Injury.Severity
Fatal        85098
Incident      2214
Unknown       1465
Name: count, dtype: int64
```

In [158]:

```
df1["Injury.Severity"].isna().sum()
```

Out[158]:

```
0
```

In [159]:

```
#dealing with total injuries, since median is 0 for Fatal,Serious and Minor injuries,
# i will replace the missing values with 0. The percentages of missing values in these co
lumns are
# approximately 13%,14%,13% respectively
df1["Total.Injuries"] = (df1["Total.Fatal.Injuries"].fillna(0) + df1["Total.Serious.Inju
ries"].fillna(0) +
                         df1["Total.Minor.Injuries"].fillna(0))
df1["Total.Injuries"].isna().sum()
```

Out[159]:

```
0
```

In [160]:

```
# Total uninjured mean is 5.3 and median is 1 suggesting that data is right-skewed, very
few but high values pulling the mean up.
# i will fillna with median to avoid overestimating values due to outliers
df1["Total.Uninjured"].fillna(1, inplace=True)
df1["Total.Uninjured"].isna().sum()
```

Out[160]:

```
0
```

Data Analysis

To answer question 2, i need to analyze;

- **High-risk airplane: Many events, high fatality rate**
- **Low-risk (safe) airplane: Many events, low fatality rate**
- **Get total number of accidents/incidents**
- **Identify top recommended safe airplane types**

In [161]:

```
columns_2a = ["Aircraft.Simple", "Event.Id", "Investigation.Type", "Accident.Number",
              "Injury.Severity", "Location", "Country","Total.Injuries", "Total.Uninjure
d",
              "Event.Date", "Publication.Date"]
```

```
Aircraft_Event = df1[columns_2a].dropna(subset=["Aircraft.Simple", "Event.Id"])
Aircraft_Event.head(10)
```

Out[161]:

	Aircraft.Simple	Event.Id	Investigation.Type	Accident.Number	Injury.Severity	Location	Country	To
0	Stinson108-3	20001218X45444	Accident	SEA87LA080	Fatal	MOOSE CREEK, ID	United States	
1	PiperPA24-180	20001218X45447	Accident	LAX94LA336	Fatal	BRIDGEPORT, CA	United States	
2	Cessna172M	20061025X01555	Accident	NYC07LA005	Fatal	Saltville, VA	United States	
3	Rockwell112	20001218X45448	Accident	LAX96LA321	Fatal	EUREKA, CA	United States	
4	Cessna501	20041105X01764	Accident	CHI79FA064	Fatal	Canton, OH	United States	
5	Mcdonnell.DouglasDC9	20170710X52551	Accident	NYC79AA106	Fatal	BOSTON, MA	United States	
6	Cessna180	20001218X45446	Accident	CHI81LA106	Fatal	COTTON, MN	United States	
7	Cessna140	20020909X01562	Accident	SEA82DA022	Fatal	PULLMAN, WA	United States	
8	Cessna401B	20020909X01561	Accident	NYC82DA015	Fatal	EAST HANOVER, NJ	United States	
9	North.AmericanNAVION L-17B	20020909X01560	Accident	MIA82DA029	Fatal	JACKSONVILLE, FL	United States	

In [162]:

```
# updating Aircraft Simple as string
df1["Aircraft.Simple"] = df1["Make"].astype(str) + " " + df1["Model"].astype(str)
```

In [163]:

```
# calculating severity counts
Severity_Counts= df1.groupby(["Aircraft.Simple", "Injury.Severity"])["Event.Id"].count()
.unstack(fill_value=0)
```

In [164]:

```
# adding total events and fatal rates
Severity_Counts["Total.Events"] = Severity_Counts.sum(axis=1)
Severity_Counts["Fatal.Rate"] = (Severity_Counts.get("Fatal", 0) / Severity_Counts["Tota
l.Events"]).round(2)
Severity_Counts
```

Out[164]:

Injury.Severity	Fatal	Incident	Unknown	Total.Events	Fatal.Rate
Aircraft.Simple					
107.5.Flying.Corporation One Design DR 107	1	0	0	1	1.0
1200 G103	1	0	0	1	1.0
177Mf.Llc PITTS MODEL 12	1	0	0	1	1.0
1977.Colfer-Chan STEEN SKYBOLT	1	0	0	1	1.0
1St.Ftr.Gp FOCKE-WULF 190	1	0	0	1	1.0
...	...	...	...	...	...
Zubair.S.Khan RAVEN	1	0	0	1	1.0
Zuber.Thomas.P ZUBER SUPER DRIFTER	1	0	0	1	1.0

Injury.Severity	Fatal	Incident	Unknown	Total.Events	Fatal.Rate
Aircraft.Simple					
Zukowski FAA BIPLANE	1	0	0	1	1.0
Zwart KIT FOX VIXEN	1	0	0	1	1.0
Zwicker.Murray.R GLASTAR	1	0	0	1	1.0

18465 rows × 5 columns

In [165]:

```
# resetting Aircraft Simple back to column
Severity_Counts = Severity_Counts.reset_index()
```

In [166]:

```
# Filtering fatal rate more than 10% for at least 5 events to get safe airplanes
Safe_Airplanes = Severity_Counts[(Severity_Counts["Fatal.Rate"] <= 0.10) & (Severity_Cou
nts["Total.Events"] >= 5)]
Safe_Airplanes
```

Out[166]:

Injury.Severity	Aircraft.Simple	Fatal	Incident	Unknown	Total.Events	Fatal.Rate
949	Airbus.Industrie A300-600	0	8	0	8	0.0
3239	Boeing 737-130	0	6	0	6	0.0
3337	Boeing 747-123	0	7	0	7	0.0
6917	Douglas DC-8-71	0	5	0	5	0.0
6933	Douglas DC-9-51	0	7	0	7	0.0
11683	Mcdonnell.Douglas DC-10-40	0	7	0	7	0.0
11700	Mcdonnell.Douglas DC-9	0	5	0	5	0.0

In [167]:

```
# getting the top 10 safe
Top_Safe = Safe_Airplanes.sort_values(by=["Total.Events", "Fatal.Rate"], ascending=[Fals
e, True]).head(10)
Top_Safe
```

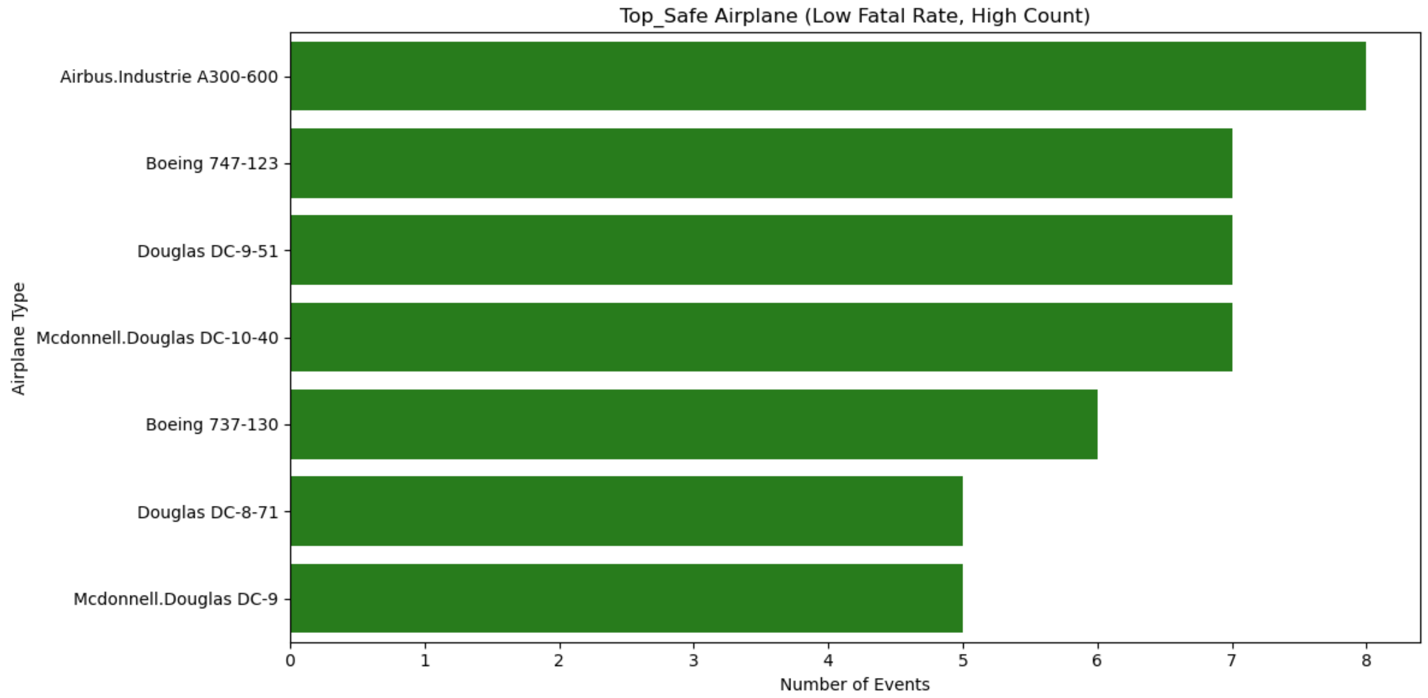
Out[167]:

Injury.Severity	Aircraft.Simple	Fatal	Incident	Unknown	Total.Events	Fatal.Rate
949	Airbus.Industrie A300-600	0	8	0	8	0.0
3337	Boeing 747-123	0	7	0	7	0.0
6933	Douglas DC-9-51	0	7	0	7	0.0
11683	Mcdonnell.Douglas DC-10-40	0	7	0	7	0.0
3239	Boeing 737-130	0	6	0	6	0.0
6917	Douglas DC-8-71	0	5	0	5	0.0
11700	Mcdonnell.Douglas DC-9	0	5	0	5	0.0

Univariate Analysis: Top 10 Safe Airplanes Vs Number of Events

In [168]:

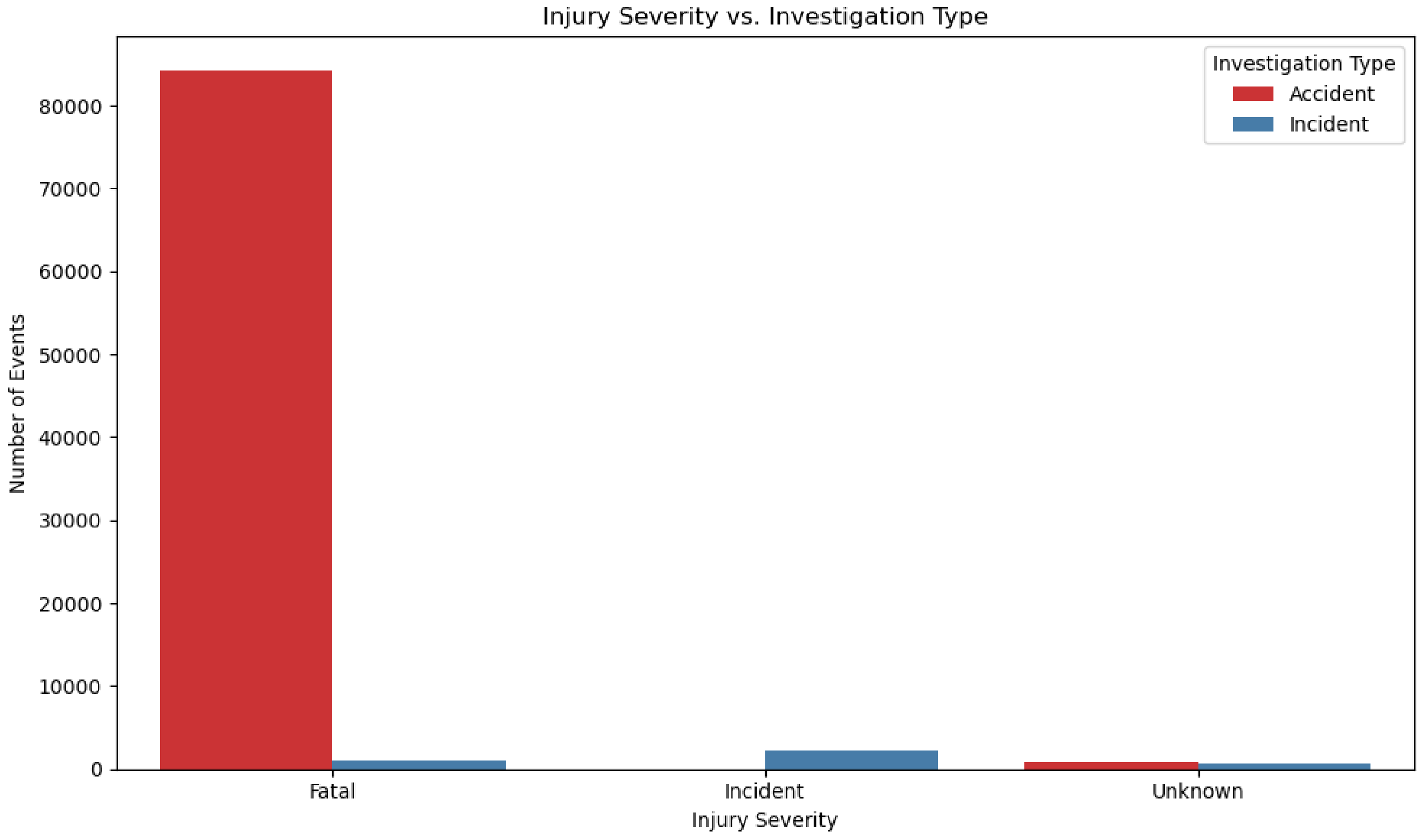
```
plt.figure(figsize=(12, 6))
sns.barplot(data=Top_Safe, y="Aircraft.Simple", x="Total.Events", color="#1c8c0c")
plt.title("Top Safe Airplane (Low Fatal Rate, High Count)")
plt.xlabel("Number of Events")
plt.ylabel("Airplane Type")
plt.tight_layout()
plt.show()
```



Bivariate Analysis: Injury Severity by Investigation Type

In [169]:

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df1, x="Injury.Severity", hue="Investigation.Type", palette="Set1")
plt.title("Injury Severity vs. Investigation Type")
plt.xlabel("Injury Severity")
plt.ylabel("Number of Events")
plt.legend(title="Investigation Type")
plt.tight_layout()
plt.show()
```



In [170]:

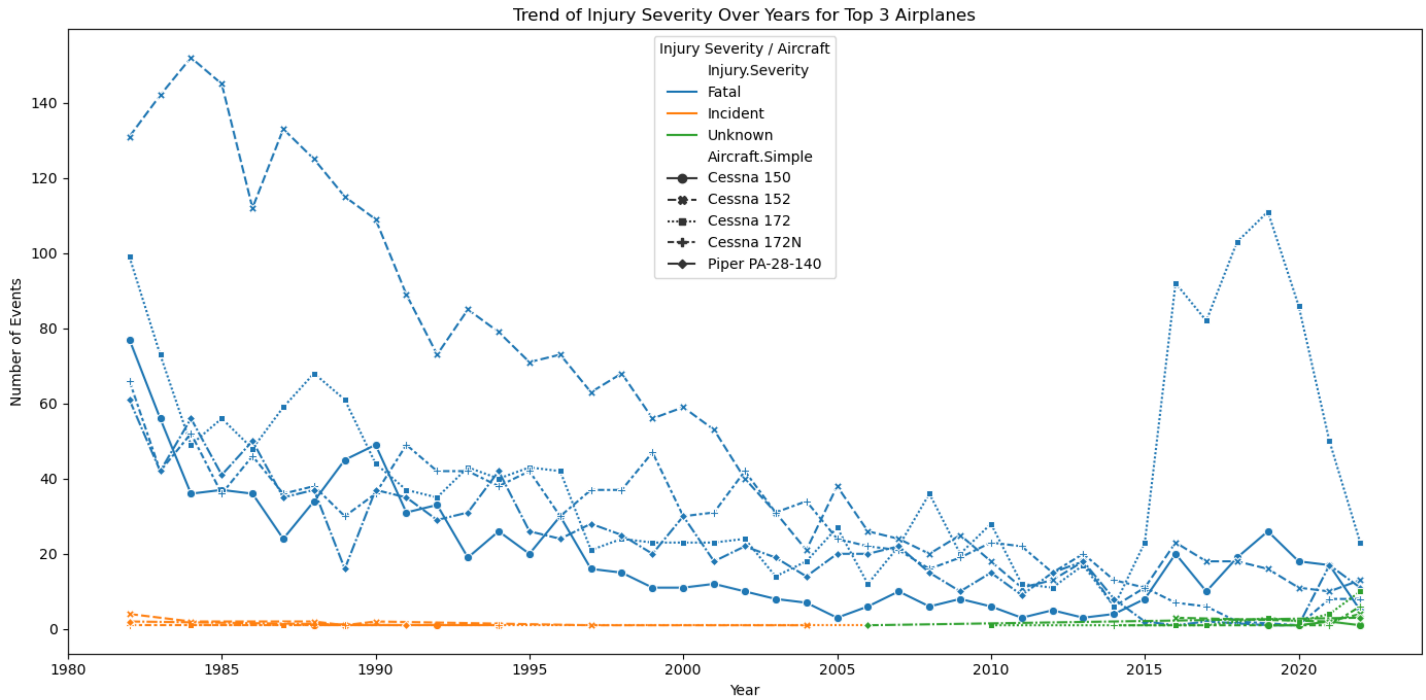
```
df1["Event.Date"] = pd.to_datetime(df1["Event.Date"], errors="coerce")
df1.loc[:, "Event.Year"] = df1["Event.Date"].dt.year
```

Multivariate Analysis: Trend of Injury Severity Over the Years, grouped by Aircraft Type

In [171]:

```
df1["Event.Date"] = pd.to_datetime(df1["Event.Date"], errors="coerce")
df1.loc[:, "Event.Year"] = df1["Event.Date"].dt.year
df1 = df1[(df1["Event.Year"] >= 1980) & (df1["Event.Year"] <= 2023)]
top_aircrafts = df1["Aircraft.Simple"].value_counts().head(5).index
filtered_df = df1[df1["Aircraft.Simple"].isin(top_aircrafts)]
line_df = filtered_df.groupby(["Event.Year", "Aircraft.Simple", "Injury.Severity"]).size
().reset_index(name="Count")
plt.figure(figsize=(14, 7))
sns.lineplot(data=line_df, x="Event.Year", y="Count", hue="Injury.Severity", style="Airc
raft.Simple", markers=True)
plt.title("Trend of Injury Severity Over Years for Top 3 Airplanes")
plt.xlabel("Year")
plt.ylabel("Number of Events")
plt.legend(title="Injury Severity / Aircraft")
plt.tight_layout()
plt.show()
```

```
C:\Users\USER\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_in
f_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\USER\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_in
f_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Key Insights

Analysis of top safe airplanes in regards to severity of events involves indicate that:

- **Airbus.Industrie A300, Boeing 747-123, Douglas DC-9-51, Mcdonnell.Douglas DC-10-40,Boeing 737-130 are top 5 most safe airplanes with low fatalities but high counts.**
- **Accidents where the highest causes of fatalities compared to incidents.**
- **Cessna models have had high counts of fatal accidents over time.**

Question 3 : What were the causes of the accidents or the incidents and the level of damage sustained on the airplane?

To determine the causes of accidents and the level of damage on the airplane i will use `Weather.Condition` `Report.Status` `Broad.Phase.Of.Flight`. `Aircraft.Damage`

Data Preparation

In [172]:

```
columns3= ["Report.Status", "Broad.Phase.Of.Flight", "Aircraft.Damage","Weather.Condition
"]
unique_values ={col: df1[col].unique() for col in columns3}
for col, values in unique_values.items():
    print(f"\n{col}:\n{values}\n")
```

```
Report.Status:
['Probable Cause' 'Factual' 'Foreign' ...
 'The pilot did not ensure adequate clearance from construction vehicles during taxi.'
 'The pilot’s failure to secure the magneto switch before attempting to hand rotate the e
ngine which resulted in an inadvertent engine start, a runaway airplane, and subsequent i
mpact with parked airplanes. Contributing to the accident was the failure to properly sec
ure the airplane with chocks.'
 'The pilot’s loss of control due to a wind gust during landing.']


Broad.Phase.Of.Flight:
['Unknown' 'Takeoff' 'Landing' 'Cruise' 'Approach' 'Taxi' 'Descent'
 'Maneuvering' 'Climb' 'Standing' 'Go-around' 'Other' nan]


Aircraft.Damage:
['Destroyed' 'Substantial' 'Minor' nan 'Unknown']


Weather.Condition:
['IMC' 'VMC' 'UNK' nan 'Unk']
```

In [173]:

```
df1[columns3].isna().sum()
```

Out[173]:

```
Report.Status             6338
Broad.Phase.Of.Flight    27094
Aircraft.Damage           3172
Weather.Condition         4439
dtype: int64
```

In [174]:

```
df1[columns3].value_counts()
```

Out[174]:

```
Report.Status   Broad.Phase.Of.Flight  Aircraft.Damage  Weather.Condition
Probable Cause  Landing                Substantial      VMC                  13694
                Takeoff                Substantial      VMC                   8988
                Cruise                 Substantial      VMC                   5861
                Maneuvering            Substantial      VMC                   4231
                Approach               Substantial      VMC                   4003
                                                                              ...
                Other                  Substantial      UNK                      1
                                       Minor            UNK                      1
                                       Destroyed        IMC                      1
Foreign         Takeoff                Destroyed        VMC                      1
                Approach               Destroyed        VMC                      1
Name: count, Length: 105, dtype: int64
```

Data Cleaning

Data Cleaning

In [175]:

```
# filtering the unique values in weather condition
df1["Weather.Condition"].unique()
```

Out[175]:

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

- **Domain knowledge in aviation suggest that the abbreviation represented in the Weather conditions are;**

  `UNK` **alias Unknown, weather not recorded.**

  `IMC` **alias Instrument Meteorological Conditions meaning, poor weather**

  `VMC` **alias Visual Meteorological Conditions meaning good visibility**

In [176]:

```
# fillna missing values with UNK that is Unknown
df1["Weather.Condition"]= df1["Weather.Condition"].str.upper().fillna("UNK")
df1["Weather.Condition"].isna().sum()
```

Out[176]:

```
0
```

In [177]:

```
# Phase of flight replace - with_ and fillna with unknown since no records were available
and although it represents 44% of the records, it is truthful
df1["Broad.Phase.Of.Flight"] = df1["Broad.Phase.Of.Flight"].str.replace("-","_").fillna("
Unknown")
df1["Broad.Phase.Of.Flight"].unique()
```

Out[177]:

```
array(['Unknown', 'Takeoff', 'Landing', 'Cruise', 'Approach', 'Taxi',
       'Descent', 'Maneuvering', 'Climb', 'Standing', 'Go_around',
       'Other'], dtype=object)
```

In [178]:

```
# Aircraft Damage fill na with Unknown
df1["Aircraft.Damage"] = df1["Aircraft.Damage"].fillna("Unknown")
df1["Aircraft.Damage"].value_counts()
```

Out[178]:

```
Aircraft.Damage
Substantial    64096
Destroyed      18592
Unknown         3291
Minor           2792
Name: count, dtype: int64
```

In [179]:

```
df1["Investigation.Type"].value_counts()
```

Out[179]:

```
Investigation.Type
Accident    84931
Incident     3840
Name: count, dtype: int64
```

In [180]:

```
# Report Status top 20 counts
```

```python
df1["Report.Status"].value_counts().head(20)
```

Out[180]:

```
Report.Status
Probable Cause
61707
Foreign
1986
<br /><br />
167
Factual
145
The pilot's failure to maintain directional control during the landing roll.
56
A loss of engine power for undetermined reasons.
52
The pilot's failure to maintain directional control during landing.
44
A total loss of engine power for undetermined reasons.
39
The loss of engine power for undetermined reasons.
29
The pilot’s failure to maintain directional control during the landing roll.\r\n\r
21
The pilot’s failure to maintain directional control during the landing roll.
19
The pilot's improper recovery from a bounced landing.
19
The pilot's failure to maintain directional control during takeoff.
17
None.
17
The pilot's failure to maintain directional control of the airplane during landing.
17
The pilot's improper landing flare, which resulted in a hard landing.
16
The pilot’s failure to maintain directional control during landing.
16
The student pilot's improper recovery from a bounced landing.
16
The pilot's failure to maintain directional control during the takeoff roll.
15
.
15
Name: count, dtype: int64
```

In [181]:

```python
# defining valid status values of report status
valid_status = ["Probable Cause", "Factual", "Foreign"]
def clean_report_status(status):
    if pd.isna(status) or status.strip() in ["<br /><br />", "", " "]:
        return "Missing"
    elif status in valid_status:
        return status
    else:
        return "Narrative/Other"

df1["Cleaned.Report.Status"] = df1["Report.Status"].apply(clean_report_status)
df1["Cleaned.Report.Status"].value_counts()
```

Out[181]:

```
Cleaned.Report.Status
Probable Cause     61707
Narrative/Other    18428
Missing             6505
Foreign             1986
Factual              145
Name: count, dtype: int64
```

Data Analysis

In [182]:

```
# categorizing report status into cause types by defining key words for each cause type
def classify_cause(report):
    if pd.isna(report) or report.strip() in ["<br /><br />", "", "None."]:
        return "Unknown"
    r = report.lower()
    if "pilot" in r or "student" in r or "control" in r or "landing" in r or "takeoff" i
n r or "flare" in r:
        return "Human Error"
    if "engine" in r or "mechanical" in r or "system" in r or "power" in r:
        return "Mechanical Failure"
    if "maintenance" in r:
        return "Maintenance Issue"
    if "weather" in r or "wind" in r or "gust" in r or "imc" in r:
        return "Weather-Related"
    if report.strip() in ["Probable Cause", "Factual", "Foreign"]:
        return "General"
    return "Other"
df1["Cause.Category"] = df1["Report.Status"].apply(classify_cause)
```

In [183]:

```
# grouping by cause and damage per airplane
Summary = df1.groupby(["Aircraft.Simple","Cause.Category","Weather.Condition",
                      "Broad.Phase.Of.Flight","Aircraft.Damage"])["Event.Id"].count().r
eset_index()
Summary.columns = ["Aircraft Type", "Cause Category", "Weather", "Flight Phase", "Damage
Level", "Event Count"]
Summary = Summary.sort_values(by="Event Count", ascending=False)
print(Summary.head(10))
```

```
      Aircraft Type Cause Category Weather Flight Phase Damage Level  \
11150    Cessna 152        General     VMC      Landing  Substantial
11370    Cessna 172    Human Error     VMC      Unknown  Substantial
11350    Cessna 172        General     VMC      Landing  Substantial
11844   Cessna 172N        General     VMC      Landing  Substantial
11162    Cessna 152        General     VMC      Takeoff  Substantial
11141    Cessna 152        General     VMC       Cruise  Substantial
8141     Boeing 737        Unknown     UNK      Unknown      Unknown
11906   Cessna 172P        General     VMC      Landing  Substantial
11778   Cessna 172M        General     VMC      Landing  Substantial
12313    Cessna 180        General     VMC      Landing  Substantial

       Event Count
11150          753
11370          475
11350          346
11844          320
11162          298
11141          279
8141           248
11906          229
11778          214
12313          202
```

In [184]:

```
# finding the top causes of accidents or incident in airplanes
top_causes = Summary.groupby("Cause Category")["Event Count"].sum().sort_values(ascendin
g=False).head(10)
print(top_causes)
```

```
Cause Category
General               63838
Human Error           15487
Unknown                6522
Mechanical Failure     1893
Other                   924
Maintenance Issue        64
```

```
Maintenance Issue        64
Weather-Related          43
Name: Event Count, dtype: int64
```

In [185]:

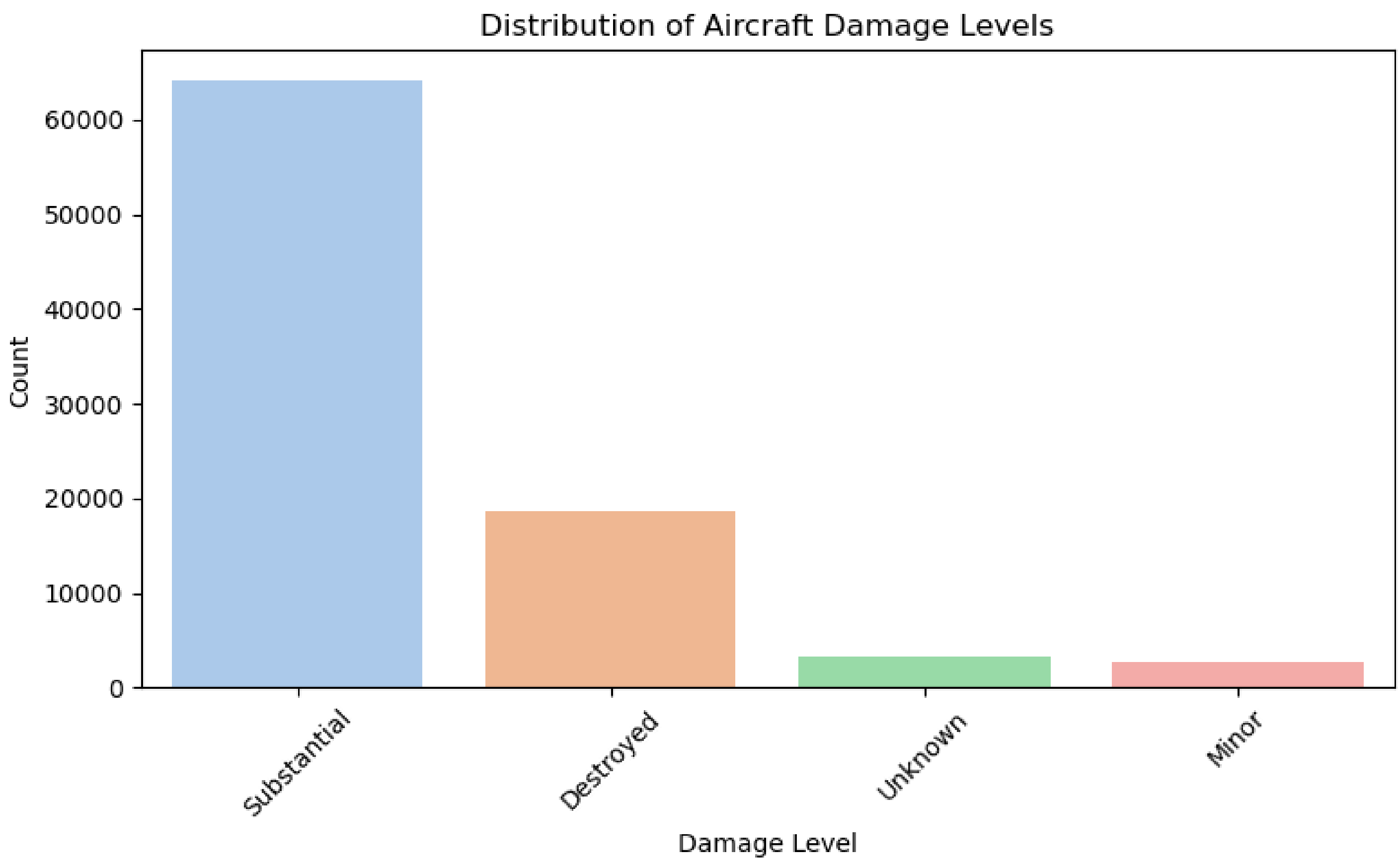
```python
# assessing airplanes with least severe damage
least_damage = Summary[Summary["Damage Level"].isin(["Minor", "None"])].groupby("Aircraft Type")["Event Count"].sum().sort_values(ascending=False).head(10)

print(least_damage)
```

```
Aircraft Type
Boeing 737                     124
Boeing 747                      38
Boeing 777                      32
Cessna 152                      29
Cessna 402C                     24
Piper PA-31-350                 21
Beech 1900D                     20
Boeing 767                      17
Boeing 727-200                  17
Mcdonnell.Douglas DC-10-10      16
Name: Event Count, dtype: int64
```

Univariate Analysis: Aircraft Damage Severity

In [186]:

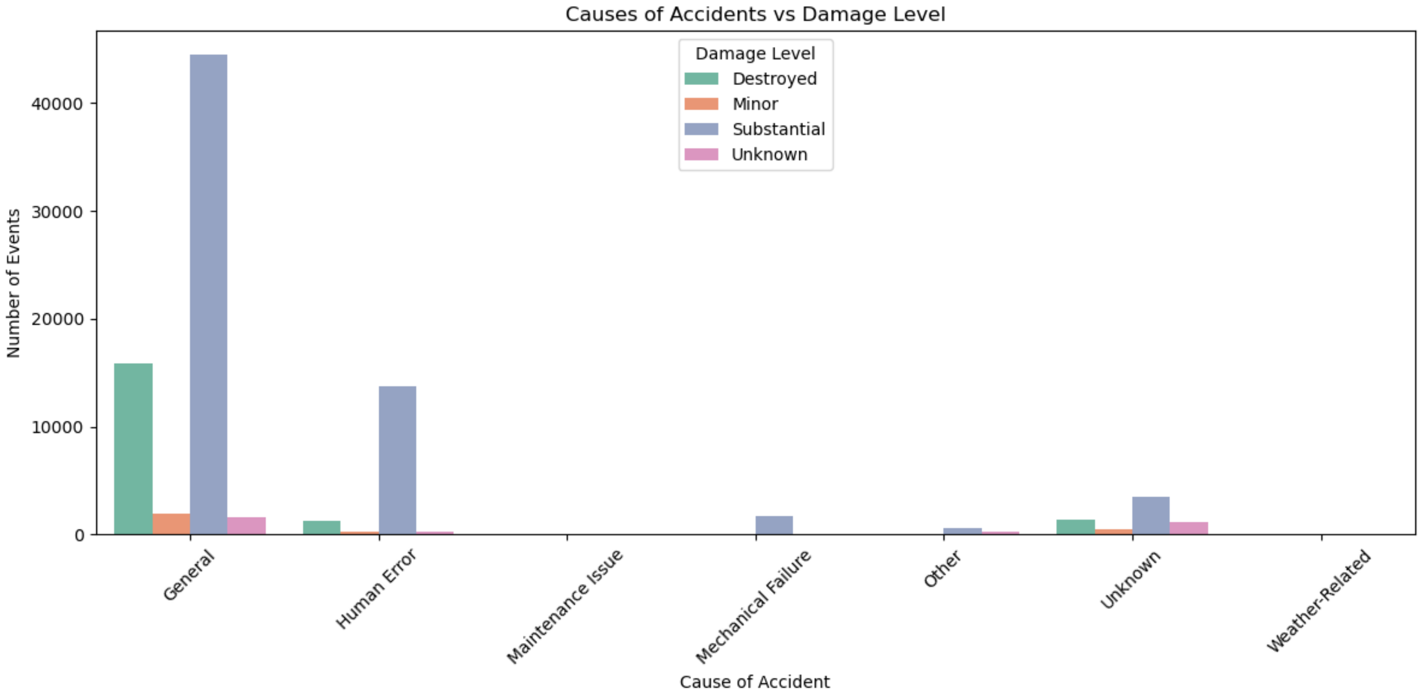
```python
plt.figure(figsize=(8,5))
sns.countplot(data=df1, x="Aircraft.Damage", order=df1["Aircraft.Damage"].value_counts()
.index, palette="pastel")
plt.title("Distribution of Aircraft Damage Levels")
plt.xlabel("Damage Level")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Bivariate Analysis: Causes Vs Damage

In [187]:

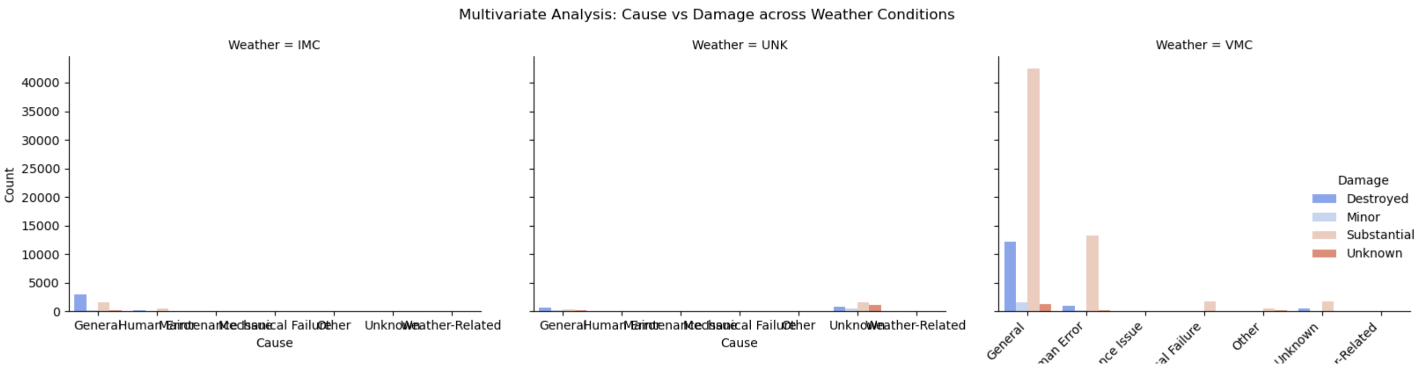
```
plot_df = df1.groupby(["Cause.Category", "Aircraft.Damage"])["Event.Id"].count().reset_i
ndex()
plot_df.columns = ["Cause Category", "Damage Level", "Event Count"]

plt.figure(figsize=(12,6))
sns.barplot(data=plot_df, x="Cause Category", y="Event Count", hue="Damage Level", palet
te="Set2")
plt.title("Causes of Accidents vs Damage Level")
plt.xlabel("Cause of Accident")
plt.ylabel("Number of Events")
plt.xticks(rotation=45)
plt.legend(title="Damage Level")
plt.tight_layout()
plt.show()
```

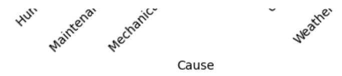


Multivariate Analysis: Cause vs Damage vs Weather

In [188]:

```
multi_df = df1.groupby(["Cause.Category", "Aircraft.Damage", "Weather.Condition"])["Even
t.Id"].count().reset_index()
multi_df.columns = ["Cause", "Damage", "Weather", "Count"]


sns.catplot(data=multi_df, x="Cause", y="Count", hue="Damage", col="Weather", kind="bar"
, palette="coolwarm")
plt.subplots_adjust(top=0.85)
plt.suptitle("Multivariate Analysis: Cause vs Damage across Weather Conditions")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```





Key Insights

- **Major causes of accidents and incidents are general, human error and mechanical failure.**
- **Most airplanes were substantially damaged in the events.**
- **Unexpectedly most events occurred when the weather was generally good.**

Question 4: Are the said safest airplanes useful for commercial and private operations?

To determine the uses of the top safe airplanes, if it can be used for both commercial and private enterprises, i will use `Purpose.Of.Flight` **and** `Far.Description`

Data Preparation

In [189]:

```
columns4= ["Purpose.Of.Flight", "Far.Description"]
unique_values ={col: df1[col].unique() for col in columns4}
for col, values in unique_values.items():
    print(f"\n{col}:\n{values}\n")
```

```
Purpose.Of.Flight:
['Personal' 'Business' 'Instructional' 'Unknown' 'Ferry'
 'Executive/corporate' 'Aerial Observation' 'Aerial Application' nan
 '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']


Far.Description:
['Unknown' 'Part 91: General Aviation' 'Part 135: Air Taxi'
 'Part 125: Large Aircraft' 'Part 121: Air Carrier' 'Part 129: Foreign'
 'Part 137: Agricultural' 'Part 133: Rotorcraft'
 'Part 91F: Special Flight Ops' 'Foreign: Non-Commercial' 'Public Use'
 'Foreign: Commercial' 'Military' 'Part 91: Fractional' 'Other']
```

In [190]:

```
df1[columns4].value_counts()
```

Out[190]:

```
Purpose.Of.Flight          Far.Description
Personal                   Unknown                      31654
                           Part 91: General Aviation    17433
Instructional              Unknown                       6717
Unknown                    Unknown                       5645
Instructional              Part 91: General Aviation     3795
                                                         ...
Public Aircraft - Federal  Military                         1
Other Work Use             Part 125: Large Aircraft         1
Executive/corporate        Part 135: Air Taxi               1
                           Part 125: Large Aircraft         1
Positioning                Part 135: Air Taxi               1
Name: count, Length: 136, dtype: int64
```

In [191]:

```
df1[columns4].isna().sum()
```

Out[191]:

```
Purpose.Of.Flight    6137
Far.Description         0
dtype: int64
```

Data Cleaning

In [192]:

```
# cleaning Purpose.Of.Flight by filling missing values with unknown, preserve data interg
rity
df1["Purpose.Of.Flight"].fillna("Unknown", inplace=True)
df1["Purpose.Of.Flight"].isna().sum()
```

Out[192]:

```
0
```

In [193]:

```
# normalizing to remove white spaces and cases
df1["Purpose.Of.Flight"]= (df1["Purpose.Of.Flight"].str.strip().str.title()
                           .str.replace("/","_").str.replace("-","_").str.replace("\s+",
"_",regex=True)
                           .str.replace("_+","_", regex=True))
df1["Purpose.Of.Flight"].unique()
```

Out[193]:

```
array(['Personal', 'Business', 'Instructional', 'Unknown', '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', 'Pubs', 'Asho', 'Publ'], dtype=object)
```

In [194]:

```
# Categorizing each item in the unique items according to the domain knowledge
Private_use =["Personal", "Instructional", "Executive_Corporate",
              "Skydiving","Air_Race_Show", "Air_Drop", "Glider_Tow","Flight_Test","Asho"
]

Commercial_use =["Business","Ferry", "Aerial_Observation","Aerial_Application",
                 "Other_Work_Use", "Positioning","Banner_Tow","External_Load"]

Public =["Public_Aircraft_Federal","Firefighting","Public_Aircraft_State",
         "Public_Aircraft","Public_Aircraft_Local"]

Unknown =["Unknown","Pubs","Publ"]
```

In [195]:

```
df1["Flight.Purpose.Category"]= df1["Purpose.Of.Flight"].apply(
    lambda i: "Private" if i in Private_use
    else "Commercial" if i in Commercial_use
    else "Public" if i in Public
    else "Unknown")

df1["Flight.Purpose.Category"].value_counts()
```

Out[195]:

```
Flight.Purpose.Category
Private       61374
Commercial    13465
Unknown       12929
Public         1003
Name: count, dtype: int64
```

In [196]:

```
#  cleaning Far Description by mapping to purpose that is unknown,private, commercial or government or other according to the domain knowledge on aviation
def map_far_description(desc):
    if "91" in desc:
        return "Private"
    elif any(code in desc for code in ["135", "121", "137", "133"]):
        return "Commercial"
    elif any(x in desc for x in ["Public", "Military", "Foreign"]):
        return "Government"
    elif "Unknown" in desc:
        return "Unknown"
    else:
        return "Other"
df1["FAR.Desc"] = df1["Far.Description"].apply(map_far_description)
```

In [197]:

```
#rechecking counts in Far Description
df1["FAR.Desc"].value_counts()
```

Out[197]:

```
FAR.Desc
Unknown       57220
Private       24698
Commercial     3466
Government     3370
Other            17
Name: count, dtype: int64
```

Data Analysis

In [198]:

```
# getting list of airplane names and filtering their full data
top_aircraft_list = Top_Safe["Aircraft.Simple"].tolist()
Top_Safe_df1 = df1[df1["Aircraft.Simple"].isin(top_aircraft_list)]
Top_Safe_df1
```

Out[198]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Injury.Severity	Aircraft.I
3702	20001214X42024	Incident	LAX83IA073	1983-01-16	LOS ANGELES, CA	United States	Incident	
4350	20001214X42620	Incident	CHI83IA162	1983-04-10	MINNEAPOLIS, MN	United States	Incident	
4791	20001214X42899	Incident	CHI83IA228	1983-05-26	CLARION, PA	United States	Incident	U
5529	20001214X43650	Incident	CHI83IA327	1983-07-20	CHICAGO, IL	United States	Incident	U
7278	20001214X38628	Incident	MIA84IA064	1984-01-21	TAMPA, FL	United States	Incident	
8264	20001214X39496	Incident	CHI84IA196	1984-05-17	MINNEAPOLIS, MN	United States	Incident	
8782	20001214X40098	Incident	NYC84IA225	1984-06-27	PORTLAND, ME	United States	Incident	
10838	20001214X35647	Incident	CHI85IA118	1985-02-10	CHAMPAIGN, IL	United States	Incident	U
11103	20001214X35949	Incident	LAX85IA175B	1985-03-15	SAN JOSE, CA	United States	Incident	U
11445	20001214X36146	Incident	DCA85IA019	1985-04-25	DETROIT, MI	United States	Incident	U

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Injury.Severity	Aircraft.D
12924	20001214X37840	Incident	MIA85IA246	1985-09-08	WEST PALM BCH, FL	United States	Incident	
13866	20010110X00217	Incident	MIA86IA066	1986-01-27	MIAMI, FL	United States	Incident	U
15579	20001213X34445	Incident	DCA86IA037	1986-08-10	CHICAGO, IL	United States	Incident	De
16430	20001213X35323	Incident	CHI87IA039	1986-12-01	MADISON, WI	United States	Incident	
18859	20001213X32283	Incident	CHI88IA003	1987-10-05	MILWAUKEE, WI	United States	Incident	
19954	20001213X25244	Incident	DCA88IA044	1988-03-30	BOSTON, MA	United States	Incident	
20200	20001213X25661	Incident	DCA88IA056	1988-05-02	NR TOKYO, Japan	Japan	Incident	
20249	20001213X25749	Incident	MIA88IA167B	1988-05-10	CHICAGO, IL	United States	Incident	U
20756	20001213X26169	Incident	CHI88IA159	1988-07-06	INDIANAPOLIS, IN	United States	Incident	U
21075	20001213X26439	Incident	ATL88IA227	1988-08-05	ATLANTA, GA	United States	Incident	
21076	20001213X26438	Incident	ATL88IA226B	1988-08-05	ATLANTA, GA	United States	Incident	
21077	20001213X26438	Incident	ATL88IA226A	1988-08-05	ATLANTA, GA	United States	Incident	
22590	20001213X27893	Incident	FTW89IA070	1989-03-23	HOUSTON, TX	United States	Incident	
23733	20001213X29088	Incident	DCA89IA066	1989-08-09	DENVER, CO	United States	Incident	
24296	20001213X29581	Incident	DEN90IA012	1989-10-18	MONTE VISTA, CO	United States	Incident	
25818	20001212X23322	Incident	ATL90IA136	1990-06-21	ATLANTA, GA	United States	Incident	
29832	20001211X14149	Incident	MIA92IA077B	1992-02-08	MIAMI, FL	United States	Incident	
30046	20001211X14235	Incident	BFO92IA046	1992-03-19	LOUISVILLE, KY	United States	Incident	
32202	20001211X11785	Incident	BFO93IA026	1993-02-13	PORTLAND, ME	United States	Incident	U
32360	20001211X12035	Incident	NYC93IA059	1993-03-15	NEWARK, NJ	United States	Incident	U
33220	20001211X12828	Incident	CHI93IA248	1993-07-10	DETROIT, MI	United States	Incident	U
34493	20001206X00724	Incident	CHI94IA081	1994-02-09	CHICAGO, IL	United States	Incident	
37248	20001207X03525	Incident	NYC95IA106	1995-05-13	LOUISVILLE, KY	United States	Incident	
39109	20001208X05264	Incident	IAD96IA044	1996-02-20	WASHINGTON, DC	United States	Incident	
40290	20001208X06543	Incident	LAX96IA300	1996-08-08	HONOLULU, HI	United States	Incident	
42113	20001208X08107	Incident	DCA99WA071	1997-06-30	SAUDIA ARABIA, Saudi Arabia	Saudi Arabia	Incident	U
43735	20001211X09833	Incident	DCA98IA035	1998-04-20	ATLANTIC OCEAN	ATLANTIC OCEAN	Incident	U
43934	20001211X09999	Incident	CHI98IA164	1998-05-18	MINNEAPOLIS, MN	United States	Incident	U

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Injury.Severity	Aircraft.D
44997	200012[illegible]	Incident	[illegible]	1998-09-28	PARIS, France	France	Incident	
45312	20001211X11406	Incident	DCA99WA011	1998-11-27	JAKARTA, Indonesia	Indonesia	Incident	U
51162	20040914X01416	Incident	ENG01WA007	2001-07-30	Jeddah, Saudi Arabia	Saudi Arabia	Incident	U
51841	20020124X00124	Incident	DCA02WA011	2001-11-28	Lima, Peru	Peru	Incident	U
58004	20050106X00021	Incident	ANC05IA020	2004-12-29	Anchorage, AK	United States	Incident	
58909	20071218X01959	Incident	ENG05RA017	2005-06-21	Singapore, Singapore	Singapore	Incident	U
61972	20070803X01090	Incident	ENG07WA024	2007-01-23	Kota Kinabalu, Malaysia	Malaysia	Incident	U

45 rows × 33 columns

In [199]:

```
# grouping airplanes and their purpose
purpose_summary = Top_Safe_df1.groupby(["Aircraft.Simple", "Purpose.Of.Flight"])["Event.
Id"].count().reset_index()
purpose_summary.columns = ["Airplane", "Purpose of Flight", "Event Count"]
purpose_summary.columns
```

Out[199]:

```
Index(['Airplane', 'Purpose of Flight', 'Event Count'], dtype='object')
```

In [200]:

```
# grouping airplanes and Far desc
far_summary =Top_Safe_df1.groupby(["Aircraft.Simple", "FAR.Desc"])["Event.Id"].count().r
eset_index()
far_summary.columns = ["Airplane", "FAR Description", "Event Count"]
far_summary.columns
```

Out[200]:

```
Index(['Airplane', 'FAR Description', 'Event Count'], dtype='object')
```

In [201]:

```
# defining purpose for private or commercial
def classify_purpose(purpose):
    if purpose in ["Business", "Personal", "Instructional"]:
        return "Private"
    elif purpose in ["Cargo", "Commuter", "Ferry", "Other Work Use", "Aerial Observation
", "Positioning"]:
        return "Commercial"
    else:
        return "Unknown"

purpose_summary["Use Type"] = purpose_summary["Purpose of Flight"].apply(classify_purpose
)
```

In [202]:

```
# defining far desc
def classify_far(far):
    if "91" in str(far):
        return "Private"
    elif any(code in str(far) for code in ["121", "135", "129"]):
        return "Commercial"
    else:
        return "Unknown"
```

```
far_summary["Use Type"] = far_summary["FAR Description"].apply(classify_far)
```

In [203]:

```
# summary of purpose use
purpose_use = purpose_summary.groupby(["Airplane", "Use Type"])["Event Count"].sum().res
et_index()
purpose_use
```

Out[203]:

	Airplane	Use Type	Event Count
0	Airbus.Industrie A300-600	Unknown	8
1	Boeing 737-130	Unknown	6
2	Boeing 747-123	Private	1
3	Boeing 747-123	Unknown	6
4	Douglas DC-8-71	Unknown	5
5	Douglas DC-9-51	Unknown	7
6	Mcdonnell.Douglas DC-10-40	Unknown	7
7	Mcdonnell.Douglas DC-9	Unknown	5

In [204]:

```
# summary of far use
far_use = far_summary.groupby(["Airplane", "Use Type"])["Event Count"].sum().reset_index
()
far_use
```

Out[204]:

	Airplane	Use Type	Event Count
0	Airbus.Industrie A300-600	Unknown	8
1	Boeing 737-130	Unknown	6
2	Boeing 747-123	Unknown	7
3	Douglas DC-8-71	Unknown	5
4	Douglas DC-9-51	Unknown	7
5	Mcdonnell.Douglas DC-10-40	Unknown	7
6	Mcdonnell.Douglas DC-9	Unknown	5

In [205]:

```
# identifying airplanes used for more than one purpose
dual_use_purpose = purpose_use.groupby("Airplane")["Use Type"].nunique().reset_index()
dual_use_purpose = dual_use_purpose[dual_use_purpose["Use Type"] > 1]
dual_use_purpose
```

Out[205]:

	Airplane	Use Type
2	Boeing 747-123	2

In [206]:

```
# identifying airplanes used for more than one far desc
dual_use_far = far_use.groupby("Airplane")["Use Type"].nunique().reset_index()
dual_use_far = dual_use_far[dual_use_far["Use Type"] > 1]
dual_use_far
```

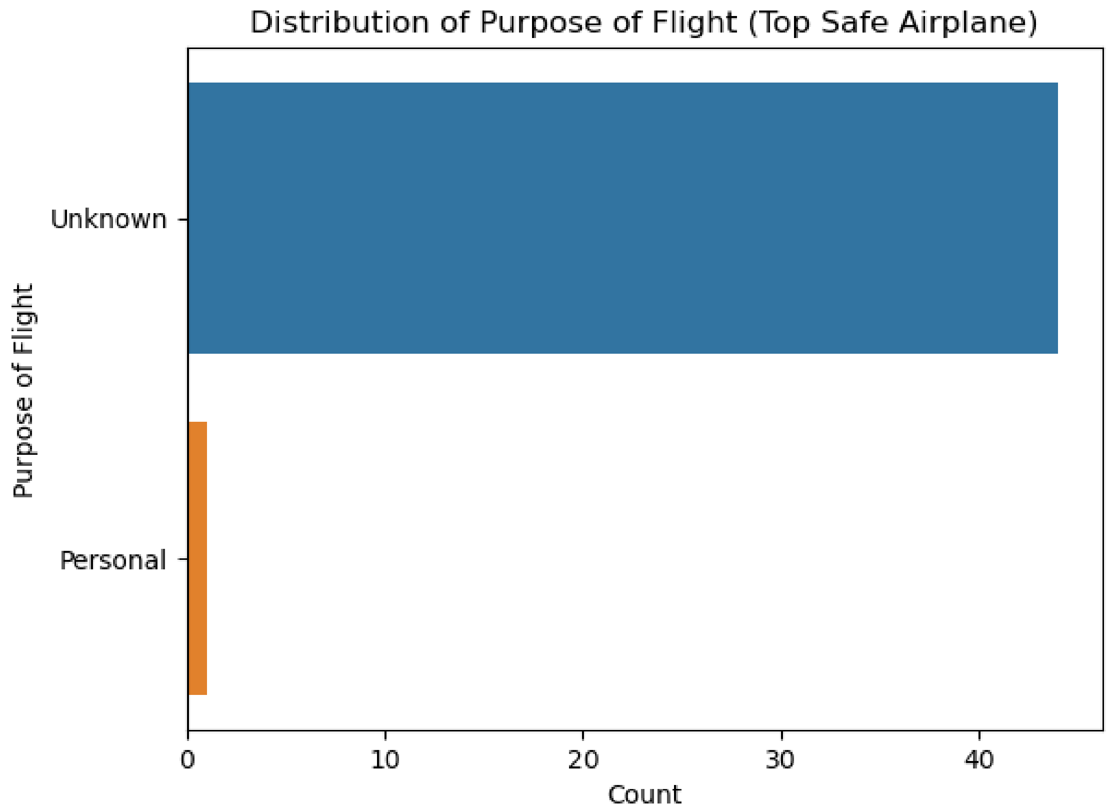
Out[206]:

Airplane	Use Type

Univariate Analysis: Distribution of Purpose of Flight

In [207]:

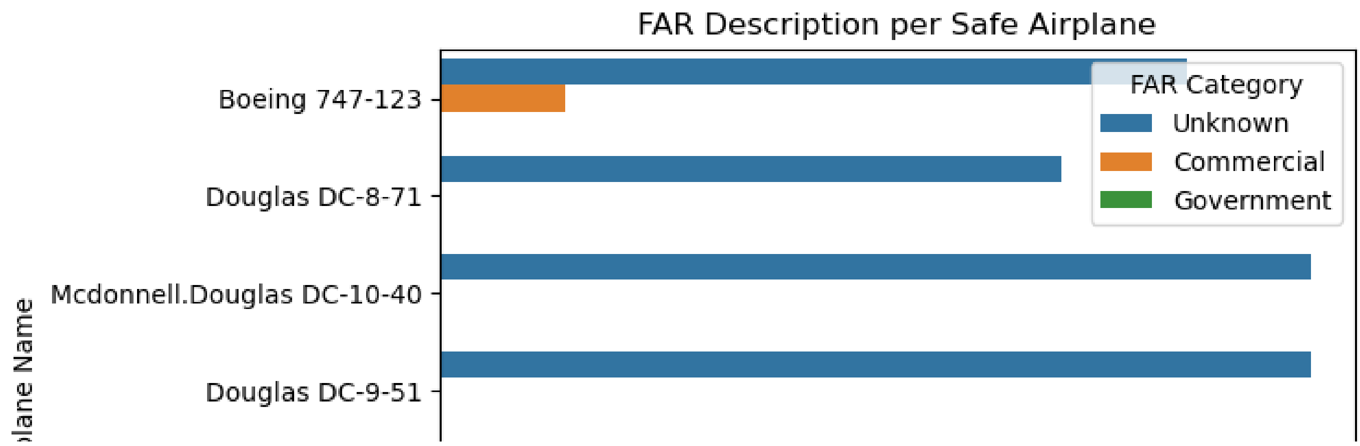
```
sns.countplot(data=Top_Safe_df1, y="Purpose.Of.Flight", order=Top_Safe_df1["Purpose.Of.Fl
ight"].value_counts().index)
plt.title("Distribution of Purpose of Flight (Top Safe Airplane)")
plt.xlabel("Count")
plt.ylabel("Purpose of Flight")
plt.show()
```

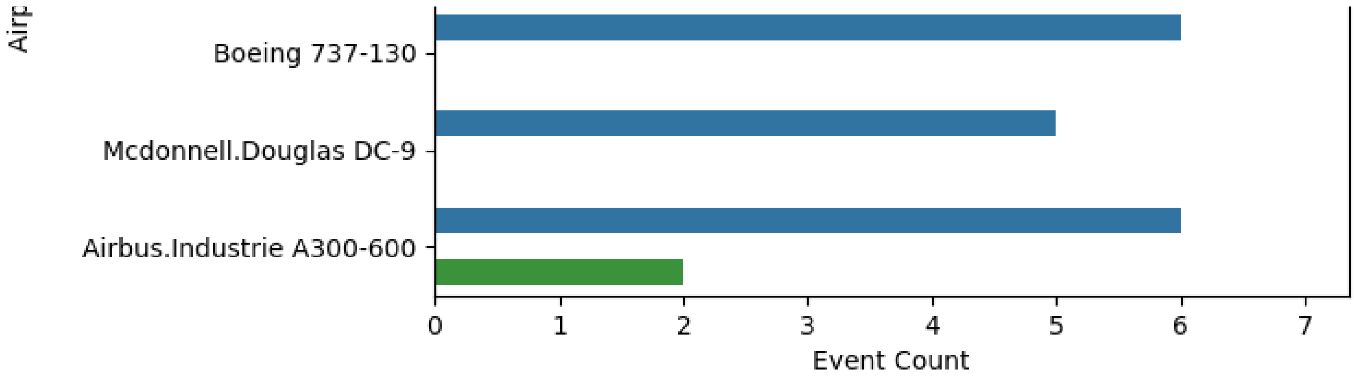


Bivariate Analysis: FAR Description by Airplanes

In [208]:

```
sns.countplot(data=Top_Safe_df1, y="Aircraft.Simple", hue="FAR.Desc")
plt.title("FAR Description per Safe Airplane")
plt.xlabel("Event Count")
plt.ylabel("Airplane Name")
plt.legend(title="FAR Category")
plt.show()
```

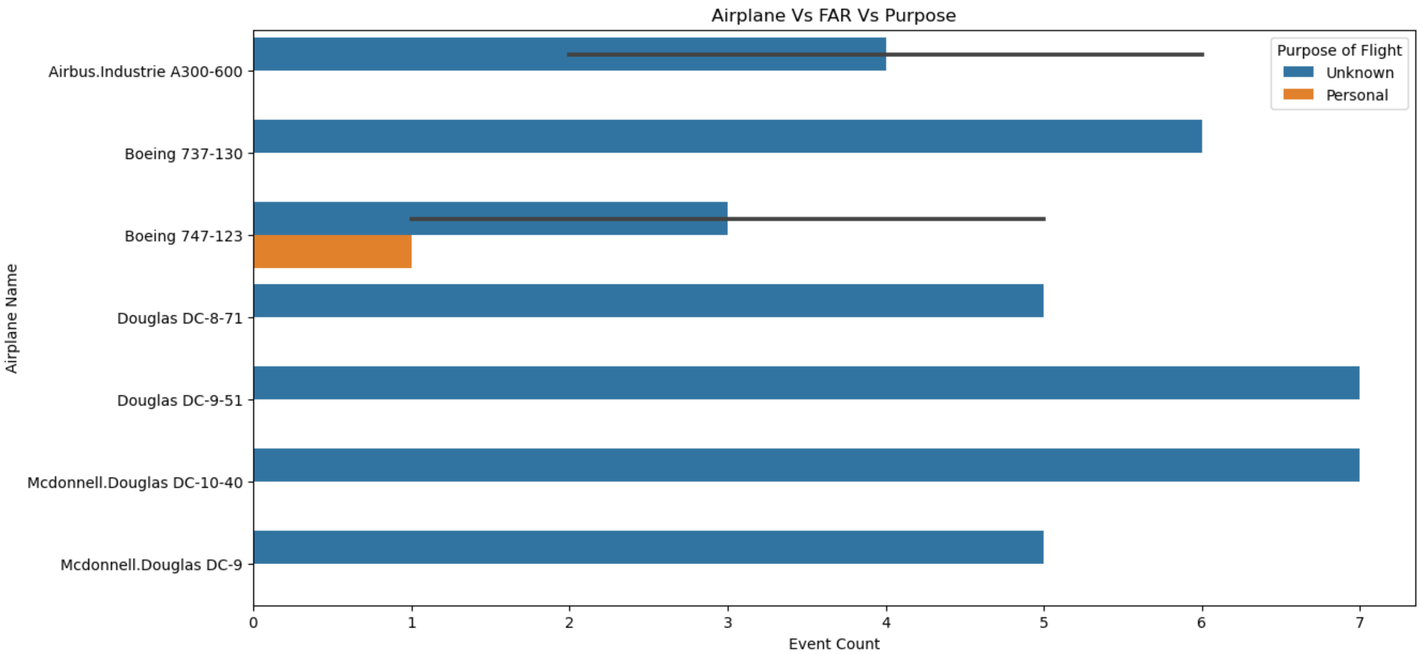




Multivariate Analysis: Aircraft Vs FAR Vs Purpose

In [209]:

```
grouped = Top_Safe_df1.groupby(["Aircraft.Simple", "FAR.Desc", "Purpose.Of.Flight"])["Event.Id"].count().reset_index()
grouped.columns = ["Airplane", "FAR Description", "Purpose", "Event Count"]

plt.figure(figsize=(14, 7))
sns.barplot(data=grouped, x="Event Count", y="Airplane", hue="Purpose")
plt.title("Airplane Vs FAR Vs Purpose")
plt.xlabel("Event Count")
plt.ylabel("Airplane Name")
plt.legend(title="Purpose of Flight")
plt.show()
```



Key Insights

Are the said safest airplanes useful for commercial and private operations?

- **The data on airplanes purpose were missing, hence the outcome of unknown use high counts.**
- `Boeing 747-123` **clearly determined by bivariate, multivariate analysis is used for both commercial and personal.**

Final Recommendation

Based on the dataset provided and the criteria of low injury severity and minimal damage, the `Boeing 747-123` **emerges as the most suitable aircraft for the new aviation division. It is not only identified as safe, but is also positively confirmed to serve both commercial and private operations, making it the ideal candidate for acquisition.**

acquisition.

Note: There were significant data gaps in the Purpose of Flight column for many airplane types, with a high proportion of records marked as “Unknown.” This may limit full operational visibility for some aircraft.

Nevertheless, based on their strong safety profiles, defined by high event counts, low fatality rates, and low levels of aircraft damage, the following airplanes are also recommended (despite their purpose being mostly unknown):

`Airbus Industrie A300-600`

`McDonnell Douglas DC-10-40`

`Douglas DC-9-51`

`Boeing 737-130`

These airplanes are ranked among the top five safest based on injury and damage metrics, and merit consideration for future expansion once operational data is clarified.

In [210]:

```
df1.to_csv("Cleaned_AviationData.csv", index=False)
```

In [211]:

```
df_cleaned= pd.read_csv("Cleaned_AviationData.csv")
df_cleaned.shape
```

Out[211]:

```
(88771, 33)
```

In [212]:

```
Top_Safe.to_csv("Top_10_Safe_Airplanes.csv")
```

In [213]:

```
Summary.to_csv("Aircraft_Event_Causes.csv", index=False)
```

In [214]:

```
purpose_use.to_csv("Purpose_Use_By_Airplane.csv", index=False)
```