Final Project Submission

```
Please fill out

Student name: Michelle Kavetza

Student pace: Full time

Scheduled project review date/time: 1st December,2024

Instructor name: William Okomba
```

Loading the Data Purpose: To import and load the aviation dataset for analysis.

Actions:

- Import necessary libraries like pandas, matplotlib, seaborn, etc.
- Load the dataset and check the first few rows to ensure it's loaded correctly.
- Verify the dataset's structure using `.info()` and `.shape`.

Outcome: Dataset loaded successfully, ready for exploration and cleaning.

In [208]:
```
# Your code here - remember to use markdown cells for comments as well!
# importing the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [209]:
```
# Displaying all columns in the data set using "pd.set_ption", "none" to mak
pd.set_option("display.max_columns", None)
df = pd.read_csv("Aviation_Data.csv", encoding='latin1')
df.head(10)
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_15732\631957593.py:3: DtypeWar
ning: Columns (6,7,28) have mixed types. Specify dtype option on import or
set low_memory=False.
  df = pd.read_csv("Aviation_Data.csv", encoding='latin1')
```

Out[209]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	United States
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	United States

In [210]:
```
#Checking the bottom columns of the dataset
df.tail()
```

Out[210]:

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

In [211]:
```
#Checking the dataset shape
df.shape
```

Out[211]:
```
(88889, 31)
```

In [212]:
```
#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 
 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
```

Data Overview

Purpose: To understand the dataset's columns, types, and initial quality.

Actions:

- Display data types for all columns.
- Check for missing values.
- Analyze basic statistics for numerical columns using `.describe()` .

Key Findings:

- Missing values in key columns like `Weather_Condition`, `Aircraft_Damage`, etc.
- Numerical columns like `Severity_Score` and `Total_Fatal_Injuries` have outliers.

In [213]:
```
#Checking the dataset data types
df.dtypes
```

Out[213]:
```
Event.Id                   object
Investigation.Type         object
Accident.Number            object
Event.Date                 object
Location                   object
Country                    object
Latitude                   object
Longitude                  object
Airport.Code               object
Airport.Name               object
Injury.Severity            object
Aircraft.damage            object
Aircraft.Category          object
Registration.Number        object
Make                       object
Model                      object
Amateur.Built              object
Number.of.Engines         float64
Engine.Type                object
FAR.Description            object
Schedule                   object
Purpose.of.flight          object
Air.carrier                object
Total.Fatal.Injuries      float64
Total.Serious.Injuries    float64
Total.Minor.Injuries      float64
Total.Uninjured           float64
Weather.Condition          object
Broad.phase.of.flight      object
Report.Status              object
Publication.Date           object
dtype: object
```

In [214]:
```
#Checking for misiing /null values, always returns a boolean
df.isnull()
```

Out[214]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude
0	False	False	False	False	False	False	True
1	False	False	False	False	False	False	True
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	True
4	False	False	False	False	False	False	True
...	...	...	...	...	...	...	...
88884	False	False	False	False	False	False	True
88885	False	False	False	False	False	False	True
88886	False	False	False	False	False	False	False
88887	False	False	False	False	False	False	True
88888	False	False	False	False	False	False	True

88889 rows × 31 columns

In [215]:
```
#Checking columns with the highest null values
df.isnull().sum().sort_values
```

Out[215]:
```
<bound method Series.sort_values of 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>
```

In [216]:
```
#Total number of null values
df.isnull().sum().sum()
```

Out[216]:
```
565032
```

In [217]:
```
#Concise summary statistics
df.describe()
```

Out[217]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.U
count	82805.000000	77488.000000	76379.000000	76956.000000	8297
mean	1.146585	0.647855	0.279881	0.357061	
std	0.446510	5.485960	1.544084	2.235625	2
min	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	380.000000	69

In [218]:
```
#Concise summary statistics by changing the position of the columns
df.describe().T
```

Out[218]:

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

In [219]:
```
#Checking for unique values in the entire dataframe
for column in df:
    unique_values = df[column].unique()
    print(f"Unique values in column '{column}', '\n': {unique_values}", '\n
```

```
Unique values in column 'Event.Id', '
': ['20001218X45444' '20001218X45447' '20061025X01555' ... '2022122710649
7'
 '20221227106498' '20221230106513']

Unique values in column 'Investigation.Type', '
': ['Accident' 'Incident']

Unique values in column 'Accident.Number', '
': ['SEA87LA080' 'LAX94LA336' 'NYC07LA005' ... 'WPR23LA075' 'WPR23LA076'
 'ERA23LA097']

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

Unique values in column 'Location', '
': ['MOOSE CREEK, ID' 'BRIDGEPORT, CA' 'Saltville, VA' ... 'San Manual, A
Z'
 'Auburn Hills, MI' 'Brasnorte, ']

Unique values in column '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' 'Belarus' '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' 'Guinea'
 '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' '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']

Unique values in column 'Latitude', '
': [nan 36.922223 42.445277 ... '321814N' '039101N' '373829N']

Unique values in column 'Longitude', '
': [nan -81.878056 -70.758333 ... '1114536W' '0835218W' '0121410W']

Unique values in column 'Airport.Code', '
': [nan 'N58' 'JAX' ... 'SKMD' 'OMAA' 'EIKH']

Unique values in column 'Airport.Name', '
': [nan 'BLACKBURN AG STRIP' 'HANOVER' ... 'HAWKINSVILLE-PULASKI COUNTY'
 'Lewiston Municipal Airport' 'WICHITA DWIGHT D EISENHOWER NT']

Unique values in column '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']

Unique values in column 'Aircraft.damage', '
': ['Destroyed' 'Substantial' 'Minor' nan 'Unknown']

Unique values in column 'Aircraft.Category', '
': [nan 'Airplane' 'Helicopter' 'Glider' 'Balloon' 'Gyrocraft' 'Ultraligh
t'
 'Unknown' 'Blimp' 'Powered-Lift' 'Weight-Shift' 'Powered Parachute'
 'Rocket' 'WSFT' 'UNK' 'ULTR']

Unique values in column 'Registration.Number', '
': ['NC6404' 'N5069P' 'N5142R' ... 'N749PJ' 'N210CU' 'N9026P']

Unique values in column 'Make', '
': ['Stinson' 'Piper' 'Cessna' ... 'JAMES R DERNOVSEK' 'ORLICAN S R O'
 'ROYSE RALPH L']

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

Unique values in column 'Amateur.Built', '
': ['No' 'Yes' nan]
```

```
Unique values in column 'Number.of.Engines', '
': [ 1. nan  2.  0.  3.  4.  8.  6.]

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

Unique values in column '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']

Unique values in column 'Schedule', '
': [nan 'SCHD' 'NSCH' 'UNK']

Unique values in column 'Purpose.of.flight', '
': ['Personal' nan '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' 'Air Race show'
 'PUBS' 'ASHO' 'PUBL']

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

Unique values in column '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.]

Unique values in column '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.]

Unique values in column '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.]

Unique values in column '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.]

Unique values in column 'Weather.Condition', '
': ['UNK' 'IMC' 'VMC' nan 'Unk']

Unique values in column 'Broad.phase.of.flight', '
': ['Cruise' 'Unknown' 'Approach' 'Climb' 'Takeoff' 'Landing' 'Taxi'
 'Descent' 'Maneuvering' 'Standing' 'Go-around' 'Other' nan]

Unique values in column 'Report.Status', '
': ['Probable Cause' 'Factual' 'Foreign' ...
 'The pilot did not ensure adequate clearance from construction vehicles d
uring taxi.'
 'The pilot\x92s failure to secure the magneto switch before attempting to
hand rotate the engine which resulted in an inadvertent engine start, a ru
naway airplane, and subsequent impact with parked airplanes. Contributing
to the accident was the failure to properly secure the airplane with chock
s.'
 'The pilot\x92s loss of control due to a wind gust during landing.']

Unique values in column 'Publication.Date', '
': [nan '19-09-1996' '26-02-2007' ... '22-12-2022' '23-12-2022' '29-12-202
2']
```

Data Cleaning

Purpose: To handle missing values, duplicates, and inconsistent data.

Actions:

1. **Handling Missing Values:**

- Dropped non-relevant columns with excessive missing data.
- Imputed missing values for categorical fields like `Weather_Condition` using mode.

2. **Removing Duplicates:**

   - Dropped duplicate entries.

3. **Standardizing Data:**

   - Corrected inconsistent entries in categorical columns like `Aircraft_Category` .
   - Converted numerical columns to appropriate types.

Outcome: Clean and consistent dataset ready for analysis

In [220]:

```
#First creating a copy to be used for data cleaning, so as to not tamper wi
df1 = df.copy()
df1.head()
```

Out[220]:

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

In [221]:

```
type(df1)
```

Out[221]:

```
pandas.core.frame.DataFrame
```

In [222]:

```
df1.columns
```

Out[222]:

```
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.Descripti
on',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuri
es',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [223]:
```
df1.rename(columns=lambda x: x.replace(".", "_").strip().title(), inplace=Tr
df1
```

Out[223]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Co
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	U S
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	U S
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	U S
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	U S
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	U S
...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	U S
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	U S
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	U S
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	U S
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	U S

88889 rows × 31 columns

In [224]:
```
df1["Make"] = df1["Make"].str.lower()
df1["Make"].unique()
```

Out[224]:
```
array(['stinson', 'piper', 'cessna', ..., 'james r dernovsek',
       'orlican s r o', 'royse ralph l'], dtype=object)
```

In [225]:
```
df1.columns = df1.columns.str.replace(" ", "")
df1.columns
```

Out[225]:
```
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_Descripti
on',
       'Schedule', 'Purpose_Of_Flight', 'Air_Carrier', 'Total_Fatal_Injuri
es',
       'Total_Serious_Injuries', 'Total_Minor_Injuries', 'Total_Uninjure
d',
       'Weather_Condition', 'Broad_Phase_Of_Flight', 'Report_Status',
       'Publication_Date'],
      dtype='object')
```

Converting the columns from integer to string for categorical data and float for numerical data

In [226]:

```
# Converting Event.Date, Publication.Date to Datetime
df1["Event_Date"] = pd.to_datetime(df1["Event_Date"], errors='coerce')

#Converting Number.of.Engines to integer
df1["Number_Of_Engines"] = df1["Number_Of_Engines"].fillna(0).astype(int)
```

In [227]:

```
df1["Year"] = df1["Event_Date"].dt.year
df1[["Year"]].head(10)
```

Out[227]:

	Year
0	1948
1	1962
2	1974
3	1977
4	1979
5	1979
6	1981
7	1982
8	1982
9	1982

Dealing with missing values

In [228]:
```
df1.isnull().sum().sort_values()
```

Out[228]:
```
Event_Id                    0
Number_Of_Engines           0
Event_Date                  0
Year                        0
Investigation_Type          0
Accident_Number             0
Location                   52
Make                       63
Model                      92
Amateur_Built             102
Country                   226
Injury_Severity          1000
Registration_Number      1382
Aircraft_Damage          3194
Weather_Condition        4492
Total_Uninjured          5912
Purpose_Of_Flight        6192
Report_Status            6384
Engine_Type              7096
Total_Fatal_Injuries    11401
Total_Minor_Injuries    11933
Total_Serious_Injuries  12510
Publication_Date        13771
Broad_Phase_Of_Flight   27165
Airport_Name            36185
Airport_Code            38757
Latitude                54507
Longitude               54516
Aircraft_Category       56602
Far_Description         56866
Air_Carrier             72241
Schedule                76307
dtype: int64
```

In [229]:
```
# Calculate percentage of missing values
missing_percentage = df1.isnull().mean() * 100

# Identify columns with more than 50% missing values
high_missing_cols = missing_percentage[missing_percentage > 50]

# Display the columns and their missing percentages
print(high_missing_cols)
```

```
Latitude             61.320298
Longitude            61.330423
Aircraft_Category    63.677170
Far_Description      63.974170
Schedule             85.845268
Air_Carrier          81.271023
dtype: float64
```

In [230]:
```
#List of columns to drop
columns_to_drop = ["Longitude","Latitude", "Publication_Date","Air_Carrier",

# Drop the specified columns from the dataframe
df1 = df1.drop(columns=columns_to_drop, axis=1, errors="ignore")

# Validate the resulting dataframe
print(f"Columns dropped: {columns_to_drop}")
print("Remaining columns:")
print(df1.columns)
print(f"Dataset size after dropping columns: {df1.shape}")
```

```
Columns dropped: ['Longitude', 'Latitude', 'Publication_Date', 'Air_Carrie
r', 'Report_Status', 'Airport_Code', 'Airport_Name', 'Registration_Numbe
r', 'Schedule', 'Far_Description']
Remaining columns:
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage',
       'Aircraft_Category', 'Make', 'Model', 'Amateur_Built',
       'Number_Of_Engines', 'Engine_Type', 'Purpose_Of_Flight',
       'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Weather_Condition',
       'Broad_Phase_Of_Flight', 'Year'],
      dtype='object')
Dataset size after dropping columns: (88889, 22)
```

In [231]:
```
# List categorical columns for imputation
categorical_columns = ['Injury_Severity', 'Aircraft_Category', 'Weather_Conc

# Impute missing values with the most frequent value (mode)
for col in categorical_columns:
    df1[col].fillna(df1[col].mode()[0], inplace=True)
```

In [232]:
```
#List numeric columns for imputation
numeric_columns = ['Total_Fatal_Injuries', 'Total_Serious_Injuries', 'Total_

# Impute missing values with the median value (less sensitive to outliers)
for col in numeric_columns:
    df1[col].fillna(df1[col].median(), inplace=True)
```

In [ ]:

In [233]:
```
# Fill missing year with the median value
df1['Year'].fillna(df1['Year'].median(), inplace=True)
df1['Year'] = df1['Year'].round(0).astype(int)
df1[['Year']]
```

Out[233]:

	Year
0	1948
1	1962
2	1974
3	1977
4	1979
...	...
88884	2022
88885	2022
88886	2022
88887	2022
88888	2022

88889 rows × 1 columns

In [234]:
```
#filter the dataset from 1962 as for our analysis
df1 = df1[df1['Year'] >= 1962]
df1
```

Out[234]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Co
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	U S
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	U S
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	U S
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	U S
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	U S
...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	U S
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	U S
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	U S
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	U S
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	U S

88888 rows × 22 columns

In [235]:
```
# Recalculate the percentage of missing values
missing_percentage_after = df1.isnull().mean() * 100

# Display any remaining columns with missing values
remaining_missing_columns = missing_percentage_after[missing_percentage_afte
print(remaining_missing_columns)
```

```
Location                 0.058501
Country                  0.254253
Aircraft_Damage          3.593286
Make                     0.070876
Model                    0.103501
Amateur_Built            0.114751
Engine_Type              7.983080
Purpose_Of_Flight        6.966070
Broad_Phase_Of_Flight   30.560931
dtype: float64
```

In [236]:
```
# Impute 'Air_Carrier' and 'Broad_Phase_Of_Flight' with 'Unknown' for missir
df1['Broad_Phase_Of_Flight'].fillna('Unknown', inplace=True)
```

In [237]:
```
# Impute columns with low missing values (less than 10%)
df1['Location'].fillna(df1['Location'].mode()[0], inplace=True)
df1['Country'].fillna(df1['Country'].mode()[0], inplace=True)
df1['Aircraft_Damage'].fillna(df1['Aircraft_Damage'].mode()[0], inplace=Tru
df1['Make'].fillna(df1['Make'].mode()[0], inplace=True)
df1['Model'].fillna(df1['Model'].mode()[0], inplace=True)

# Impute columns with moderate missing values (between 10% and 50%)
df1['Number_Of_Engines'].fillna(df1['Number_Of_Engines'].median(), inplace=
#df1['Engine_Type'].fillna(df1['Engine_Type'].mode()[0], inplace=True)
df1['Purpose_Of_Flight'].fillna(df1['Purpose_Of_Flight'].mode()[0], inplace
```

In [238]:
```
df1.dropna(inplace=True)
```

In [239]:
```
df1.isnull().sum()
```

Out[239]:
```
Event_Id                  0
Investigation_Type        0
Accident_Number           0
Event_Date                0
Location                  0
Country                   0
Injury_Severity           0
Aircraft_Damage           0
Aircraft_Category         0
Make                      0
Model                     0
Amateur_Built             0
Number_Of_Engines         0
Engine_Type               0
Purpose_Of_Flight         0
Total_Fatal_Injuries      0
Total_Serious_Injuries    0
Total_Minor_Injuries      0
Total_Uninjured           0
Weather_Condition         0
Broad_Phase_Of_Flight     0
Year                      0
dtype: int64
```

In [240]:
```
df1.shape
```

Out[240]:
```
(81771, 22)
```

In [241]:
```
df1.columns
```

Out[241]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage',
       'Aircraft_Category', 'Make', 'Model', 'Amateur_Built',
       'Number_Of_Engines', 'Engine_Type', 'Purpose_Of_Flight',
       'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Weather_Condition',
       'Broad_Phase_Of_Flight', 'Year'],
      dtype='object')
```

In [242]:
```
# Check remaining missing values
missing_percentage_after = df1.isnull().mean() * 100
remaining_missing_columns = missing_percentage_after[missing_percentage_afte
print(remaining_missing_columns)
```

```
Series([], dtype: float64)
```

In [243]:
```
df1.isnull().sum()
```

Out[243]:
```
Event_Id                 0
Investigation_Type       0
Accident_Number          0
Event_Date               0
Location                 0
Country                  0
Injury_Severity          0
Aircraft_Damage          0
Aircraft_Category        0
Make                     0
Model                    0
Amateur_Built            0
Number_Of_Engines        0
Engine_Type              0
Purpose_Of_Flight        0
Total_Fatal_Injuries     0
Total_Serious_Injuries   0
Total_Minor_Injuries     0
Total_Uninjured          0
Weather_Condition        0
Broad_Phase_Of_Flight    0
Year                     0
dtype: int64
```

Adding Columns

Severity score is to quantify the overall severity of each accident using the data from the columns Total_Fatal_Injuries, Total_Serious_Injuries, Total_Minor_Injuries. we assign; 3 to fatal injuries 2 to serious injuries 1 to minor injuries

In [244]:
```
df1.columns
```

Out[244]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage',
       'Aircraft_Category', 'Make', 'Model', 'Amateur_Built',
       'Number_Of_Engines', 'Engine_Type', 'Purpose_Of_Flight',
       'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Weather_Condition',
       'Broad_Phase_Of_Flight', 'Year'],
      dtype='object')
```

In [245]:
```
df1["Severity_Score"] = (
    df1["Total_Fatal_Injuries"].fillna(0) * 3 +
    df1["Total_Serious_Injuries"].fillna(0) * 2 +
    df1["Total_Minor_Injuries"].fillna(0) * 1
  )
df1[["Severity_Score"]]
```

Out[245]:

	Severity_Score
1	12.0
2	9.0
3	6.0
5	1.0
6	12.0
...	...
88639	0.0
88647	0.0
88661	0.0
88735	2.0
88767	0.0

81771 rows × 1 columns

In [246]:
```
#we categorise the Severity_score as either high, low or medium

def risk_category(score):
    if score <=5:
        return "Low"
    elif score <=10:
        return "Medium"
    else:
        return "High"
    df1["Risk_Category"] = df1["Severity_Score"].apply(risk_category)
```

In [247]:
```
df1['Month'] = df1['Event_Date'].dt.month

# Import required library
import calendar

# Replace month numbers with month names
df1['Month'] = df1['Month'].apply(lambda x: calendar.month_name[x])

# Check the DataFrame to confirm the replacement
df1[['Month']].head()
```

Out[247]:

	Month
1	July
2	August
3	June
5	September
6	August

In [248]:
```
#Creating a new column for Month and Days

df1["Day"] = df1["Event_Date"].dt.day

df1[["Day"]]
```

Out[248]:

	Day
1	19
2	30
3	19
5	17
6	1
...	...
88639	6
88647	8
88661	13
88735	29
88767	9

81771 rows × 1 columns

In [249]:
```
#Splitting Location into City and state
df1[["City","State"]] = df1["Location"].str.split(',', expand=True , n=1)
# Removing leading/trailing spaces (if any)
df1["City"] = df1["City"].str.strip()
df1["State"] = df1["State"].str.strip()

# Check the result
print(df1[["Location", "City", "State"]].head())
```

```
         Location        City State
1  BRIDGEPORT, CA  BRIDGEPORT    CA
2   Saltville, VA   Saltville    VA
3      EUREKA, CA      EUREKA    CA
5      BOSTON, MA      BOSTON    MA
6      COTTON, MN      COTTON    MN
```

In [250]:
```
df1.dtypes
```

Out[250]:
```
Event_Id                          object
Investigation_Type                object
Accident_Number                   object
Event_Date                datetime64[ns]
Location                          object
Country                           object
Injury_Severity                   object
Aircraft_Damage                   object
Aircraft_Category                 object
Make                              object
Model                             object
Amateur_Built                     object
Number_Of_Engines                  int32
Engine_Type                       object
Purpose_Of_Flight                 object
Total_Fatal_Injuries             float64
Total_Serious_Injuries           float64
Total_Minor_Injuries             float64
Total_Uninjured                  float64
Weather_Condition                 object
Broad_Phase_Of_Flight             object
Year                               int32
Severity_Score                   float64
Month                             object
Day                                int32
City                              object
State                             object
dtype: object
```

In [251]:
```
df1.isnull().sum()
```

Out[251]:
```
Event_Id                   0
Investigation_Type         0
Accident_Number            0
Event_Date                 0
Location                   0
Country                    0
Injury_Severity            0
Aircraft_Damage            0
Aircraft_Category          0
Make                       0
Model                      0
Amateur_Built              0
Number_Of_Engines          0
Engine_Type                0
Purpose_Of_Flight          0
Total_Fatal_Injuries       0
Total_Serious_Injuries     0
Total_Minor_Injuries       0
Total_Uninjured            0
Weather_Condition          0
Broad_Phase_Of_Flight      0
Year                       0
Severity_Score             0
Month                      0
Day                        0
City                       0
State                    499
dtype: int64
```

In [252]:
```
df1['City'].fillna('Unknown', inplace=True)
df1['State'].fillna('Unknown', inplace=True)
```

In [253]:
```
print(df1[['City', 'State']].isnull().sum())
```

```
City     0
State    0
dtype: int64
```

In [254]:
```
df1.dropna(subset=['City', 'State'], inplace=True)
```

In [255]:
```
df1.isnull().sum()
```

Out[255]:
```
Event_Id                  0
Investigation_Type        0
Accident_Number           0
Event_Date                0
Location                  0
Country                   0
Injury_Severity           0
Aircraft_Damage           0
Aircraft_Category         0
Make                      0
Model                     0
Amateur_Built             0
Number_Of_Engines         0
Engine_Type               0
Purpose_Of_Flight         0
Total_Fatal_Injuries      0
Total_Serious_Injuries    0
Total_Minor_Injuries      0
Total_Uninjured           0
Weather_Condition         0
Broad_Phase_Of_Flight     0
Year                      0
Severity_Score            0
Month                     0
Day                       0
City                      0
State                     0
dtype: int64
```

Checking for Duplicates

In [256]:
```
df1.duplicated().sum()
```

Out[256]:
```
0
```

In [257]:
```
# Identify the duplicate rows
duplicate_rows = df1[df1.duplicated(keep=False)]

# Display the duplicate rows
print("Duplicate rows:")
print(duplicate_rows)
```

```
Duplicate rows:
Empty DataFrame
Columns: [Event_Id, Investigation_Type, Accident_Number, Event_Date, Locat
ion, Country, Injury_Severity, Aircraft_Damage, Aircraft_Category, Make, M
odel, Amateur_Built, Number_Of_Engines, Engine_Type, Purpose_Of_Flight, To
tal_Fatal_Injuries, Total_Serious_Injuries, Total_Minor_Injuries, Total_Un
injured, Weather_Condition, Broad_Phase_Of_Flight, Year, Severity_Score, M
onth, Day, City, State]
Index: []
```

In [258]:
```
#Keep the first occurrence:
df2 = df1.drop_duplicates(keep='first')
```

In [259]:
```
df2.duplicated().sum().any()
```

Out[259]:
```
False
```

In [260]:
```
df2.head(10)
```

Out[260]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	C
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	
6	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	

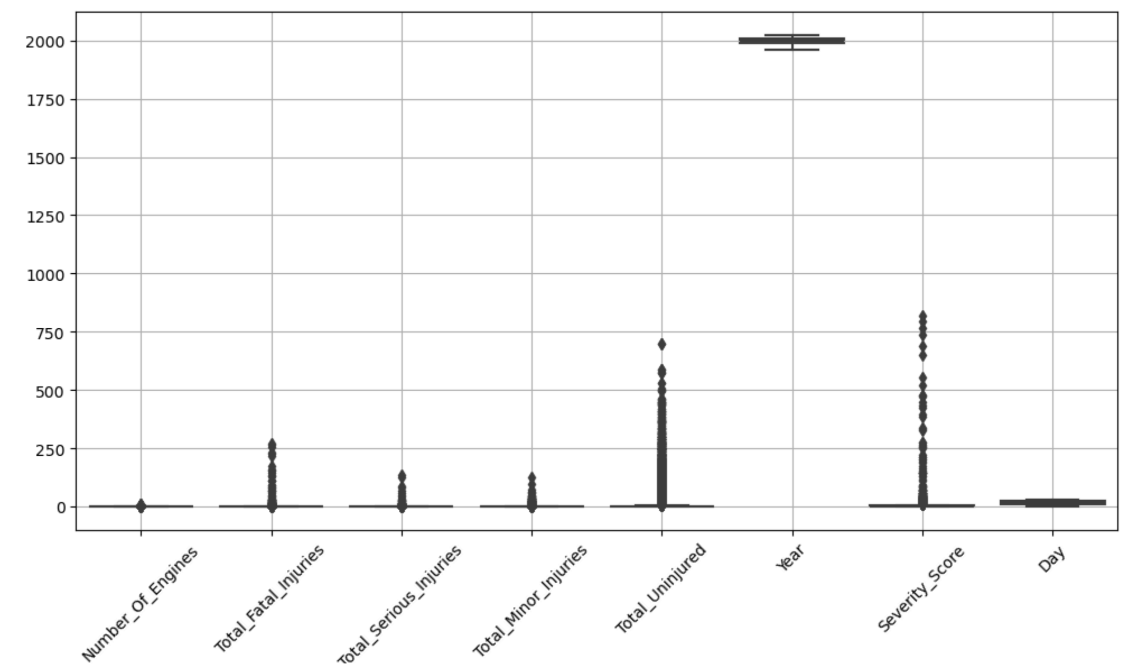
In [261]:
```
# Investigate rows with zero engines
zero_engines = df2[df2['Number_Of_Engines'] == 0]
print(zero_engines)

# Drop rows if confirmed as invalid
df2 = df2[df2['Number_Of_Engines'] != 0]
```

```
             Event_Id Investigation_Type Accident_Number Event_Date  \
62     20020917X02247           Accident      LAX82DVG13 1982-01-09
247    20020917X02190           Accident      LAX82DA098 1982-02-06
353    20020917X02298           Accident      LAX82FUJ28 1982-02-19
433    20020917X01824           Accident      CHI82DA076 1982-02-27
436    20020917X02181           Accident      LAX82DA089 1982-02-28
...               ...                ...             ...        ...
85632  20201215102416           Accident      CEN21LA087 2020-11-23
85644  20201130102350           Accident      CEN21LA069 2020-11-28
87418  20220216104650           Accident      CEN22LA122 2022-02-12
87658  20220411104927           Accident      WPR22LA150 2022-04-08
88322  20220808105678           Accident      CEN22LA363 2022-08-07

             Location        Country Injury_Severity Aircraft_Damage  \
62      CALISTOGA, CA  United States       Non-Fatal     Substantial
247      GLENDALE, AZ  United States       Non-Fatal     Substantial
353       PHOENIX, AZ  United States       Non-Fatal     Substantial
433    CINCINNATI, OH  United States       Non-Fatal     Substantial
436          NAPA, CA  United States       Non-Fatal       Destroyed
...               ...            ...             ...             ...
85632     Gilliam, LA  United States       Non-Fatal     Substantial
85644      Ottawa, KS  United States       Non-Fatal     Substantial
87418    Eldorado, IL  United States         Serious           Minor
87658    Valyermo, CA  United States       Non-Fatal     Substantial
88322      Waller, TX  United States       Non-Fatal     Substantial

      Aircraft_Category                            Make        Model  \
62               Glider                      schleicher       ASW 20
247             Balloon                           raven        S-55A
353             Balloon                   balloon works      FIREFLY
433             Balloon                          barnes    FIREFLY-7
436             Balloon                          barnes  BALLOON AX7
...                 ...                             ...          ...
85632          Airplane                           piper         PA12
85644          Airplane                           globe         GC1A
87418        Helicopter                            bell          206
87658            Glider  alexander schleicher gmbh & co       ASK 21
88322            Glider                      schleicher      ASW-20B

      Amateur_Built  Number_Of_Engines    Engine_Type Purpose_Of_Flight  \
62               No                  0        Unknown          Personal
247              No                  0        Unknown          Personal
353              No                  0        Unknown          Personal
433              No                  0        Unknown          Personal
436              No                  0        Unknown           Unknown
...             ...                ...            ...               ...
85632            No                  0  Reciprocating          Personal
85644            No                  0  Reciprocating          Personal
87418            No                  0    Turbo Shaft          Business
87658           Yes                  0           NONE          Personal
88322            No                  0           NONE          Personal

       Total_Fatal_Injuries  Total_Serious_Injuries  Total_Minor_Injuries
\
62                      0.0                     0.0                   0.0
247                     0.0                     0.0                   0.0
353                     0.0                     0.0                   0.0
433                     0.0                     1.0                   1.0
436                     0.0                     0.0                   1.0
...                     ...                     ...                   ...
85632                   0.0                     0.0                   0.0
```

```
85644              0.0              1.0              0.0
87418              0.0              0.0              1.0
87658              0.0              0.0              0.0
88322              0.0              0.0              0.0

       Total_Uninjured Weather_Condition Broad_Phase_Of_Flight  Year  \
62                 1.0               VMC               Landing  1982
247                2.0               VMC               Landing  1982
353                3.0               VMC               Landing  1982
433                2.0               VMC               Takeoff  1982
436                4.0               VMC               Landing  1982
...                ...               ...                   ...   ...
85632              1.0               Unk               Unknown  2020
85644              0.0               VMC               Unknown  2020
87418              3.0               VMC               Unknown  2022
87658              1.0               VMC               Unknown  2022
88322              1.0               VMC               Unknown  2022

       Severity_Score     Month  Day        City State
62                0.0   January    9   CALISTOGA    CA
247               0.0  February    6    GLENDALE    AZ
353               0.0  February   19     PHOENIX    AZ
433               3.0  February   27  CINCINNATI    OH
436               1.0  February   28        NAPA    CA
...               ...       ...  ...         ...   ...
85632             0.0  November   23     Gilliam    LA
85644             2.0  November   28      Ottawa    KS
87418             1.0  February   12    Eldorado    IL
87658             0.0     April    8    Valyermo    CA
88322             0.0    August    7      Waller    TX

[2721 rows x 27 columns]
```

Checking for Outliers

In [262]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(df2).grid()
plt.xticks(rotation=45);
```

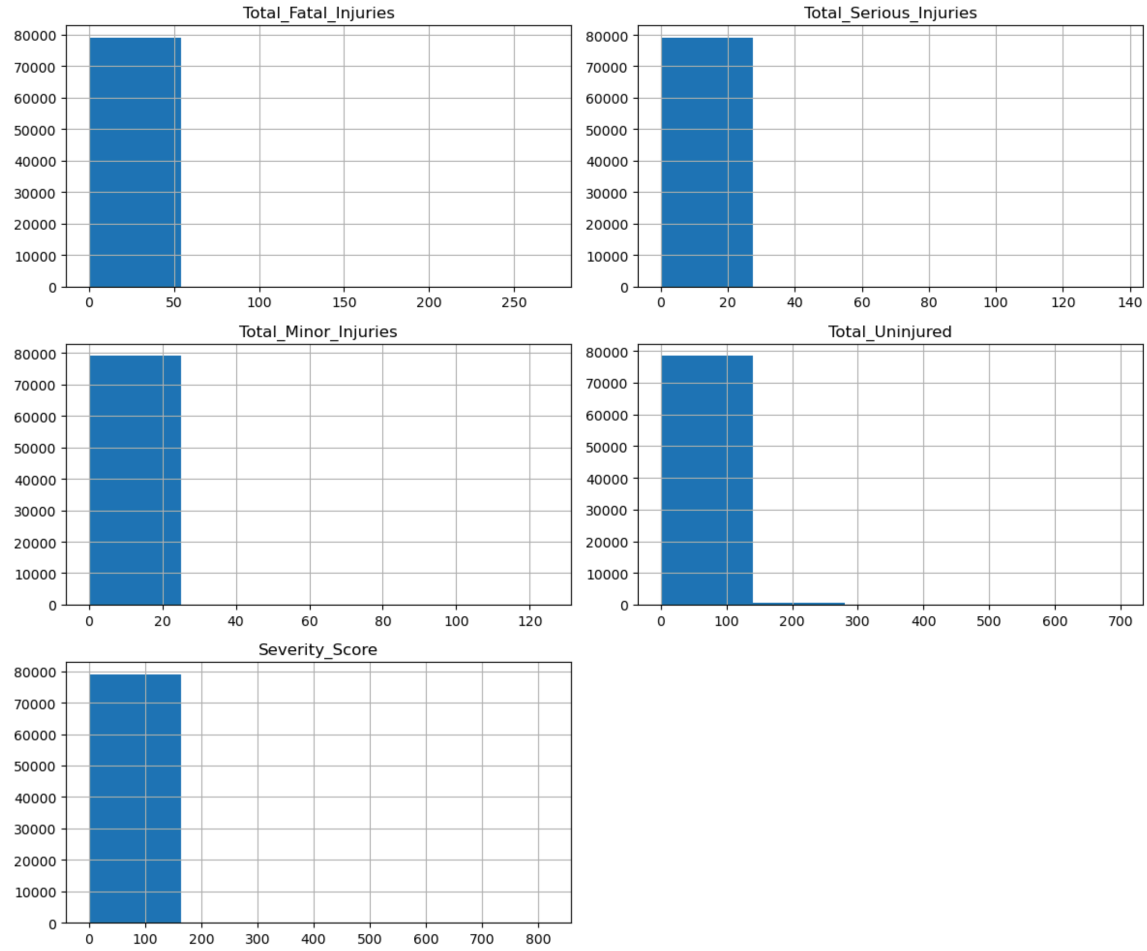


In [263]:
```
df2.columns
```

Out[263]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage',
       'Aircraft_Category', 'Make', 'Model', 'Amateur_Built',
       'Number_Of_Engines', 'Engine_Type', 'Purpose_Of_Flight',
       'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Weather_Condition',
       'Broad_Phase_Of_Flight', 'Year', 'Severity_Score', 'Month', 'Day',
       'City', 'State'],
      dtype='object')
```

In [264]:
```
numerical_cols = df2.select_dtypes(include=['float64', 'int64']).columns
```

In [265]:
```
# change x limit for evry plot
df2[numerical_cols].hist(figsize=(12, 10), bins=5)
plt.tight_layout()
plt.show()
```



In [266]:
```
# Calculate Q1 (25th percentile) and Q3 (75th percentile

Q1 = df2[numerical_cols].quantile(0.25)
Q3 = df2[numerical_cols].quantile(0.95)
IQR = Q3 - Q1

# Detect outliers using IQR
outliers_IQR = ((df2[numerical_cols] < (Q1 - 1.5 * IQR)) | (df2[numerical_co

# Display rows with outliers
outlier_rows = df2[outliers_IQR.any(axis=1)]
outlier_rows
```

Out[266]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Co
5	20170710X52551	Accident	NYC79AA106	1979-09-17	BOSTON, MA	
25	20020917X01905	Accident	DCA82AA008	1982-01-03	ASHLAND, VA	
79	20020917X01897	Incident	CHI82IA026	1982-01-12	CHICAGO, IL	
84	20020917X01907	Accident	DCA82AA011	1982-01-13	WASHINGTON, DC	
93	20020917X02538	Accident	NYC82FA021	1982-01-15	JAMAICA, NY	
...	...	...	...	...	...	
87866	20220602105173	Accident	CEN22LA222	2022-05-28	Canyon Lake, TX	
87932	20220608105217	Accident	WPR22LA201	2022-06-07	Hawthorne, CA	
88011	20220623105319	Accident	CEN22LA262	2022-06-21	Cresson, TX	
88083	20220718105497	Accident	DCA22LA151	2022-07-02	Santa Ana, CA	
88244	20220802105639	Accident	ANC22LA063	2022-07-26	Anchorage, AK	

3451 rows × 27 columns

In [267]:
```
(Q1 - 2.5 * IQR)
```

Out[267]:
```
Total_Fatal_Injuries       -5.0
Total_Serious_Injuries     -2.5
Total_Minor_Injuries       -5.0
Total_Uninjured           -10.0
Severity_Score            -17.5
dtype: float64
```

In [268]:
```
(Q3 + 1.5 * IQR)
```

Out[268]:
```
Total_Fatal_Injuries       5.0
Total_Serious_Injuries     2.5
Total_Minor_Injuries       5.0
Total_Uninjured           10.0
Severity_Score            17.5
dtype: float64
```

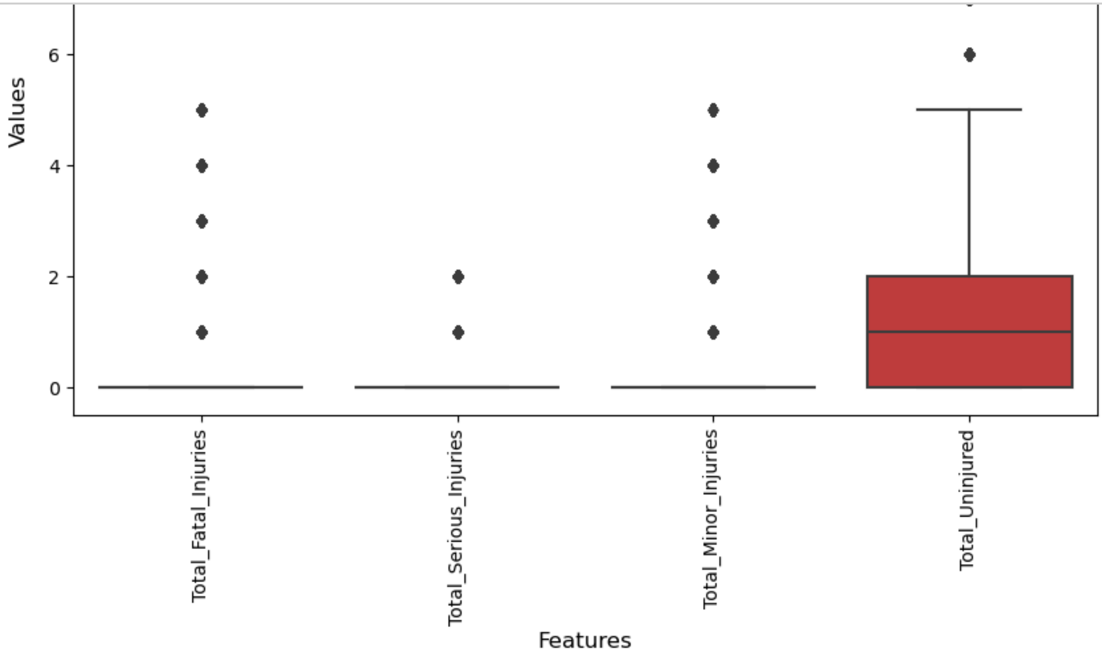
In [269]:
```
#Define numerical columns (replace with your specific column list)
numerical_cols = [
    "Number_Of_Engines", "Total_Fatal_Injuries", "Total_Serious_Injuries",
    "Total_Minor_Injuries", "Total_Uninjured", "Severity_Score"
]

# Calculate Q1, Q3, and IQR
Q1 = df2[numerical_cols].quantile(0.25)
Q3 = df2[numerical_cols].quantile(0.95)
IQR = Q3 - Q1

# Determine the lower and upper bounds for filtering
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove rows containing outliers
df_filtered = df2[
    ~((df2[numerical_cols] < lower_bound) | (df2[numerical_cols] > upper_bou
]

# Display the shape of the dataset before and after filtering
print("Original shape:", df2.shape)
print("Filtered shape:", df_filtered.shape)
```

```
Original shape: (79050, 27)
Filtered shape: (75393, 27)
```

In [270]:
```
numeric_columns
```

Out[270]:
```
['Total_Fatal_Injuries',
 'Total_Serious_Injuries',
 'Total_Minor_Injuries',
 'Total_Uninjured']
```

In [271]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_filtered[['Total_Fatal_Injuries','Total_Serious_Injuries
plt.title("After Removing Outliers")
plt.xlabel("Features", fontsize=12)
plt.ylabel("Values", fontsize=12)
plt.xticks(rotation=90)
#plt.tight_layout() # Tighten layout to prevent label cutoff
plt.show()
```



In [272]:
```
# Check for remaining outliers in the filtered dataset
remaining_outliers = (
    (df_filtered[numerical_cols] < lower_bound) |
    (df_filtered[numerical_cols] > upper_bound)
).any(axis=1)

print("Number of remaining outliers:", remaining_outliers.sum())
```

```
Number of remaining outliers: 0
```

In [273]:
```
df_filtered.shape
```

Out[273]:
```
(75393, 27)
```

In [274]:
```
state_df = pd.read_csv("USState_Codes.csv")
state_df.head()
```

Out[274]:

	US_State	Abbreviation
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA

In [275]:
```
#Ensure column names align for mapping
# Assuming data1['State'] contains abbreviations like 'CA'
df_filtered = df_filtered.merge(state_df, how='left', left_on='State', right

df_filtered.head()
```

Out[275]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country
0	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States
1	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States
2	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States
4	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States

Saving a Clean Dataset

In [276]:
```
#save the new dataframe in cvs format

df_filtered.to_csv("Aviation_Data_Clean.csv", index=False)
```

In [277]:
```
data = pd.read_csv("Aviation_Data_Clean.csv")
data.head()
```

Out[277]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country
0	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States
1	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States
2	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	United States
4	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States

In [278]:
```
data["Total_Fatal_Injuries"].unique()
```

Out[278]:
```
array([4., 3., 2., 0., 1., 5.])
```

In [279]:
```
data.isnull().sum()
```

Out[279]:
```
Event_Id                   0
Investigation_Type         0
Accident_Number            0
Event_Date                 0
Location                   0
Country                    0
Injury_Severity            0
Aircraft_Damage            0
Aircraft_Category          0
Make                       0
Model                      0
Amateur_Built              0
Number_Of_Engines          0
Engine_Type                0
Purpose_Of_Flight          0
Total_Fatal_Injuries       0
Total_Serious_Injuries     0
Total_Minor_Injuries       0
Total_Uninjured            0
Weather_Condition          0
Broad_Phase_Of_Flight      0
Year                       0
Severity_Score             0
Month                      0
Day                        0
City                       5
State                     50
US_State                1412
Abbreviation            1412
dtype: int64
```

Explaratory Data Analysis

In [280]:
```
data.isnull().sum()
```

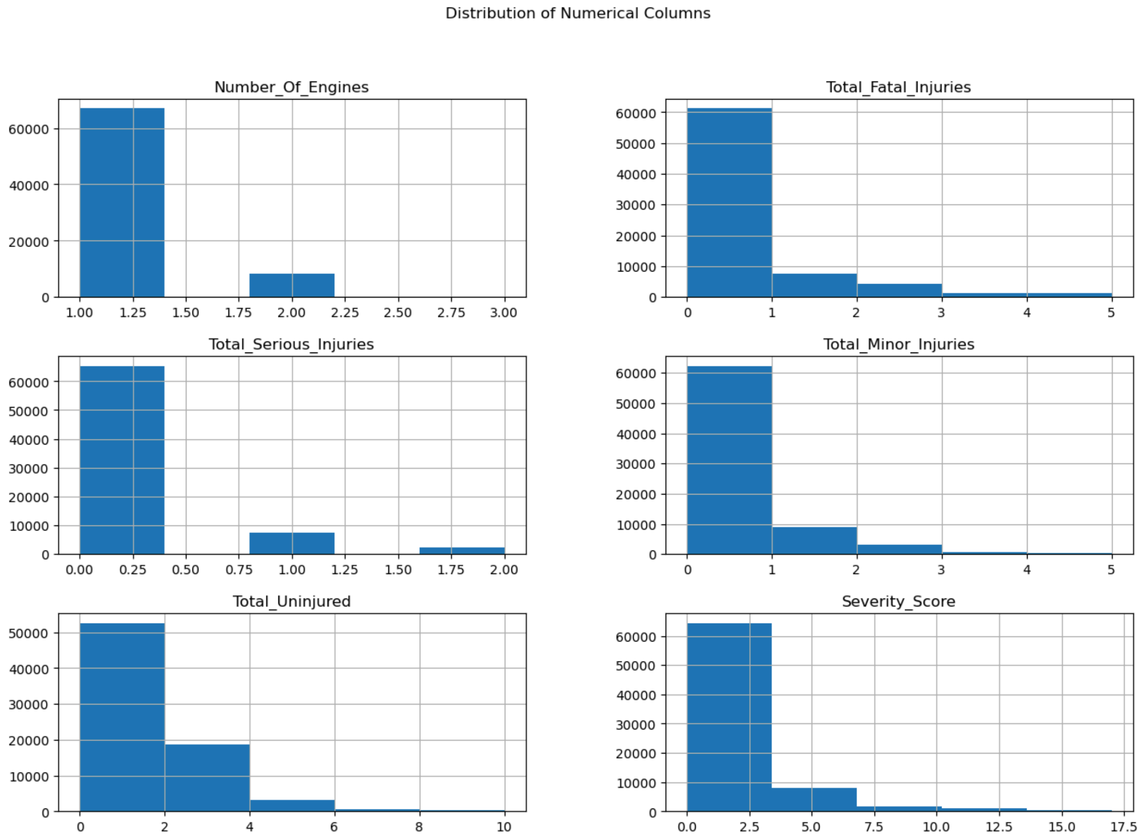
Out[280]:
```
Event_Id                     0
Investigation_Type           0
Accident_Number              0
Event_Date                   0
Location                     0
Country                      0
Injury_Severity              0
Aircraft_Damage              0
Aircraft_Category            0
Make                         0
Model                        0
Amateur_Built                0
Number_Of_Engines            0
Engine_Type                  0
Purpose_Of_Flight            0
Total_Fatal_Injuries         0
Total_Serious_Injuries       0
Total_Minor_Injuries         0
Total_Uninjured              0
Weather_Condition            0
Broad_Phase_Of_Flight        0
Year                         0
Severity_Score               0
Month                        0
Day                          0
City                         5
State                       50
US_State                  1412
Abbreviation              1412
dtype: int64
```

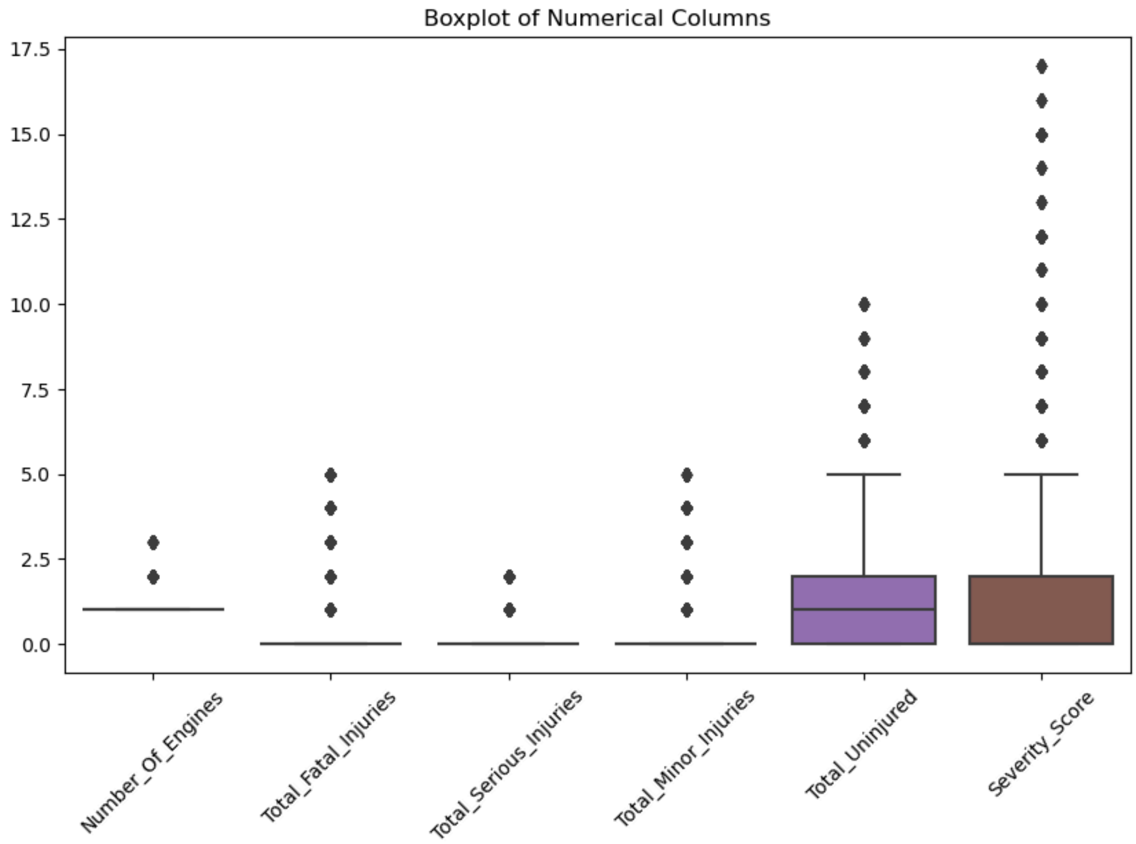
In [281]:
```
#make a copy
data1 = data.copy(deep=True)
```

In [282]:
```
data1.shape
```

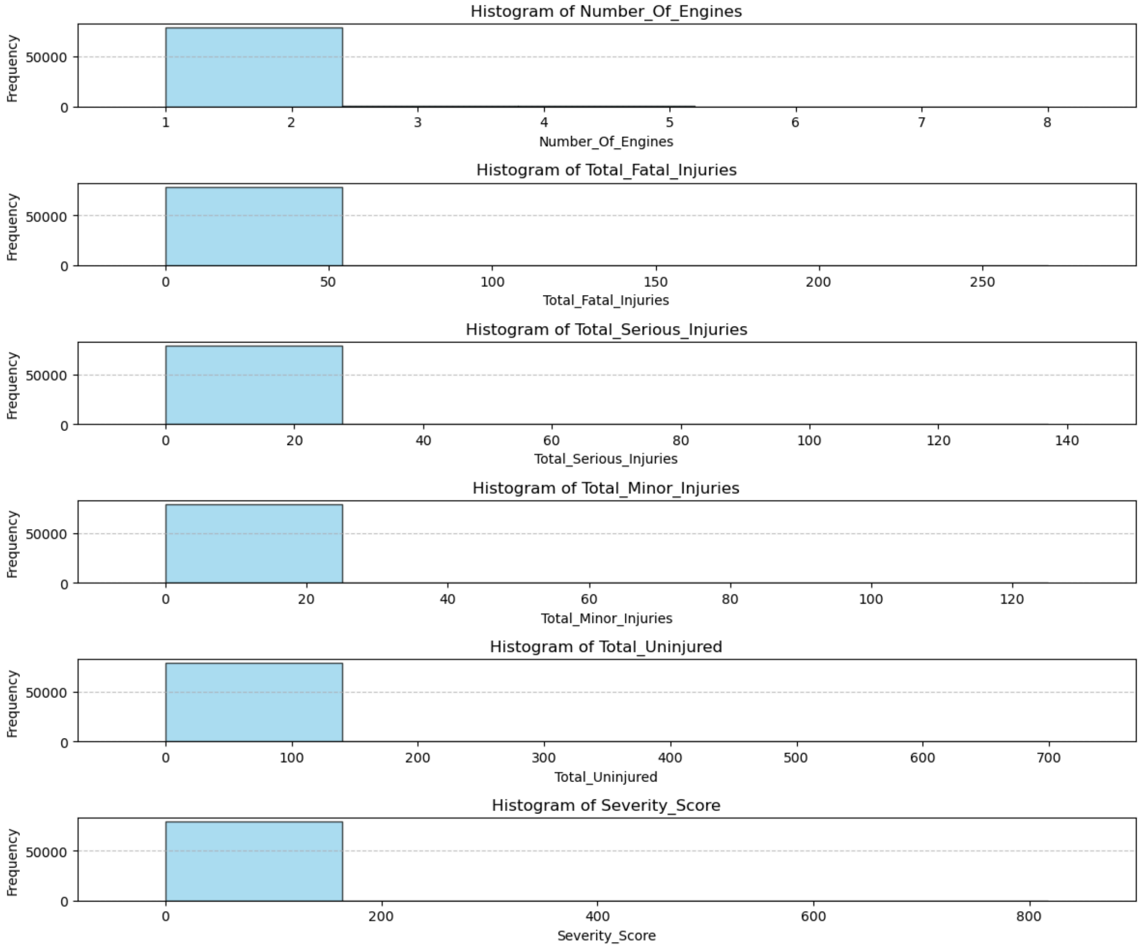
Out[282]:
```
(75393, 29)
```

In [283]:
```python
# Histograms
data1[numerical_cols].hist(bins=5, figsize=(15, 10))
plt.suptitle("Distribution of Numerical Columns")
plt.show()

# Boxplot for outliers
plt.figure(figsize=(10, 6))
sns.boxplot(data=data1[numerical_cols])
plt.xticks(rotation=45)
plt.title("Boxplot of Numerical Columns")
plt.show()
```




Boxplot of Numerical Columns
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Number_Of_Engines
Total_Fatal_Injuries
Total_Serious_Injuries
Total_Minor_Injuries
Total_Uninjured
Severity_Score

In [284]:
```
# Plot histograms for numerical columns with adjusted x-axis limits
fig, axes = plt.subplots(len(numerical_cols), 1, figsize=(12, 10))
for i, column in enumerate(numerical_cols):
    # Plot individual histogram
    axes[i].hist(df2[column], bins=5, color='skyblue', edgecolor='black', al
    axes[i].set_title(f'Histogram of {column}', fontsize=12)
    axes[i].set_xlabel(column, fontsize=10)
    axes[i].set_ylabel('Frequency', fontsize=10)

    # Set x-axis limits based on the data range
    col_min, col_max = df2[column].min(), df2[column].max()
    buffer = (col_max - col_min) * 0.1  # Add a buffer of 10% to the range
    axes[i].set_xlim([col_min - buffer, col_max + buffer])
    axes[i].grid(axis='y', linestyle='--', alpha=0.7)

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



In [285]:
```
data1.columns
```

Out[285]:
```
Index(['Event_Id', 'Investigation_Type', 'Accident_Number', 'Event_Date',
       'Location', 'Country', 'Injury_Severity', 'Aircraft_Damage',
       'Aircraft_Category', 'Make', 'Model', 'Amateur_Built',
       'Number_Of_Engines', 'Engine_Type', 'Purpose_Of_Flight',
       'Total_Fatal_Injuries', 'Total_Serious_Injuries',
       'Total_Minor_Injuries', 'Total_Uninjured', 'Weather_Condition',
       'Broad_Phase_Of_Flight', 'Year', 'Severity_Score', 'Month', 'Day',
       'City', 'State', 'US_State', 'Abbreviation'],
      dtype='object')
```

Univariate Analysis

Purpose: Explore individual variables to understand their distribution and frequency.

Actions:

- Plotted bar charts for categorical columns like `Aircraft_Category` , `Aircraft_Damage` .
- Visualized numerical columns like `Severity_Score` using histograms.
- Analyzed top aircraft makes and models using bar plots.

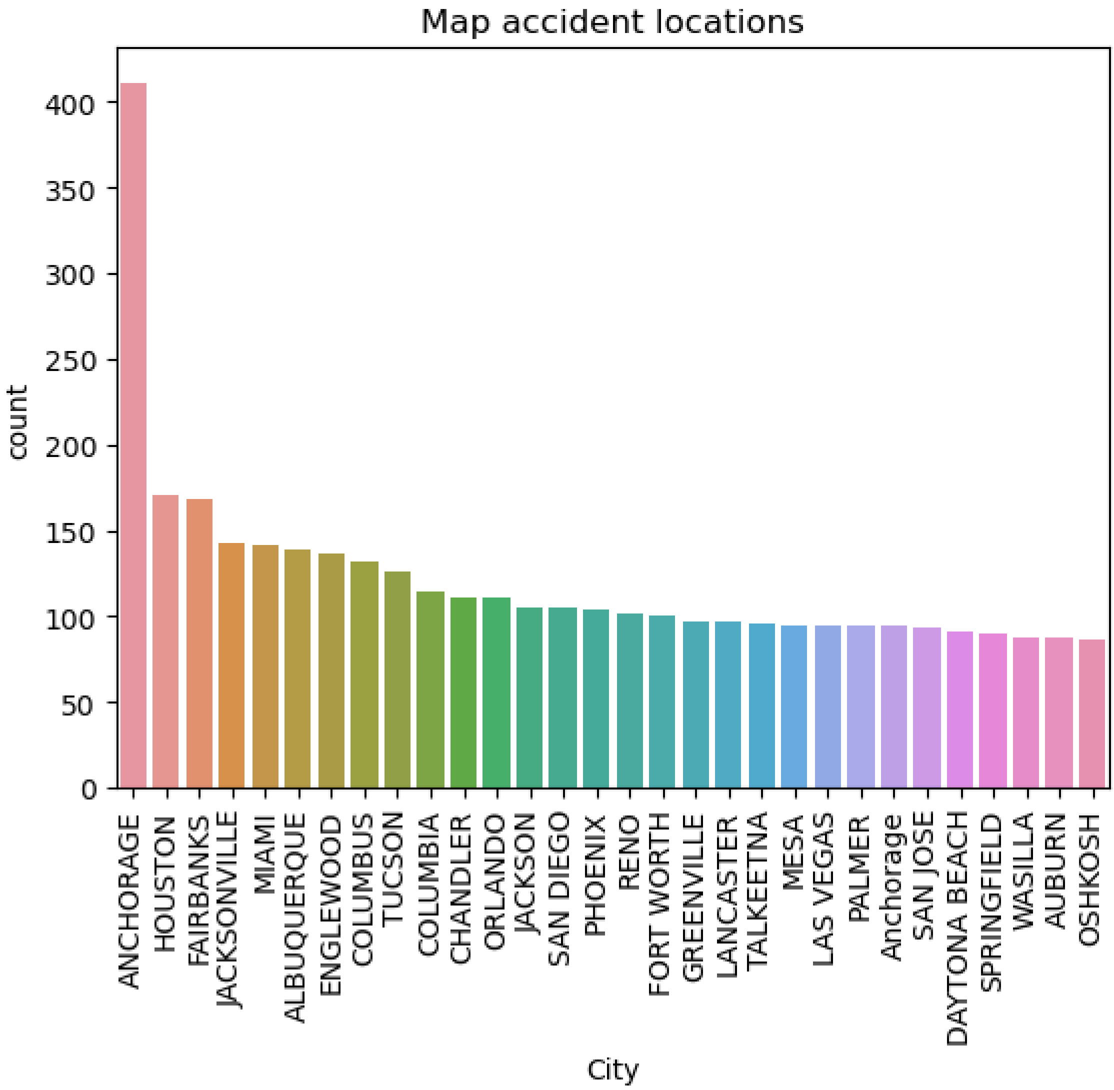
Key Findings:

- Majority of accidents involve single-engine planes.
- Certain makes/models have consistently low severity scores.

Visualizing Accident Locations by City

calculates accident counts for each city in the City column. The bar plot allows for an in-depth view at the city level, highlighting specific urban areas with significant accident occurrences. This analysis provides granular data on accident hotspots, which can be visualized on detailed city maps in to support micro-level planning and decision-making.

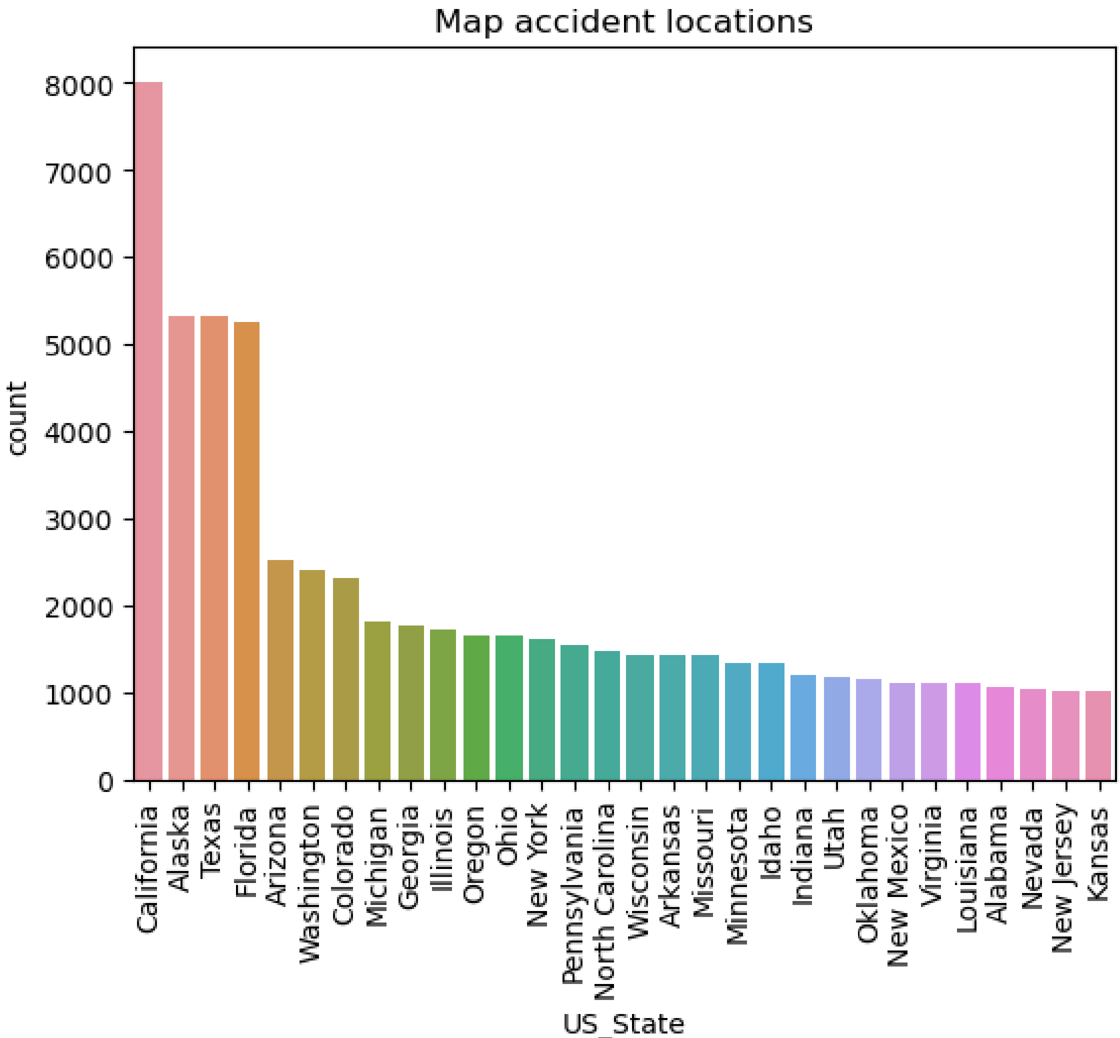
In [286]:
```
damaged_df = data1['City'].value_counts().reset_index()[:30]
sns.barplot(data=damaged_df, y="count", x='City')
plt.title("Map accident locations")
plt.xticks(rotation=90)
plt.show()
```



Visualizing Accident Locations by US State

We identify the top 30 US states with the highest number of accidents. This analysis highlights accident density on a state level, providing a regional overview.

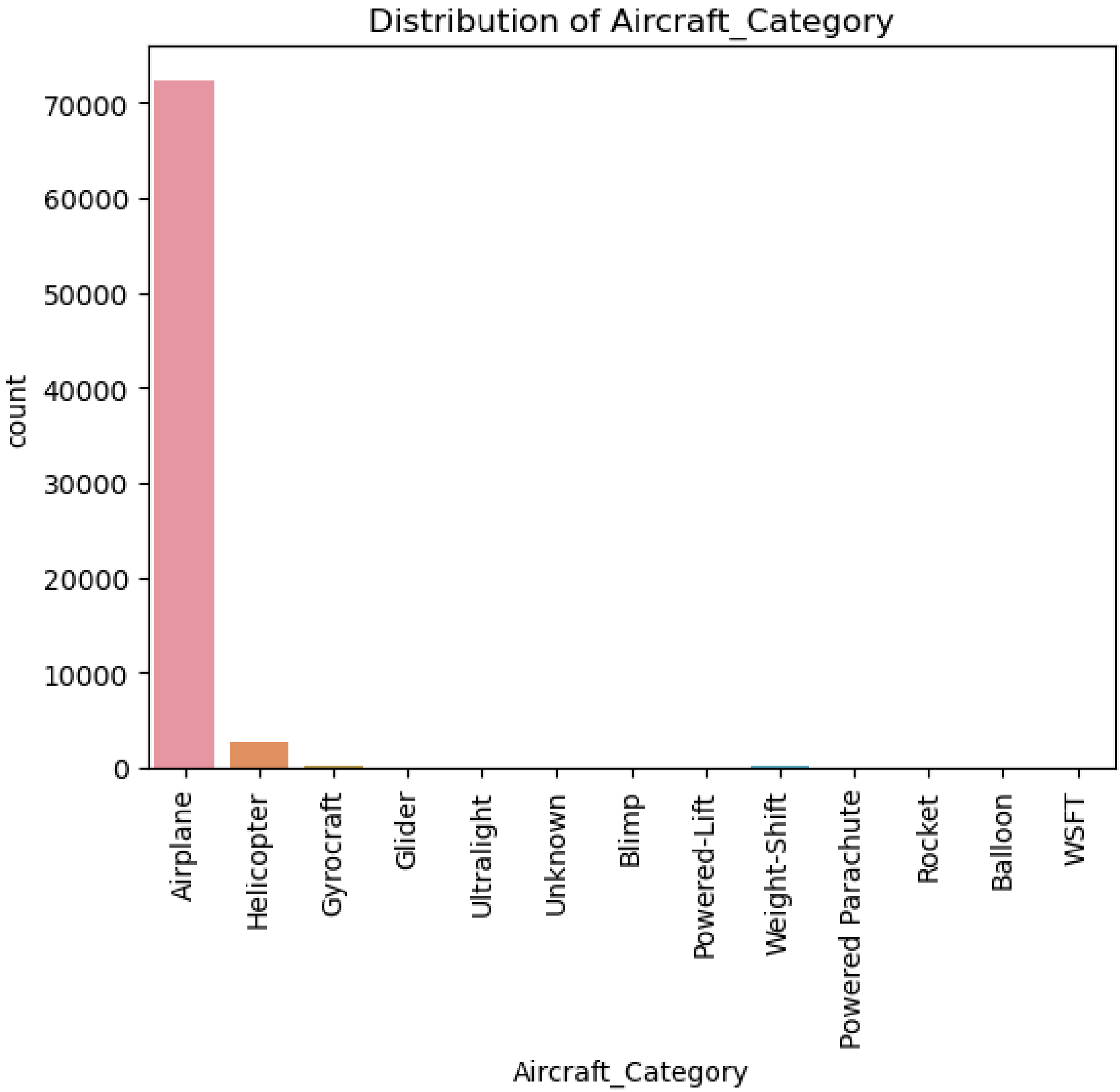
In [287]:
```
damaged_df = data1['US_State'].value_counts().reset_index()[:30]
sns.barplot(data=damaged_df, y="count", x='US_State')
plt.title("Map accident locations")
plt.xticks(rotation=90)
plt.show()
```



Distribution of Aircraft Category

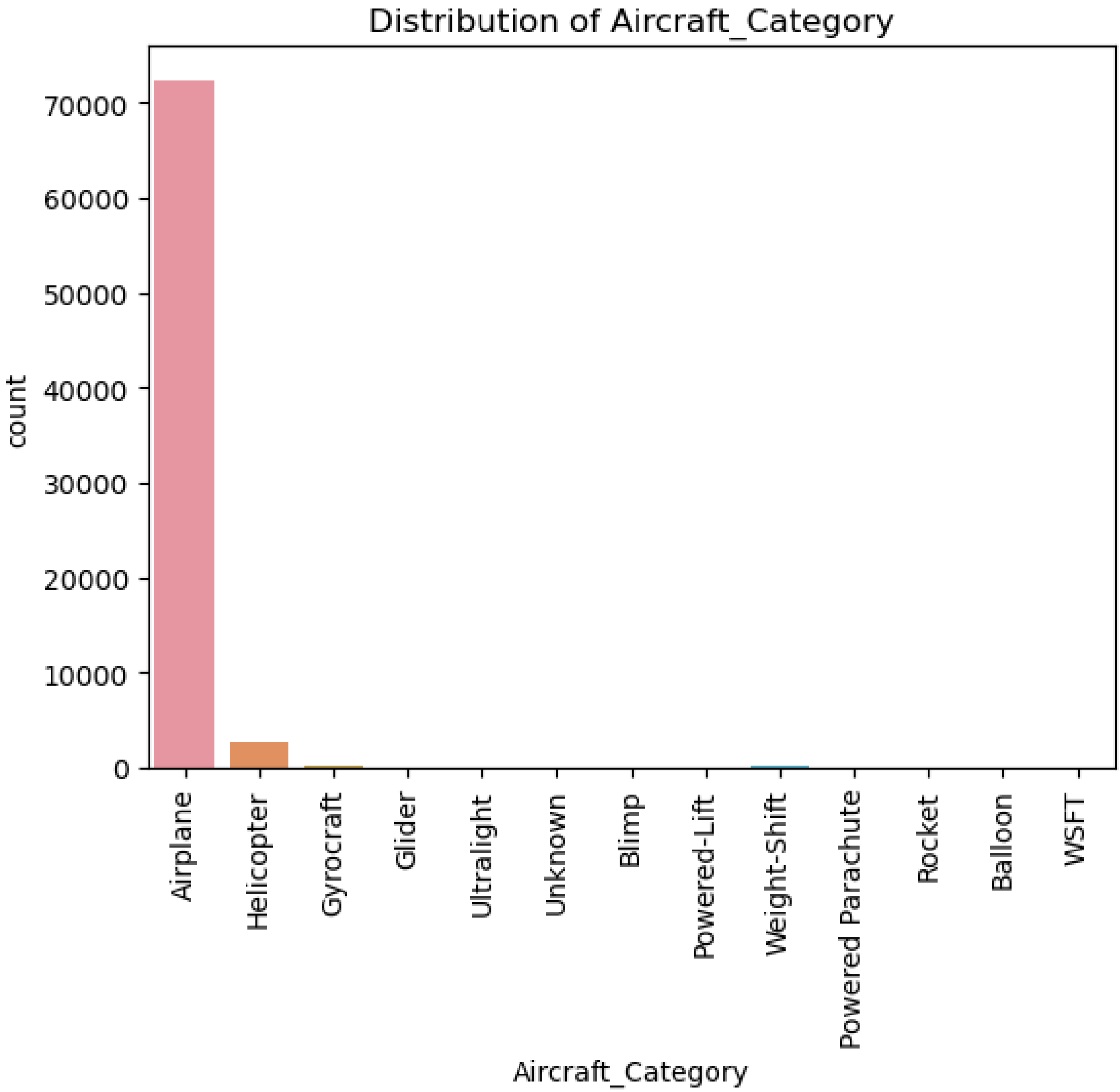
The countplot visualizes the distribution of entries across different categories in the Aircraft_Category column and the next explains with the intensity of the damage caused after an accident. This helps identify the most and least frequent aircraft categories involved in the dataset. Relevance for Tableau: The distribution can be transformed into a bar chart or pie chart to visualize category proportions and support analysis of aircraft types.

In [288]:
```
# Count plot for 'Distribution of Aircraft Category'
sns.countplot(data=data1, x='Aircraft_Category')
plt.title("Distribution of Aircraft_Category")
plt.xticks(rotation=90)
plt.show()
```



The countplot visualizes the distribution of entries across different categories in the Aircraft_Category column This helps identify the most and least frequent aircraft categories involved in the dataset. Relevance for Tableau: The distribution can be transformed into a bar chart or pie chart to visualize category proportions and support analysis of aircraft types.

In [289]:
```
# Count plot for 'Distribution of Aircraft Category'
sns.countplot(data=data1, x='Aircraft_Category')
plt.title("Distribution of Aircraft_Category")
plt.xticks(rotation=90)
plt.show()
```



Univariate Analysis

Purpose: Examine relationships between two variables for meaningful patterns.

Actions:

- Analyzed `Severity_Score` vs. `Total_Fatal_Injuries` using scatterplots:
  - Observed high Severity Scores correlate with higher fatalities.
- Plotted boxplots for `Severity_Score` by `Aircraft_Category` :
  - Helicopters exhibit higher median severity compared to fixed-wing aircraft.
- Explored the impact of `Weather_Condition` on `Severity_Score` :
  - Adverse weather conditions significantly raise severity levels.

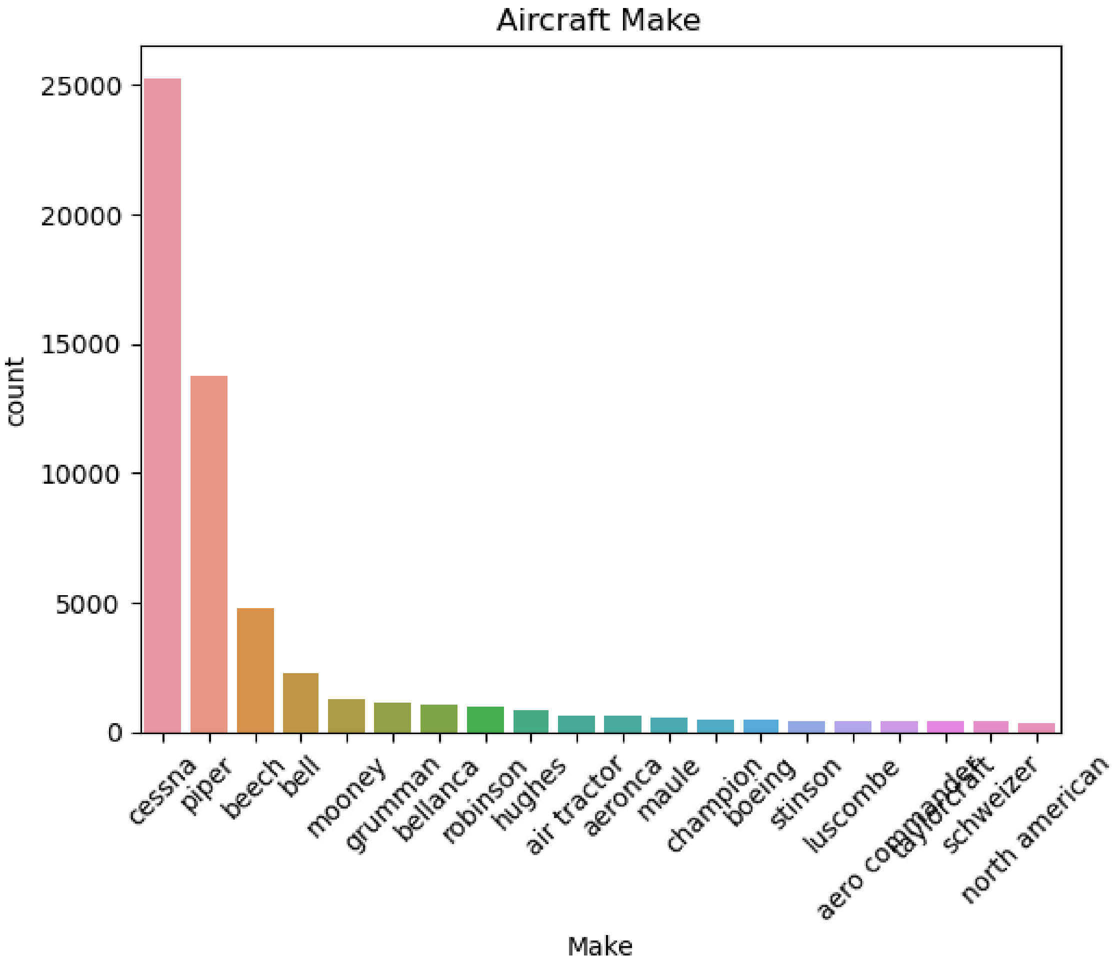
Key Findings:

- Fatalities have a strong positive relationship with Severity Scores.

Analysis of the Distribution of Make

Visualises the most occurring flight in terms of accident by the aircraft make/ manufacturer specifically the top 20 makes based on their frequency

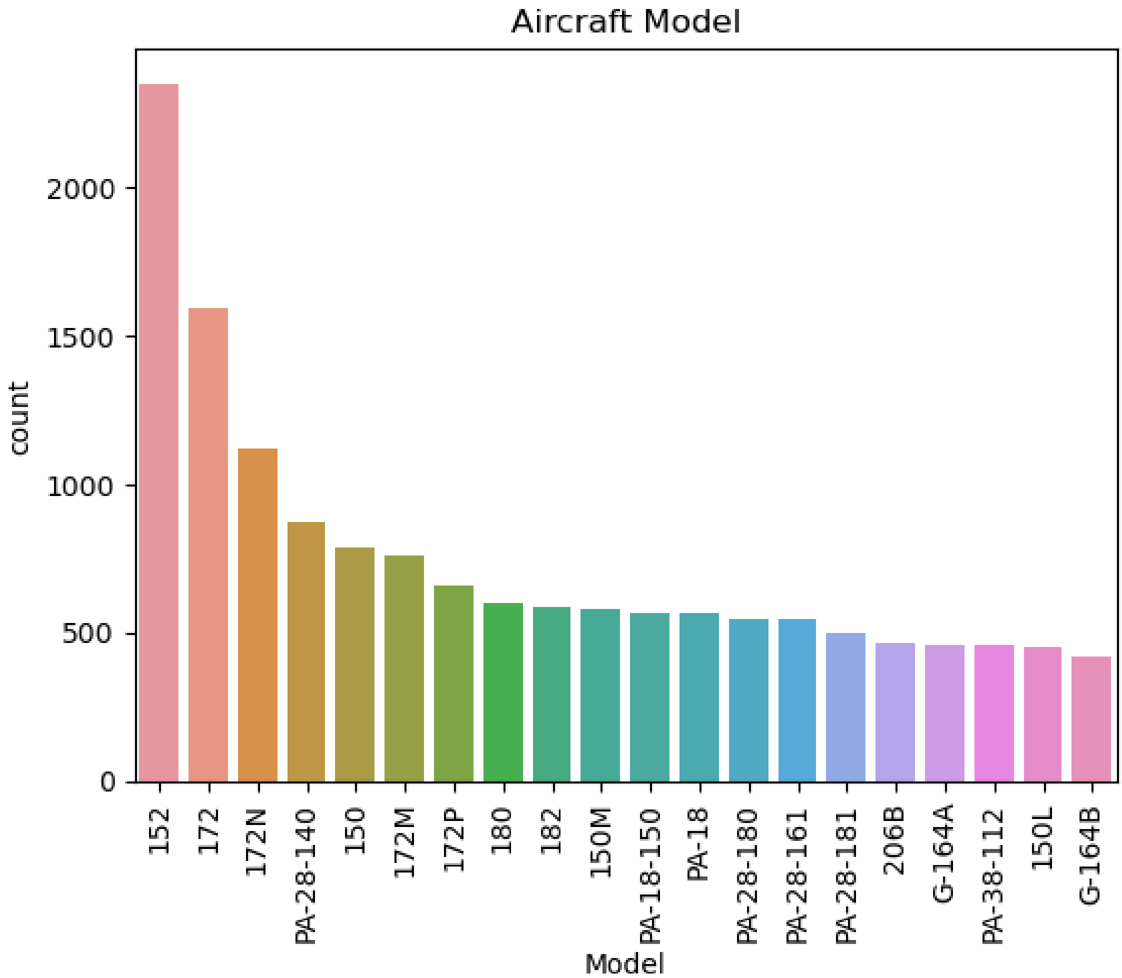
In [291]:
```
# Example for other categorical columns
damaged_df = data1['Make'].value_counts().reset_index()[:20]
sns.barplot(data=damaged_df, y="count", x='Make')
plt.xticks(rotation=45)
plt.title("Aircraft Make")
plt.show()
```



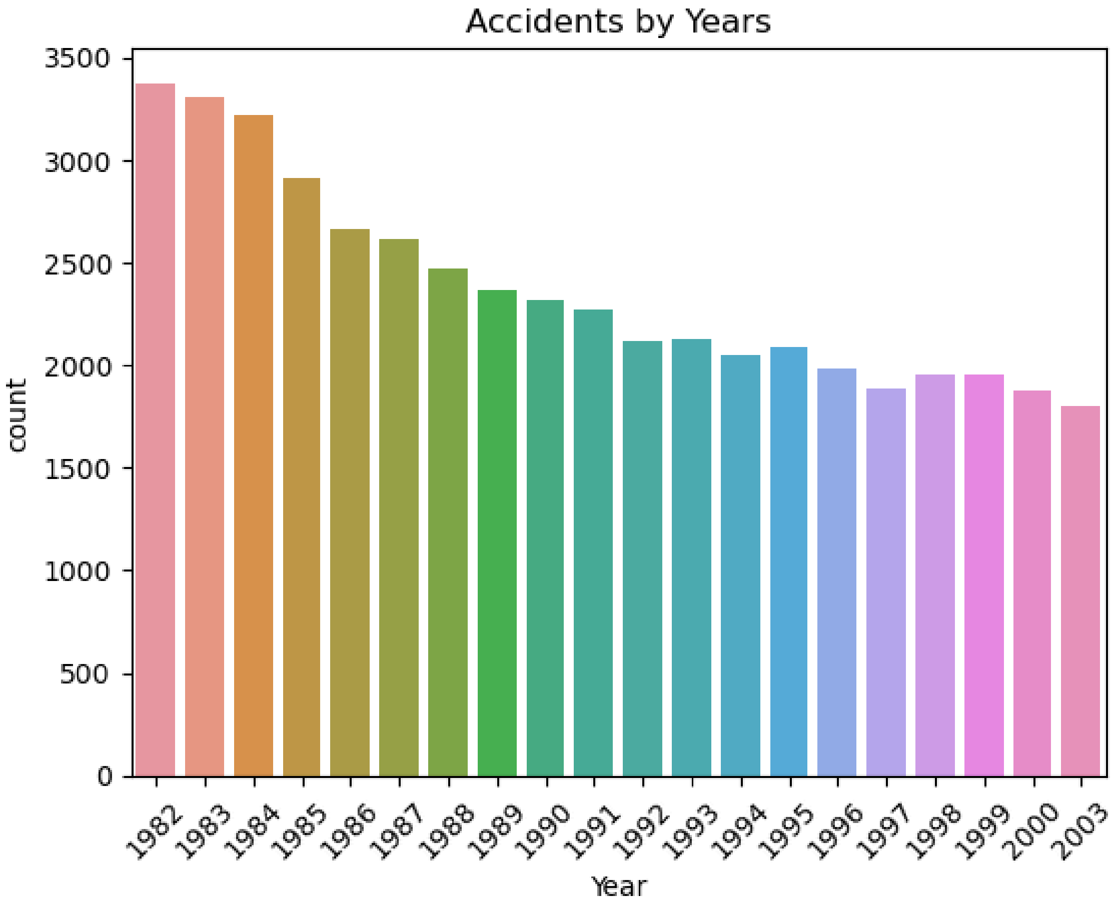
Distribution by Model

Visualises the most occurring flight in terms of accident by the aircraft model;the top 20 makes based on their frequency.

In [292]:
```
model_df = data1['Model'].value_counts().reset_index()[:20]
sns.barplot(data=model_df, y="count", x='Model')
plt.xticks(rotation=90)
plt.title("Aircraft Model")
plt.show()
```



In [293]:
```
# Example for other categorical columns
damaged_df = data1['Year'].value_counts().reset_index()[:20]
sns.barplot(data=damaged_df, y="count", x='Year')
plt.xticks(rotation=45)
plt.title("Accidents by Years")
plt.show()
```



In [295]:
```
damaged_df
```

Out[295]:

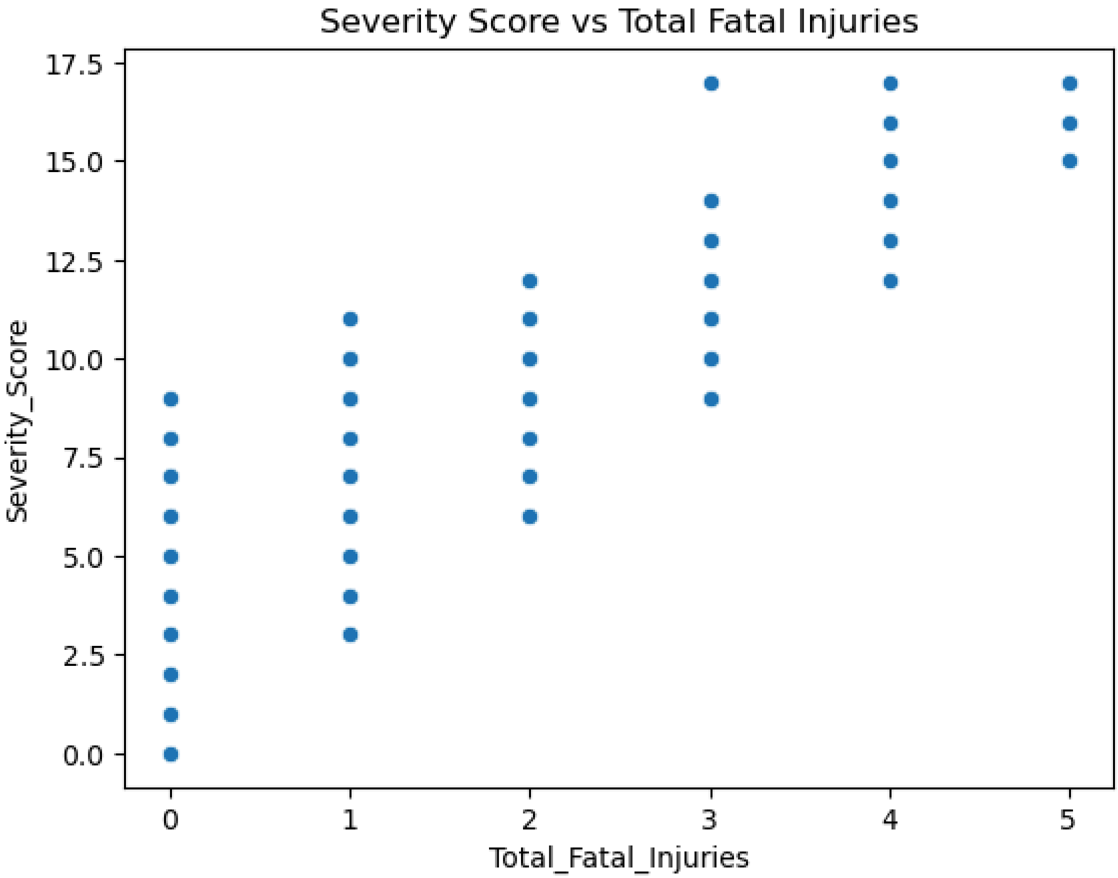
	Year	count
0	1982	3375
1	1983	3304
2	1984	3217
3	1985	2909
4	1986	2666
5	1987	2612
6	1988	2475
7	1989	2369
8	1990	2317
9	1991	2270
10	1993	2125
11	1992	2115
12	1995	2092
13	1994	2053
14	1996	1982
15	1998	1957
16	1999	1954
17	1997	1884
18	2000	1877
19	2003	1800

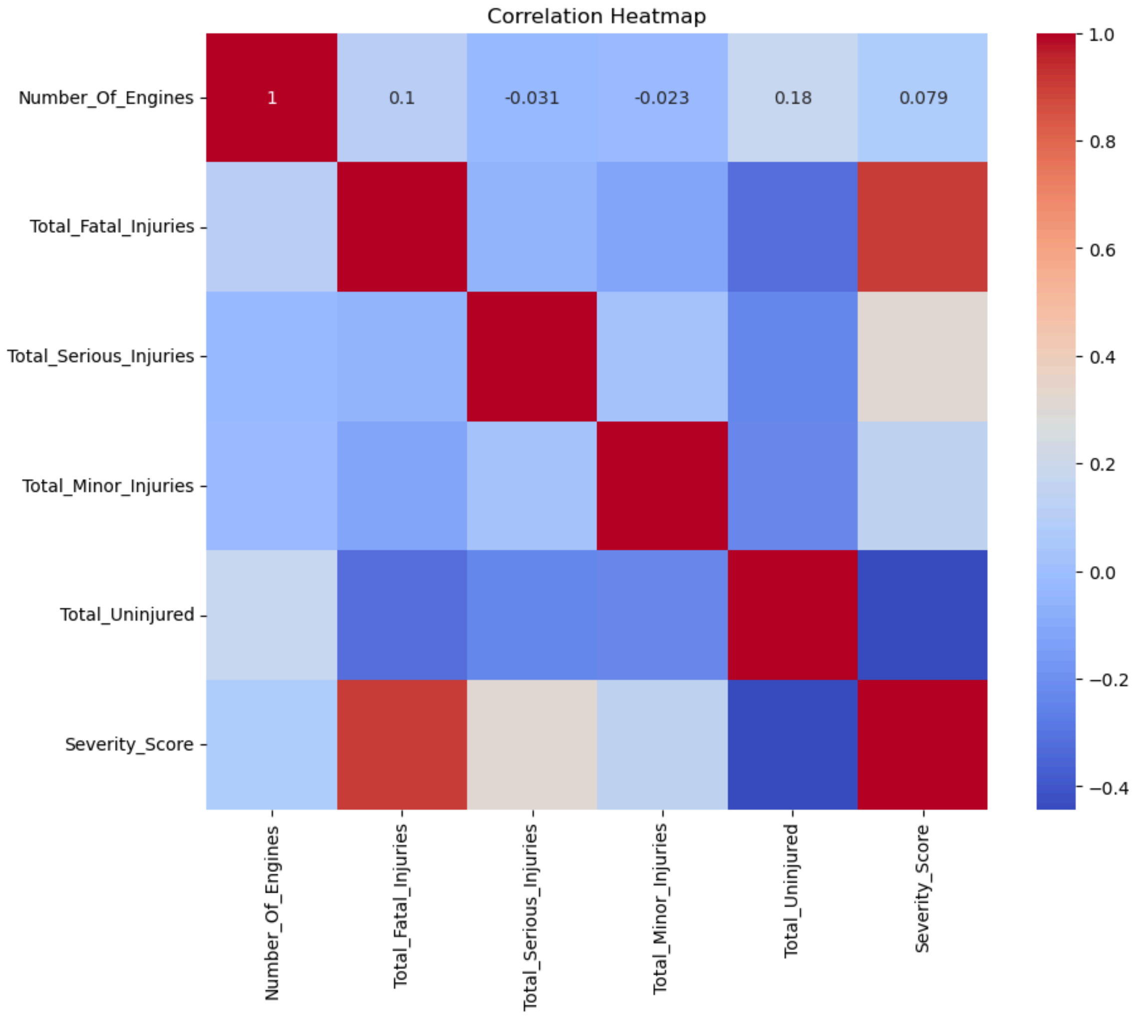
In [296]:
```
data1.head()
```

Out[296]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Country
0	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	Unitec States
1	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Unitec States
2	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	Unitec States
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	Unitec States
4	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	Unitec States

In [297]:
```
# Scatterplot of Severity_Score vs Total_Fatal_Injuries
sns.scatterplot(data=data1, x='Total_Fatal_Injuries', y='Severity_Score')
plt.title("Severity Score vs Total Fatal Injuries")
plt.show()

# Correlation heatmap
corr = data1[numerical_cols].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```





Multivariate Analysis

Purpose: Analyze complex interactions across multiple variables.

Actions:

- Line graph for yearly accident trends by `Make` :
  - Revealed specific makes with consistently high accident counts.
- Heatmap for correlation between numerical fields:
  - High correlation between `Total_Fatal_Injuries` and `Severity_Score` .
- Grouped data by `Purpose_Of_Flight` , `Make` , and `Year` to assess accident risks:
  - Training flights have higher severity scores than commercial operations.

Key Findings:

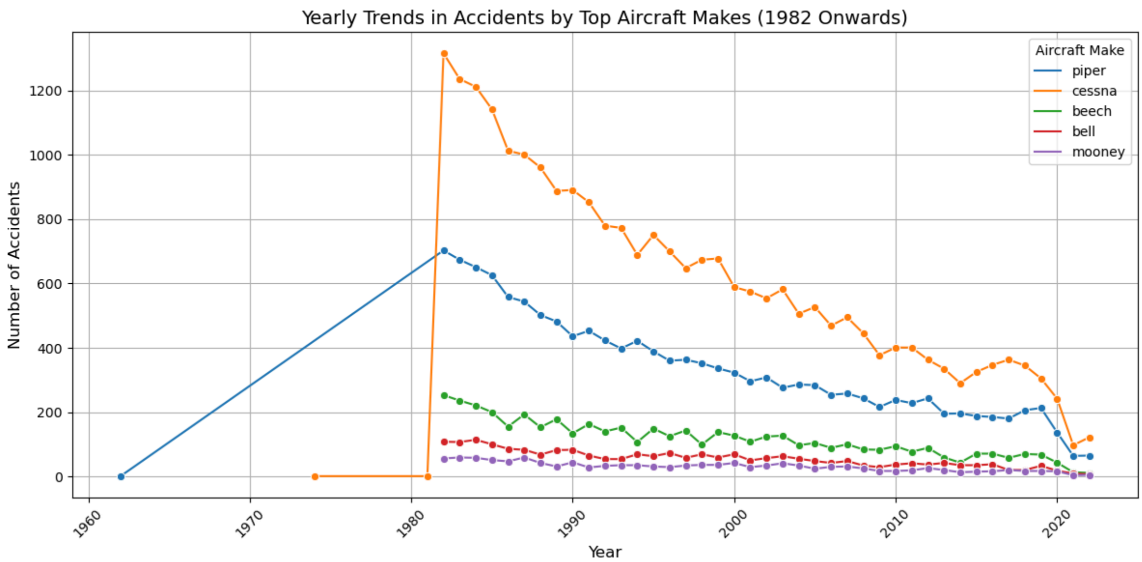
- Certain aircraft makes and flight purposes are high-risk.
- Trends highlight improvement in safety measures post-2010.

Analysing Year Trends of Number of Accidents by Aircraft Make

Focuses on identifiying the trend of accidents from 1962 onwards and aggregates the accidents occurence by year and aircraft make filtering to the top 5 aircraft make

In [299]:
```
filtered_data = data1[data1['Year'] >= 1962]

# Recalculate the yearly trends for accidents by aircraft make
yearly_trends = filtered_data.groupby(['Year', 'Make']).size().reset_index(r

# Identify top 5 aircraft makes with the highest overall accident count
top_makes = yearly_trends.groupby('Make')['Accident_Count'].sum().nlargest(5

# Filter the trends data for only the top makes
top_makes_trends = yearly_trends[yearly_trends['Make'].isin(top_makes)]

# Plotting the trends
plt.figure(figsize=(12, 6))
sns.lineplot(data=top_makes_trends, x='Year', y='Accident_Count', hue='Make

plt.title("Yearly Trends in Accidents by Top Aircraft Makes (1982 Onwards)",
plt.xlabel("Year", fontsize=12)
plt.ylabel("Number of Accidents", fontsize=12)
plt.xticks(rotation=45)
plt.legend(title="Aircraft Make")
plt.grid(True)
plt.tight_layout()
plt.show()
```

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



In [300]:
```
top_makes_trends
```

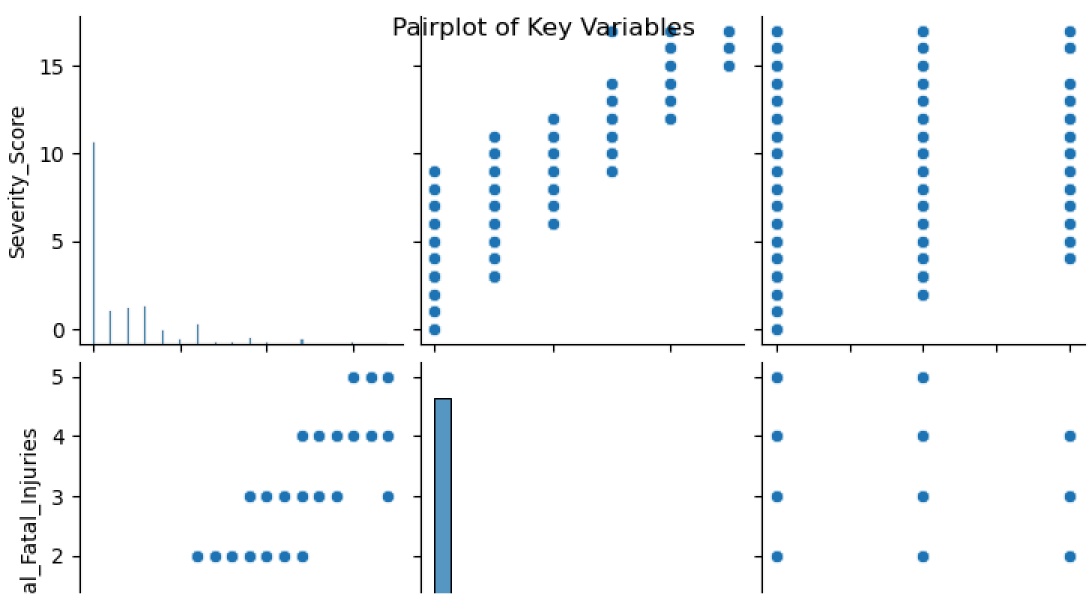
Out[300]:

	Year	Make	Accident_Count
0	1962	piper	1
1	1974	cessna	1
3	1981	cessna	1
27	1982	beech	253
29	1982	bell	108
...	...	...	...
12770	2022	beech	11
12772	2022	bell	7
12780	2022	cessna	122
12840	2022	mooney	5
12849	2022	piper	65

208 rows × 3 columns

In [301]:
```
sns.pairplot(data1, vars=['Severity_Score', 'Total_Fatal_Injuries', 'Total_S
plt.suptitle("Pairplot of Key Variables")
plt.show()
```

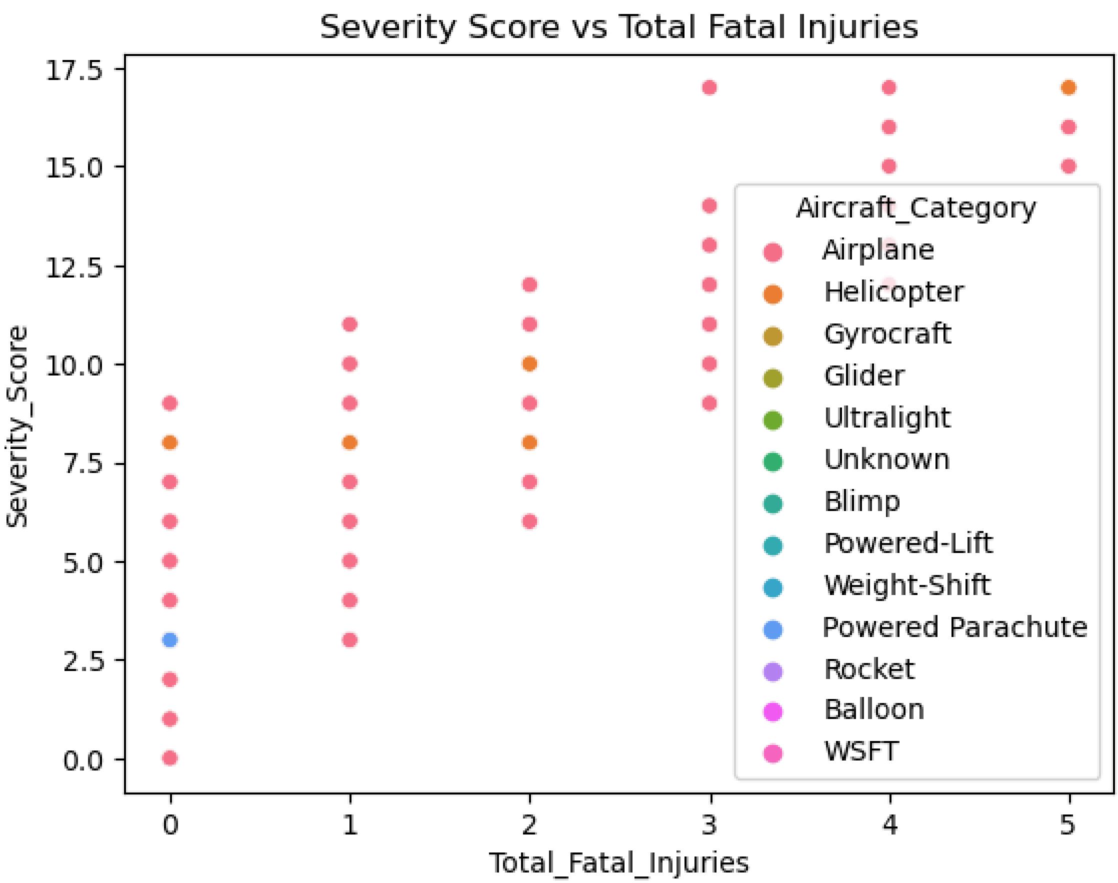
```
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



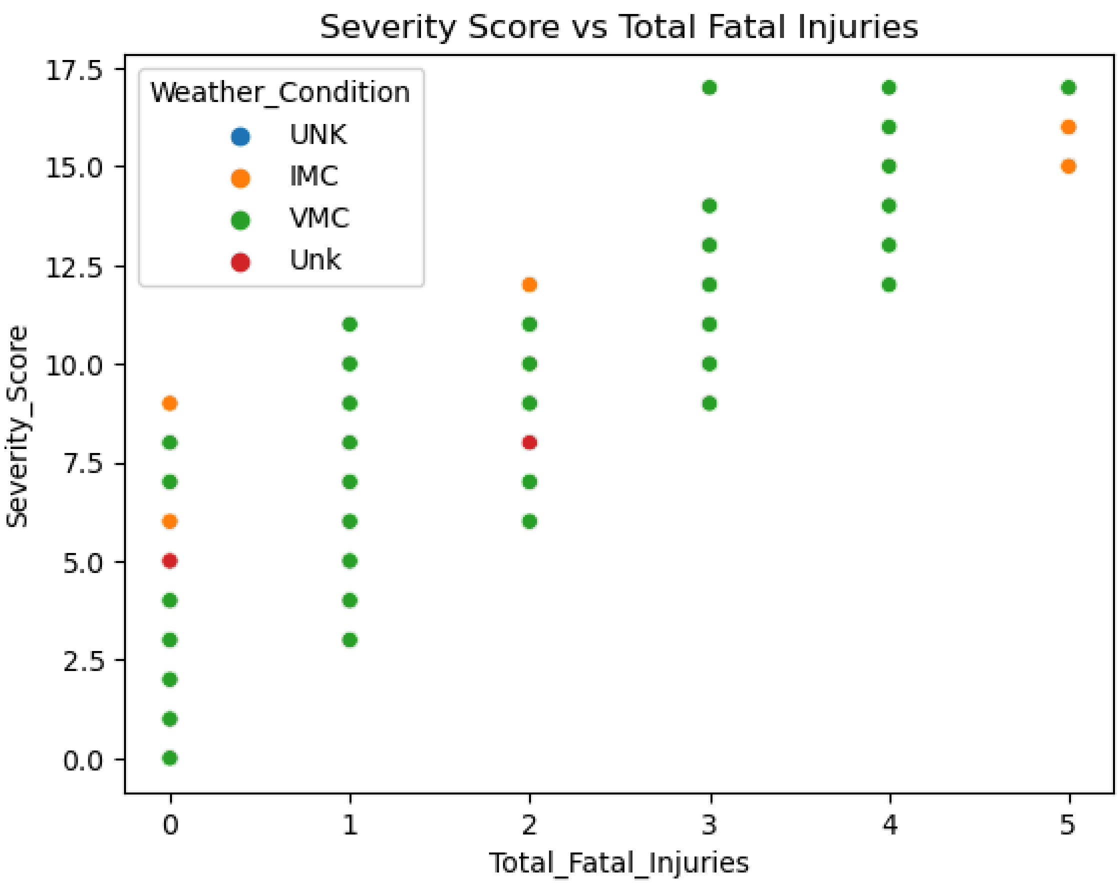
Analysing the Relationship Between Total Fatal Injuries and Severity Score

Analysing the Relationship Between Total Fatal Injuries and Severity Score differentiating by Aircraft category by color

In [302]:
```
sns.scatterplot(data=data1, x='Total_Fatal_Injuries', y='Severity_Score', hu
plt.title("Severity Score vs Total Fatal Injuries")
plt.show()
```



In [303]:
```
sns.scatterplot(data=data1, x='Total_Fatal_Injuries', y='Severity_Score', hu
plt.title("Severity Score vs Total Fatal Injuries")
plt.show()
```



In [304]:
```
damaged_df
```

Out[304]:

	Year	count
0	1982	3375
1	1983	3304
2	1984	3217
3	1985	2909
4	1986	2666
5	1987	2612
6	1988	2475
7	1989	2369
8	1990	2317
9	1991	2270
10	1993	2125
11	1992	2115
12	1995	2092
13	1994	2053
14	1996	1982
15	1998	1957
16	1999	1954
17	1997	1884
18	2000	1877
19	2003	1800

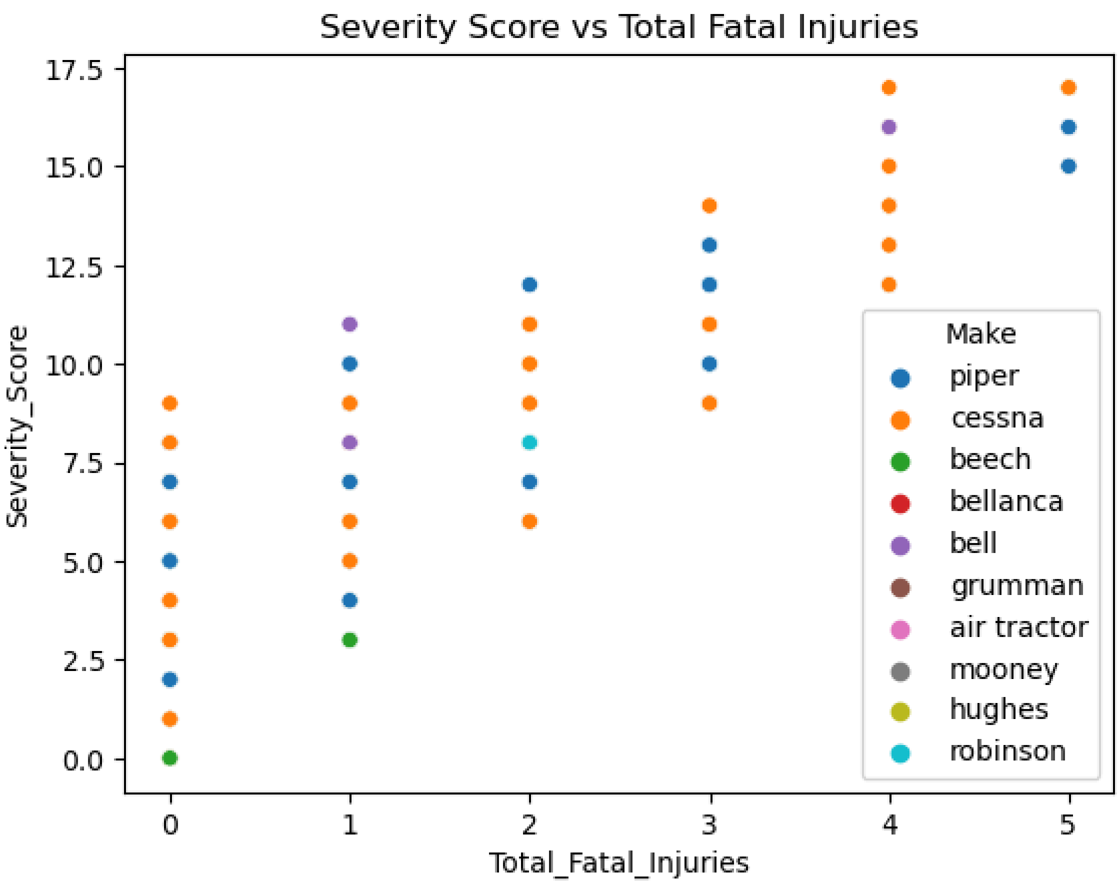
In [305]:
```
damaged_df = data1['Make'].value_counts().reset_index()[:10]
damaged_makes = damaged_df['Make']  # Extract the "Make" column names

# Filter rows in 'data1' where "Make" is in the top 10 most frequent makes
Make_df = data1[data1["Make"].isin(damaged_makes)]

# Display the filtered DataFrame
Make_df
```

Out[305]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Co
0	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	U S
1	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	U S
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	U S
4	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	U S
5	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	U S
...	...	...	...	...	...	
75384	20221031106225	Accident	CEN22LA441	2022-09-29	Shell Lake, WI	U S
75386	20221011106095	Accident	CEN23LA011	2022-10-05	Navasota, TX	U S
75388	20221011106092	Accident	CEN23LA008	2022-10-06	Iola, TX	U S
75389	20221011106098	Accident	ERA23LA014	2022-10-08	Dacula, GA	U S
75390	20221018106153	Accident	CEN23LA015	2022-10-13	Ardmore, OK	U S

51877 rows × 29 columns

In [306]:
```
sns.scatterplot(data=Make_df, x='Total_Fatal_Injuries', y='Severity_Score',
plt.title("Severity Score vs Total Fatal Injuries")
plt.show()
```



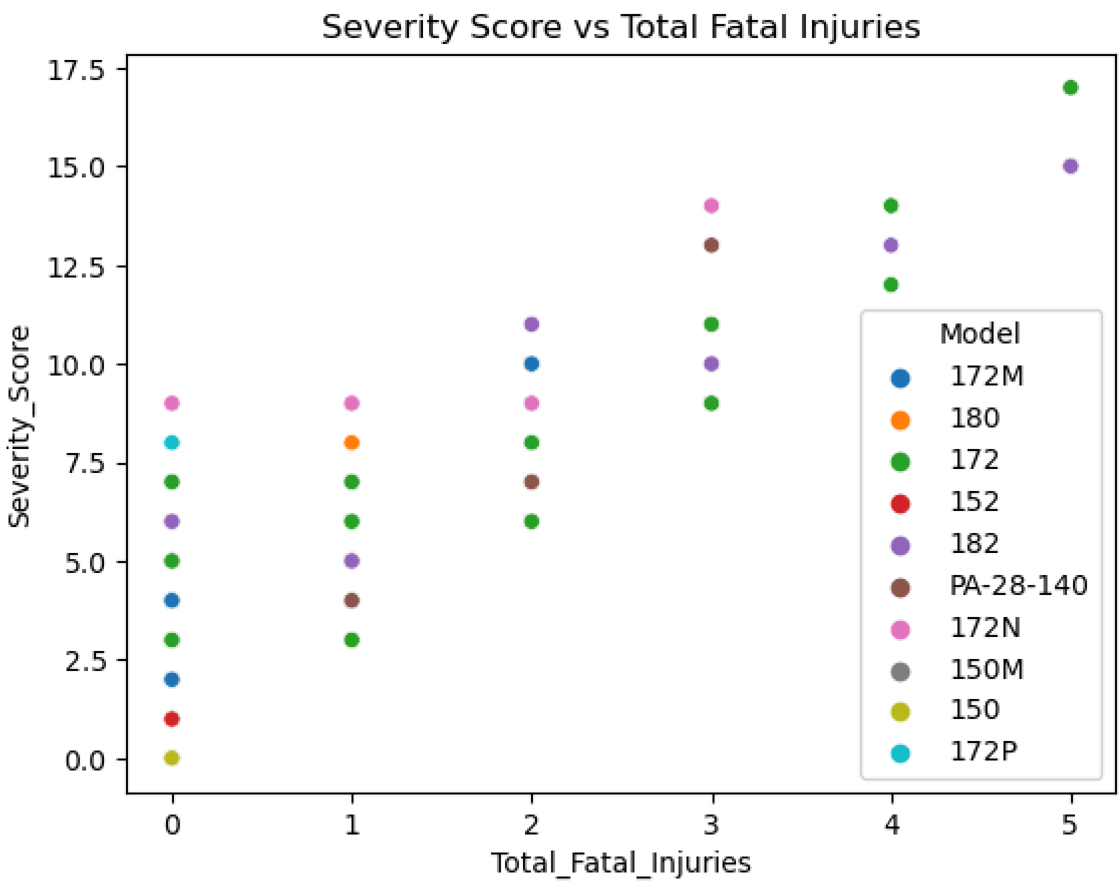
In [307]:
```
damaged_df = Make_df['Model'].value_counts().reset_index()[:10]
damaged_Models = damaged_df['Model']  # Extract the "Model" column names

# Filter rows in 'data1' where "Model" is in the top 10 most frequent Models
Model_df = Make_df[Make_df["Model"].isin(damaged_Models)]

# Display the filtered DataFrame
Model_df
```

Out[307]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Count
1	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Unit Stat
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	Unit Stat
14	20020917X01656	Accident	ANC82FAG14	1982-01-02	SKWENTA, AK	Unit Stat
15	20020917X02481	Accident	NYC82DA016	1982-01-02	GALETON, PA	Unit Stat
17	20020917X01894	Accident	CHI82FEC08	1982-01-02	YPSILANTI, MI	Unit Stat
...	...	...	...	...	...	
75342	20220831105837	Accident	WPR22LA326	2022-08-23	Egegik, AK	Unit Stat
75368	20220912105913	Accident	WPR22LA342	2022-09-07	Salt Lake City, UT	Unit Stat
75373	20220912105911	Accident	CEN22LA414	2022-09-12	Fort Collins, CO	Unit Stat
75383	20221006106076	Accident	CEN22LA439	2022-09-26	Williston, ND	Unit Stat
75388	20221011106092	Accident	CEN23LA008	2022-10-06	Iola, TX	Unit Stat

9869 rows × 29 columns

In [308]:
```
sns.scatterplot(data=Model_df, x='Total_Fatal_Injuries', y='Severity_Score',
plt.title("Severity Score vs Total Fatal Injuries")
plt.show()
```



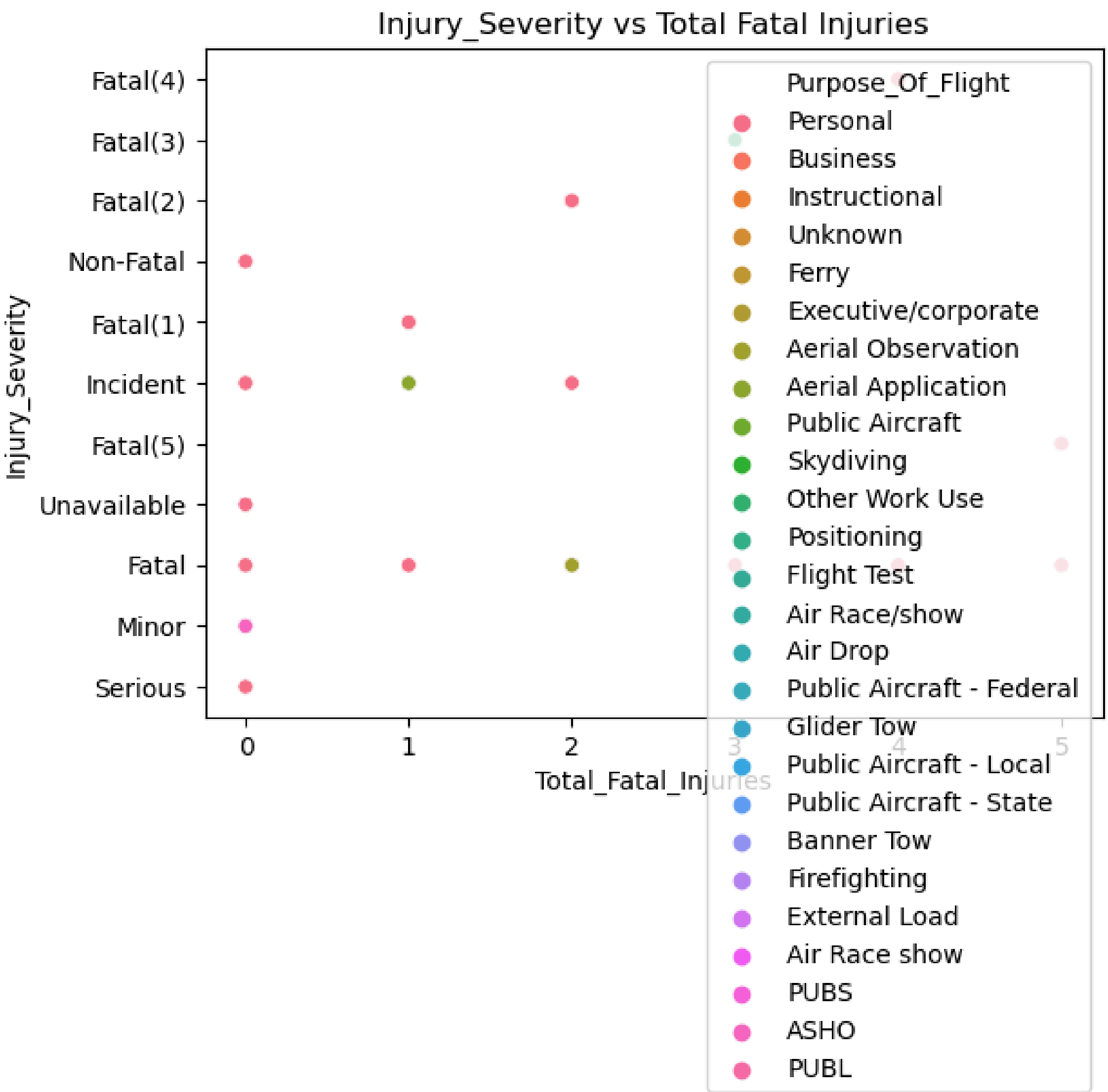
In [309]:
```
damaged_df = data1['Model'].value_counts().reset_index()[:10]
damaged_Models = damaged_df['Model']  # Extract the "Model" column names

# Filter rows in 'data1' where "Model" is in the top 10 most frequent Models
Model_df = data1[data1["Model"].isin(damaged_Models)]

# Display the filtered DataFrame
Model_df
```

Out[309]:

	Event_Id	Investigation_Type	Accident_Number	Event_Date	Location	Count
1	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	Unit Stat
3	20001218X45446	Accident	CHI81LA106	1981-08-01	COTTON, MN	Unit Stat
14	20020917X01656	Accident	ANC82FAG14	1982-01-02	SKWENTA, AK	Unit Stat
15	20020917X02481	Accident	NYC82DA016	1982-01-02	GALETON, PA	Unit Stat
17	20020917X01894	Accident	CHI82FEC08	1982-01-02	YPSILANTI, MI	Unit Stat
...	...	...	...	...	...	
75342	20220831105837	Accident	WPR22LA326	2022-08-23	Egegik, AK	Unit Stat
75368	20220912105913	Accident	WPR22LA342	2022-09-07	Salt Lake City, UT	Unit Stat
75373	20220912105911	Accident	CEN22LA414	2022-09-12	Fort Collins, CO	Unit Stat
75383	20221006106076	Accident	CEN22LA439	2022-09-26	Williston, ND	Unit Stat
75388	20221011106092	Accident	CEN23LA008	2022-10-06	Iola, TX	Unit Stat

9899 rows × 29 columns

In [310]:
```
sns.scatterplot(data=data1, x='Total_Fatal_Injuries', y='Injury_Severity', h
plt.title("Injury_Severity vs Total Fatal Injuries")
plt.show()
```



In [311]:
```
sns.scatterplot(data=Make_df, x='Purpose_Of_Flight', y='Severity_Score', hue
plt.title("Severity Score vs Total Fatal Injuries")
plt.xticks(rotation=90)
plt.show()
```