1. Introduction

This project aims to evaluate the safety risks associated with purchasing and operating different types of aircraft for commercial and private enterprises. The company is expanding into the aviation industry and needs data-driven insights to guide the decision-making process on which aircraft to acquire, with a focus on minimizing potential risks.

The dataset used in this analysis contains aviation accident records from the National Transportation Safety Board (NTSB), spanning from 1962 to 2023. This rich dataset includes information on various aircraft types, accident severity, causes, locations, and other contributing factors. The primary goal is to assess the historical performance of different aircraft models by examining accident frequency, severity, and the underlying risk factors associated with each type of aircraft.

1.1 General objective

To identify the safest aircraft that the company can purchase through analyzing aviation accident data and provide actionable insights for good decision making.

1.1.1 Specific objectives

1. To evaluate the aviation accident data with the goal of identifying the aircraft with the highest safety records and lowest risk.

Visualization: Bar Chart / Horizontal Bar Chart to compare accident frequencies for different aircraft types.

1. To analyze the data to understand factors contributing to accident frequency and severity.

Heatmap to identify correlations between different risk factors.

1. To use Geospatial Map to visualize accident distribution and risk hotspots the US and relationship between them and specific aircraft.

The primary audience for this analysis is the Head of the Aviation Division, who needs actionable insights to make informed purchasing decisions about aircraft models for the company's new venture into aviation.

2. Data Understanding

In [2]:

```
#import python libraries

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

loading data set

Using Pandas to load data set

In [3]:

```
df = pd.read_csv("C:/Users/Fluxtech/Desktop/Moringa projects/AviationData.csv", encoding=
'ISO-8859-1')
df.head()
```

```
c:\Users\Fluxtech\anaconda3\envs\learn-env\lib\site-packages\IPython\core\interactiveshel
l.py:3145: DtypeWarning: Columns (6,7,28) have mixed types.Specify dtype option on import
or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Coc
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	Na
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	Na
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9222	-81.8781	Na
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	Na
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	Na

5 rows × 31 columns

In [4]:

```
df.shape
```

Out[4]:

```
(88889, 31)
```

In [5]:

```
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
```

In [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
df.describe(include = 'O')
```

Out[7]:

	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	25592	27156	

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airpo
top	20001214X45071	Accident	ERA22FA338	1984-06-30	ANCHORAGE, AK	United States	332739N	0112457W	
freq	3	85015	2	25	434	82248	19	24	

4 rows × 26 columns

In [8]:

```
#make a copy

df2 = df.copy(deep = True)
```

In [9]:

```
#check for unique values all at once

for column in df2:
 unique_values = df2[column].unique()
 print(f"unique values in column {column}, '\n': {unique_values}", '\n')
```

```
unique values in column Event.Id, '
': ['20001218X45444' '20001218X45447' '20061025X01555' ... '20221227106497'
 '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, AZ'
 '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 36.922222999999995 42.445277000000004 ... '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' 'Ultralight'
 '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' 'None' '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 during taxi.'
 'The pilot\x92s failure to secure the magneto switch before attempting to hand rotate th
e engine which resulted in an inadvertent engine start, a runaway airplane, and subsequen
t impact with parked airplanes. Contributing to the accident was the failure to properly
secure the airplane with chocks.'
 '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' '29-12-2022']
```

3. Data wrangling

3.1. Checking columns and changing mispelled to correct name

In [10]:

```
#check columns and deal with mispelled columns

df2.columns
```

Out[10]:

```
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 [11]:

```
#for uniformity, change column name to lower case

df2.columns = df2.columns.str.lower()
df2.columns
```

Out[11]:

```
Index(['event.id', 'investigation.type', 'accident.number', 'event.date',
       'location', 'country', 'latitude', 'longitude', 'airport.code',
       'airport.name', 'injury.severity', 'aircraft.damage',
       '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 [12]:

```
#remove whitespaces if any

df2.columns = df2.columns.str.replace(" ", "")
```

In [13]:

```
#drop unncessary columns

df2.drop(['event.id', 'accident.number','airport.code', 'publication.date'], axis = 1, i
nplace = True)
```

In [14]:

```
df2.columns
```

Out[14]:

```
Index(['investigation.type', 'event.date', 'location', 'country', 'latitude',
       'longitude', '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'],
      dtype='object')
```

In [15]:

```
# Replace fullstop with lowerscore for the who data
df2.columns = df2.columns.str.replace(".", "_")
```

3.2. Checking missing values

In [16]:

```
#check the missing values and deal with them

df2.isna().sum()
```

Out[16]:

```
investigation_type            0
event_date                    0
location                     52
country                     226
latitude                  54507
longitude                 54516
airport_name              36099
injury_severity            1000
aircraft_damage            3194
aircraft_category         56602
registration_number        1317
make                         63
model                        92
amateur_built               102
number_of_engines          6084
engine_type                7077
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              6381
dtype: int64
```

In [17]:

```
#for make, mode and injury severity, we use mode to fill missings

make_mode = df2["make"].mode()[0]
df2['make'] = df2['make'].fillna(make_mode)
```

In [18]:

```
#for make, mode and injury severity, we use mode to fill missings

model_mode = df2["model"].mode()[0]
df2['model'] = df2['model'].fillna(model_mode)
```

In [19]:

```
#for make, mode and injury severity, we use mode to fill missings
injury_severity_mode = df2["injury_severity"].mode()[0]
df2['injury_severity'] = df2['injury_severity'].fillna(injury_severity_mode)
```

In [20]:

```
#check the missing values and deal with them

df2.isna().sum()
```

Out[20]:

```
investigation_type            0
event_date                    0
location                     52
country                     226
latitude                  54507
longitude                 54516
airport_name              36099
injury_severity               0
aircraft_damage            3194
aircraft_category         56602
registration_number        1317
make                          0
model                         0
amateur_built               102
number_of_engines          6084
engine_type                7077
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              6381
dtype: int64
```

In [21]:

```
columns_to_drop = ['latitude', 'longitude', 'schedule', 'far_description', 'air_carrier'
, 'aircraft_category']
```

```
columns_to_drop

df2 = df2.drop(columns=columns_to_drop)
```

In [22]:

```
df2.isna().sum()
```

Out[22]:

```
investigation_type            0
event_date                    0
location                     52
country                     226
airport_name              36099
injury_severity               0
aircraft_damage            3194
registration_number        1317
make                          0
model                         0
amateur_built               102
number_of_engines          6084
engine_type                7077
purpose_of_flight          6192
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              6381
dtype: int64
```

In [23]:

```
# Impute `location` and `registration_number` with the most frequent values
df2['location'] = df2['location'].fillna(df2['location'].mode()[0])
df2['registration_number'] = df2['registration_number'].fillna(df2['registration_number']
.mode()[0])
```

In [24]:

```
# Impute `broad_phase_of_flight` using the most frequent value
df2['broad_phase_of_flight'] = df2['broad_phase_of_flight'].fillna(df2['broad_phase_of_fl
ight'].mode()[0])
```

In [25]:

```
#country`: Impute with mode or "Unknown" if highly diverse
df2['country'] = df2['country'].fillna(df2['country'].mode()[0])

#aircraft_damage`: Impute with mode
df2['aircraft_damage'] = df2['aircraft_damage'].fillna(df2['aircraft_damage'].mode()[0])

#amateur_built`: Impute with "No" assuming majority aircraft are not amateur-built
df2['amateur_built'] = df2['amateur_built'].fillna("No")

#number of engines replace with median
df2['number_of_engines'] = df2['number_of_engines'].fillna(df2['number_of_engines'].media
n())
```

In [26]:

```
#weather_condition`: Impute with mode
df2['weather_condition'] = df2['weather_condition'].fillna(df2['weather_condition'].mode(
)[0])
```

In [27]:

```
columns_to_drop_now = ['airport_name']
columns_to_drop_now
```

```
df2 = df2.drop(columns=columns_to_drop_now)
```

In [28]:

```
#report_status`: Fill with "Unknown" if missing
df2['report_status'] = df2['report_status'].fillna("Unknown")
```

In [29]:

```
#For rows where grouping doesn't provide a mode, fill remaining with "Unknown"
df2['engine_type'] = df2['engine_type'].fillna("Unknown")

#purpose_of_flight`: Impute with mode or "Unknown"
df2['purpose_of_flight'] = df2['purpose_of_flight'].fillna("Unknown")
```

In [30]:

```
#Injury-related columns: Replace missing with 0
injury_columns = [
    'total_fatal_injuries', 'total_serious_injuries',
    'total_minor_injuries', 'total_uninjured'
]
df2[injury_columns] = df2[injury_columns].fillna(0)
```

In [31]:

```
df2.columns
```

Out[31]:

```
Index(['investigation_type', 'event_date', 'location', 'country',
       'injury_severity', 'aircraft_damage', 'registration_number', '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', 'report_status'],
      dtype='object')
```

In [32]:

```
#check for unique values all at once

for column in df2:
 unique_values = df2[column].unique()
 print(f"unique values in column {column}, '\n': {unique_values}", '\n')
```

```
unique values in column investigation_type, '
': ['Accident' 'Incident']

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, AZ'
 'Auburn Hills, MI' 'Brasnorte, ']

unique values in column country, '
': ['United States' '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 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' 'Minor' 'Serious']

unique values in column aircraft_damage, '
': ['Destroyed' 'Substantial' 'Minor' 'Unknown']

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']

unique values in column number_of_engines, '
': [1. 2. 0. 3. 4. 8. 6.]
```

```
unique values in column engine_type, '
': ['Reciprocating' 'Unknown' 'Turbo Fan' 'Turbo Shaft' 'Turbo Prop'
 'Turbo Jet' 'None' 'Electric' 'Hybrid Rocket' 'Geared Turbofan' 'LR'
 'NONE' 'UNK']

unique values in column purpose_of_flight, '
': ['Personal' 'Unknown' 'Business' 'Instructional' 'Ferry'
 'Executive/corporate' 'Aerial Observation' 'Aerial Application'
 'Public Aircraft' 'Skydiving' 'Other Work Use' 'Positioning'
 'Flight Test' 'Air Race/show' 'Air Drop' 'Public Aircraft - Federal'
 'Glider Tow' 'Public Aircraft - Local' 'External Load'
 'Public Aircraft - State' 'Banner Tow' 'Firefighting' 'Air Race show'
 'PUBS' 'ASHO' 'PUBL']

unique values in column total_fatal_injuries, '
': [  2.   4.   3.   1.   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.   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.   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.  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' 'Unk']

unique values in column broad_phase_of_flight, '
```

```
': ['Cruise' 'Unknown' 'Approach' 'Climb' 'Takeoff' 'Landing' 'Taxi'
 'Descent' 'Maneuvering' 'Standing' 'Go-around' 'Other']

unique values in column report_status, '
': ['Probable Cause' 'Factual' 'Foreign' ...
 'The pilot did not ensure adequate clearance from construction vehicles during taxi.'
 'The pilot\x92s failure to secure the magneto switch before attempting to hand rotate th
e engine which resulted in an inadvertent engine start, a runaway airplane, and subsequen
t impact with parked airplanes. Contributing to the accident was the failure to properly
secure the airplane with chocks.'
 'The pilot\x92s loss of control due to a wind gust during landing.']
```

In [33]:

```
# Drop the 'report_status' column
df2 = df2.drop(columns=["report_status"])
```

In [34]:

```
df2.columns
```

Out[34]:

```
Index(['investigation_type', 'event_date', 'location', 'country',
       'injury_severity', 'aircraft_damage', 'registration_number', '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'],
      dtype='object')
```

In [35]:

```
#check unique value for each column and count them

df2.groupby('investigation_type')['investigation_type'].count()
```

Out[35]:

```
investigation_type
Accident    85015
Incident     3874
Name: investigation_type, dtype: int64
```

In [36]:

```
#check unique value for each column and count them

df2.groupby('injury_severity')['injury_severity'].count()
```

Out[36]:

```
injury_severity
Fatal           5262
Fatal(1)        6167
Fatal(10)         32
Fatal(102)         2
Fatal(104)         2
               ...
Incident        2219
Minor            218
Non-Fatal      68357
Serious          173
Unavailable       96
Name: injury_severity, Length: 109, dtype: int64
```

In [37]:

```
# Replace rows like "Fatal(2)", "Fatal(3)", etc., with "Fatal"
df2['injury_severity'] = df2['injury_severity'].str.replace(r'Fatal\(\d+\)', 'Fatal', re
gex=True)
```

```
# Verify the changes
print(df2['injury_severity'].unique())
```

```
['Fatal' 'Non-Fatal' 'Incident' 'Unavailable' 'Minor' 'Serious']
```

In [38]:

```
#check unique value for each column and count them

df2.groupby('engine_type')['engine_type'].count()
```

Out[38]:

```
engine_type
Electric               10
Geared Turbofan        12
Hybrid Rocket           1
LR                      2
NONE                    2
None                   19
Reciprocating       69530
Turbo Fan            2481
Turbo Jet             703
Turbo Prop           3391
Turbo Shaft          3609
UNK                     1
Unknown              9128
Name: engine_type, dtype: int64
```

In [39]:

```
#Replacing unique values in engine_type

df2["engine_type"] = df2["engine_type"].str.replace("NONE", "None")

df2["engine_type"] = df2["engine_type"].str.replace("UNK", "Unknown")
```

In [40]:

```
#check unique value for each column and count them

df2.groupby('engine_type')['engine_type'].count()
```

Out[40]:

```
engine_type
Electric               10
Geared Turbofan        12
Hybrid Rocket           1
LR                      2
None                   21
Reciprocating       69530
Turbo Fan            2481
Turbo Jet             703
Turbo Prop           3391
Turbo Shaft          3609
Unknown              9129
Name: engine_type, dtype: int64
```

In [41]:

```
#check unique value for each column and count them

df2.groupby('weather_condition')['weather_condition'].count()
```

Out[41]:

```
weather_condition
IMC     5976
UNK      856
Unk      262
VMC    81795
```

```
Name: weather_condition, dtype: int64
```

In [42]:

```
df2["weather_condition"] = df2["weather_condition"].str.replace("UNK", "Unk")
```

3.3 Checking Duplicates

In [43]:

```
#check for duplicates and drop them

df2.duplicated().sum()
df2 = df2.drop_duplicates()
```

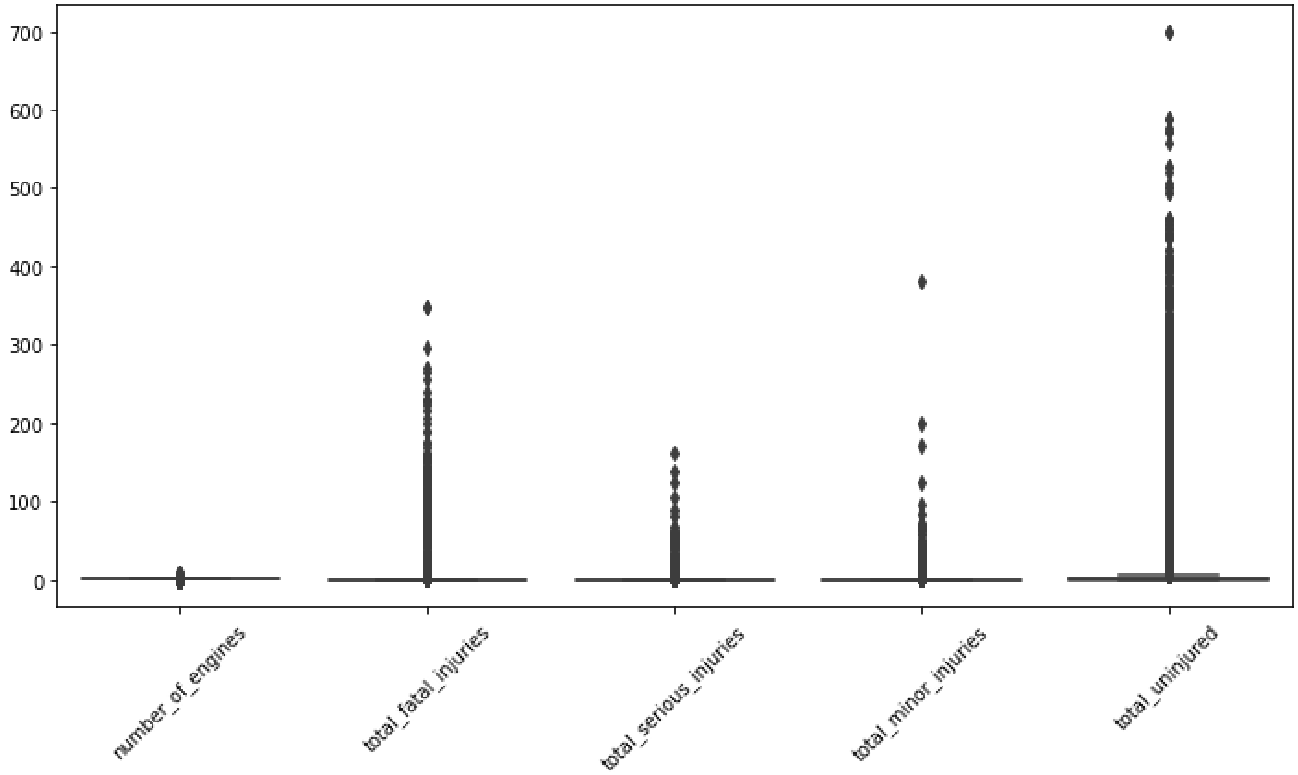
3.4. Checking Outliers

In [44]:

```
#to show all outliers

numeric_cols = df2.select_dtypes(include='number')

# Create a boxplot for each numeric column
plt.figure(figsize=(12, 6))
sns.boxplot(data=numeric_cols)

# Show the plot
plt.xticks(rotation=45)
```

Out[44]:

```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'number_of_engines'),
  Text(1, 0, 'total_fatal_injuries'),
  Text(2, 0, 'total_serious_injuries'),
  Text(3, 0, 'total_minor_injuries'),
  Text(4, 0, 'total_uninjured')])
```

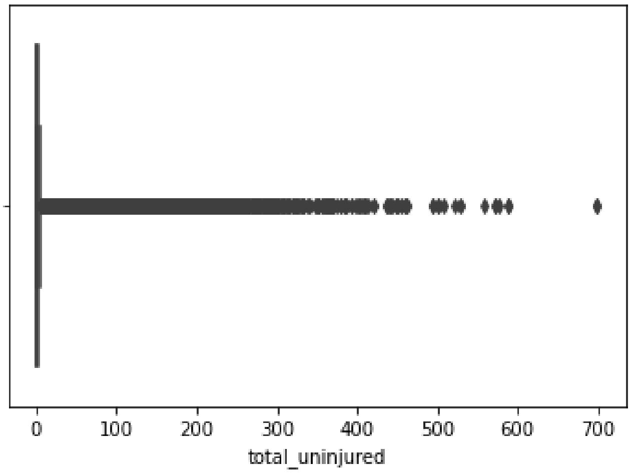


In [45]:

```
#identifying outliers in total_uninjured
```

```
sns.boxplot(df2["total_uninjured"]);
```

```
c:\Users\Fluxtech\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [46]:

```
#removing outliers
max_total_unijured = df2["total_uninjured"].quantile(0.995)
max_total_unijured

#Check outliers at max
df2[df2["total_uninjured"] > max_total_unijured]
```

Out[46]:

	investigation_type	event_date	location	country	injury_severity	aircraft_damage	registration_number	ma
2456	Incident	1982-08-21	HONOLULU, HI	United States	Incident	Minor	N104WA	Mcdonn Dougl
3578	Incident	1982-12-30	THERMAL, CA	United States	Incident	Substantial	N137AA	Mcdonne dougl
3686	Incident	1983-01-13	CHICAGO, IL	United States	Incident	Minor	N115AA	Mcdonn Dougl
3702	Incident	1983-01-16	LOS ANGELES, CA	United States	Incident	Minor	N9664	Boei
4149	Incident	1983-03-18	LOS ANGELES, CA	United States	Incident	Minor	N323EA	Lockhe
...	...	...	...	...	...	...	...	
88319	Accident	2022-08-06	Atlanta, GA	United States	Non-Fatal	Substantial	N540US	BOEII
88563	Incident	2022-09-22	Los Angeles, CA	United States	Non-Fatal	Substantial	N393HA	AIRBI
88605	Incident	2022-10-01	Manila,	Philippines	Non-Fatal	Minor	HZ-AK28	BOEII
88726	Incident	2022-10-27	Buenos Aires,	Argentina	Non-Fatal	Substantial	N765AN	BOEII
88742	Incident	2022-11-02	LIberia,	Costa Rica	Non-Fatal	Substantial	N6714Q	BOEII

440 rows × 19 columns

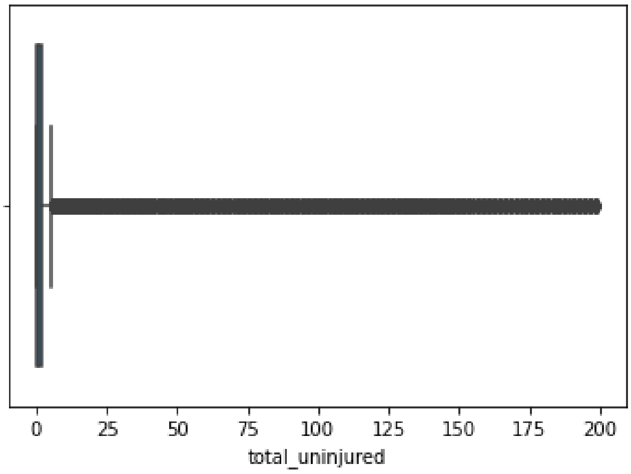
In [47]:

```
df3 = df2[df2["total_uninjured"] < max_total_unijured]
```

In [48]:

```
sns.boxplot(df3["total_uninjured"]);
```

```
c:\Users\Fluxtech\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

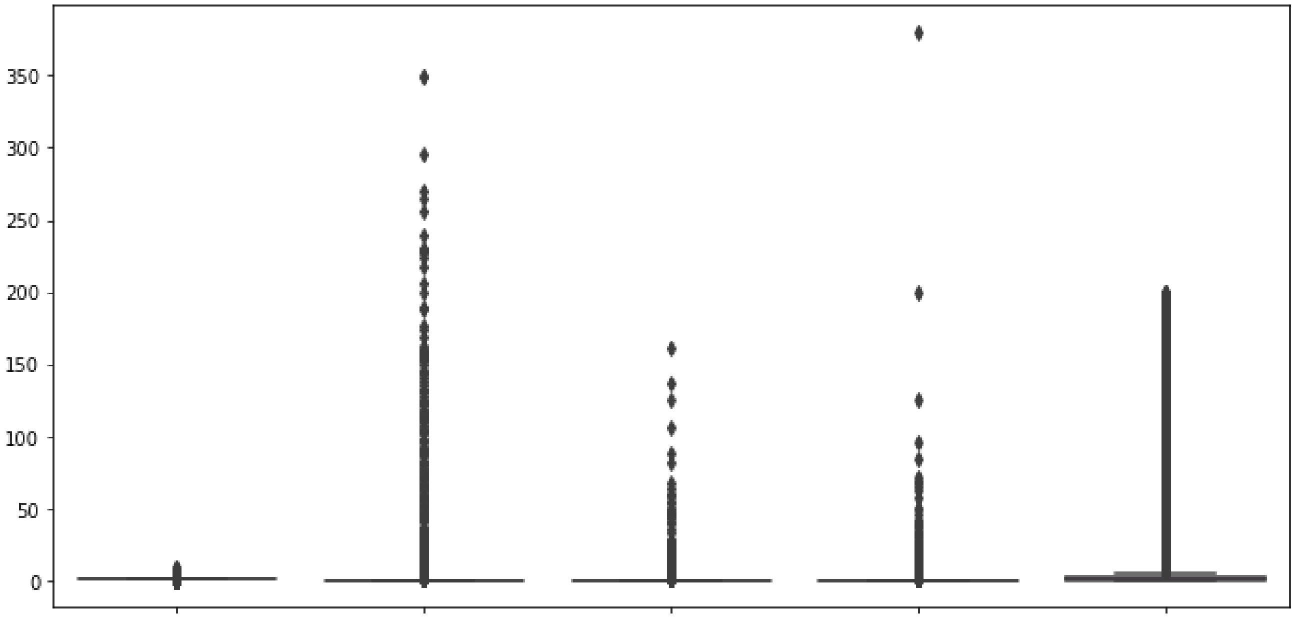


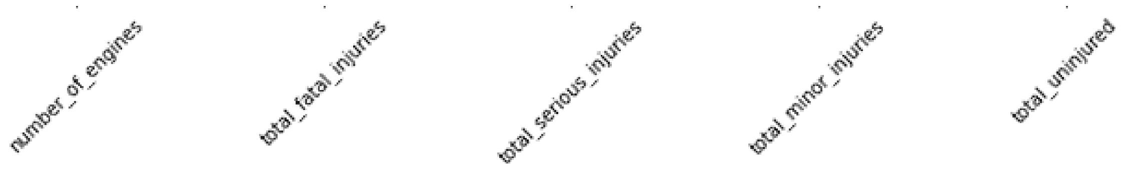
In [49]:

```
#to show all outliers

numeric_cols = df3.select_dtypes(include='number')

# Create a boxplot for each numeric column
plt.figure(figsize=(12, 6))
sns.boxplot(data=numeric_cols)

# Show the plot
plt.xticks(rotation=45)
```

Out[49]:

```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'number_of_engines'),
  Text(1, 0, 'total_fatal_injuries'),
  Text(2, 0, 'total_serious_injuries'),
  Text(3, 0, 'total_minor_injuries'),
  Text(4, 0, 'total_uninjured')])
```





In [50]:

```
#removing outliers
max_total_minor_injuries = df3["total_minor_injuries"].quantile(0.995)
max_total_minor_injuries

#Check outliers at max
df3[df3["total_minor_injuries"] > max_total_minor_injuries]
```

Out[50]:

	investigation_type	event_date	location	country	injury_severity	aircraft_damage	registration_number	ma
155	Accident	1982-01-23	BOSTON, MA	United States	Fatal	Destroyed	N113WA	Mcdonn Dougl
229	Accident	1982-02-03	HAYDEN, CO	United States	Non-Fatal	Destroyed	N149JA	Mitsubis
552	Accident	1982-03-12	SYRACUSE, NY	United States	Non-Fatal	Substantial	N260BB	Pip
1343	Accident	1982-05-26	SAN FRANCISCO, CA	United States	Non-Fatal	Substantial	N1833U	Dougl
1347	Incident	1982-05-27	NEAR GOSHEN, IN	United States	Incident	Substantial	N8088U	Dougl
...	...	...	...	...	...	...	...	
86385	Accident	2021-06-14	Madisonville, TX	United States	Fatal	Substantial	N3258W	PIPI
86814	Accident	2021-08-28	Kuserua, Spain	Spain	Non-Fatal	Unknown	HC-CMQ	BRITTI NORM/
86864	Accident	2021-09-09	Provincetown, MA	United States	Non-Fatal	Substantial	N88833	CESSI
87788	Accident	2022-05-11	BOITUVA, OF	Brazil	Fatal	Substantial	PT-OQR	CESSI
88025	Accident	2022-06-22	Papua,	Indonesia	Serious	Substantial	PK-BVM	PILATI

400 rows × 19 columns

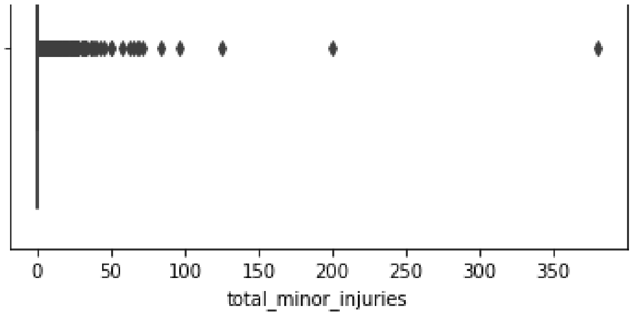
In [51]:

```
df4 = df3[df3["total_minor_injuries"] < max_total_minor_injuries]

sns.boxplot(df3["total_minor_injuries"])
```

```
c:\Users\Fluxtech\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[51]:

```
<AxesSubplot:xlabel='total_minor_injuries'>
```



In [52]:

```
#removing outliers
max_total_serious_injuries = df4["total_serious_injuries"].quantile(0.995)
max_total_serious_injuries

#Check outliers at max
df4[df4["total_serious_injuries"] > max_total_serious_injuries]
```

Out[52]:

	investigation_type	event_date	location	country	injury_severity	aircraft_damage	registration_number	ma
84	Accident	1982-01-13	WASHINGTON, DC	United States	Fatal	Destroyed	N62AF	Boei
214	Accident	1982-02-01	GROTON, CT	United States	Non-Fatal	Destroyed	N451C	Bee
377	Accident	1982-02-21	PROVIDENCE, RI	United States	Fatal	Destroyed	N127PM	Havilla
1216	Accident	1982-05-16	HOOPER BAY, AK	United States	Non-Fatal	Destroyed	N103AQ	Havilla
1465	Accident	1982-06-06	ST. PETERSBURG, FL	United States	Non-Fatal	Destroyed	N95C	Dougl
...	...	...	...	...	...	...	...	
85149	Accident	2020-08-03	Jackson, WY	United States	Non-Fatal	Substantial	N971LB	Lindstra
86455	Accident	2021-06-25	New Orleans, LA	United States	Non-Fatal	Substantial	N926UW	AIRBI
87537	Accident	2022-03-16	Baía de Camamu, BA, OF	Brazil	Fatal	Substantial	PR-LCT	SIKORSI
87861	Accident	2022-05-27	Lahore,	Pakistan	Serious	Substantial	A6-BLF	BOEIN
88505	Accident	2022-09-10	Oriximina, OF	Brazil	Fatal	Substantial	PT-MES	CESSI

203 rows × 19 columns

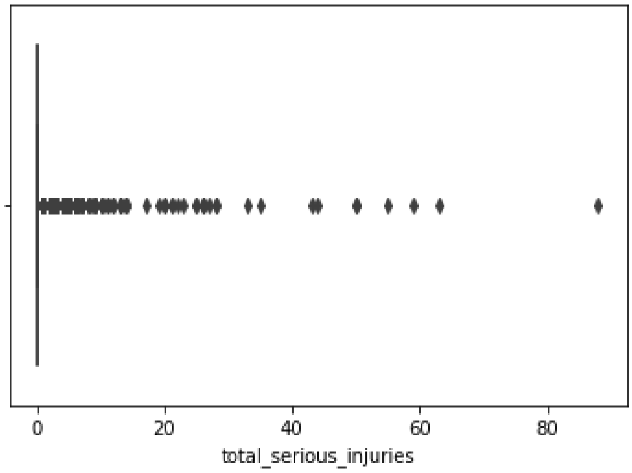
In [53]:

```
df5 = df4[df4["total_serious_injuries"] < max_total_serious_injuries]

sns.boxplot(df4["total_serious_injuries"])
```

```
c:\Users\Fluxtech\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[53]:

```
<AxesSubplot:xlabel='total_serious_injuries'>
```



In [54]:

```
#removing outliers
max_total_fatal_injuries = df5["total_fatal_injuries"].quantile(0.995)
max_total_fatal_injuries

#Check outliers at max
df4[df4["total_fatal_injuries"] > max_total_fatal_injuries]
```

Out[54]:

	investigation_type	event_date	location	country	injury_severity	aircraft_damage	registration_number	
25	Accident	1982-01-03	ASHLAND, VA	United States	Fatal	Destroyed	N2620L	
84	Accident	1982-01-13	WASHINGTON, DC	United States	Fatal	Destroyed	N62AF	
165	Accident	1982-01-24	LAREDO, TX	United States	Fatal	Destroyed	N4244Z	Rol
254	Accident	1982-02-07	W. OF HOMESTEAD, FL	United States	Fatal	Destroyed	N7361P	
255	Accident	1982-02-07	W. OF HOMESTEAD, FL	United States	Fatal	Destroyed	N2280G	
...	...	...	...	...	...	...	...	
87471	Accident	2022-02-26	Comoros,	China	Fatal	Unknown	5H-MZA	C
87562	Accident	2022-03-21	Wuzhou,	China	Fatal	Destroyed	B-1791	E
88468	Accident	2022-09-04	Freeland, WA	United States	Fatal	Substantial	N725TH	DEHAVI
88689	Accident	2022-10-18	Rudraprayag,	India	Fatal	Destroyed	VT-RPN	
88806	Accident	2022-11-21	Medellin,	Colombia	Fatal	Destroyed	HK 5121	

462 rows × 19 columns

In [55]:

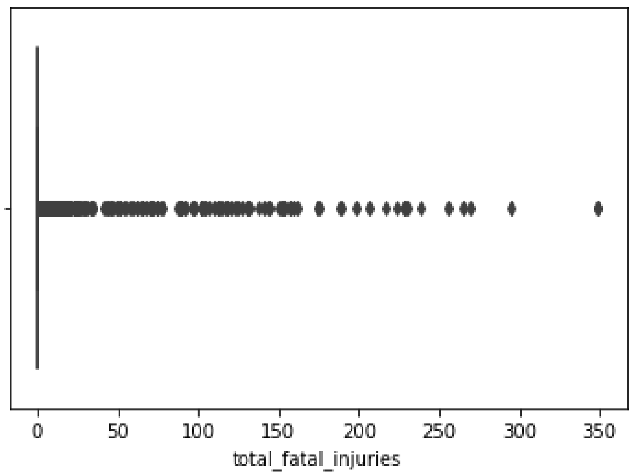
```
df6 = df5[df5["total_fatal_injuries"] < max_total_fatal_injuries]

sns.boxplot(df4["total_fatal_injuries"])
```

```
c:\Users\Fluxtech\anaconda3\envs\learn-env\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an explic
```

```
y valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[55]:

```
<AxesSubplot:xlabel='total_fatal_injuries'>
```



In [56]:

```
df6.columns
```

Out[56]:

```
Index(['investigation_type', 'event_date', 'location', 'country',
       'injury_severity', 'aircraft_damage', 'registration_number', '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'],
      dtype='object')
```

In [57]:

```
#confirming data types

df6.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 86590 entries, 0 to 88888
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   investigation_type      86590 non-null  object 
 1   event_date              86590 non-null  object 
 2   location                86590 non-null  object 
 3   country                 86590 non-null  object 
 4   injury_severity         86590 non-null  object 
 5   aircraft_damage         86590 non-null  object 
 6   registration_number     86590 non-null  object 
 7   make                    86590 non-null  object 
 8   model                   86590 non-null  object 
 9   amateur_built           86590 non-null  object 
 10  number_of_engines       86590 non-null  float64
 11  engine_type             86590 non-null  object 
 12  purpose_of_flight       86590 non-null  object 
 13  total_fatal_injuries    86590 non-null  float64
 14  total_serious_injuries  86590 non-null  float64
 15  total_minor_injuries    86590 non-null  float64
 16  total_uninjured         86590 non-null  float64
 17  weather_condition       86590 non-null  object 
 18  broad_phase_of_flight   86590 non-null  object 
dtypes: float64(5), object(14)
memory usage: 13.2+ MB
```

In [58]:

```
# Filter the rows where the country is 'United States'
df6 = df6[df6['country'] == 'United States']
```

In [59]:

```
# Use str.split and limit to 2 splits (expand=True ensures two separate columns)
df6[['City', 'State']] = df6['location'].str.split(', ', expand=True, n=1)

# Fill missing 'State' values with 'Unknown' to handle inconsistencies
df6['State'] = df6['State'].fillna('Unknown')
```

In [60]:

```
df6.to_csv("AviationData_cleansetfinal.csv", index = False)
```

In [61]:

```
df6 = pd.read_csv("AviationData_cleansetfinal.csv")
df6.head()
```

Out[61]:

	investigation_type	event_date	location	country	injury_severity	aircraft_damage	registration_number	make	moc
0	Accident	1948-10-24	MOOSE CREEK, ID	United States	Fatal	Destroyed	NC6404	Stinson	108
1	Accident	1962-07-19	BRIDGEPORT, CA	United States	Fatal	Destroyed	N5069P	Piper	PA2 1
2	Accident	1974-08-30	Saltville, VA	United States	Fatal	Destroyed	N5142R	Cessna	17
3	Accident	1977-06-19	EUREKA, CA	United States	Fatal	Destroyed	N1168J	Rockwell	1
4	Accident	1979-08-02	Canton, OH	United States	Fatal	Destroyed	N15NY	Cessna	5

5 rows × 21 columns

The research aims at first indentifying the safest Aircraft based on the frequency of the accidents.

It was noted that CESSNA 72 and Cessna 72 is the same plane. That means the data may require further cleaning

In [62]:

```
df6["make"] = df6["make"].str.replace("CESSNA", "Cessna")
```

In [63]:

```
# Extract state abbreviation (last two characters after a comma)
df6['State_Abbreviation'] = df6['location'].str.extract(r',\s*([A-Z]{2})$')

# Fill missing values with 'Unknown'
df6['State_Abbreviation'] = df6['State_Abbreviation'].fillna('Unknown')

# Check the results
print(df6[['location', 'State_Abbreviation']].head())
```

```
          location State_Abbreviation
0  MOOSE CREEK, ID                 ID
1   BRIDGEPORT, CA                 CA
2    Saltville, VA                 VA
3       EUREKA, CA                 CA
4       Canton, OH                 OH
```

In [64]:

```python
# Find rows without valid state abbreviations
missing_states = df6[df6['State_Abbreviation'] == 'Unknown']
print(missing_states)
```

```
      investigation_type  event_date           location        country  \
35              Accident  1982-01-04        SAINT CROIX  United States
443             Accident  1982-03-02           HUMA CAO  United States
444             Accident  1982-03-02    MUSTANG BLK A11  United States
697             Accident  1982-03-31               MOCA  United States
802             Accident  1982-04-13    WEST DELTA 105D  United States
...                  ...         ...                ...            ...
76833           Accident  2019-07-10            Saipan,  United States
77140           Accident  2019-09-12  Charlotte Amalie,  United States
77150           Accident  2019-09-15    Gulf of Mexico,  United States
77268           Accident  2019-10-17           Unknown,  United States
77829           Accident  2020-06-01    GULF OF MEXICO,  United States

      injury_severity aircraft_damage registration_number    make    model  \
35              Fatal       Destroyed              N5151U  Cessna      206
443             Fatal       Destroyed              N2741J  Cessna      150
444         Non-Fatal     Substantial              N1080S    Bell       12
697         Non-Fatal     Substantial              N309MJ   Piper   PA-32R
802         Non-Fatal       Destroyed               N41AL    Bell     206B
...               ...             ...                 ...     ...      ...
76833       Non-Fatal     Substantial              N6733F   Piper     PA28
77140       Non-Fatal     Substantial              N269KW   Piper     PA23
77150       Non-Fatal       Destroyed              N218MW   Piper     PA46
77268           Fatal       Destroyed              N778PA   Piper     PA23
77829       Non-Fatal     Substantial               N619J  JABIRU  J250-SP

      amateur_built  ...   purpose_of_flight total_fatal_injuries  \
35               No  ...            Business                  1.0
443              No  ...            Personal                  1.0
444              No  ...             Unknown                  0.0
697              No  ...             Unknown                  0.0
802              No  ...               Ferry                  0.0
...             ...  ...                 ...                  ...
76833            No  ...  Aerial Observation                  0.0
77140            No  ...            Personal                  0.0
77150            No  ...            Personal                  0.0
77268            No  ...            Personal                  1.0
77829            No  ...            Personal                  0.0

      total_serious_injuries  total_minor_injuries  total_uninjured  \
35                       0.0                   1.0              0.0
443                      0.0                   0.0              0.0
444                      1.0                   0.0              5.0
697                      0.0                   0.0              1.0
802                      0.0                   1.0              0.0
...                      ...                   ...              ...
76833                    0.0                   0.0              3.0
77140                    0.0                   0.0              1.0
77150                    0.0                   0.0              1.0
77268                    0.0                   0.0              0.0
77829                    0.0                   0.0              2.0

       weather_condition  broad_phase_of_flight               City    State  \
35                   VMC                   Taxi        SAINT CROIX  Unknown
443                  VMC                Descent           HUMA CAO  Unknown
444                  VMC               Standing    MUSTANG BLK A11  Unknown
697                  Unk                Landing               MOCA  Unknown
802                  VMC                Takeoff    WEST DELTA 105D  Unknown
...                  ...                    ...                ...      ...
76833                VMC                Landing            Saipan,  Unknown
77140                VMC                Landing  Charlotte Amalie,  Unknown
77150                VMC                Landing    Gulf of Mexico,  Unknown
77268                VMC                Landing           Unknown,  Unknown
77829                VMC                Landing    GULF OF MEXICO,  Unknown

      State_Abbreviation
35               Unknown
443              Unknown
```

```
444               Unknown
697               Unknown
802               Unknown
...                   ...
76833             Unknown
77140             Unknown
77150             Unknown
77268             Unknown
77829             Unknown

[266 rows x 22 columns]
```

In [65]:

```
# Dictionary to map state abbreviations to full names
state_mapping = {
    "AL": "Alabama", "AK": "Alaska", "AZ": "Arizona", "AR": "Arkansas", "CA": "California",
    "CO": "Colorado", "CT": "Connecticut", "DE": "Delaware", "FL": "Florida", "GA": "Georgia",
    "HI": "Hawaii", "ID": "Idaho", "IL": "Illinois", "IN": "Indiana", "IA": "Iowa",
    "KS": "Kansas", "KY": "Kentucky", "LA": "Louisiana", "ME": "Maine", "MD": "Maryland",
    "MA": "Massachusetts", "MI": "Michigan", "MN": "Minnesota", "MS": "Mississippi",
    "MO": "Missouri", "MT": "Montana", "NE": "Nebraska", "NV": "Nevada", "NH": "New Hampshire",
    "NJ": "New Jersey", "NM": "New Mexico", "NY": "New York", "NC": "North Carolina",
    "ND": "North Dakota", "OH": "Ohio", "OK": "Oklahoma", "OR": "Oregon", "PA": "Pennsylvania",
    "RI": "Rhode Island", "SC": "South Carolina", "SD": "South Dakota", "TN": "Tennessee",
    "TX": "Texas", "UT": "Utah", "VT": "Vermont", "VA": "Virginia", "WA": "Washington",
    "WV": "West Virginia", "WI": "Wisconsin", "WY": "Wyoming"
}
```

In [66]:

```
# Extract state abbreviation (last two characters after a comma)
df6['State_Abbreviation'] = df6['location'].str.extract(r',\s*([A-Z]{2})$')

# Map the abbreviations to full state names
df6['State'] = df6['State_Abbreviation'].map(state_mapping)

# Fill missing state names with 'Unknown'
df6['State'] = df6['State'].fillna('Unknown')

# Check the results
print(df6[['location', 'State_Abbreviation', 'State']].head())
```

```
          location State_Abbreviation       State
0  MOOSE CREEK, ID                 ID       Idaho
1   BRIDGEPORT, CA                 CA  California
2    Saltville, VA                 VA    Virginia
3       EUREKA, CA                 CA  California
4       Canton, OH                 OH        Ohio
```

In [67]:

```
df6.drop(columns=['State_Abbreviation'], inplace=True)
```

In [68]:

```
# Remove the state abbreviation and keep only the city name
df6['City'] = df6['location'].str.replace(r',\s*[A-Z]{2}$', '', regex=True)

# Check the results
print(df6[['location', 'City']].head())
```

```
          location         City
0  MOOSE CREEK, ID  MOOSE CREEK
1   BRIDGEPORT, CA   BRIDGEPORT
2    Saltville, VA    Saltville
```

```
3       EUREKA, CA       EUREKA
4       Canton, OH       Canton
```

In [69]:

```
df6.drop(columns=['location'], inplace=True)
```

In [70]:

```
# Rename the 'City' column to 'location'
df6.rename(columns={'City': 'location'}, inplace=True)

# Check the updated DataFrame
print(df6.head())
```

```
  investigation_type  event_date        country injury_severity  \
0           Accident  1948-10-24  United States           Fatal   
1           Accident  1962-07-19  United States           Fatal   
2           Accident  1974-08-30  United States           Fatal   
3           Accident  1977-06-19  United States           Fatal   
4           Accident  1979-08-02  United States           Fatal   

  aircraft_damage registration_number      make     model amateur_built  \
0       Destroyed              NC6404   Stinson     108-3            No   
1       Destroyed              N5069P     Piper  PA24-180            No   
2       Destroyed              N5142R    Cessna      172M            No   
3       Destroyed              N1168J  Rockwell       112            No   
4       Destroyed               N15NY    Cessna       501            No   

   number_of_engines    engine_type purpose_of_flight  total_fatal_injuries  \
0                1.0  Reciprocating          Personal                   2.0   
1                1.0  Reciprocating          Personal                   4.0   
2                1.0  Reciprocating          Personal                   3.0   
3                1.0  Reciprocating          Personal                   2.0   
4                1.0        Unknown          Personal                   1.0   

   total_serious_injuries  total_minor_injuries  total_uninjured  \
0                     0.0                   0.0              0.0   
1                     0.0                   0.0              0.0   
2                     0.0                   0.0              0.0   
3                     0.0                   0.0              0.0   
4                     2.0                   0.0              0.0   

  weather_condition broad_phase_of_flight     location       State  
0               Unk                Cruise  MOOSE CREEK       Idaho  
1               Unk               Unknown   BRIDGEPORT  California  
2               IMC                Cruise    Saltville    Virginia  
3               IMC                Cruise       EUREKA  California  
4               VMC              Approach       Canton        Ohio  
```

In [71]:

```
df6.columns
```

Out[71]:

```
Index(['investigation_type', 'event_date', 'country', 'injury_severity',
       'aircraft_damage', 'registration_number', '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', 'location', 'State'],
      dtype='object')
```

In [72]:

```
df6.head()
```

Out[72]:

	investigation_type	event_date	country	injury_severity	aircraft_damage	registration_number	make	model	amateur_bu

	investigation_type	event_date	country	injury_severity	aircraft_damage	registration_number	make	model	amateur_bu
0	Accident	1948-10-24	United States	Fatal	Destroyed	NC6404	Stinson	108-3	N
1	Accident	1962-07-19	United States	Fatal	Destroyed	N5069P	Piper	PA24-180	N
2	Accident	1974-08-30	United States	Fatal	Destroyed	N5142R	Cessna	172M	N
3	Accident	1977-06-19	United States	Fatal	Destroyed	N1168J	Rockwell	112	N
4	Accident	1979-08-02	United States	Fatal	Destroyed	N15NY	Cessna	501	N

In [73]:

```
#Save Final document

df6.to_csv('AviationData_updatedupdatedupdated.csv', index=False)
```

4.0 Data Analysis and Results

4.1. Objective 1

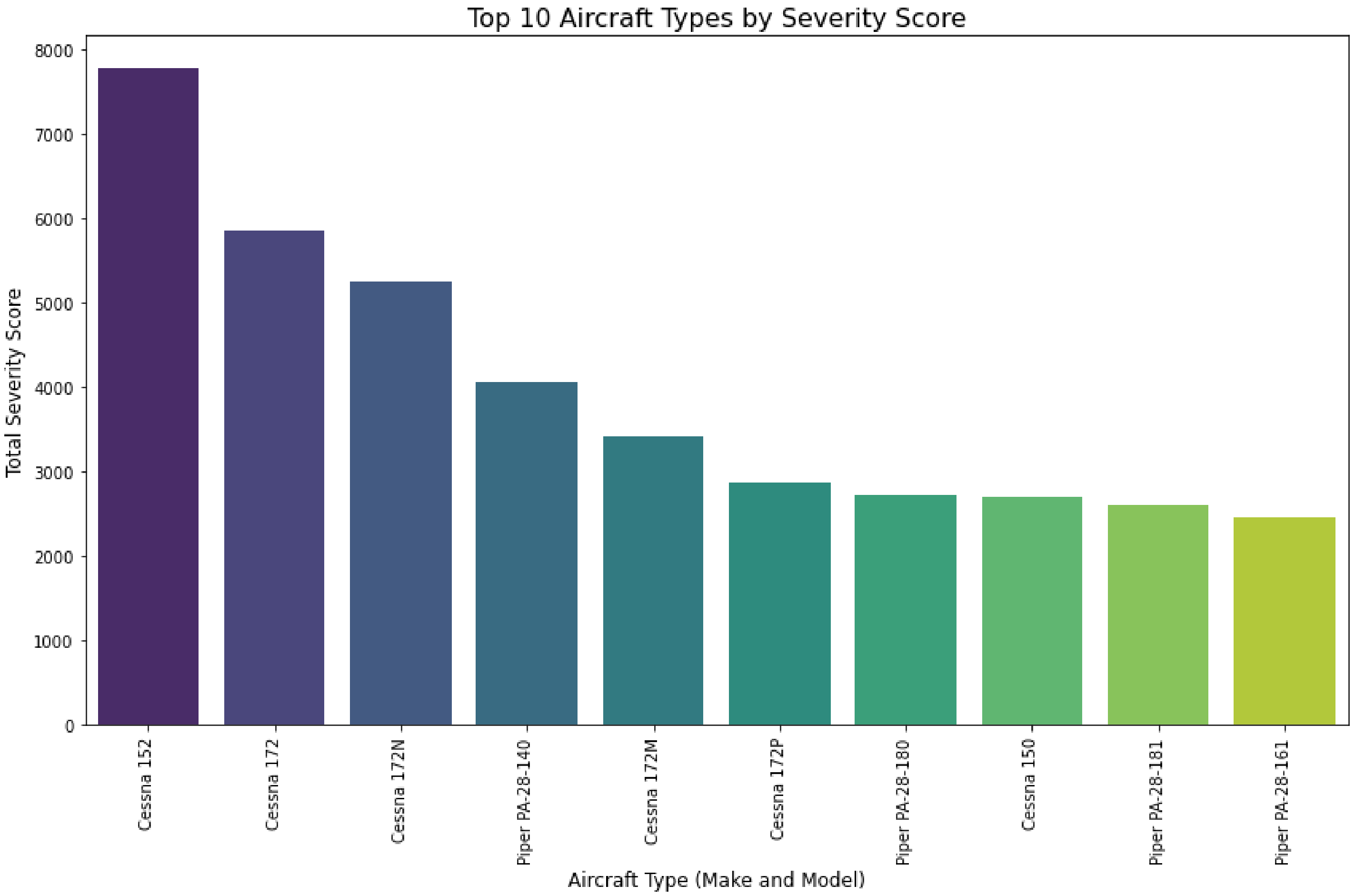
The goal of this object is to evaluate the aviation accident data with the goal of identifying the aircraft with the highest safety records and lowest risk.

The top 10 most common used aircraft were selected for the pupose of this report. Thereafter severity score was calculated as shown in the code. The severity score was calculated to quantify the seriousness of aviation accidents based on various injury categories and the extent of aircraft damage. This score helped in identifying which aircraft types are involved in more severe accidents highlighting the reasons as to why not to invest in those planes.

In [74]:

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

# Assuming df6 is already loaded and contains the necessary columns

# Calculate severity score
df6['severity_score'] = (
    (df6['total_fatal_injuries'] * 5) +
    (df6['total_serious_injuries'] * 3) +
    (df6['total_minor_injuries'] * 1) +
    df6['aircraft_damage'].map({
        'Destroyed': 3,
        'Substantial': 2,
        'Minor': 0
    })
)

# Combine 'make' and 'model' to create 'type_of_aircraft'
df6['type_of_aircraft'] = df6['make'] + " " + df6['model']

# Group by 'type_of_aircraft' and sum severity scores
make_model_severity = df6.groupby('type_of_aircraft').agg(
    total_severity_score=('severity_score', 'sum'),
    accident_count=('type_of_aircraft', 'size')
).reset_index()

# Sort by severity score in descending order
make_model_severity = make_model_severity.sort_values(by='total_severity_score', ascendin
g=False)
```

```
# Select the top 10 aircraft by severity score
top_10_severity = make_model_severity.head(10)

# Plot a vertical bar chart for the top 10 aircraft types by severity score
plt.figure(figsize=(12, 8))
sns.barplot(
    x=top_10_severity['type_of_aircraft'],
    y=top_10_severity['total_severity_score'],
    palette='viridis'
)

# Add titles and labels
plt.title('Top 10 Aircraft Types by Severity Score', fontsize=16)
plt.xlabel('Aircraft Type (Make and Model)', fontsize=12)
plt.ylabel('Total Severity Score', fontsize=12)
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability

# Display the plot
plt.tight_layout()
plt.show()
```



4.2. Objective 2

The purpose of this objective is to analyze the data to understand factors contributing to accident frequency and severity of the selected 10 most common used aircrafts.

The severity is first calculated in order to find factor that contributes to most accidents.

In [75]:

```
import pandas as pd

# Assuming df6 is already loaded

# Calculate severity score based on injuries
df6['Severity Score'] = (
    (df6['total_fatal_injuries'] * 5) +
    (df6['total_serious_injuries'] * 3) +
    (df6['total_minor_injuries'] * 1) +
    df6['aircraft_damage'].map({
        'Destroyed': 3,
        'Substantial': 2,
        'Minor': 0
    })
)

# Display first few rows to check the calculation
print(df6[['make', 'model', 'weather_condition', 'broad_phase_of_flight', 'engine_type',
'Severity Score']].head())
```

```
       make     model weather_condition broad_phase_of_flight    engine_type  \
0   Stinson     108-3               Unk                Cruise  Reciprocating
1     Piper  PA24-180               Unk               Unknown  Reciprocating
2    Cessna      172M               IMC                Cruise  Reciprocating
3  Rockwell       112               IMC                Cruise  Reciprocating
4    Cessna       501               VMC              Approach        Unknown

   Severity Score
0            13.0
1            23.0
2            18.0
3            13.0
4            14.0
```

In [76]:

```
# Count accidents by aircraft model
accident_counts = df6.groupby(['make', 'model']).size().reset_index(name='Accident Count
')

# Sort by accident count in descending order
top_10_models = accident_counts.sort_values(by='Accident Count', ascending=False).head(1
0)

# Display top 10 models
print(top_10_models)
```

```
         make      model  Accident Count
4680   Cessna        152            2332
4704   Cessna        172            1631
4753   Cessna       172N            1123
4653   Cessna        150             791
13538   Piper  PA-28-140             791
4751   Cessna       172M             762
4756   Cessna       172P             665
4786   Cessna        180             614
4809   Cessna        182             580
4679   Cessna       150M             578
```

In [77]:

```
# Filter df6 to only include the top 10 models
top_10_df = df6[df6['make'].isin(top_10_models['make']) & df6['model'].isin(top_10_models
['model'])]

# Group by aircraft model and contributing factors to calculate the average severity scor
e
factors_analysis = top_10_df.groupby(['make', 'model', 'weather_condition', 'broad_phase
_of_flight', 'engine_type']).agg(
    accident_count=('make', 'size'),
    avg_severity_score=('Severity Score', 'mean')
).reset_index()

# Display the results
print(factors_analysis.head())
```

```
     make model weather_condition broad_phase_of_flight    engine_type  \
0  Cessna   150               IMC              Approach  Reciprocating
1  Cessna   150               IMC                Cruise  Reciprocating
```

```
2  Cessna   150            IMC          Descent  Reciprocating
3  Cessna   150            IMC          Landing  Reciprocating
4  Cessna   150            IMC          Takeoff  Reciprocating

   accident_count  avg_severity_score
0               4                 8.0
1              10                 6.0
2               1                 4.0
3               5                 6.4
4               3                 5.0
```

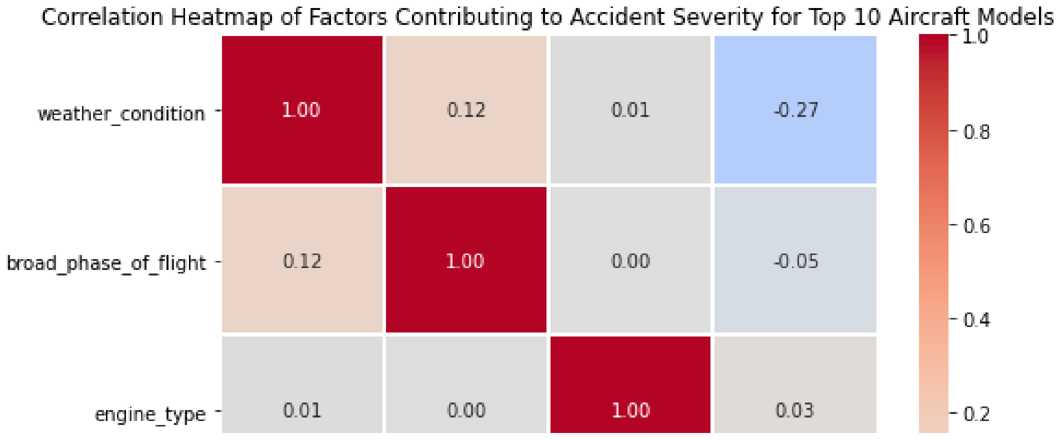
In [78]:

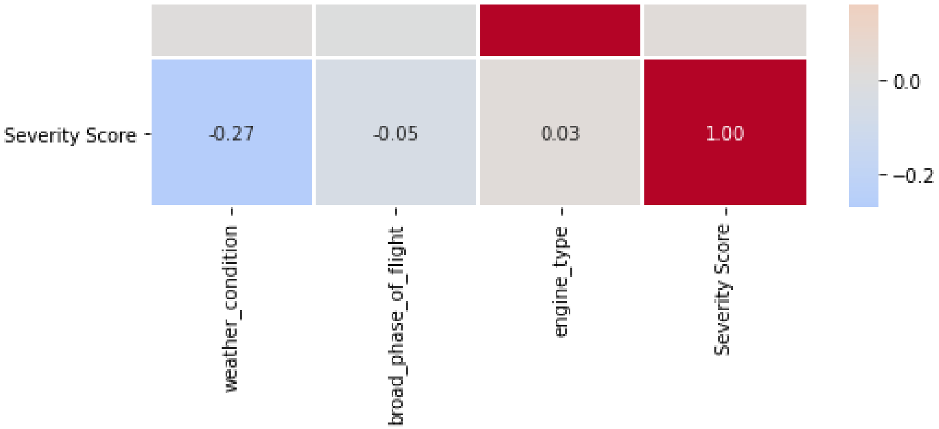
```python
# Encode categorical variables for correlation analysis
top_10_df_encoded = top_10_df.copy()

# Convert categorical variables to category codes
top_10_df_encoded['weather_condition'] = pd.Categorical(top_10_df_encoded['weather_condition']).codes
top_10_df_encoded['broad_phase_of_flight'] = pd.Categorical(top_10_df_encoded['broad_phase_of_flight']).codes
top_10_df_encoded['engine_type'] = pd.Categorical(top_10_df_encoded['engine_type']).codes

# Calculate the correlation matrix for the factors and severity score
correlation_matrix = top_10_df_encoded[['weather_condition', 'broad_phase_of_flight', 'engine_type', 'Severity Score']].corr()

# Display correlation matrix
print(correlation_matrix)
```

```
                       weather_condition  broad_phase_of_flight  engine_type  \
weather_condition               1.000000               0.115083     0.012540   
broad_phase_of_flight           0.115083               1.000000     0.000839   
engine_type                     0.012540               0.000839     1.000000   
Severity Score                 -0.271399              -0.050023     0.028046   

                       Severity Score  
weather_condition           -0.271399  
broad_phase_of_flight       -0.050023  
engine_type                  0.028046  
Severity Score               1.000000  
```

In [79]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0, fmt='.2f', linewidths=1)

# Set title and labels
plt.title('Correlation Heatmap of Factors Contributing to Accident Severity for Top 10 Aircraft Models')
plt.show()
```





4.3. Objective 3

The purpose of this objective is to use Geospatial Map to visualize accident distribution and risk hotspots the US.

By translating these findings into actionable insights, this project will assist the head of the aviation division in making informed, risk-conscious decisions when selecting aircraft for the company's new business endeavor.

The mapping was done using tableau and the link is provide below.

a) Private aircraft, such as the Piper-PA-28-161 and Cessna 150, tend to have lower severity scores. However, this is still higher than for commercial aircraft. The overall risk of accidents is higher in private aviation due to lower pilot experience and fewer safety features. Therefore, the risk of investing in private plane is higher than in commercial planes

b) Weather conditions are the dominant factor influencing accident severity, with a clear correlation to severe weather events causing higher-risk incidents. The correction among Engine type, broad-phase of flight and weather condition is also very low.

c) Most accidents happen during landing, followed by cruise and then takeoff. Very few accidents happen when a plane is standing and when taxiing to the Runway

d) California has the highest severity scores, followed by Florida and Texas, suggesting these states have higher accident rates compared to others. North and South Dakota have the lowest accident severity, indicating a lower risk for operations in these states.

5. Recommendations

a). Prioritize Commercial Aircraft Over Private Aircraft: Commercial aircraft generally exhibit a lower risk of accidents compared to private aircraft. Investing in commercial aircraft would align with the company’s goal of minimizing risk and enhancing safety for the new aviation venture.

b). Invest in Aircraft Resilient to Extreme Weather: Weather conditions are the leading cause of aviation accidents. The company should consider aircraft that are designed to withstand extreme weather, such as those equipped with better de-icing and weather detection systems.

c). Focus on Landing Infrastructure and Safety Protocols: The landing phase of flight is where most accidents occur. The company should invest in improving airport infrastructure and implement robust safety protocols for landing procedures. This will reduce the overall risk during this critical phase of flight.

d). Invest in North and south Dakota: These best place to start the company since very few accidents happen there

6. Next Steps

The following next steps should be taken to operationalize these findings:

a). Aircraft Review: The company should review various commercial aircraft models that meet safety and

weather-resilience criteria.

b). Weather Resilience Investment: Identify and prioritize aircraft that have been proven to perform well in adverse weather conditions.

c). Landing Infrastructure: Conduct a feasibility study on improving landing infrastructure, focusing on critical accident hotspots, particularly in states like California and Florida.