Final Project Submission

Please fill out:

- Student name: Mary Njeri Kamithi
- Student pace: self paced / part time / full time : Full Time Remote (DSFT13)
- Scheduled project review date/time: 29/6/2025
- Instructor name: William Okomba
- Blog post URL:

Aviation Accident Risk Analysis

Introduction

```
'''
Welcome to my aviation accident analysis project!

In this project, I analyzed aviation accident data from the National Transportation Safety Board (NTSB) to help a company decide which types
The dataset includes accidents from 1962 to 2022, with information about flight phases, aircraft damage, fatalities, and more.

My goal was to clean and explore the data to find patterns in aviation accidents and identify which aircraft characteristics are linked to h
I created visualizations and summarized the most important findings to support business decisions and reduce potential risks when purchasing


'''
```

```
'\nWelcome to my aviation accident analysis project!\n\nIn this project, I analyzed aviation accident data from the National Transporta
tion Safety Board (NTSB) to help a company decide which types of aircraft might be the safest investment as they enter the aviation ind
ustry.\nThe dataset includes accidents from 1962 to 2022, with information about flight phases, aircraft damage, fatalities, and mor
e.\n\nMy goal was to clean and explore the data to find patterns in aviation accidents and identify which aircraft characteristics are
```

Data Preparation

```
#All neccessary imports
import pandas as pd
import numpy
import seaborn as sns
import matplotlib.pyplot as plt


#Loading the dataset
df = pd.read_csv('AviationData.csv', encoding='latin1')
state_codes = pd.read_csv('USState_Codes.csv')
```

```
/tmp/ipython-input-75-2741206010.py:2: DtypeWarning: Columns (6,7,28) have mixed types. Specify dtype option on import or set low_memory
  df = pd.read_csv('AviationData.csv', encoding='latin1')
```

```
df.head()
```

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

5 rows × 31 columns

```
df.shape
```

```
(88889, 31)
```

```
df.info(verbose=True, show_counts=True)
```

```
<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
```

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

```
': [  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']
```

```
df.isnull().sum()
```

	0
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

```
df.head()
```

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

5 rows × 31 columns

```
df.dtypes
```

	0
Event.Id	object
Investigation.Type	object
Accident.Number	object
Event.Date	datetime64[ns]
Location	object
Country	object
Airport.Code	object
Airport.Name	object
Injury.Severity	object
Aircraft.damage	object
Registration.Number	object
Make	object
Model	object
Amateur.Built	object
Number.of.Engines	float64
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
Report.Status	object
Make_Grouped	object
Is_Fatal	int64

dtype: object

Handling null values

```
#handling the columns with too many null values
high_null_cols = ['Latitude','Longitude','Aircraft.Category','FAR.Description','Schedule','Air.carrier']
df.drop(columns = high_null_cols, inplace =True)


#Filling numeric cols with 0
num_cols = ['Total.Fatal.Injuries','Total.Serious.Injuries','Total.Minor.Injuries','Total.Uninjured']
df[num_cols] = df[num_cols].fillna(0)


#Filling categorical cols with 'Unknown'
cat_cols =[ 'Location', 'Country', 'Airport.Code', 'Airport.Name','Injury.Severity', 'Aircraft.damage', 'Registration.Number','Make', 'Model
df[cat_cols] = df[cat_cols].fillna('Unknown')


#Converting dates and handling null vals
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors = 'coerce')
df['Publication.Date'] = pd.to_datetime(df['Publication.Date'], errors = 'coerce')
```

```
/tmp/ipython-input-85-3081802418.py:3: UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the default) was specified. Pa
  df['Publication.Date'] = pd.to_datetime(df['Publication.Date'], errors = 'coerce')
```

```
#Rechecking missing vals
df.isnull().sum()
```

	0
Event.Id	0
Investigation.Type	0
Accident.Number	0
Event.Date	0
Location	0
Country	0
Airport.Code	0
Airport.Name	0
Injury.Severity	0
Aircraft.damage	0
Registration.Number	0
Make	0
Model	0
Amateur.Built	0
Number.of.Engines	6084
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
Report.Status	0
Publication.Date	13771

dtype: int64

```
#Handling the remaining columns that have null values
# Fill with median
df['Number.of.Engines'] = df['Number.of.Engines'].fillna(df['Number.of.Engines'].median())

#Handling publication date
df.drop(columns=['Publication.Date'], inplace=True)



df.isnull().sum().sum()
```

```
np.int64(0)
```

EXPLORATORY DATA ANALYSIS (EDA)

Univariate Analysis

```
#Numerical cols
numerical_cols = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured', 'Number.of.Engines']
df[numerical_cols].describe()
```

	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Number.of.Engines
count	88889.000000	88889.000000	88889.000000	88889.000000	88889.000000
mean	0.564761	0.240491	0.309127	4.971245	1.136552
std	5.126649	1.434614	2.083715	27.002011	0.432545
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	1.000000	1.000000
75%	0.000000	0.000000	0.000000	2.000000	1.000000
max	349.000000	161.000000	380.000000	699.000000	8.000000

```
# Convert to numeric in case there are non-numeric entries
df[numerical_cols] = df[numerical_cols].apply(pd.to_numeric, errors='coerce')

# Display summary statistics
summary = df[numerical_cols].describe().T[['min', 'max', 'mean', '50%', 'std']]
summary.rename(columns={'50%': 'median'}, inplace=True)
print(summary)
```

```
                       min    max      mean  median        std
Total.Fatal.Injuries    0.0  349.0  0.564761     0.0   5.126649
Total.Serious.Injuries  0.0  161.0  0.240491     0.0   1.434614
Total.Minor.Injuries    0.0  380.0  0.309127     0.0   2.083715
Total.Uninjured         0.0  699.0  4.971245     1.0  27.002011
Number.of.Engines       0.0    8.0  1.136552     1.0   0.432545
```

```
#Analysis of the model with the no of incidents
df['Model'].value_counts().head(10) # Check the top 10 models with the most accidents and incidents
df['Model'].value_counts().tail(10)
```

Model	count
DH-82 Tiger Moth	1
707-330C	1
Mustang MII	1
Bird CK	1
Tom Cat Mark 5A	1
2L	1
32RT-300T	1
RUATAN	1
SPECIAL 51C	1
Micro Mong	1

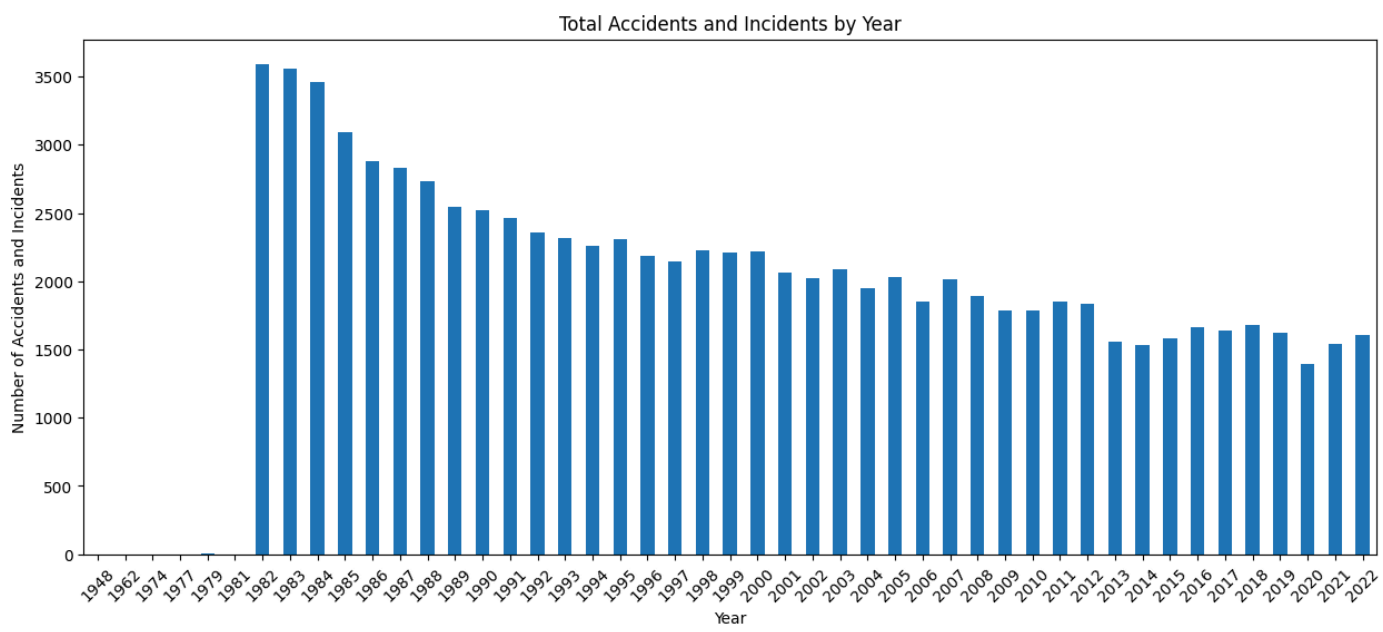
dtype: int64

```
# Take the year from the 'Event.Date' column and create a new 'Year' column
df['Year'] = pd.to_datetime(df['Event.Date'], errors='coerce').dt.year

# Total fatalities and injuries by year
df['Year'].value_counts().head(10) # Check the top 10 years with the most accidents and incidents
df['Year'].value_counts().tail(10) # Check the bottom 10 years with the most accidents and incidents
```

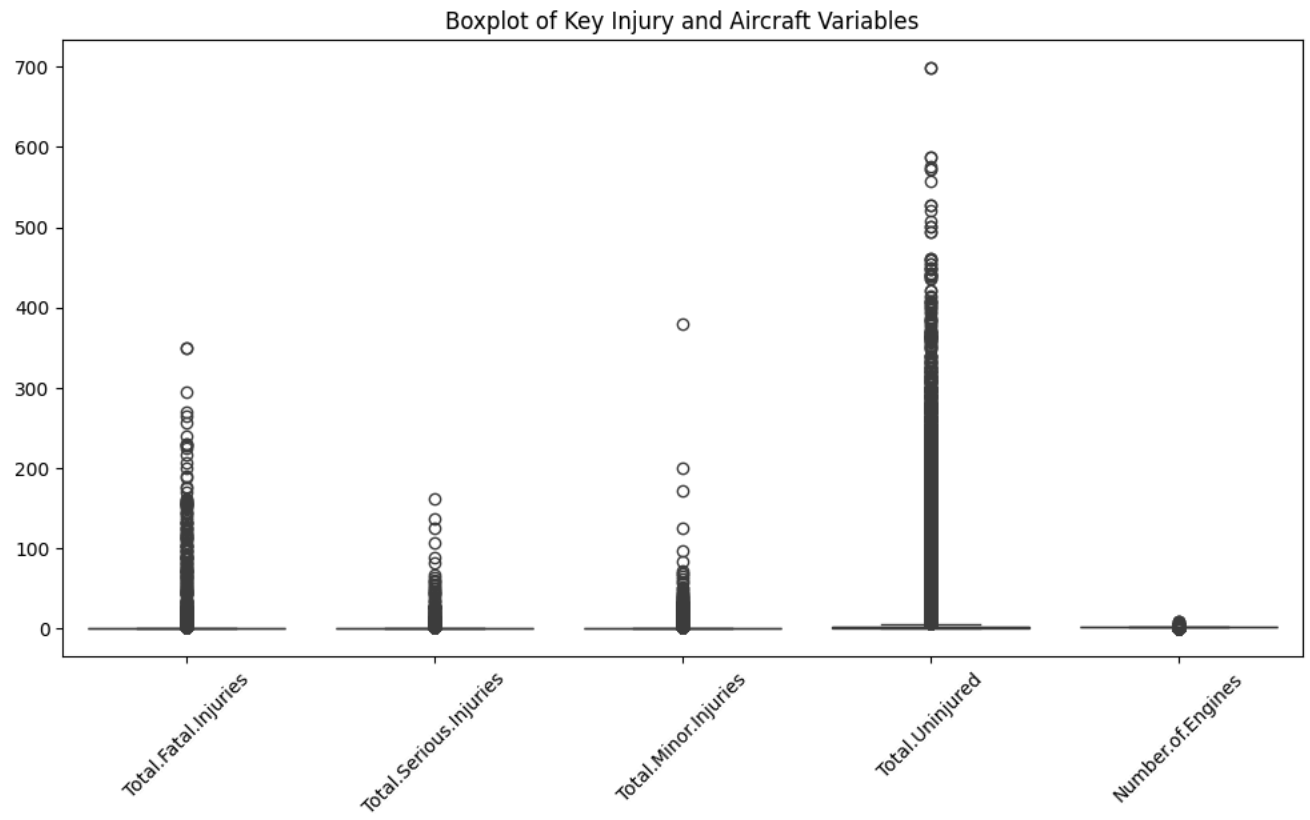
Year	count
2013	1561
2021	1545
2014	1535
2020	1392
1979	2
1974	1
1981	1
1962	1
1977	1
1948	1

dtype: int64

```
#Total accidents and incidents by year
df['Year'].value_counts().sort_index().plot(kind='bar', figsize=(15, 6))
plt.title('Total Accidents and Incidents by Year')
plt.xlabel('Year')
plt.ylabel('Number of Accidents and Incidents')
plt.xticks(rotation=45)
plt.show()
```




```
#boxplot of key injury and aircraft variables
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['Total.Fatal.Injuries', 'Total.Serious.Injuries',
                     'Total.Minor.Injuries', 'Total.Uninjured',
                     'Number.of.Engines']])
plt.xticks(rotation=45)
plt.title('Boxplot of Key Injury and Aircraft Variables')
plt.show()
```




```
#Handling skewed data
df['Total.Serious.Injuries'] = pd.to_numeric(df['Total.Serious.Injuries'], errors='coerce')
df['Total.Serious.Injuries'].fillna(0, inplace=True)
```

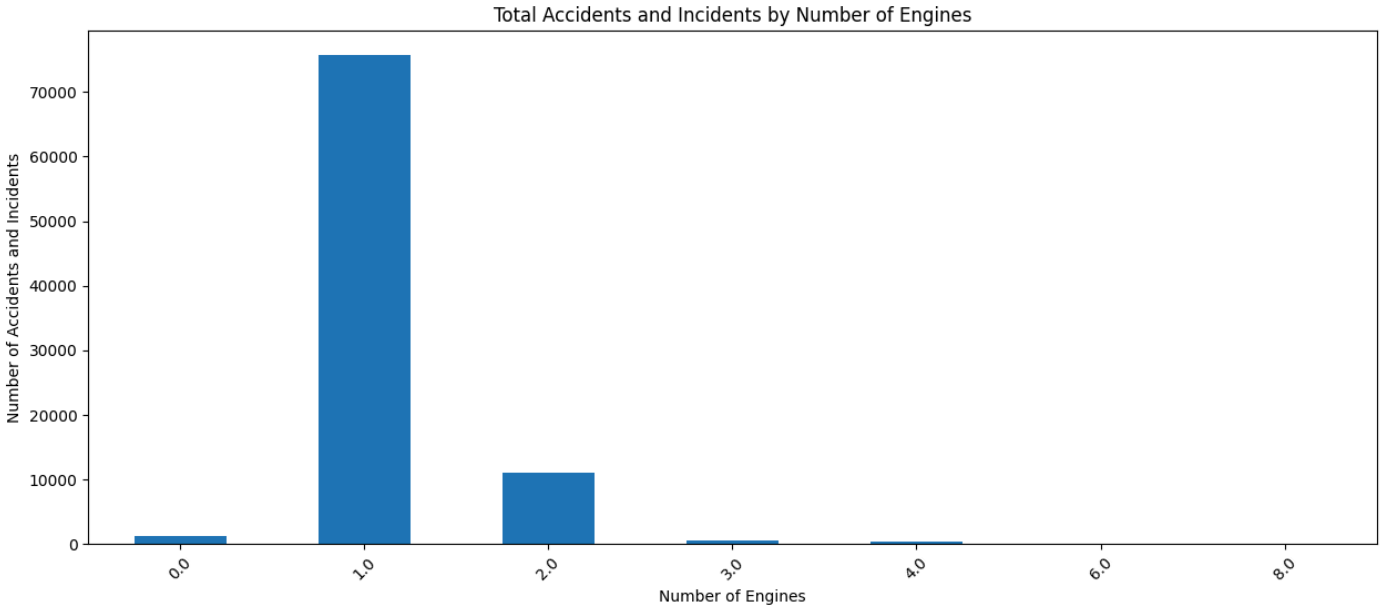
```
/tmp/ipython-input-93-3799509882.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Total.Serious.Injuries'].fillna(0, inplace=True)
```

```
#zero counts and skewness
for col in numerical_cols:
    zero_count = (df[col] == 0).sum()
    skew = df[col].skew()
    print(f"{col}: Zeros = {zero_count}, Skewness = {skew:.2f}")
```

```
Total.Fatal.Injuries: Zeros = 71076, Skewness = 35.32
Total.Serious.Injuries: Zeros = 75799, Skewness = 53.01
Total.Minor.Injuries: Zeros = 73387, Skewness = 93.38
Total.Uninjured: Zeros = 35791, Skewness = 9.41
Number.of.Engines: Zeros = 1226, Skewness = 2.71
```

Bivariate Aanalysis

```
# Total accidents by engine type and number of engines
df['Engine.Type'].value_counts().head(10) # Check the top 10 engine types with the most accidents and incidents
df['Number.of.Engines'].value_counts().head(10) # Check the top 10 engine counts with the most accidents and incidents
df['Number.of.Engines'].value_counts().tail(10) # Check the bottom 10 engine counts with the most accidents and incidents
df['Number.of.Engines'].value_counts().sort_index().plot(kind='bar', figsize=(15, 6))
plt.title('Total Accidents and Incidents by Number of Engines')
plt.xlabel('Number of Engines')
plt.ylabel('Number of Accidents and Incidents')
plt.xticks(rotation=45)
plt.show()
```



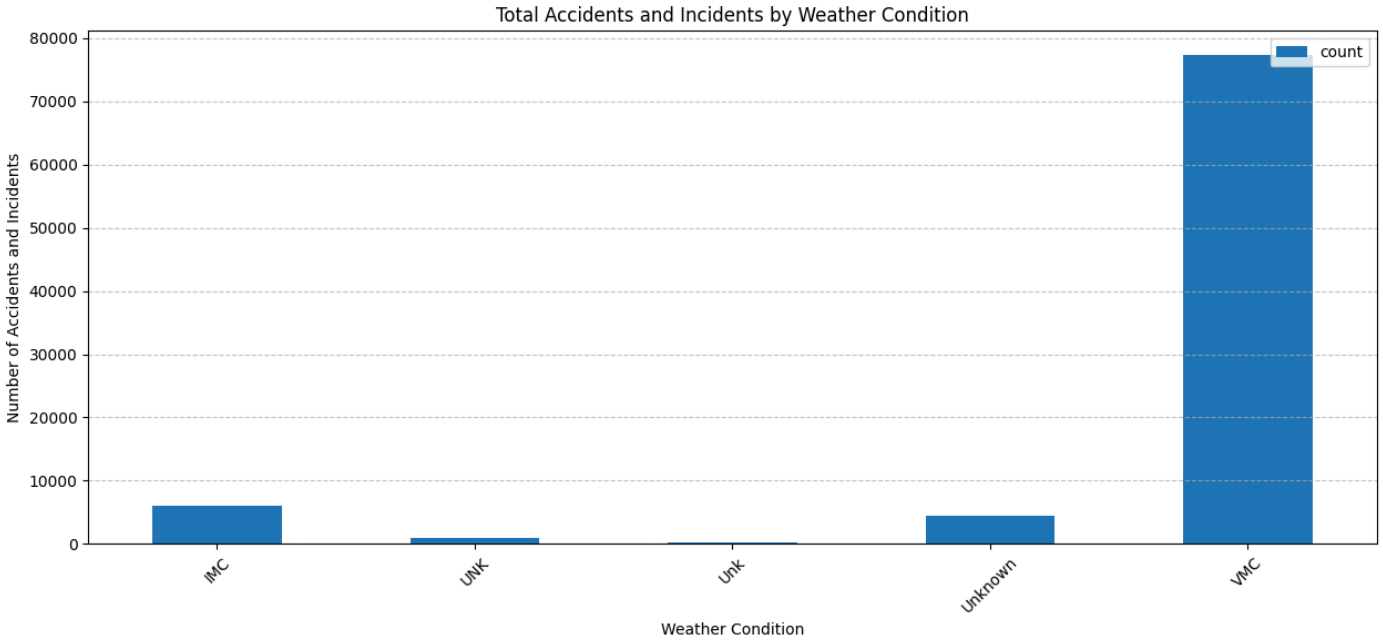
weather conditions which cause the most incidences

```
#Weather conditions analysis
df['Weather.Condition'].value_counts().head(10) # Check the top 10 weather conditions with the most accidents and incidents
df['Weather.Condition'].value_counts().tail(10) # Check the bottom 10 weather conditions with the most accidents and incidents
```

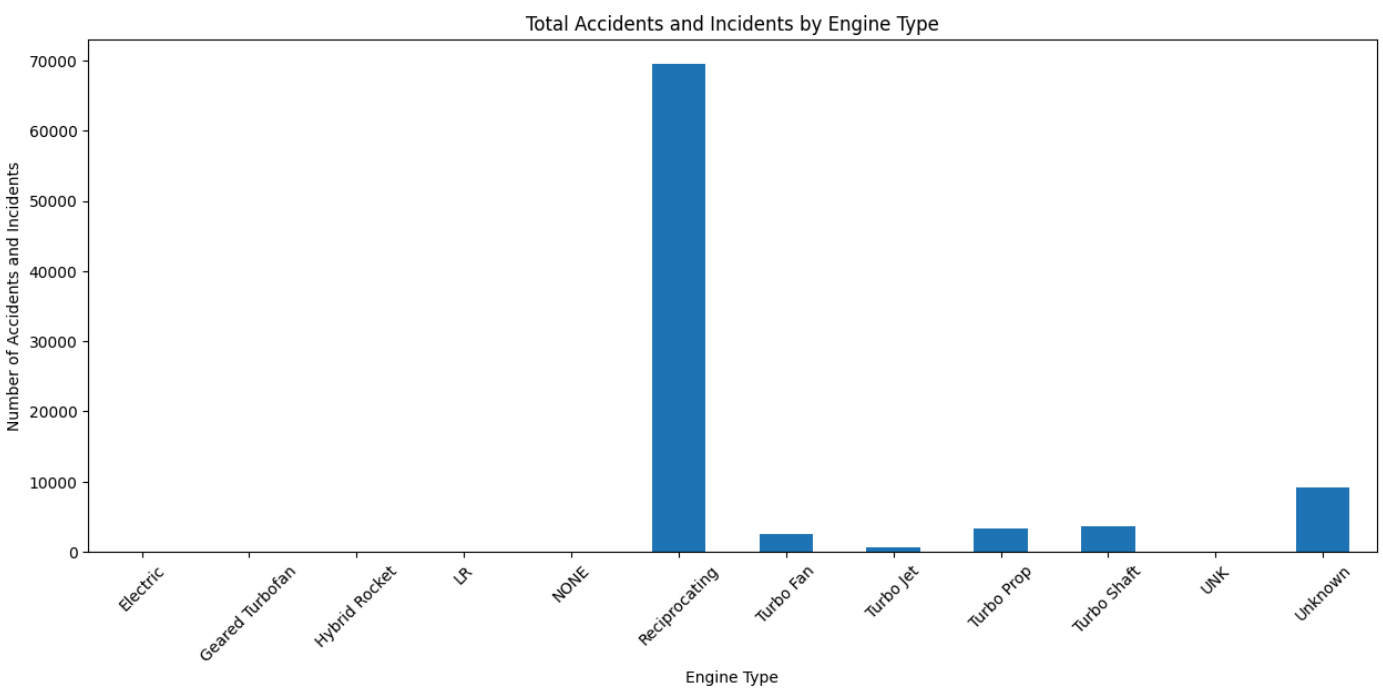
Weather.Condition	count
VMC	77303
IMC	5976
Unknown	4492
UNK	856
Unk	262

dtype: int64

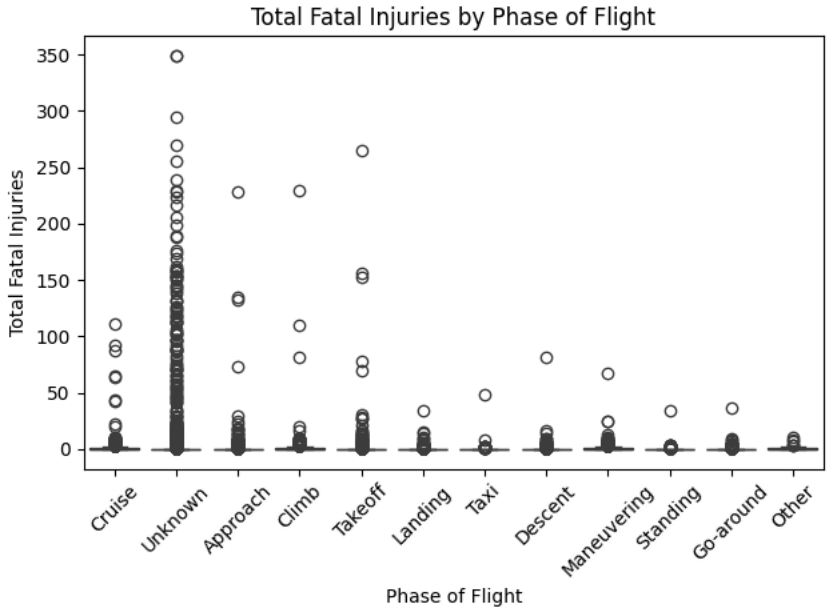
```
#Weather conditions and accidents
df['Weather.Condition'].value_counts().sort_index().plot(kind='bar', figsize=(15, 6))
plt.title('Total Accidents and Incidents by Weather Condition')
plt.xlabel('Weather Condition')
plt.ylabel('Number of Accidents and Incidents')
plt.xticks(rotation=45)
plt.legend (loc='upper right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



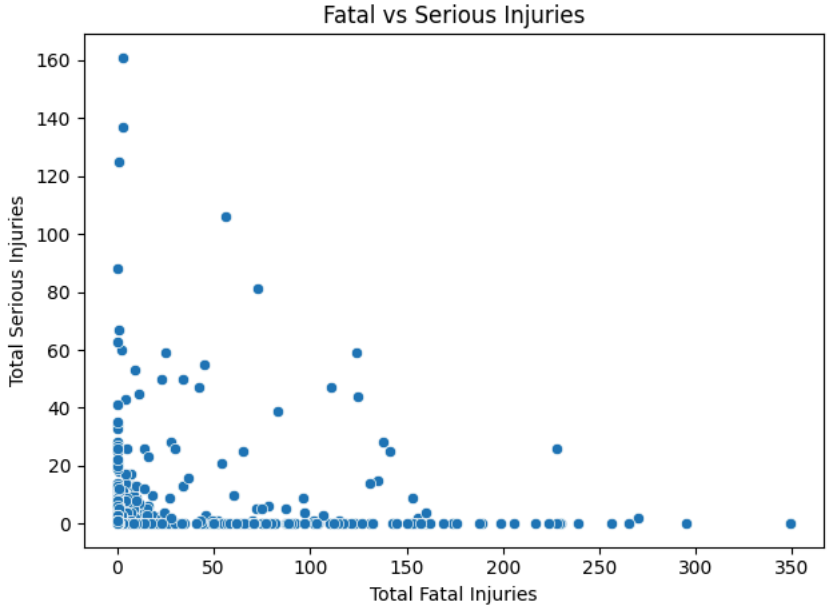

```
#Engine type and accidents
df['Engine.Type'].value_counts().sort_index().plot(kind='bar', figsize=(15, 6))
plt.title('Total Accidents and Incidents by Engine Type')
plt.xlabel('Engine Type')
plt.ylabel('Number of Accidents and Incidents')
plt.xticks(rotation=45)
plt.show()
```



```
#categorical vs numerical
sns.boxplot(x='Broad.phase.of.flight', y='Total.Fatal.Injuries', data=df)
plt.title("Total Fatal Injuries by Phase of Flight")
plt.xlabel("Phase of Flight")
plt.ylabel("Total Fatal Injuries")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

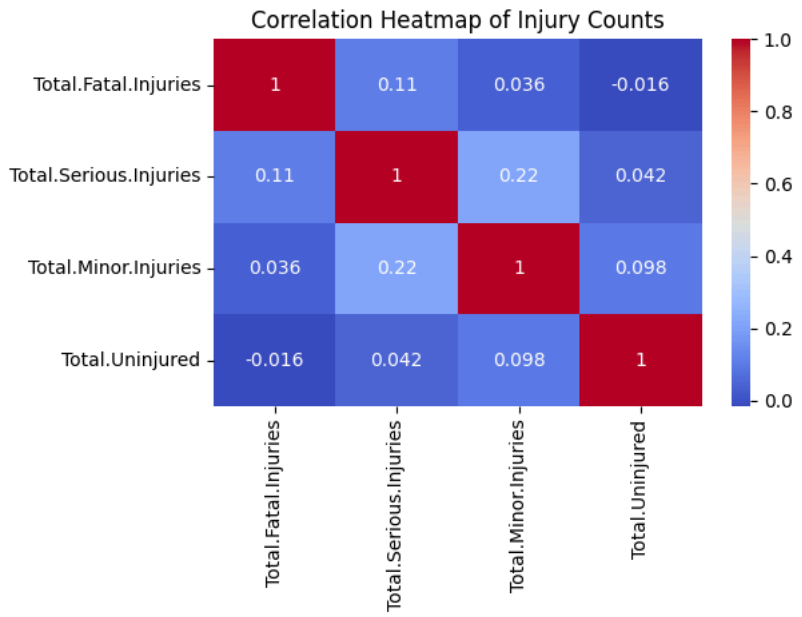



```
#umerical vs numerical
# Scatter plot between Serious and Fatal Injuries
sns.scatterplot(x='Total.Fatal.Injuries', y='Total.Serious.Injuries', data=df)
plt.title("Fatal vs Serious Injuries")
plt.xlabel("Total Fatal Injuries")
plt.ylabel("Total Serious Injuries")
plt.tight_layout()
plt.show()
```



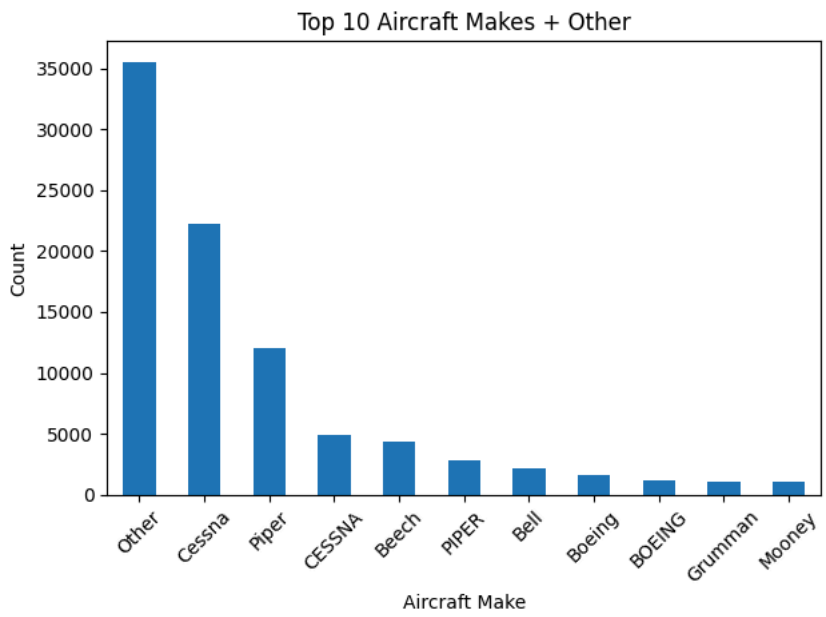

```
#correlation heatmap of injury columns
injury_cols = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']
corr = df[injury_cols].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap of Injury Counts")
```

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



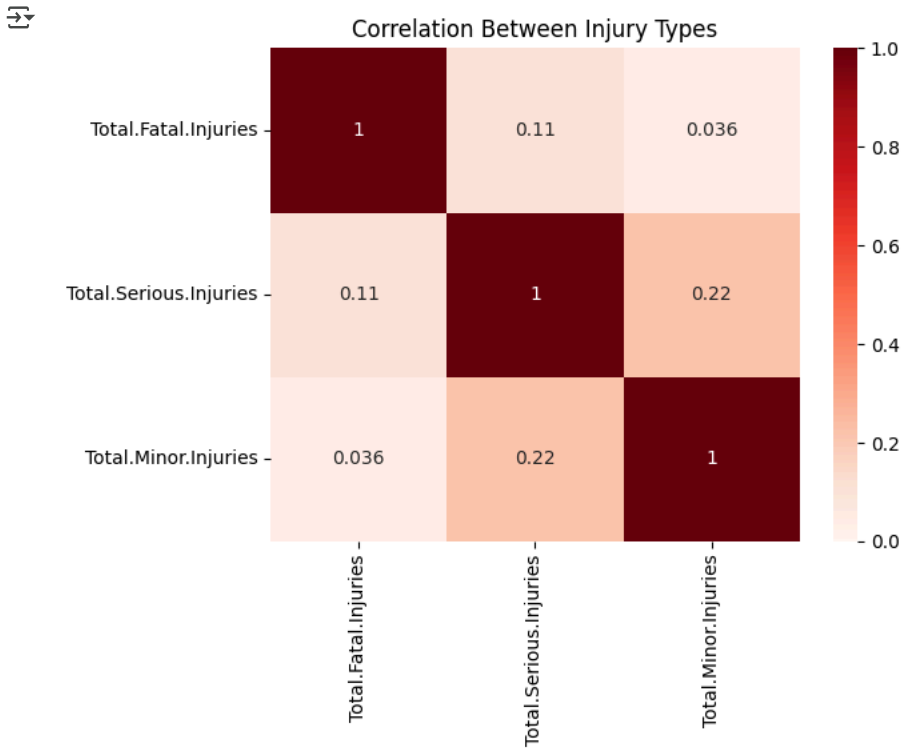
Mulltivariate Analysis

```
# Count the top 10 most common aircraft makes
top_10_makes = df['Make'].value_counts().nlargest(10).index

# Create a new column that groups lesser makes as 'Other'
df['Make_Grouped'] = df['Make'].apply(lambda x: x if x in top_10_makes else 'Other')


df['Make_Grouped'].value_counts().plot(kind='bar', title='Top 10 Aircraft Makes + Other')
plt.xlabel('Aircraft Make')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

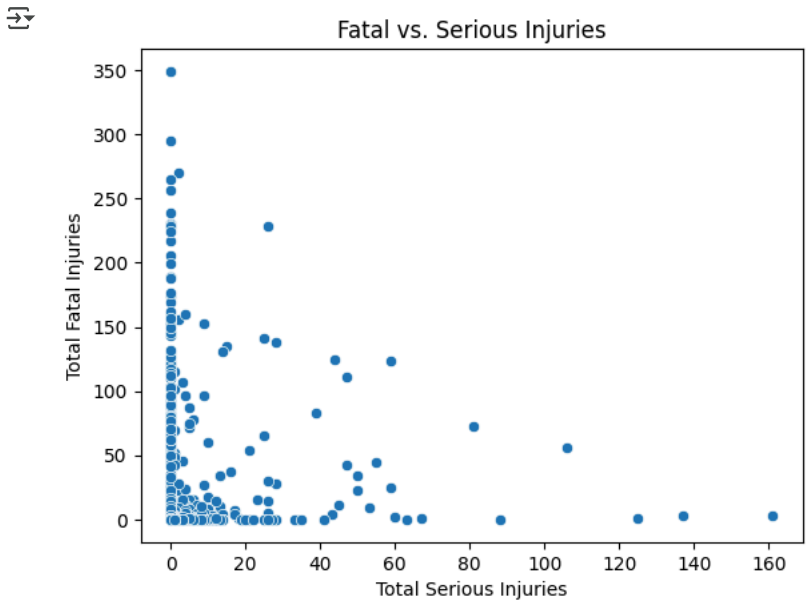



```
injury_vars = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries']

# Convert to numeric in case of issues
```

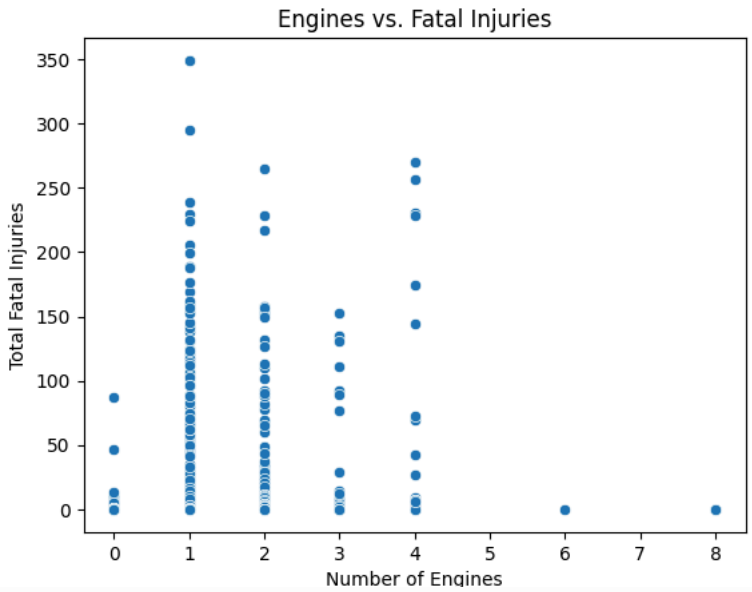
```
df[injury_vars] = df[injury_vars].apply(pd.to_numeric, errors='coerce')

# Correlation matrix and heatmap
corr = df[injury_vars].corr()
sns.heatmap(corr, annot=True, cmap='Reds', vmin=0, vmax=1)
plt.title('Correlation Between Injury Types')
plt.show()
```




```
sns.scatterplot(data=df, x='Total.Serious.Injuries', y='Total.Fatal.Injuries')
plt.title('Fatal vs. Serious Injuries')
plt.xlabel('Total Serious Injuries')
plt.ylabel('Total Fatal Injuries')
plt.show()
```




```
sns.scatterplot(data=df, x='Number.of.Engines', y='Total.Fatal.Injuries')
plt.title('Engines vs. Fatal Injuries')
plt.xlabel('Number of Engines')
plt.ylabel('Total Fatal Injuries')
plt.show()
```




```
# Create a binary column 'Is_Fatal'
df['Is_Fatal'] = df['Total.Fatal.Injuries'].fillna(0).apply(lambda x: 1 if x > 0 else 0)

# Optional: preview value counts to confirm
print(df['Is_Fatal'].value_counts())
```

```
Is_Fatal
0    71076
1    17813
Name: count, dtype: int64
```

```
df.groupby('Is_Fatal')[['Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].mean()
```

Is_Fatal	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
0	0.232681	0.337737	6.108222
1	0.271656	0.194970	0.434570

```
sns.countplot(data=df, x='Is_Fatal', palette='Set2')
plt.title('Count of Fatal vs Non-Fatal Accidents')
```