dsc-phase-1-project-v3 / student.ipynb

irushawn final project 638aead · 1 minute ago

2489 lines (2489 loc) · 307 KB

Final Project Submission

Please fill out:

- Student name: Shawn J Irungu
- Student pace: Hybrid
- Scheduled project review date/time:
- Instructor name: Samuel Karu
- Blog post URL:

PROJECT INTRODUCTION

The aviation industry is one of the most capital-intensive and highly regulated sectors, requiring strategic planning and data-driven decision-making. Airlines, charter companies, and new aviation startups face significant challenges when selecting aircraft for purchase or lease. These challenges include assessing safety records, operational risks, maintenance costs, and long-term profitability.

Many companies make investment decisions without fully understanding the historical performance and accident trends of different aircraft models, leading to financial losses, increased safety risks, and inefficient operations. This project aims to develop a data-driven aviation consulting framework that provides expert guidance to aviation companies before purchasing aircraft. By analyzing historical aviation data, accident trends, and operational metrics, the consulting service will help clients make informed aircraft acquisition decisions, minimizing risks and optimizing costs.

BUSINESS PROBLEM

The company is seeking to expand its portfolio by entering the aviation industry, with a focus on purchasing and operating aircraft for both commercial and private use. A key challenge is identifying aircraft models that present the lowest operational and safety risks. To make strategic and data-driven investment decisions, the company requires a comprehensive analysis of historical aviation data, accident trends, and maintenance records. This will ensure optimal aircraft selection, minimizing risks while maximizing efficiency and profitability in this new market segment.

MAIN OBJECTIVE

- To identify the safest and most reliable aircraft models for commercial and private operations, enabling the company to make data-driven investment decisions while minimizing operational risks and maximizing profitability.

SPECIFIC OBJECTIVE

- Which aircraft models have the lowest accident rates?
- What are the most common causes of aviation accidents?
- How does the number of engines affect accident frequency?
- What is the relationship between number of engines and accident rates?

Libraries Importation

In [64]:

```
# Import Libraries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
import datetime
```

Loading our Data Set - Aviation Dataset

In [65]:

```
df = pd.read_csv('./data/AviationData.csv', encoding='ISO-8859-1', low_memory=
df.head()
```

Out[65]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Co
0	20001218X45444	Accident	SEA87LA080	10/24/1948	MOOSE CREEK, ID	U
1	20001218X45447	Accident	LAX94LA336	7/19/1962	BRIDGEPORT, CA	U
2	20061025X01555	Accident	NYC07LA005	8/30/1974	Saltville, VA	U
3	20001218X45448	Accident	LAX96LA321	6/19/1977	EUREKA, CA	U
4	20041105X01764	Accident	CHI79FA064	8/2/1979	Canton, OH	U

5 rows × 31 columns

Data Wrangling Process

In [66]:

```
# check for duplicates
df.duplicated().value_counts()

# this returns a true of 1390 . meaning we have 1390 duplicated rows
```

Out[66]:

```
False    88889
dtype: int64
```

In [67]:

```
# check for shape
df.shape

# this shows that our df has 90348 rows(including the dupliacted) and 31 colum
```

Out[67]: (88889, 31)

In [68]:

```
# check information
df.info()

# this shows the data types and also columns that don`t count to 90348
# indicates that they contain missing values
# also shows that we need to change dtypes of some columns
```

```
<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            50249 non-null  object
 9   Airport.Name            52790 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     87572 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             81812 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           82508 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [69]:

```
# print the column names
df.columns
```

Out[69]:
```
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 [70]:
```
# show the summary statistics
df.describe()
```

Out[70]:

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

Data Cleaning

In [71]:
```
# drop duplicated rows
df.drop_duplicates(inplace=True)
```

In [72]:
```
df.shape
```

Out[72]: (88889, 31)

In [73]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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            50249 non-null  object
 9   Airport.Name            52790 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     87572 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             81812 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           82508 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.7+ MB
```

This to Note

1.We note that in our data the data type for Event.Date is an object instead of date.

2.There are several missing values in some columns

In [74]:

```
# DROPING COLUMNS
# dropping of unnecessary columns for our analysis
# first create a list of the columns we are interested in

c = ['Event.Id', 'Make', 'Model', 'Accident.Number', 'Total.Fatal.Injuries','F
 'Broad.phase.of.flight', 'Number.of.Engines', 'Accident.Number', 'Total.Fatal
  'Aircraft.Category', 'Accident.Number', 'Total.Fatal.Injuries']
c=set(c)
columns_to_keep=list(c)
print(columns_to_keep)
type(columns_to_keep)
```

dsc-phase-1-project-v3 / student.ipynb ↑ Top

Preview Code Blame Raw

In [75]:

```
# pass the list to the dataframe
df=df[columns_to_keep]
df.set_index(('Event.Id'))
```

Out[75]: Make FAR.Description Accident.Number Broad.phase.of.flight

Out[75]:

	Make	FAR.Description	Accident.Number	Broad.phase.of.flight
Event.Id				
20001218X45444	Stinson	NaN	SEA87LA080	Cruise
20001218X45447	Piper	NaN	LAX94LA336	Unknown
20061025X01555	Cessna	NaN	NYC07LA005	Cruise
20001218X45448	Rockwell	NaN	LAX96LA321	Cruise
20041105X01764	Cessna	NaN	CHI79FA064	Approach
...	...	...	...	...
2.02212E+13	PIPER	91	ERA23LA093	NaN
2.02212E+13	BELLANCA	NaN	ERA23LA095	NaN
2.02212E+13	AMERICAN CHAMPION AIRCRAFT	91	WPR23LA075	NaN
2.02212E+13	CESSNA	91	WPR23LA076	NaN
2.02212E+13	PIPER	91	ERA23LA097	NaN

88889 rows × 8 columns

In [76]:

```
# using list comprehension
# this is an alternative way to drop columns
# df = df.drop(columns=[col for col in df.columns if col not in columns_to_kee
```

In [77]:

```
df.shape
```

Out[77]: (88889, 9)

In [78]:

```
# Look number of missing values per column.
df.isna().sum()
```

Out[78]:

```
Event.Id                     0
Make                        63
FAR.Description          56866
Accident.Number              0
Broad.phase.of.flight    27165
Model                       92
Number.of.Engines         6084
Aircraft.Category        56602
Total.Fatal.Injuries     11401
dtype: int64
```

Checking for Data Completness

In [79]:
```
categorical_columns = df.select_dtypes(include=['object']).columns

for column in categorical_columns:
    print(column , df[column].nunique())
```

```
Event.Id 84468
Make 8237
FAR.Description 31
Accident.Number 88863
Broad.phase.of.flight 12
Model 12315
Aircraft.Category 15
```

This tells that Event Id has duplicated Values since unique counts=87951 while our df rows = 88889

In [80]:
```
# Check for duplicates in the column Event ID
df.duplicated(subset='Event.Id').sum()
```

Out[80]: 4421

In [81]:
```
#Drop the Duplicates
df.drop_duplicates(subset='Event.Id',inplace=True)
```

In [82]:
```
# Recheck Again
df.duplicated(subset='Event.Id').sum()
```

Out[82]: 0

Dealing with Misiing Values

In [83]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84468 entries, 0 to 88377
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Event.Id               84468 non-null  object 
 1   Make                   84405 non-null  object 
 2   FAR.Description        28516 non-null  object 
 3   Accident.Number        84468 non-null  object 
 4   Broad.phase.of.flight  60837 non-null  object 
 5   Model                  84376 non-null  object 
 6   Number.of.Engines      79193 non-null  float64
 7   Aircraft.Category      28884 non-null  object 
 8   Total.Fatal.Injuries   73201 non-null  float64
dtypes: float64(2), object(7)
memory usage: 6.4+ MB
```

In [84]:

```
# Standardize and Filling Missing Values
df['Make'] = df['Make'].str.strip().str.title().fillna('unknown')
df['Model'] = df['Model'].str.strip().str.title().fillna('unknown')
```

In [85]:

```
# Dealing with missing Values
df.fillna({'Aircraft.Category': 'Unknown'}, inplace=True)
df.fillna({'Total.Fatal.Injuries': df['Total.Fatal.Injuries'].mean() }, inplac
df.fillna({'Number.of.Engines': df['Number.of.Engines'].median() }, inplace=Tr
df.fillna({'Broad.phase.of.flight': 'Unknown'},inplace = True)
df.fillna({'FAR.Description': 'Unknown' }, inplace=True)
```

The FAR Description appears to 2 differentvalues that appear to mean one cause of accidents, which has some prefixes that are meaningless. So Lets rename them to Appropriate description to make it easier for us to understand.

In [86]:

```
df['FAR.Description'].replace({
    "Part 91: General Aviation": "General Aviation",
    "91": "General Aviation",
    "Part 135: Air Taxi & Commuter": "Air Taxi & Commuter",
    "Part 135": "Air Taxi & Commuter",
    "135": "Air Taxi & Commuter",
    "Part 125: 20+ Pax,6000+ lbs": "20+ Pax,6000+ lbs",
    "Part 125": "20+ Pax,6000+ lbs",
    "125": "20+ Pax,6000+ lbs",
    "103": "20+ Pax,6000+ lbs",
    "107": "20+ Pax,6000+ lbs",
    "129": "Foreign",
    "Part 129: Foreign": "Foreign",
    "Part 129": "Foreign",
    "Part 133: Rotorcraft Ext. Load": "Rotorcraft Ext. Load",
    "Part 133": "Rotorcraft Ext. Load",
    "133": "Rotorcraft Ext. Load",
    "Part 121: Air Carrier": "Air Carrier",
    "Part 121": "Air Carrier",
    "121": "Air Carrier",
    "Part 137: Agricultural": "Agricultural",
    "137": "Agricultural",
    "Part 137": "Agricultural",
    "Part 91 Subpart K: Fractional": "Subpart K: Fractional",
    "Part 91F: Special Flt Ops.": "Special Flt Ops",
    "091K": "Special Flt Ops",
    "437": "Special Flt Ops",
    "UNK": "Unknown",
    "Non-U.S., Commercial": "Commercial",
    "Non-U.S., Non-Commercial": "Non-Commercial",
}, inplace=True)
```

In [87]:

```
# Verify no missing values remain
print("Missing Values After Handling:\n", df.isnull().sum())
```

```
Missing Values After Handling:
 Event.Id                0
Make                    0
FAR.Description         0
```

```
Accident.Number          0
Broad.phase.of.flight    0
Model                    0
Number.of.Engines        0
Aircraft.Category        0
Total.Fatal.Injuries     0
dtype: int64
```

Check for Extraneous Value

In [88]:

```python
for col in df.columns:
    print(col, '\n', df[col].value_counts(), '\n')
```

```
Event.Id
 20001211X11226    1
20001214X40034    1
20001213X26015    1
20001214X37058    1
20001212X22874    1
                  ..
20001213X26042    1
20001208X09208    1
20020917X01764    1
20001213X33237    1
20001213X32670    1
Name: Event.Id, Length: 84468, dtype: int64

Make
 Cessna                  25987
Piper                   14254
Beech                    5165
Bell                     2606
Boeing                   2461
                         ...
Griner                      1
Williams Helicopter         1
Larry Schindler             1
Robinson Helicopters        1
Giackino Donald W           1
Name: Make, Length: 7188, dtype: int64

FAR.Description
 Unknown                 56273
General Aviation        22282
Agricultural             1330
NUSN                     1182
Air Taxi & Commuter       939
NUSC                      833
Air Carrier               758
Foreign                   279
PUBU                      225
Rotorcraft Ext. Load      123
Non-Commercial             96
Commercial                 91
Public Use                 19
Special Flt Ops            14
20+ Pax,6000+ lbs          14
ARMF                        7
Public Aircraft             2
```

```
Armed Forces                1
Name: FAR.Description, dtype: int64

Accident.Number
 MIA93LA017    1
NYC99WA006    1
SEA86LA088    1
FTW96FA075    1
LAX84LA205    1
             ..
NYC00LA031    1
LAX95LA179    1
DCA07WA011    1
WPR20FA031    1
LAX92LA340    1
Name: Accident.Number, Length: 84468, dtype: int64

Broad.phase.of.flight
 Unknown        24178
Landing        15320
Takeoff        12404
Cruise         10141
Maneuvering     8052
Approach        6389
Climb           1995
Descent         1870
Taxi            1786
Go-around       1345
Standing         872
Other            116
Name: Broad.phase.of.flight, dtype: int64

Model
 152             2283
172             1627
172N            1121
Pa-28-140        893
150              792
                ...
Cl-65              1
Fa150              1
Eaa Special        1
Airbike Rx40       1
Cwii               1
Name: Model, Length: 11282, dtype: int64

Number.of.Engines
 1.0    71893
2.0    10565
0.0     1153
3.0      446
4.0      408
8.0        3
Name: Number.of.Engines, dtype: int64

Aircraft.Category
 Unknown              55596
Airplane             24713
Helicopter            3062
Glider                 457
Balloon                209
```

```
Gyrocraft              154
Weight-Shift           150
Powered Parachute       87
Ultralight              30
Powered-Lift             5
Blimp                    4
Rocket                   1
Name: Aircraft.Category, dtype: int64

Total.Fatal.Injuries
 0.000000      56337
0.645169      11267
1.000000       8426
2.000000       4898
3.000000       1510
               ...
295.000000         1
37.000000          1
144.000000         1
112.000000         1
349.000000         1
Name: Total.Fatal.Injuries, Length: 126, dtype: int64
```

By checking the extraneous value we see that there are 1210 planes with 0 number of engines. This is IMPOSIIBLE. We need to replace this with mean

In [89]:

```
df['Number.of.Engines']=df['Number.of.Engines'].replace(0,df['Number.of.Engine
```

In [90]:

```
#check for extraneous value again and you will see now our colum is clean with
for col in df.columns:
    print(col, '\n', df[col].value_counts(), '\n')
```

```
Event.Id
 20001211X11226    1
20001214X40034    1
20001213X26015    1
20001214X37058    1
20001212X22874    1
                 ..
20001213X26042    1
20001208X09208    1
20020917X01764    1
20001213X33237    1
20001213X32670    1
Name: Event.Id, Length: 84468, dtype: int64

Make
 Cessna                  25987
Piper                   14254
Beech                    5165
Bell                     2606
Boeing                   2461
                         ...
Griner                      1
```

```
Williams Helicopter          1
Larry Schindler              1
Robinson Helicopters         1
Giackino Donald W            1
Name: Make, Length: 7188, dtype: int64

FAR.Description
 Unknown                  56273
General Aviation        22282
Agricultural             1330
NUSN                     1182
Air Taxi & Commuter       939
NUSC                      833
Air Carrier               758
Foreign                   279
PUBU                      225
Rotorcraft Ext. Load      123
Non-Commercial             96
Commercial                 91
Public Use                 19
Special Flt Ops            14
20+ Pax,6000+ lbs          14
ARMF                        7
Public Aircraft             2
Armed Forces                1
Name: FAR.Description, dtype: int64

Accident.Number
 MIA93LA017    1
NYC99WA006    1
SEA86LA088    1
FTW96FA075    1
LAX84LA205    1
             ..
NYC00LA031    1
LAX95LA179    1
DCA07WA011    1
WPR20FA031    1
LAX92LA340    1
Name: Accident.Number, Length: 84468, dtype: int64

Broad.phase.of.flight
 Unknown        24178
Landing        15320
Takeoff        12404
Cruise         10141
Maneuvering     8052
Approach        6389
Climb           1995
Descent         1870
Taxi            1786
Go-around       1345
Standing         872
Other            116
Name: Broad.phase.of.flight, dtype: int64

Model
 152             2283
172             1627
172N            1121
Pa-28-140        893
```

```
150              792
                ...
Cl-65              1
Fa150              1
Eaa Special        1
Airbike Rx40       1
Cwii               1
Name: Model, Length: 11282, dtype: int64

Number.of.Engines
 1.000000    71893
2.000000    10565
1.136726     1153
3.000000      446
4.000000      408
8.000000        3
Name: Number.of.Engines, dtype: int64

Aircraft.Category
 Unknown              55596
Airplane             24713
Helicopter            3062
Glider                 457
Balloon                209
Gyrocraft              154
Weight-Shift           150
Powered Parachute       87
Ultralight              30
Powered-Lift             5
Blimp                    4
Rocket                   1
Name: Aircraft.Category, dtype: int64

Total.Fatal.Injuries
 0.000000      56337
0.645169      11267
1.000000       8426
2.000000       4898
3.000000       1510
              ...
295.000000        1
37.000000         1
144.000000        1
112.000000        1
349.000000        1
Name: Total.Fatal.Injuries, Length: 126, dtype: int64
```

Data Type Conversion

In [91]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84468 entries, 0 to 88377
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                84468 non-null  object
```

```
 1   Make                   84468 non-null  object
 2   FAR.Description        84468 non-null  object
 3   Accident.Number        84468 non-null  object
 4   Broad.phase.of.flight  84468 non-null  object
 5   Model                  84468 non-null  object
 6   Number.of.Engines      84468 non-null  float64
 7   Aircraft.Category      84468 non-null  object
 8   Total.Fatal.Injuries   84468 non-null  float64
dtypes: float64(2), object(7)
memory usage: 6.4+ MB
```

In [92]:

```
# Convert categorical columns to category dtype
categorical_columns = ['Make', 'Aircraft.Category', 'Broad.phase.of.flight', '
                       'Accident.Number', 'Event.Id', 'FAR.Description']
for column in categorical_columns:
    df[column] = df[column].astype('category').str.strip().str.title()

# Convert Number.of.Engines to integer
df['Number.of.Engines'] = df['Number.of.Engines'].astype(int)

# Convert Total Fatal Injuries to integer
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].astype(int)
```

In [93]:

```
# Run to confirm dtype have been chaged to Category dtype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84468 entries, 0 to 88377
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Event.Id               84468 non-null  object
 1   Make                   84468 non-null  object
 2   FAR.Description        84468 non-null  object
 3   Accident.Number        84468 non-null  object
 4   Broad.phase.of.flight  84468 non-null  object
 5   Model                  84468 non-null  object
 6   Number.of.Engines      84468 non-null  int32
 7   Aircraft.Category      84468 non-null  object
 8   Total.Fatal.Injuries   84468 non-null  int32
dtypes: int32(2), object(7)
memory usage: 5.8+ MB
```

In [94]:

```
# Statistics Summary of Numerical Columns
numerical_columns = ['Total.Fatal.Injuries', 'Number.of.Engines']
print("Numerical Data Description:\n", df[numerical_columns].describe())
```

```
Numerical Data Description:
        Total.Fatal.Injuries  Number.of.Engines
count          84468.000000       84468.000000
mean               0.559111           1.150376
std                5.029975           0.410851
min                0.000000           1.000000
25%                0.000000           1.000000
50%                0.000000           1.000000
75%                0.000000           1.000000
max              349.000000           8.000000
```

In [95]:
```
# VERIFY NO DUPLICATES
duplicate_rows = df.duplicated().sum()
print("Duplicate Rows:\n", duplicate_rows)
```

```
Duplicate Rows:
 0
```

Export Cleaned CSV

In [96]:
```
AvCleaned = df.to_csv('./data/AVCleaned.csv')
```

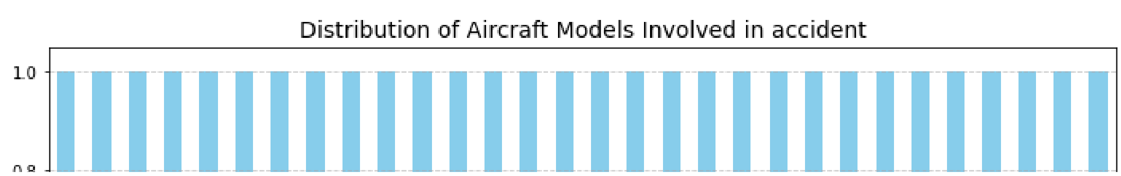
EDA

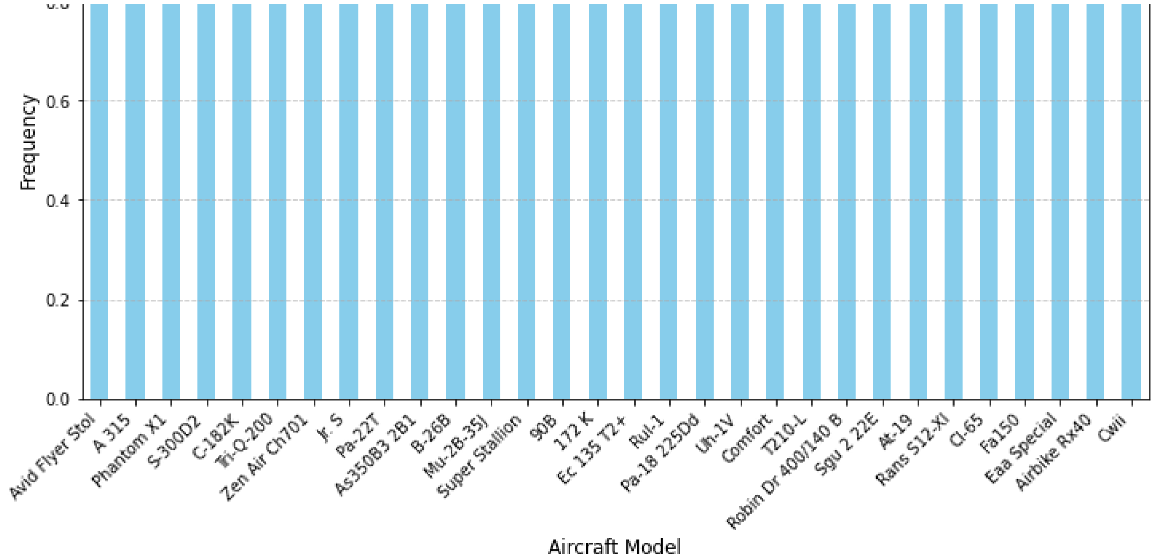
Distribution of Aircraft Model

In [97]:
```
df['Model'].value_counts().iloc[0:]
```

Out[97]:
```
152             2283
172             1627
172N            1121
Pa-28-140        893
150              792
                ...
Cl-65              1
Fa150              1
Eaa Special        1
Airbike Rx40       1
Cwii               1
Name: Model, Length: 11281, dtype: int64
```

In [98]:
```
# Plot distribution of aircraft model
# Count occurrences of each aircraft model
model_counts = df['Model'].value_counts().tail(30)  # Top 15 models with most

# Plot the distribution
plt.figure(figsize=(12, 6))
model_counts.plot(kind='bar', color='skyblue')

# Customize the plot
plt.title("Distribution of Aircraft Models Involved in accident", fontsize=14)
plt.xlabel("Aircraft Model", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(rotation=45, ha='right')  # Rotate labels for readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```





Obj 1

Which aircraft models have the lowest accident rates?

This Distribution Indicates that the Model with low Frequency has low Accident Rates: THis are :

In [99]:
```
models_with_one_occurrence = model_counts[model_counts == 1].index.tolist()
models_with_one_occurrence
```

Out[99]:
```
['Avid Flyer Stol',
 'A 315',
 'Phantom X1',
 'S-300D2',
 'C-182K',
 'Tri-Q-200',
 'Zen Air Ch701',
 'Jr. S',
 'Pa-22T',
 'As350B3 2B1',
 'B-26B',
 'Mu-2B-35J',
 'Super Stallion',
 '90B',
 '172 K',
 'Ec 135 T2+',
 'Rul-1',
 'Pa-18 225Dd',
 'Uh-1V',
 'Comfort',
 'T210-L',
 'Robin Dr 400/140 B',
 'Sgu 2 22E',
 'At-19',
 'Rans S12-Xl',
 'Cl-65',
 'Fa150',
 'Eaa Special',
 'Airbike Rx40'
```

```
 'Airbike Rx40',
 'Cwii']
```

In [100…

```
df['Make'].value_counts()
```

Out[100…

```
Cessna                     25987
Piper                      14254
Beech                       5165
Bell                        2606
Boeing                      2461
                           ...
Griner                         1
Williams Helicopter            1
Larry Schindler                1
Robinson Helicopters           1
Giackino Donald W              1
Name: Make, Length: 7187, dtype: int64
```

In [101…

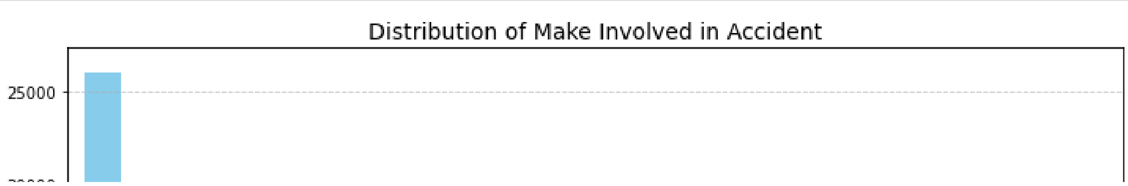
```
df['Make'].value_counts()
```

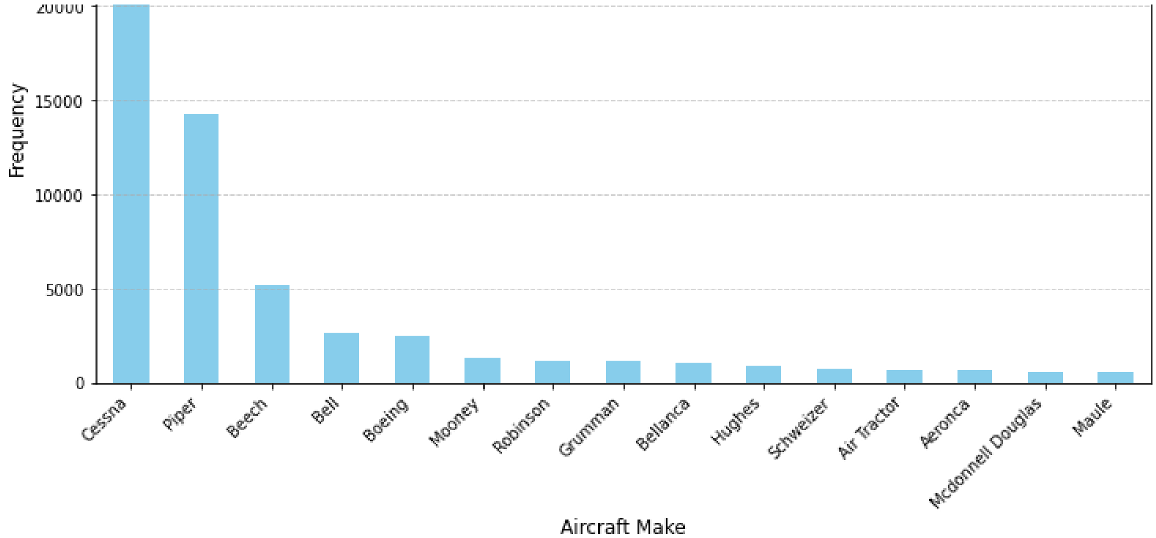
Out[101…

```
Cessna                     25987
Piper                      14254
Beech                       5165
Bell                        2606
Boeing                      2461
                           ...
Griner                         1
Williams Helicopter            1
Larry Schindler                1
Robinson Helicopters           1
Giackino Donald W              1
Name: Make, Length: 7187, dtype: int64
```

In [102…

```
# Plot distribution of aircraft make
# Count occurrences of each aircraft make
make_all_count = df['Make'].value_counts()
make_counts = df['Make'].value_counts().head(15)  # Top 15 make with most acci

# Plot the distribution
plt.figure(figsize=(12, 6))
make_counts.plot(kind='bar', color='skyblue')

# Customize the plot
plt.title("Distribution of Make Involved in Accident", fontsize=14)
plt.xlabel("Aircraft Make", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(rotation=45, ha='right')  # Rotate labels for readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```





In [103…

```
make_with_one_occurrence = make_all_count[make_all_count == 1].index.tolist()
make_with_one_occurrence[:5]
```

Out[103…

```
['Diemert/Rotorway',
 'Bosco Don',
 'Mason Robert',
 'Mcgrath Robert F',
 'Lampman']
```

In [104…

```
df.groupby(['Make', 'Model'])['Accident.Number'].count().sort_values()
```

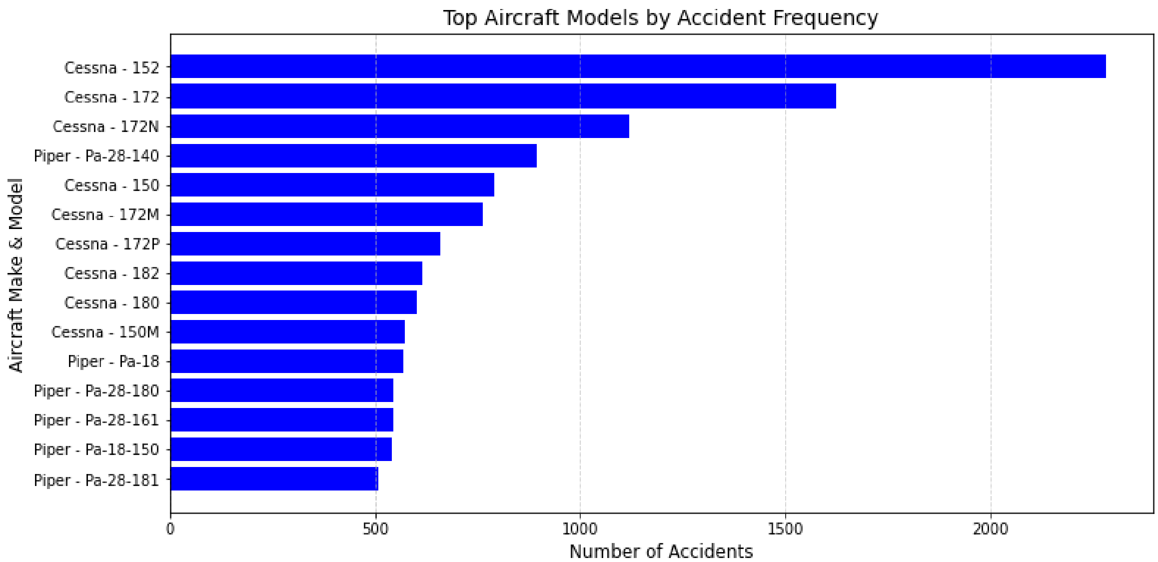
Out[104…

```
Make                       Model
107.5 Flying Corporation   One Design Dr 107       1
Maule                      M5-210Tc                1
                           M5C                     1
                           M6235                   1
                           M7-235                  1
                                                 ...
Cessna                     150                   792
Piper                      Pa-28-140             893
Cessna                     172N                 1120
                           172                  1625
                           152                  2282
Name: Accident.Number, Length: 17538, dtype: int64
```

In [105…

```
# Group by 'Make' and 'Model', then count accidents
accident_counts = df.groupby(['Make', 'Model'])['Accident.Number'].count().res

# Rename the count column for clarity
accident_counts.rename(columns={'Accident.Number': 'Accident_Count'}, inplace=

# Sort by accident count (highest first)
accident_counts = accident_counts.sort_values(by='Accident_Count', ascending=F

# Select top 15 (modify as needed)
top_accidents = accident_counts.head(15)

# Create bar chart
plt.figure(figsize=(12, 6))
plt.barh(top_accidents['Make'] + " - " + top_accidents['Model'], top_accidents
```

```
# Customize the plot
plt.title("Top Aircraft Models by Accident Frequency", fontsize=14)
plt.xlabel("Number of Accidents", fontsize=12)
plt.ylabel("Aircraft Make & Model", fontsize=12)
plt.gca().invert_yaxis()  # Invert y-axis to show highest count at the top
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



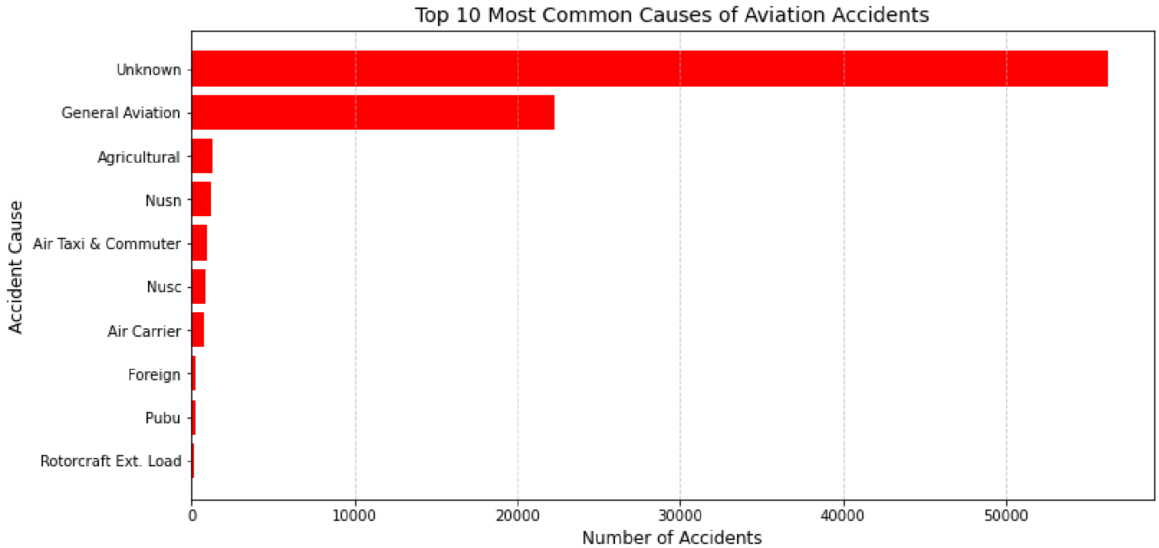
The above analysis shows the aircrafts and models with the highest frequency of Accidents. Thi are top 15 models corresponding to their makes. It appears Cessna and Piper Make have high chances of getting accidents.

OBJ 2

What are the most common causes of aviation accidents?

In [106…

```
cause_counts = df['FAR.Description'].value_counts().reset_index()
cause_counts.columns = ['Accident Cause', 'Count']

# Select top 10 most common causes
top_causes = cause_counts.head(10)

# Plot bar chart
plt.figure(figsize=(12, 6))
plt.barh(top_causes['Accident Cause'], top_causes['Count'], color='red')
plt.xlabel("Number of Accidents", fontsize=12)
plt.ylabel("Accident Cause", fontsize=12)
plt.title("Top 10 Most Common Causes of Aviation Accidents", fontsize=14)
plt.gca().invert_yaxis()  # Highest count at the top
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Show plot
plt.show()
```



The Graph shows from the known Cause of Accidents, General aviation leads followed by Agriculture. But the Most cause appears to be unknown.

Rotorcraft Ext Load appears to have few cases of accident

OBJ 3

How does the number of engines affect accident frequency and severity?

In [107…

```
df.groupby(['Number.of.Engines'])['Total.Fatal.Injuries'].count()
```

Out[107…

```
Number.of.Engines
1    73046
2    10565
3      446
4      408
8        3
Name: Total.Fatal.Injuries, dtype: int64
```

In [108…

```
# How often do accidents occur for different engine numbers?
# here we Count accidents per Number.of.Engines

engine_accident_counts = df['Number.of.Engines'].value_counts().sort_index()
engine_accident_counts
```

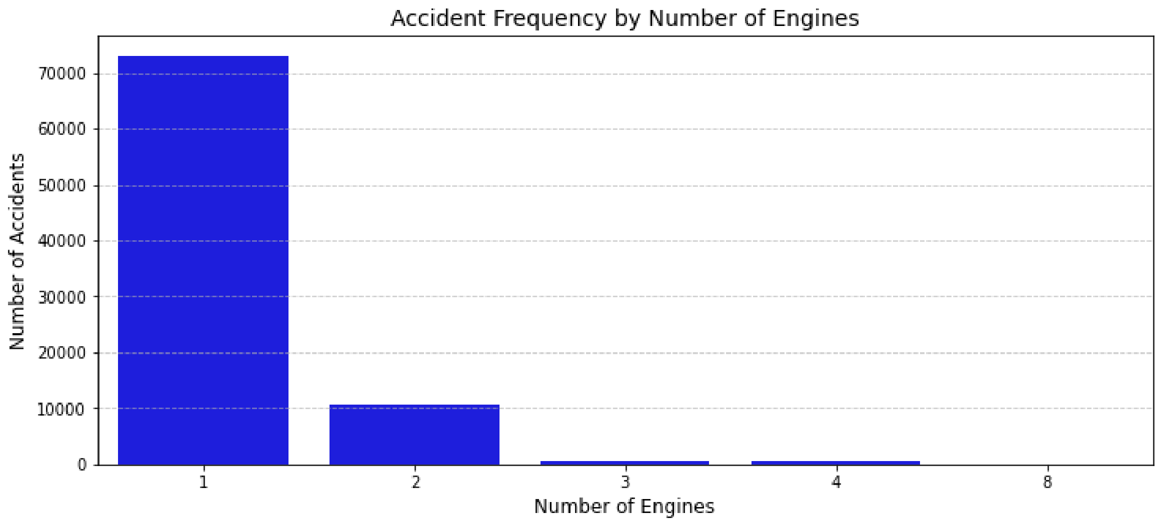
Out[108…

```
1    73046
2    10565
3      446
4      408
8        3
Name: Number.of.Engines, dtype: int64
```

In [109…

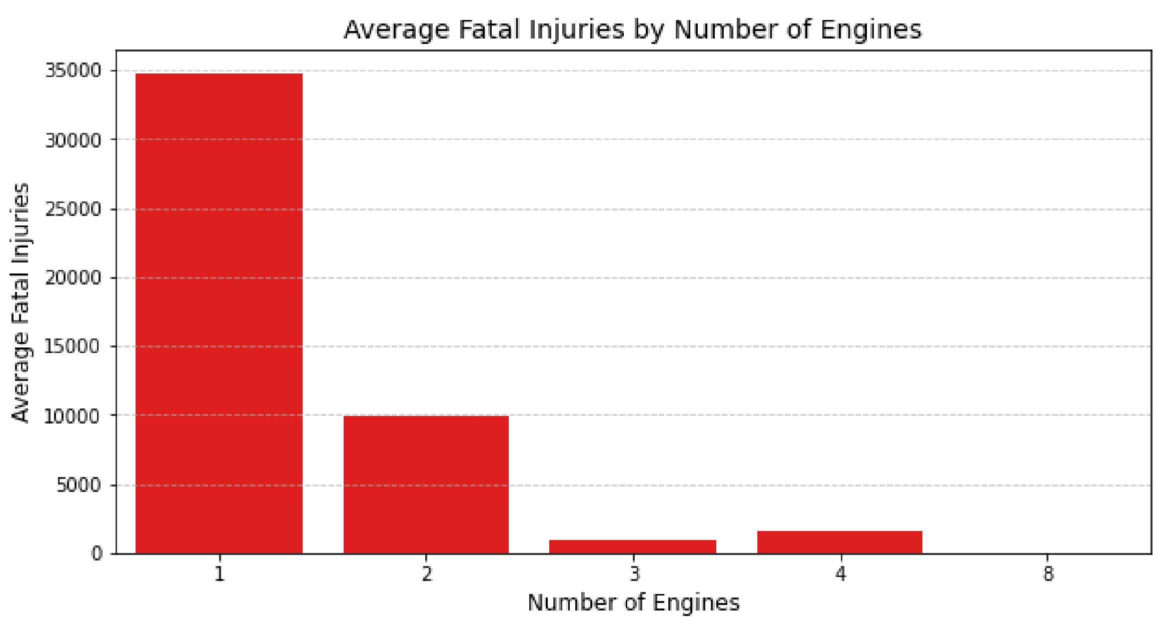
```
plt.figure(figsize=(12, 5))
sns.barplot(x=engine_accident_counts.index, y=engine_accident_counts.values, c
plt.xlabel("Number of Engines", fontsize=12)
plt.ylabel("Number of Accidents", fontsize=12)
plt.title("Accident Frequency by Number of Engines", fontsize=14)
```

```
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



In [110…

```
# Are accidents with more engines more severe?
# We Compare injury Total.Fatal.Injuries
# Analyze the average number of Total.Fatal.Injuries per engine type.

fatalities_per_engine = df.groupby('Number.of.Engines')['Total.Fatal.Injuries'
fatalities_per_engine
plt.figure(figsize=(10, 5))
sns.barplot(x=fatalities_per_engine['Number.of.Engines'], y=fatalities_per_eng
plt.xlabel("Number of Engines", fontsize=12)
plt.ylabel("Average Fatal Injuries", fontsize=12)
plt.title("Average Fatal Injuries by Number of Engines", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



Here we see that : Single-engine planes crash more often than multi-engine planes?

multi-engine plane accidents are less involved in accidents hence less Fatal injuries.

This will helps aviation companies decide to purchase aircrafts with multi engines since they reduce risk.

OBJ 4

What is the relationship between Number of Engines and Fatal Injuries?

In [111…

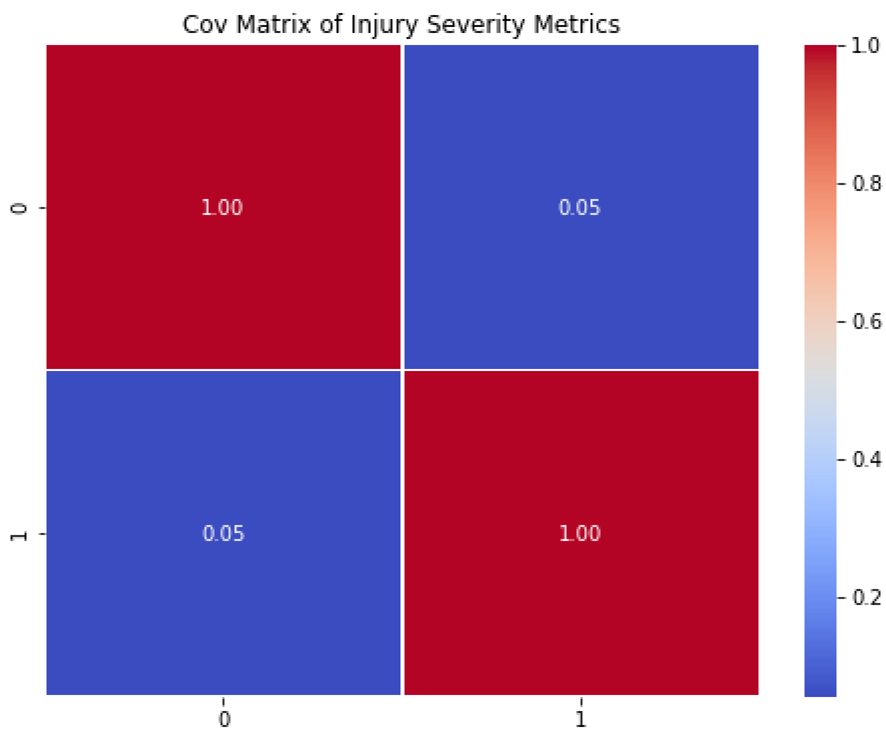
```
cov_matrix = np.corrcoef(df['Number.of.Engines'],df['Total.Fatal.Injuries'] )
cov_matrix[0][1]
```

Out[111…

```
0.05449318588777948
```

In [112…

```
# Heatmap of correlation
plt.figure(figsize=(8, 6))
sns.heatmap(cov_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5
plt.title("Cov Matrix of Injury Severity Metrics")
plt.show()
```



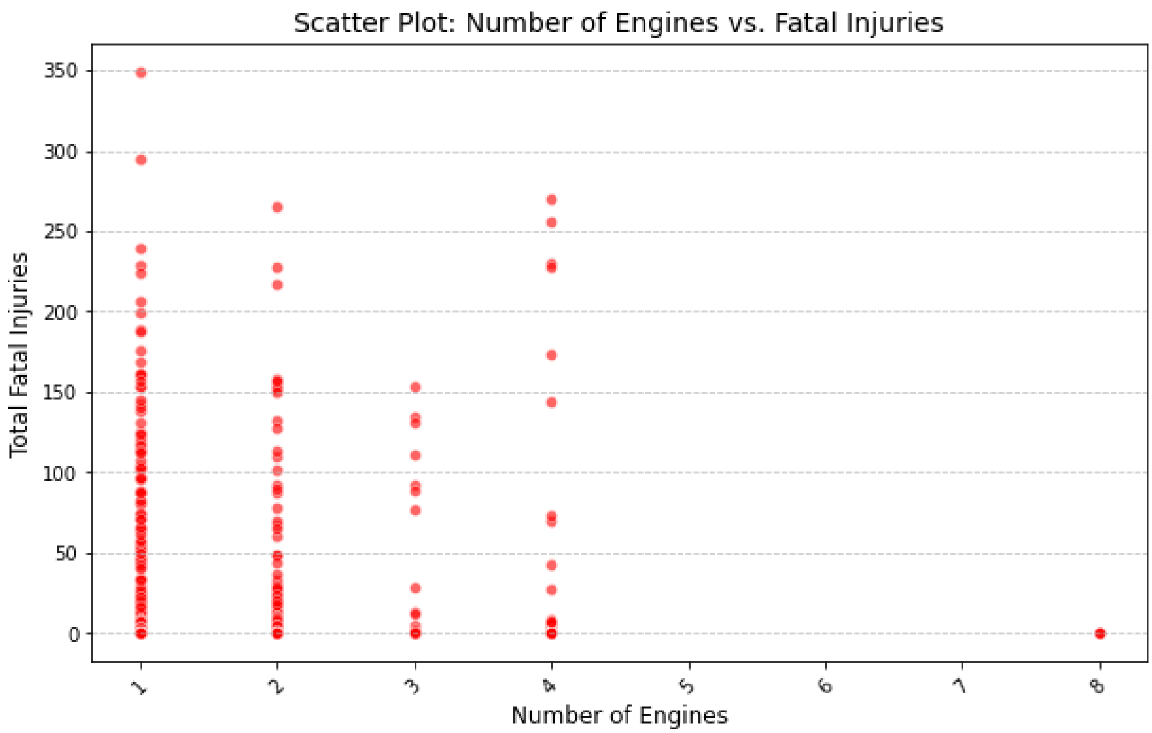
The correlation coefficient is (0.05449318588777948) which indicates a week positive coreelation. meaning there is a week positive relationship between the number of engines and the number of fatal injuries.

this is because if an aircraft has multiple engines, it might still operate after one fails, this could reduce the severity of crashes, making the correlation weak. And this was seen from the above analysis that when number of engines were high - the fatal injuries were low. we concluded that aircrafts with multi engines experience few accidents.

Other factors discussed above might be more important in determining accident severity.

In [113…

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Number.of.Engines', y='Total.Fatal.Injuries', alph
#sns.regplot(data=df, x='Number.of.Engines', y='Total.Fatal.Injuries', scatter_
plt.xlabel("Number of Engines", fontsize=12)
plt.ylabel("Total Fatal Injuries", fontsize=12)
plt.title("Scatter Plot: Number of Engines vs. Fatal Injuries", fontsize=14)
plt.xticks(rotation=45)  # Rotate category labels for better visibility
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()
```



CONCLUSION

Aircraft models with lower accident counts than others, indicating they may be safer or less frequently used.

Aircraft makes with the lowest accident counts indicate better safety records

Planes with Multi engine are best to purchase for our business

Recommendation

Conclusion The analysis of the aviation dataset has led to several key insights regarding the safety of different aircraft models, specifically focusing on airplanes and helicopters. By evaluating the uninjured outcomes, we can identify the safest models for both personal and commercial aviation purposes.

Key Findings

Top 10 Airplane Models with Highest Uninjured Outcomes Boeing models dominate the list, indicating a strong safety record. Airbus A320 and A321 also show high safety with substantial uninjured outcomes. Top 10 Airplane Models:

Boeing 737 Boeing 767 Boeing 757 Boeing 777 Boeing 737 7H4 Boeing 737-7H4 Airbus A320 Boeing 777-222 Boeing 747 Airbus A321 Top 10 Helicopter Models with Highest Uninjured Outcomes Bell and Robinson Helicopter Company models are prevalent in the top 10. These models show consistent safety performance. Top 10 Helicopter Models:

Bell 206B Robinson R44 II Robinson R44 Robinson R22 Beta Robinson R22 Bell 206 Bell 407 Schweizer 269C Robinson R44 Robinson R22 Beta

Recommendations

Focus on High-Safety Models For airplane acquisitions, prioritize models such as Boeing 737, 767, 757, 777, and Airbus A320/A321. For helicopters, focus on Bell 206B, Robinson R44 II, and other top-performing models. Ensure Regular Maintenance and Training Continuous maintenance of aircraft to ensure they remain in top safety condition. Regular training for pilots to handle various flight scenarios and emergencies effectively. Invest in Safety Upgrades Upgrade older models with the latest safety technologies. Implement advanced monitoring systems for real-time assessment of aircraft health. Data-Driven Decision Making Use data analytics continuously to monitor the safety performance of the fleet. Regularly update the safety protocols based on the latest data insights. Further Investigation To enhance the safety and operational efficiency, consider the following steps:

Longitudinal Study on Safety Improvements Conduct a study over time to evaluate how safety improvements and technological advancements impact the safety of specific models. Comparative Analysis with Global Data Compare the findings with global