Comprehensive Aircraft Risk Assessment for a startup business operations

Project Overview

In this project, I will carry out a series of tasks, such as data cleaning, data analysis, and visualization, to extract valuable insights that can inform and guide decision-making for business stakeholders. This process involves preparing the data by removing inconsistencies, exploring patterns and trends through analysis, and using visual tools to present findings in a clear and actionable manner.

Business Understanding

PROBLEM STATEMENT

In order to diversify our business portfolio,we decided to expand into the aviation industry by acquiring aircrafts for both commercial and private operations.This expansion though an excellent idea presents significant challenges of a risky nature that come along with owning and operating aircraft.Since the company has little experience in the aviation sector,there is therefore a need to identify which aircraft models have the lowest risks.This will therefore help in ensuring that the company will be on the right track as it ventures into this market.

The goal of this project is to conduct a comprehensive analysis of various aircrafts as well as evaluating the probable risks therefore determining which models are the most reliable to be used by the company.By leveraging data-driven insights, we aim to identify aircrafts that will minimize potential risks and maximize return on investment. The findings from this analysis will be translated into actionable insights that the head of the new aviation division can use to make informed decisions on aircraft acquisition. These insights will help the company make strategic choices about which aircrafts are best suited for commercial and private enterprises, ensuring a smooth and successful entry into the aviation industry while mitigating risks and optimizing profitability.

This project will benefit stakeholders by providing data-driven insights that inform better decision-making regarding aircraft purchases, ultimately minimizing financial risks. By identifying low-risk aircraft, the company can enhance operational efficiency, ensure regulatory compliance, and improve customer satisfaction through reliable service. Additionally, these efforts will strengthen the company’s reputation in the aviation industry and position it for strategic growth, benefiting investors and shareholders with the potential for increased revenue and market share.

GOALS

Find the variables that make an airplane the safest and provide business recommendations based on these findings.

OBJECTIVES

1.To identify the lowest-risk aircraft to be bought by the company as it ventures into the aviation business. 2.To minimize any financial losses and ensure that the business is profitable to the company.

LIMITATION

The company is relatively new to the aviation sector and therefore it lacks knowledge on how to navigate the market as well as the types of aircrafts that will be low risk to the company.

#Source

The data to be used for this project was gathered from Kaggle.The dataset originates from the National Transportation Safety Board (NTSB) aviation accident database, which has been documenting civil aviation accidents and selected incidents in the United States and its territories since 1962. This database provides preliminary reports shortly after accidents occur, with factual information being updated as investigations progress. Once investigations are complete, preliminary reports are replaced with final reports that detail the accident and its probable causes. It's important to note that full narrative descriptions may not be available for incidents prior to 1993, those under revision, or cases not primarily investigated by the NTSB. The goal of this dataset is to enhance the quality and safety of air travel by analyzing aviation incidents.

Shape and Size

Data Types

Data Understanding

The first step in any data analysis process is to understand the dataset.This helps to get insights about the data set in order to be able to analyse the data.Gaining a deep understanding of the data is vital because it enables analysts to discern which features may influence outcomes and which might need further cleaning or transformation.

In [1]:
```
#Importing libraries to be used
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
#reading the csv file into a pandas DataFrame
df= pd.read_csv('aviation_data.csv',index_col=0,encoding='ISO-8859-1',low_m
df.head()
```

Out[2]:

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

5 rows × 30 columns

In [3]:
```
# what is in the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88889 entries, 20001218X45444 to 20221230106513
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Investigation.Type      88889 non-null  object 
 1   Accident.Number         88889 non-null  object 
 2   Event.Date              88889 non-null  object 
 3   Location                88837 non-null  object 
 4   Country                 88663 non-null  object 
 5   Latitude                34382 non-null  object 
 6   Longitude               34373 non-null  object 
 7   Airport.Code            50249 non-null  object 
 8   Airport.Name            52790 non-null  object 
 9   Injury.Severity         87889 non-null  object 
 10  Aircraft.damage         85695 non-null  object 
 11  Aircraft.Category       32287 non-null  object 
 12  Registration.Number     87572 non-null  object 
 13  Make                    88826 non-null  object 
 14  Model                   88797 non-null  object 
 15  Amateur.Built           88787 non-null  object 
 16  Number.of.Engines       82805 non-null  float64
 17  Engine.Type             81812 non-null  object 
 18  FAR.Description         32023 non-null  object 
 19  Schedule                12582 non-null  object 
 20  Purpose.of.flight       82697 non-null  object 
 21  Air.carrier             16648 non-null  object 
 22  Total.Fatal.Injuries    77488 non-null  float64
 23  Total.Serious.Injuries  76379 non-null  float64
 24  Total.Minor.Injuries    76956 non-null  float64
 25  Total.Uninjured         82977 non-null  float64
 26  Weather.Condition       84397 non-null  object 
 27  Broad.phase.of.flight   61724 non-null  object 
 28  Report.Status           82508 non-null  object 
 29  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(25)
memory usage: 21.0+ MB
```

In [4]:
```
# view numerical columns
df.describe()
```

Out[4]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.
count	82805.000000	77488.000000	76379.000000	76956.000000	829
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	6

In [5]:
```
df.shape
```

Out[5]: (88889, 30)

In [6]:
```
df.size
```

Out[6]: 2666670

In [7]:
```
df.dtypes
```

Out[7]:
```
Investigation.Type         object
Accident.Number            object
Event.Date                 object
Location                   object
Country                    object
Latitude                   object
Longitude                  object
Airport.Code               object
Airport.Name               object
Injury.Severity            object
Aircraft.damage            object
Aircraft.Category          object
Registration.Number        object
Make                       object
Model                      object
Amateur.Built              object
Number.of.Engines         float64
Engine.Type                object
FAR.Description            object
Schedule                   object
Purpose.of.flight          object
Air.carrier                object
Total.Fatal.Injuries      float64
Total.Serious.Injuries    float64
Total.Minor.Injuries      float64
Total.Uninjured           float64
Weather.Condition          object
Broad.phase.of.flight      object
Report.Status              object
Publication.Date           object
dtype: object
```

In [8]:
```
df.keys()
```

Out[8]:
```
Index(['Investigation.Type', 'Accident.Number', 'Event.Date', 'Location',
       'Country', 'Latitude', 'Longitude', 'Airport.Code', 'Airport.Nam
e',
       '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.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

DATA PREPARATION

Data preparation is an important step in data science that affects how successful a project is. It involves cleaning the data to make sure it's accurate, making it easier to use, and getting it ready for analysis.When data is well-prepared, data scientists can really make the most out of it,

which is why it's a crucial part of the data science process.

Data Cleaning

In [9]:
```
#view present columns with their uniqu values
unique_values_all = {col: df[col].unique() for col in df.columns}
unique_values_all
```

Out[9]:
```
{'Investigation.Type': array(['Accident', 'Incident'], dtype=object),
 'Accident.Number': array(['SEA87LA080', 'LAX94LA336', 'NYC07LA005',
..., 'WPR23LA075',
        'WPR23LA076', 'ERA23LA097'], dtype=object),
 'Event.Date': array(['1948-10-24', '1962-07-19', '1974-08-30', ..., '2
022-12-22',
        '2022-12-26', '2022-12-29'], dtype=object),
 'Location': array(['MOOSE CREEK, ID', 'BRIDGEPORT, CA', 'Saltville, V
A', ...,
        'San Manual, AZ', 'Auburn Hills, MI', 'Brasnorte, '], dtype=obj
ect),
 'Country': array(['United States', nan, 'GULF OF MEXICO', 'Puerto Ric
o',
        'ATLANTIC OCEAN', 'HIGH ISLAND', 'Bahamas', 'MISSING', 'Pakista
n',
        'Angola', 'Germany', 'Korea, Republic Of', 'Martinique',
        'American Samoa', 'PACIFIC OCEAN', 'Canada', 'Bolivia', 'Mexic
o',
        'Dominica', 'Netherlands Antilles', 'Iceland', 'Greece', 'Gua
 '
```

Identifying Missing Data

In [10]:
```
# is there any missing data
df.isna().sum()
```

Out[10]:
```
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
Latitude                54507
Longitude               54516
Airport.Code            38640
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
Publication.Date        13771
dtype: int64
```

In [11]:
```
#get the percentage of missing data
df.isna().sum()/len(df)* 100
```

Out[11]:
```
Investigation.Type         0.000000
Accident.Number            0.000000
Event.Date                 0.000000
Location                   0.058500
Country                    0.254250
Latitude                  61.320298
Longitude                 61.330423
Airport.Code              43.469946
Airport.Name              40.611324
Injury.Severity            1.124999
Aircraft.damage            3.593246
Aircraft.Category         63.677170
Registration.Number        1.481623
Make                       0.070875
Model                      0.103500
Amateur.Built              0.114750
Number.of.Engines          6.844491
Engine.Type                7.961615
FAR.Description           63.974170
Schedule                  85.845268
Purpose.of.flight          6.965991
Air.carrier               81.271023
Total.Fatal.Injuries      12.826109
Total.Serious.Injuries    14.073732
Total.Minor.Injuries      13.424608
Total.Uninjured            6.650992
Weather.Condition          5.053494
Broad.phase.of.flight     30.560587
Report.Status              7.178616
Publication.Date          15.492356
dtype: float64
```

In [12]:
```
#handling missing data
# dropping data that has too many missing values preferably more than 40%
dropped_columns=[]
for x in df:
    if (df[x].isna().sum() / len(df[x])) * 100 > 40:
        dropped_columns.append(x)
dropped_columns
```

Out[12]:
```
['Latitude',
 'Longitude',
 'Airport.Code',
 'Airport.Name',
 'Aircraft.Category',
 'FAR.Description',
 'Schedule',
 'Air.carrier']
```

In [13]:
```
# view the columns without the columns earlier dropped
df= df.drop(columns=dropped_columns)
df.columns
```

Out[13]:
```
Index(['Investigation.Type', 'Accident.Number', 'Event.Date', 'Location',
       'Country', 'Injury.Severity', 'Aircraft.damage', 'Registration.Num
ber',
       'Make', 'Model', 'Amateur.Built', 'Number.of.Engines', 'Engine.Typ
e',
       'Purpose.of.flight', 'Total.Fatal.Injuries', 'Total.Serious.Injuri
es',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight', 'Report.Status', 'Publication.Date'],
      dtype='object')
```

In [14]:
```
#check remaining columns that still have missing data
df.isna().sum()/len(df)* 100
```

Out[14]:
```
Investigation.Type         0.000000
Accident.Number            0.000000
Event.Date                 0.000000
Location                   0.058500
Country                    0.254250
Injury.Severity            1.124999
Aircraft.damage            3.593246
Registration.Number        1.481623
Make                       0.070875
Model                      0.103500
Amateur.Built              0.114750
Number.of.Engines          6.844491
Engine.Type                7.961615
Purpose.of.flight          6.965991
Total.Fatal.Injuries      12.826109
Total.Serious.Injuries    14.073732
Total.Minor.Injuries      13.424608
Total.Uninjured            6.650992
Weather.Condition          5.053494
Broad.phase.of.flight     30.560587
Report.Status              7.178616
Publication.Date          15.492356
dtype: float64
```

Handling Missing Data

In [15]:
```
#imputing numerical data by filling up the missing values in the various co

df['Total.Fatal.Injuries'].fillna(df['Total.Fatal.Injuries'].median(),inpla
df['Total.Minor.Injuries'].fillna(df['Total.Minor.Injuries'].median(),inpla
df['Total.Serious.Injuries'].fillna(df['Total.Serious.Injuries'].median(),i
df['Total.Uninjured'].fillna(df['Total.Uninjured'].median(),inplace=True)
df['Number.of.Engines'].fillna(df['Number.of.Engines'].mode(),inplace=True)
```

In [16]:
```
#Handling remaining missing values by filling them with suitable data  usir
df=df.fillna(method='ffill').fillna(method='bfill')
```

In [17]:
```
df.isna().sum()
```

Out[17]:
```
Investigation.Type         0
Accident.Number            0
Event.Date                 0
Location                   0
Country                    0
Injury.Severity            0
Aircraft.damage            0
Registration.Number        0
Make                       0
Model                      0
Amateur.Built              0
Number.of.Engines          0
Engine.Type                0
Purpose.of.flight          0
Total.Fatal.Injuries       0
Total.Serious.Injuries     0
Total.Minor.Injuries       0
Total.Uninjured            0
Weather.Condition          0
Broad.phase.of.flight      0
Report.Status              0
Publication.Date           0
dtype: int64
```

In [18]:
```
#count the occurrences of each unique value in the 'Make' column of the Dat
df['Make'].value_counts()
```

Out[18]:
```
Cessna           22232
Piper            12034
CESSNA            4930
Beech             4332
PIPER             2843
                 ...
SACKMAN              1
Ramsay               1
Dehaan               1
Spreuer              1
STROUP ROBERT        1
Name: Make, Length: 8237, dtype: int64
```

In [19]:
```
#converts all the values in the 'Make' column of the DataFrame to lowercase
df['Make'] = df['Make'].str.lower()
df['Make'].value_counts()
```

Out[19]:
```
cessna                  27162
piper                   14877
beech                    5375
boeing                   2749
bell                     2724
                        ...
barger steve m              1
kashpureff                  1
bergmann                    1
bradford pietenpol          1
zlin aviation s.r.o.        1
Name: Make, Length: 7587, dtype: int64
```

In [20]:
```
#counts the occurrences of each unique combination of values across the 'We
df[['Weather.Condition','Model','Aircraft.damage']].value_counts()
```

Out[20]:
```
Weather.Condition  Model          Aircraft.damage
VMC                152            Substantial        1939
                   172            Substantial        1483
                   172N           Substantial         889
                   PA-28-140      Substantial         678
                   150            Substantial         674
                                                     ...
                   Glasair II     Substantial           1
                                  Destroyed             1
                   Glasair I TD   Destroyed             1
                   Glasair FT II  Substantial           1
IMC                -737-222       Destroyed             1
Length: 18509, dtype: int64
```

In [21]:
```
df['Engine.Type']
```

Out[21]:
```
Event.Id
20001218X45444    Reciprocating
20001218X45447    Reciprocating
20061025X01555    Reciprocating
20001218X45448    Reciprocating
20041105X01764    Reciprocating
                      ...
20221227106491    Reciprocating
20221227106494    Reciprocating
20221227106497    Reciprocating
20221227106498    Reciprocating
20221230106513    Reciprocating
Name: Engine.Type, Length: 88889, dtype: object
```

In [22]:
```
df.info(['Engine.Type'])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88889 entries, 20001218X45444 to 20221230106513
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Investigation.Type      88889 non-null  object
 1   Accident.Number         88889 non-null  object
 2   Event.Date              88889 non-null  object
 3   Location                88889 non-null  object
 4   Country                 88889 non-null  object
 5   Injury.Severity         88889 non-null  object
 6   Aircraft.damage         88889 non-null  object
 7   Registration.Number     88889 non-null  object
 8   Make                    88889 non-null  object
 9   Model                   88889 non-null  object
 10  Amateur.Built           88889 non-null  object
 11  Number.of.Engines       88889 non-null  float64
 12  Engine.Type             88889 non-null  object
 13  Purpose.of.flight       88889 non-null  object
 14  Total.Fatal.Injuries    88889 non-null  float64
 15  Total.Serious.Injuries  88889 non-null  float64
 16  Total.Minor.Injuries    88889 non-null  float64
 17  Total.Uninjured         88889 non-null  float64
 18  Weather.Condition       88889 non-null  object
 19  Broad.phase.of.flight   88889 non-null  object
 20  Report.Status           88889 non-null  object
 21  Publication.Date        88889 non-null  object
dtypes: float64(5), object(17)
memory usage: 18.1+ MB
```

In [23]:
```
df['Engine.Type'].value_counts()
```

Out[23]:
```
Reciprocating      75277
Turbo Shaft         4293
Turbo Prop          3708
Turbo Fan           2732
Unknown             2062
Turbo Jet            750
None                  25
Geared Turbofan       18
Electric              16
NONE                   3
LR                     2
UNK                    2
Hybrid Rocket          1
Name: Engine.Type, dtype: int64
```

In [24]:
```
df['Number.of.Engines'].value_counts()
```

Out[24]:
```
1.0    74599
2.0    12013
0.0     1301
3.0      510
4.0      462
8.0        3
6.0        1
Name: Number.of.Engines, dtype: int64
```

In [25]:
```
df['Aircraft.damage'].value_counts()
```

Out[25]:
```
Substantial    66154
Destroyed      19631
Minor           2976
Unknown          128
Name: Aircraft.damage, dtype: int64
```

In [26]:
```
df['Purpose.of.flight'].value_counts()
```

Out[26]:
```
Personal                  53396
Instructional             11502
Unknown                    6930
Aerial Application         4977
Business                   4262
Positioning                1823
Other Work Use             1386
Aerial Observation          874
Ferry                       861
Public Aircraft             754
Executive/corporate         592
Flight Test                 475
Skydiving                   211
External Load               143
Public Aircraft - Federal   122
Air Race show               113
Banner Tow                  109
Public Aircraft - Local      84
Public Aircraft - State      69
Glider Tow                   64
Air Race/show                62
Firefighting                 48
Air Drop                     14
ASHO                          9
PUBS                          8
PUBL                          1
Name: Purpose.of.flight, dtype: int64
```

In [27]:
```
df['Weather.Condition'].value_counts()
```

Out[27]:
```
VMC    81501
IMC     6184
UNK      856
Unk      348
Name: Weather.Condition, dtype: int64
```

In [28]:
```
df['Publication.Date'].value_counts()
```

Out[28]:
```
25-09-2020    17096
26-09-2020     1805
03-11-2020     1166
10-03-1988     1011
06-02-1995      848
              ...
05-06-2013        1
02-06-2011        1
03-10-2005        1
29-06-2007        1
02-09-1998        1
Name: Publication.Date, Length: 2924, dtype: int64
```

In [29]:
```
df.info(['Event.Date'])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88889 entries, 20001218X45444 to 20221230106513
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Investigation.Type      88889 non-null  object 
 1   Accident.Number         88889 non-null  object 
 2   Event.Date              88889 non-null  object 
 3   Location                88889 non-null  object 
 4   Country                 88889 non-null  object 
 5   Injury.Severity         88889 non-null  object 
 6   Aircraft.damage         88889 non-null  object 
 7   Registration.Number     88889 non-null  object 
 8   Make                    88889 non-null  object 
 9   Model                   88889 non-null  object 
 10  Amateur.Built           88889 non-null  object 
 11  Number.of.Engines       88889 non-null  float64
 12  Engine.Type             88889 non-null  object 
 13  Purpose.of.flight       88889 non-null  object 
 14  Total.Fatal.Injuries    88889 non-null  float64
 15  Total.Serious.Injuries  88889 non-null  float64
 16  Total.Minor.Injuries    88889 non-null  float64
 17  Total.Uninjured         88889 non-null  float64
 18  Weather.Condition       88889 non-null  object 
 19  Broad.phase.of.flight   88889 non-null  object 
 20  Report.Status           88889 non-null  object 
 21  Publication.Date        88889 non-null  object 
dtypes: float64(5), object(17)
memory usage: 18.1+ MB
```

In [30]:
```
df['Injury.Severity'] = ['Fatal' if item[0:5]=='Fatal' else item for item i
```

In [31]:
```
df['Injury.Severity'].value_counts()
```

Out[31]:
```
Non-Fatal      68124
Fatal          18035
Incident        2219
Minor            233
Serious          182
Unavailable       96
Name: Injury.Severity, dtype: int64
```

In [32]:
```
df[['Injury.Severity','Total.Fatal.Injuries','Total.Serious.Injuries', 'Tot
```

Out[32]:

Event.Id	Injury.Severity	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries
20001218X45444	Fatal	2.0	0.0	0.0
20001218X45447	Fatal	4.0	0.0	0.0
20061025X01555	Fatal	3.0	0.0	0.0
20001218X45448	Fatal	2.0	0.0	0.0
20041105X01764	Fatal	1.0	2.0	0.0
...	...	...	...	...
20221227106491	Minor	0.0	1.0	0.0
20221227106494	Minor	0.0	0.0	0.0
20221227106497	Non-Fatal	0.0	0.0	0.0
20221227106498	Non-Fatal	0.0	0.0	0.0
20221230106513	Minor	0.0	1.0	0.0

88889 rows × 5 columns

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88889 entries, 20001218X45444 to 20221230106513
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Investigation.Type      88889 non-null  object 
 1   Accident.Number         88889 non-null  object 
 2   Event.Date              88889 non-null  object 
 3   Location                88889 non-null  object 
 4   Country                 88889 non-null  object 
 5   Injury.Severity         88889 non-null  object 
 6   Aircraft.damage         88889 non-null  object 
 7   Registration.Number     88889 non-null  object 
 8   Make                    88889 non-null  object 
 9   Model                   88889 non-null  object 
 10  Amateur.Built           88889 non-null  object 
 11  Number.of.Engines       88889 non-null  float64
 12  Engine.Type             88889 non-null  object 
 13  Purpose.of.flight       88889 non-null  object 
 14  Total.Fatal.Injuries    88889 non-null  float64
 15  Total.Serious.Injuries  88889 non-null  float64
 16  Total.Minor.Injuries    88889 non-null  float64
 17  Total.Uninjured         88889 non-null  float64
 18  Weather.Condition       88889 non-null  object 
 19  Broad.phase.of.flight   88889 non-null  object 
 20  Report.Status           88889 non-null  object 
 21  Publication.Date        88889 non-null  object 
dtypes: float64(5), object(17)
memory usage: 18.1+ MB
```

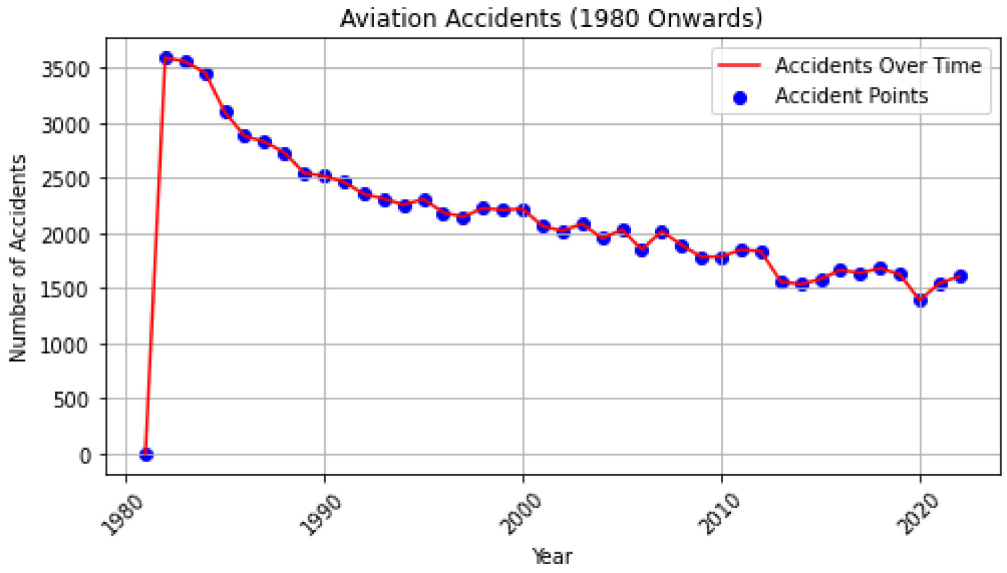
In [34]:
```
df.to_csv('df.csv', index=False)
```

Data Analysis

Type *Markdown* and LaTeX: α^2

We start by finding out the trend of accidents across the years by counting the number of accidents per year using a scatter plot and then joining the scatter plots to create a line graph.

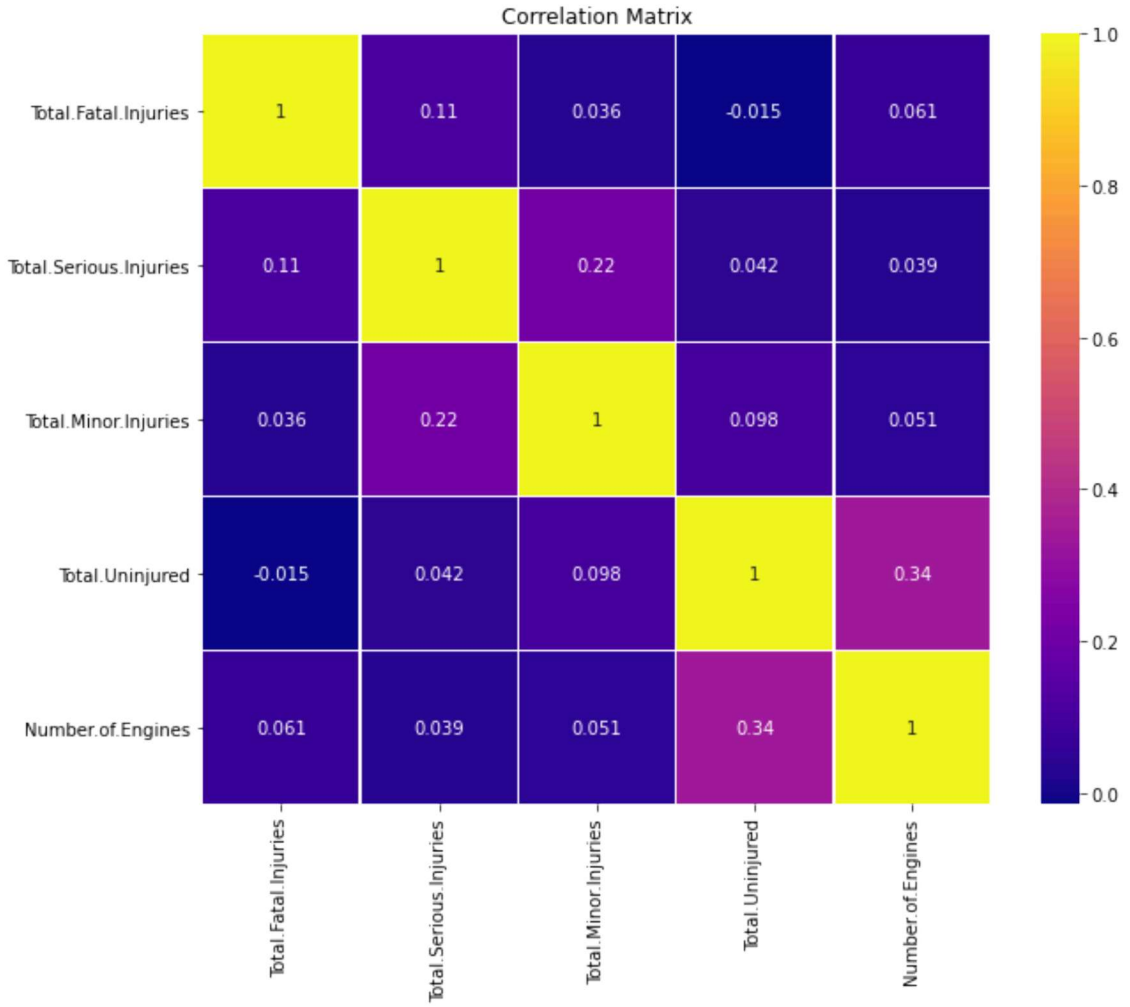
In [35]:
```
# Convert 'Event.Date' to datetime format
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')

# Filter for years from 1980 onwards
df_filtered = df[df['Event.Date'].dt.year >= 1980]

# Count accidents by year
accidents_by_year = df_filtered['Event.Date'].dt.year.value_counts().sort_i

# Plot accidents over time
plt.figure(figsize=(8, 4))

# Plot the line
plt.plot(accidents_by_year.index, accidents_by_year.values, linestyle='-',

# Add points for each year
plt.scatter(accidents_by_year.index, accidents_by_year.values, color='blue'

# Adding titles and labels
plt.title('Aviation Accidents (1980 Onwards)')
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.grid(True)
plt.xticks(rotation=45)

# Adding a legend to differentiate between the line and points
plt.legend()

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



Overall, there's a clear downward trend. This means that there were fewer accidents over time which shows that flying has become safer compared to past years

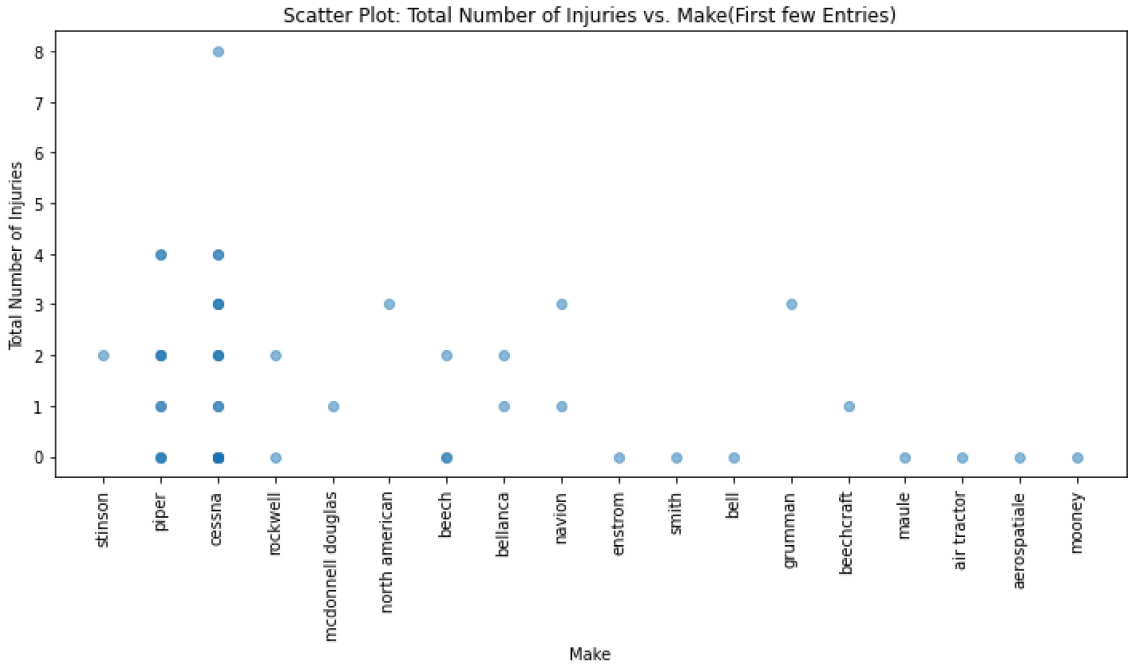
Next we check the correlation between the numerical data to have an idea of the reltionships to explore in your data

In [36]:
```
main_columns = [
    'Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries
    'Total.Uninjured', 'Number.of.Engines','Weather.Condition'
]
# Create a DataFrame with these columns
df_corr = df[main_columns]

# Compute the correlation matrix
corr_matrix = df_corr.corr()

# Plot the correlation matrix using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='plasma', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



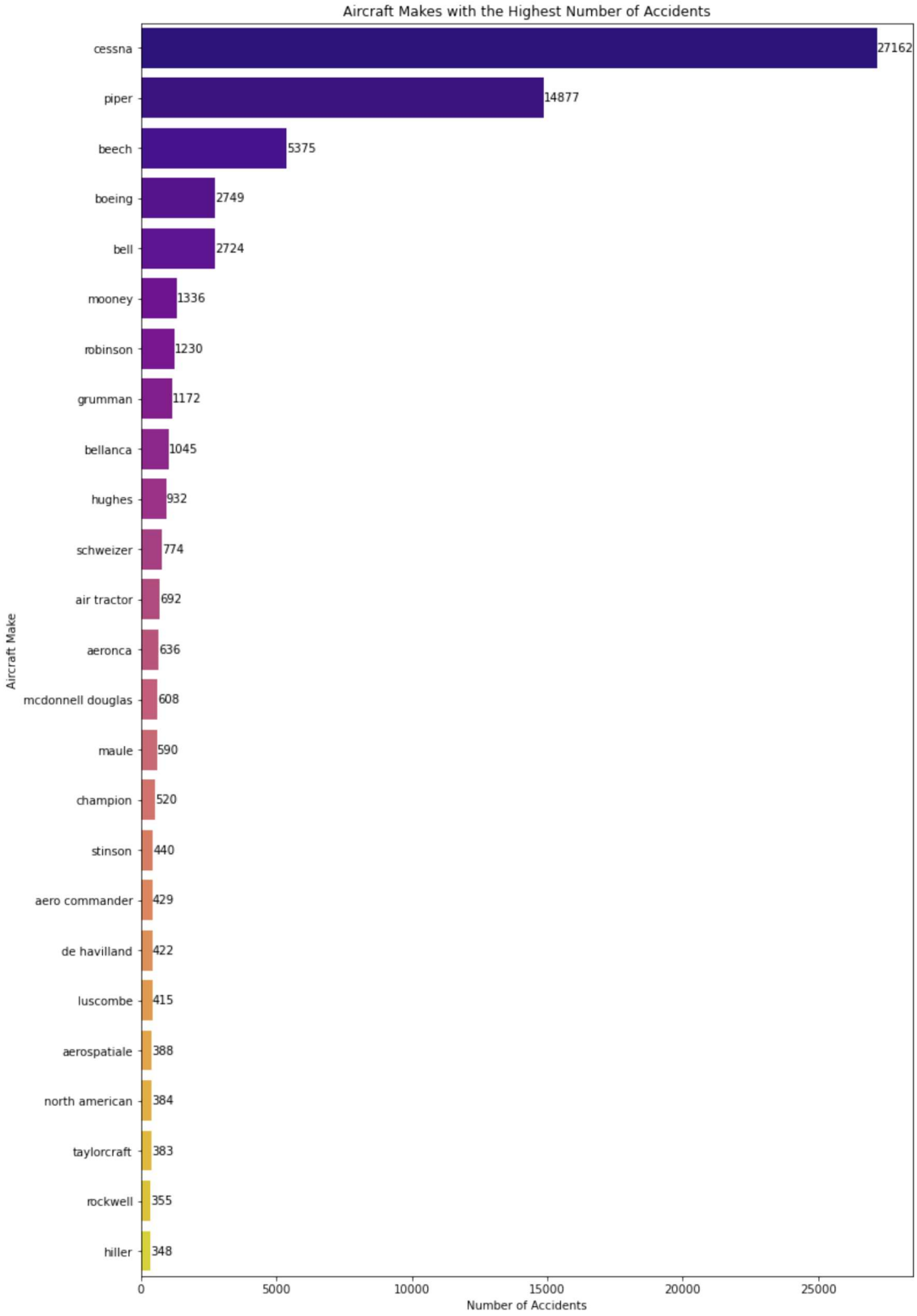
Now let's look at the Make of the aircrafts and the viability for the company

In [37]:
```
# scatter plot to visualize the total number of injuries against the 'Make
subset_data =df.head(50)

# Adding the injuries
Total_injuries = df['Total.Fatal.Injuries'] + df['Total.Minor.Injuries'] +

# Plotting the scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(subset_data['Make'], Total_injuries.head(50), alpha=0.5)
plt.xlabel('Make')
plt.ylabel('Total Number of Injuries')
plt.title('Scatter Plot: Total Number of Injuries vs. Make(First few Entrie
plt.xticks(rotation=90)  # Rotate x-axis labels for readability
plt.tight_layout()
plt.show()
```



In [38]:

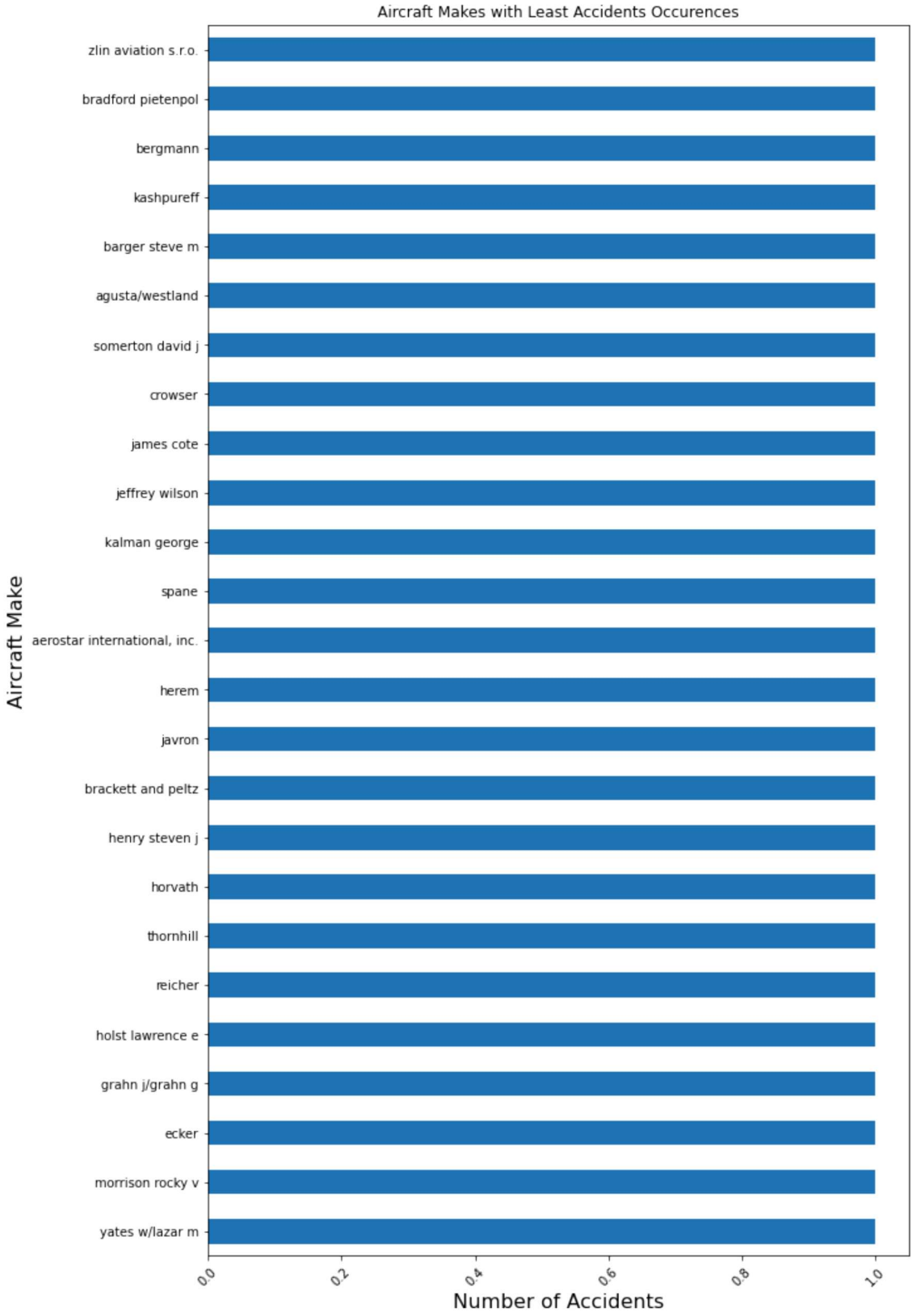
```
# Count accidents by 'Make' with a horizontal bar chart showing the top 25
accident_counts = df.groupby('Make').size().reset_index(name='accident_cour

# Sort and select the top 25 makes
top_accidents = accident_counts.sort_values(by='accident_count', ascending=

# Plotting the horizontal bar chart for the top aircraft makes with the mos
plt.figure(figsize=(12, 20))
sns.barplot(y='Make', x='accident_count', data=top_accidents, palette='plas

# Adding data labels
for index, value in enumerate(top_accidents['accident_count']):
    plt.text(value, index, str(value), va='center', ha='left', fontsize=10)

plt.title('Aircraft Makes with the Highest Number of Accidents')
plt.xlabel('Number of Accidents')
plt.ylabel('Aircraft Make')
plt.show()
```



Cessna experienced the highest number of accidents among the listed aircraft makes, this is likely due to their widespread use, large fleet size, frequent use in training, and thorough incident reporting. This doesn't necessarily indicate that Cessna aircraft are less safe; instead, it reflects their significant presence and popularity within the aviation industry.

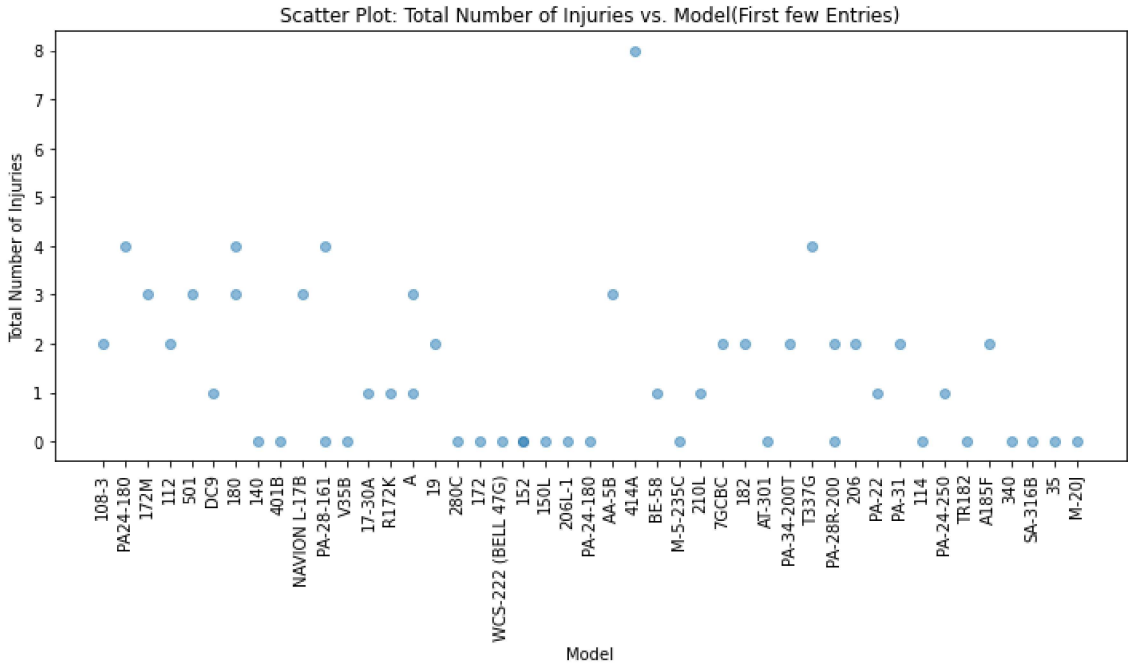
In [39]:

```
# least accidents by the 'Make' with a horizontal bar chart displaying the
least_accident = df ['Make'].value_counts().tail(25)
plt.figure(figsize=(10, 18))
least_accident.plot(kind='barh')
plt.title('Aircraft Makes with Least Accidents Occurences')
plt.ylabel('Aircraft Make',fontsize=16)
plt.xlabel('Number of Accidents',fontsize=16)
plt.xticks(rotation=45)
plt.show()
```

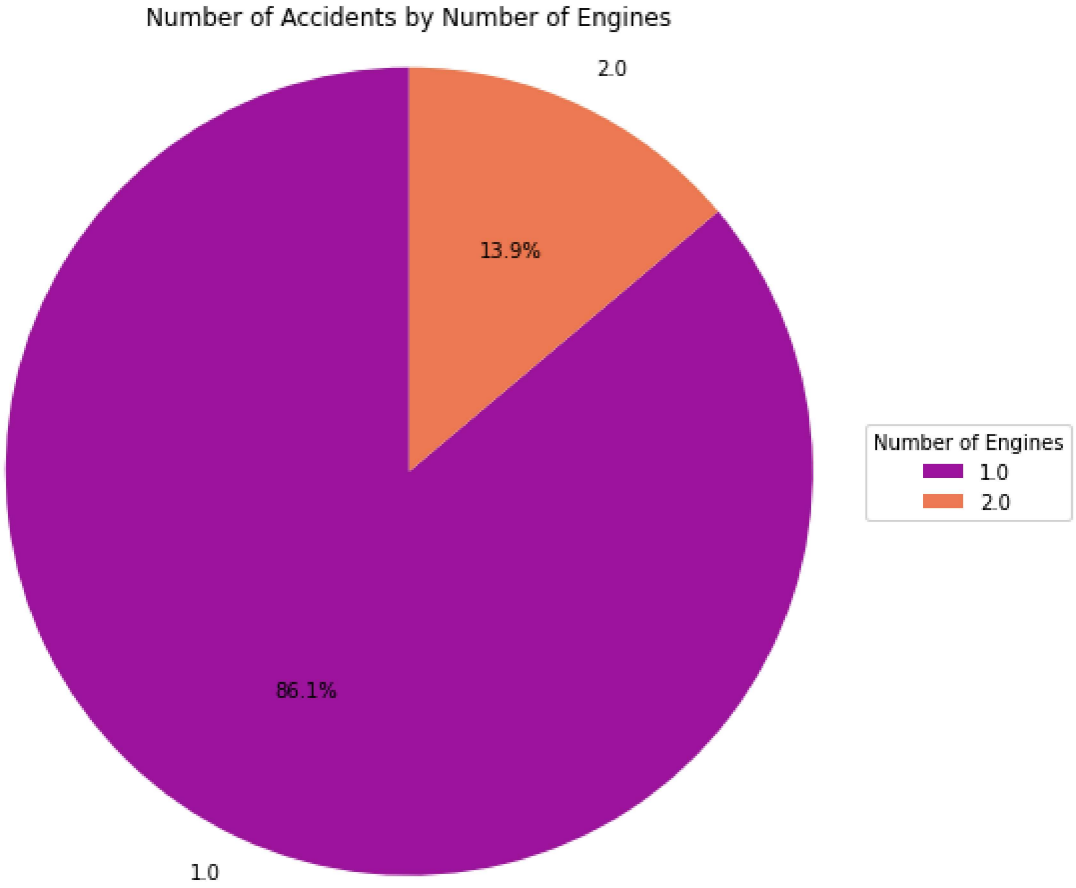


This graph shows a list of alternative aircrafts that are less risky to consider when selecting a make

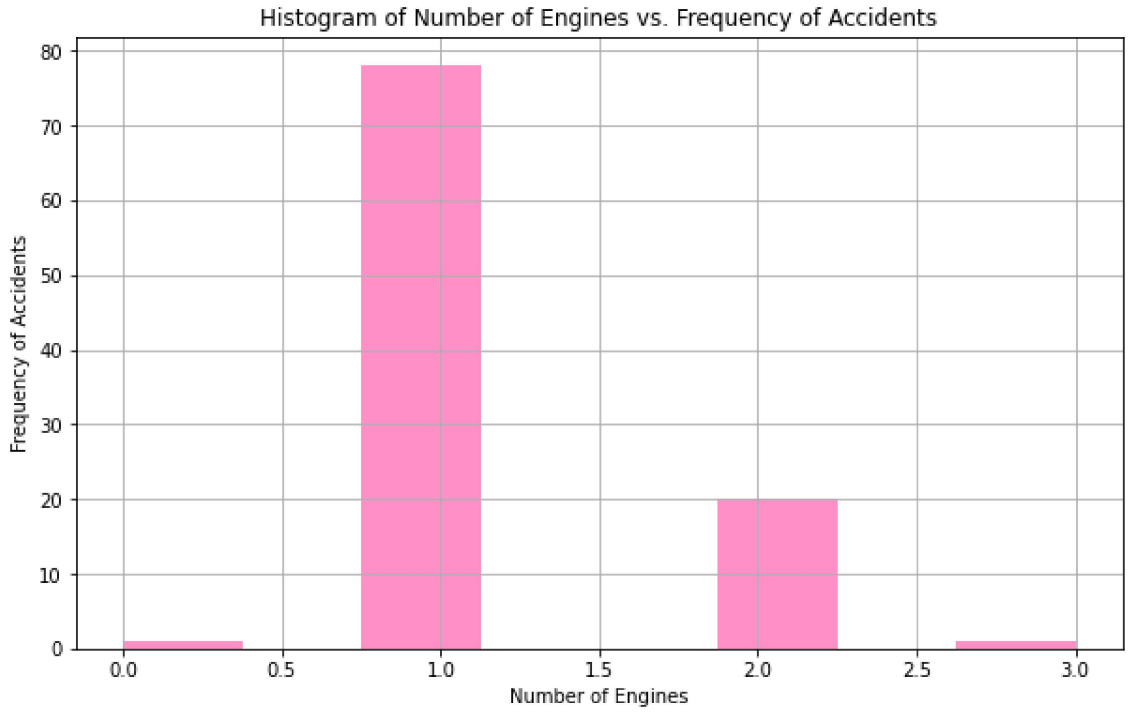
In [40]:
```
#A scatter plot displaying the total number of injuries for the first 50 ai
subset_data =df.head(50)

# Adding the injuries
Total_injuries = df['Total.Fatal.Injuries'] + df['Total.Minor.Injuries'] +

# Plotting the scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(subset_data['Model'], Total_injuries.head(50), alpha=0.5)
plt.xlabel('Model')
plt.ylabel('Total Number of Injuries')
plt.title('Scatter Plot: Total Number of Injuries vs. Model(First few Entri
plt.xticks(rotation=90)  # Rotate x-axis labels for readability
plt.tight_layout()
plt.show()
```



In [41]:
```
# a pie chart illustrating the distribution of aviation accidents based on
engine_counts = df[df['Number.of.Engines'].isin([1, 2])].groupby('Number.of

# Plot the pie chart
plt.figure(figsize=(8, 8))
colors = sns.color_palette('plasma', len(engine_counts))

# Include autopct to get percentage labels
wedges, texts, autotexts = plt.pie(engine_counts,
                                   labels=engine_counts.index,  # Use the
                                   startangle=90,
                                   colors=colors,
                                   autopct='%1.1f%%')  # Show percentage o

plt.title('Number of Accidents by Number of Engines')
plt.axis('equal')  # Make the pie chart circular

# Add a legend to label what the colors represent
plt.legend(wedges,
           engine_counts.index,
           title='Number of Engines',
           loc='center left',
           bbox_to_anchor=(1, 0, 0.5, 1))  # Position the legend outside th

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



In [42]:
```
# The frequency of aviation accidents based on the number of engines in the
subset_data = df['Number.of.Engines'].head(100)

# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(subset_data, bins='auto', color='hotpink', alpha=0.7)
plt.xlabel('Number of Engines')
plt.ylabel('Frequency of Accidents')
plt.title('Histogram of Number of Engines vs. Frequency of Accidents')
plt.grid(True)
plt.show()
```



Aircrafts with more engines have the least amount of accidents compared to those with a few engines.It is therefore a lower risk to have aircrafts with more engines

In [43]:

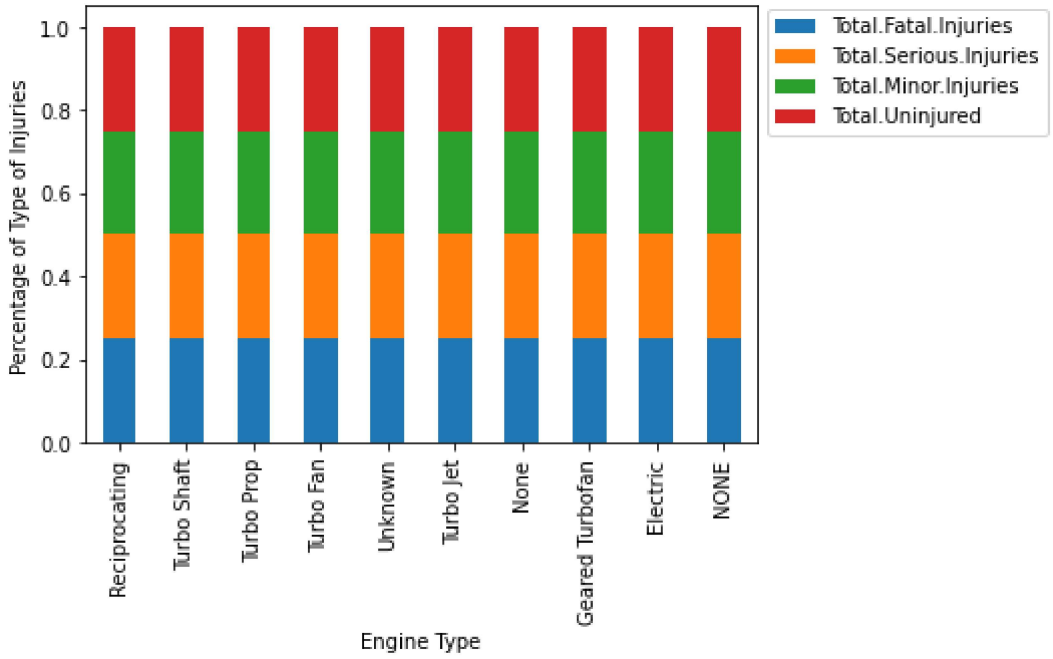
```
#count injuries and uninjured individuals by engine type in aviation accide
engine_accidents = df.groupby('Engine.Type')[['Total.Fatal.Injuries',
'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured']].count

engine_accidents['Total'] = engine_accidents.sum(axis=1)
engine_accidents = engine_accidents.sort_values(by=['Total'],ascending=Fals
engine_accidents_sorted_top_10 = engine_accidents.head(10)
engine_accidents_final= engine_accidents_sorted_top_10.drop(columns=['Total

Total = engine_accidents_final.sum(axis = 1)

engine_percentage = engine_accidents_final.div(Total,axis=0)

print(engine_accidents)
print(engine_accidents_final)
print(engine_percentage)
```

```
                 Total.Fatal.Injuries  Total.Serious.Injuries  \
Engine.Type
Reciprocating                   75277                   75277
Turbo Shaft                      4293                    4293
Turbo Prop                       3708                    3708
Turbo Fan                        2732                    2732
Unknown                          2062                    2062
Turbo Jet                         750                     750
None                               25                      25
Geared Turbofan                    18                      18
Electric                           16                      16
NONE                                3                       3
LR                                  2                       2
UNK                                 2                       2
Hybrid Rocket                       1                       1

                 Total.Minor.Injuries  Total.Uninjured   Total
Engine.Type
Reciprocating                   75277            75277  301108
Turbo Shaft                      4293             4293   17172
Turbo Prop                       3708             3708   14832
Turbo Fan                        2732             2732   10928
Unknown                          2062             2062    8248
Turbo Jet                         750              750    3000
None                               25               25     100
Geared Turbofan                    18               18      72
Electric                           16               16      64
NONE                                3                3      12
LR                                  2                2       8
UNK                                 2                2       8
Hybrid Rocket                       1                1       4
                 Total.Fatal.Injuries  Total.Serious.Injuries  \
Engine.Type
Reciprocating                   75277                   75277
Turbo Shaft                      4293                    4293
Turbo Prop                       3708                    3708
Turbo Fan                        2732                    2732
Unknown                          2062                    2062
Turbo Jet                         750                     750
None                               25                      25
Geared Turbofan                    18                      18
Electric                           16                      16
NONE                                3                       3

                 Total.Minor.Injuries  Total.Uninjured
Engine.Type
Reciprocating                   75277            75277
Turbo Shaft                      4293             4293
Turbo Prop                       3708             3708
Turbo Fan                        2732             2732
Unknown                          2062             2062
Turbo Jet                         750              750
None                               25               25
Geared Turbofan                    18               18
Electric                           16               16
NONE                                3                3
                 Total.Fatal.Injuries  Total.Serious.Injuries  \
```

Engine.Type		
Reciprocating	0.25	0.25
Turbo Shaft	0.25	0.25
Turbo Prop	0.25	0.25
Turbo Fan	0.25	0.25
Unknown	0.25	0.25
Turbo Jet	0.25	0.25
None	0.25	0.25
Geared Turbofan	0.25	0.25
Electric	0.25	0.25
NONE	0.25	0.25

Engine.Type	Total.Minor.Injuries	Total.Uninjured
Reciprocating	0.25	0.25
Turbo Shaft	0.25	0.25
Turbo Prop	0.25	0.25
Turbo Fan	0.25	0.25
Unknown	0.25	0.25
Turbo Jet	0.25	0.25
None	0.25	0.25
Geared Turbofan	0.25	0.25
Electric	0.25	0.25
NONE	0.25	0.25

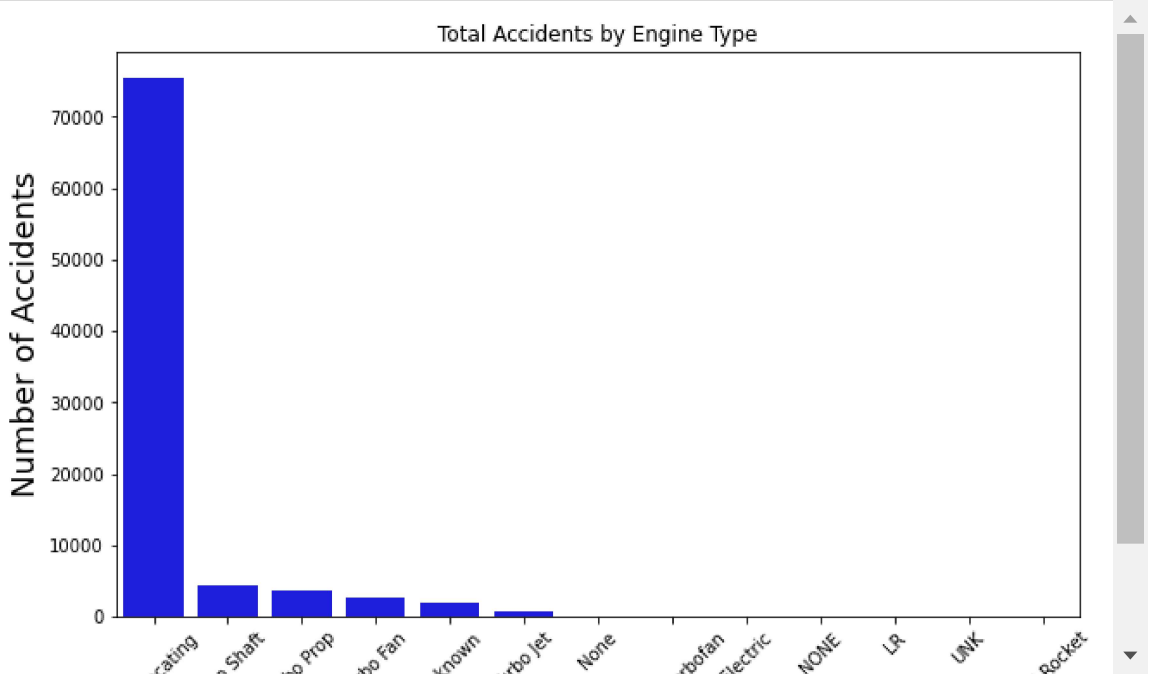
In [44]:
```
#bar chart displaying the percentage of different types of injuries for eac
ax= engine_percentage.plot(kind='bar',stacked='True')
ax.legend(loc='center left', bbox_to_anchor=(1, .85), ncol=1)
plt.xlabel('Engine Type')
plt.ylabel('Percentage of Type of Injuries')
plt.show()
```



The percentage of injuries across the different engine types appears to be of minimal significance

In [45]:

```
# Count of accidents by engine type
plt.figure(figsize=(10, 6))

# Create a count DataFrame for the number of accidents by Engine Type
count_data = df['Engine.Type'].value_counts().reset_index()
count_data.columns = ['Engine.Type', 'Accident_Count']

# Create a bar plot using the count DataFrame
sns.barplot(x='Engine.Type', y='Accident_Count', data=count_data,color='blu

plt.xticks(rotation=45)
plt.title('Total Accidents by Engine Type')
plt.ylabel('Number of Accidents',fontsize =18)
plt.xlabel('Engine Type',fontsize=18)
plt.legend
plt.show()
```



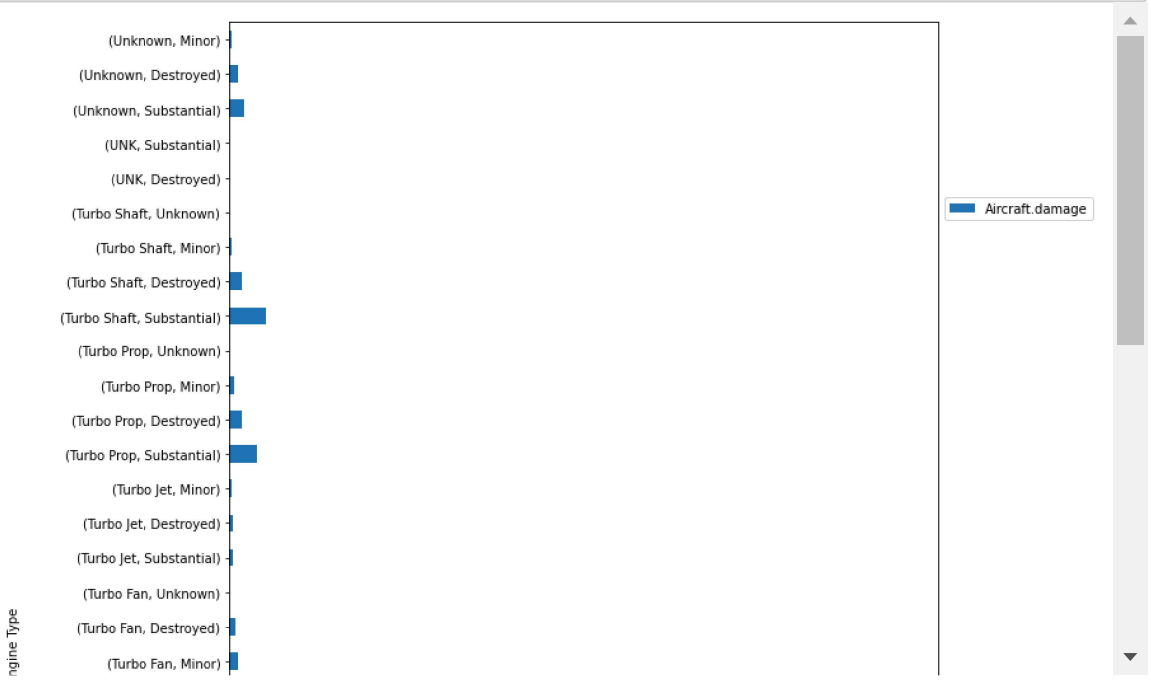
In [46]:
```
engine_damage = df.groupby('Engine.Type')['Aircraft.damage'].value_counts()
engine_damage
```

Out[46]:

Engine.Type	Aircraft.damage	
Electric	Substantial	13
	Destroyed	2
	Minor	1
Geared Turbofan	Substantial	12
	Destroyed	5
	Minor	1
Hybrid Rocket	Destroyed	1
LR	Substantial	2
NONE	Substantial	3
None	Substantial	20
	Destroyed	4
	Minor	1
Reciprocating	Substantial	57677
	Destroyed	16092
	Minor	1398
	Unknown	110
Turbo Fan	Substantial	1423
	Minor	765
	Destroyed	541
	Unknown	3
Turbo Jet	Substantial	329
	Destroyed	246
	Minor	175
Turbo Prop	Substantial	2356
	Destroyed	1000
	Minor	351
	Unknown	1
Turbo Shaft	Substantial	3092
	Destroyed	1053
	Minor	134
	Unknown	14
UNK	Destroyed	1
	Substantial	1
Unknown	Substantial	1226
	Destroyed	686
	Minor	150

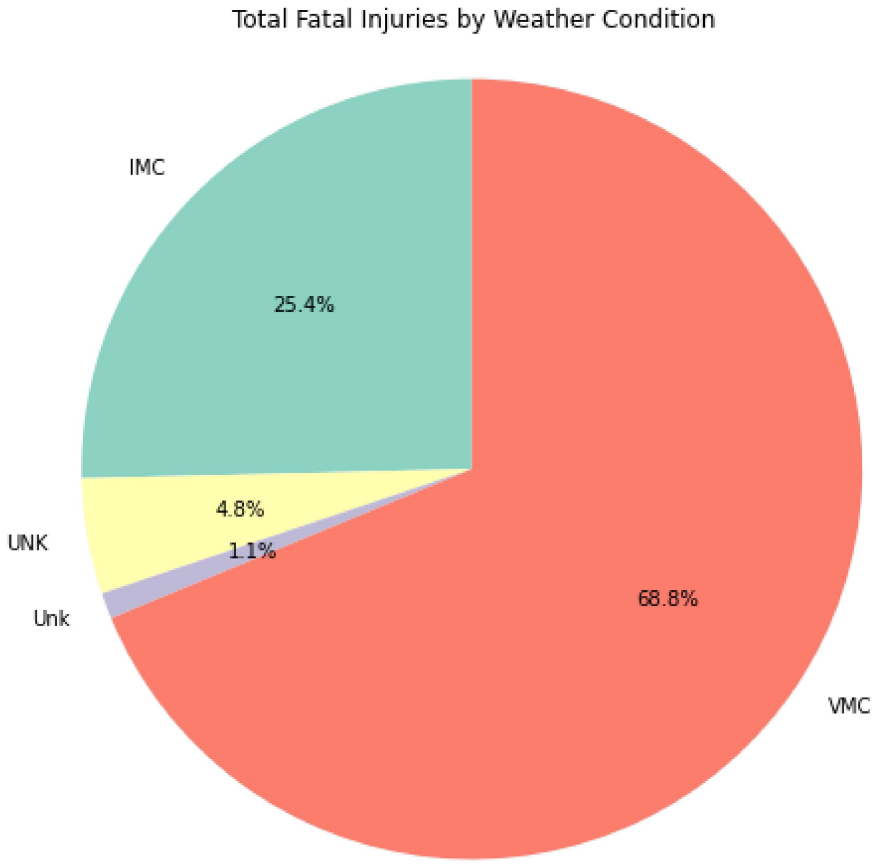
Name: Aircraft.damage, dtype: int64

In [47]:
```
# a horizontal  bar chart showing the percentage of aircraft damage for eac
plt.figure(figsize=(10,18))
ax= engine_damage.plot(kind='barh',stacked='True')
ax.legend(loc='center left', bbox_to_anchor=(1, .85), ncol=1)
plt.ylabel('Engine Type')
plt.xlabel('Aircraft Damage')
plt.show()
```



From the graphs above it is clear that the reciprocating engine has the highest number of accidents as well as the higest percentage of damage which poses a great risk to the company.It is therfore diqualified to reduce the risks caused.

In [48]:
```
#a display of the distribution of total fatal injuries in aviation accident
fatalities_by_weather = df.groupby('Weather.Condition')['Total.Fatal.Injuri

# Plotting the pie chart
plt.figure(figsize=(10, 8))
plt.pie(fatalities_by_weather,
        labels=fatalities_by_weather.index,
        autopct='%1.1f%%',
        startangle=90,
        colors=sns.color_palette('Set3', len(fatalities_by_weather)))

plt.title('Total Fatal Injuries by Weather Condition')
plt.axis('equal')  # Equal aspect ratio ensures that pie chart is circular
plt.show()
```



The chart above shows that there are more fatal injuries when the weather is VMC .This is a risky time to travel.

Type *Markdown* and LaTeX: α^2