stanProject

November 23, 2024

0.1 1. Business Understanding

A company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating airplanes for commercial and private enterprises, but do not know anything about the potential risks of aircraft. To evaluate the potential risks associated with purchasing and operating airplanes, the company will consider historical accident trends. Data will be sourced from National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents.

0.1.1 a. General Objective

To determine which aircraft are the lowest risk for the company to start this new business endeavor.

0.1.2 b. Research questions

i. Aircraft Make and model: What is the relationship between Accidents and Aircraft Make and model?

ii. Amateur Built: Is there a correlation between accidents trends of amateur built and professional built aircrafts.

iii. Number of Engines: Does number of engines determine safety.

iv. Engine types: Is there a relationship between engine type and number of accidents.

0.2 2. Data Understanding

[ ]:
```
#importing the libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

[ ]:
```
# Ignore all the warnings
warnings.filterwarnings('ignore')
```

0.2.1 Loading the data

[ ]:
```
#loading the data and set encoding because there are Non-ASCII characters
df = pd.read_csv('AviationData.csv', encoding='latin1')
#preview the first five rows
df.head()
```

[ ]:
```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country   Latitude  Longitude Airport.Code  \
0  MOOSE CREEK, ID  United States        NaN        NaN          NaN
1   BRIDGEPORT, CA  United States        NaN        NaN          NaN
2    Saltville, VA  United States  36.922223 -81.878056          NaN
3       EUREKA, CA  United States        NaN        NaN          NaN
4       Canton, OH  United States        NaN        NaN          NaN

  Airport.Name  … Purpose.of.flight Air.carrier Total.Fatal.Injuries  \
0          NaN  …          Personal         NaN                  2.0
1          NaN  …          Personal         NaN                  4.0
2          NaN  …          Personal         NaN                  3.0
3          NaN  …          Personal         NaN                  2.0
4          NaN  …          Personal         NaN                  1.0

  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0

  Weather.Condition  Broad.phase.of.flight   Report.Status Publication.Date
0               UNK                 Cruise  Probable Cause              NaN
1               UNK                Unknown  Probable Cause       19-09-1996
2               IMC                 Cruise  Probable Cause       26-02-2007
3               IMC                 Cruise  Probable Cause       12-09-2000
4               VMC               Approach  Probable Cause       16-04-1980

[5 rows x 31 columns]
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
```

```
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      88889 non-null  object
 2   Accident.Number         88889 non-null  object
 3   Event.Date              88889 non-null  object
 4   Location                88837 non-null  object
 5   Country                 88663 non-null  object
 6   Latitude                34382 non-null  object
 7   Longitude               34373 non-null  object
 8   Airport.Code            50132 non-null  object
 9   Airport.Name            52704 non-null  object
 10  Injury.Severity         87889 non-null  object
 11  Aircraft.damage         85695 non-null  object
 12  Aircraft.Category       32287 non-null  object
 13  Registration.Number     87507 non-null  object
 14  Make                    88826 non-null  object
 15  Model                   88797 non-null  object
 16  Amateur.Built           88787 non-null  object
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object
 19  FAR.Description         32023 non-null  object
 20  Schedule                12582 non-null  object
 21  Purpose.of.flight       82697 non-null  object
 22  Air.carrier             16648 non-null  object
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object
 28  Broad.phase.of.flight   61724 non-null  object
 29  Report.Status           82505 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

[ ]:
```
df.shape
```

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

[ ]:
```
# See all value occurences across all columns
cols = ['Investigation.Type', 'Location', 'Country', 'Airport.Code', 'Airport.
 ↪Name', 'Injury.Severity', 'Registration.Number', 'Make', 'Amateur.Built',␣
 ↪'Publication.Date', 'Weather.Condition', 'Purpose.of.flight']
for col in df[cols].columns:
    print(df[col].value_counts().nlargest(5))
```

```python
    print('\n---\n')
```

```
Investigation.Type
Accident    85015
Incident     3874
Name: count, dtype: int64

---

Location
ANCHORAGE, AK      434
MIAMI, FL          200
ALBUQUERQUE, NM    196
HOUSTON, TX        193
CHICAGO, IL        184
Name: count, dtype: int64

---

Country
United States     82248
Brazil              374
Canada              359
Mexico              358
United Kingdom      344
Name: count, dtype: int64

---

Airport.Code
NONE    1488
PVT      485
APA      160
ORD      149
MRI      137
Name: count, dtype: int64

---

Airport.Name
Private             240
PRIVATE             224
Private Airstrip    153
NONE                146
PRIVATE STRIP       111
Name: count, dtype: int64

---
```

```
Injury.Severity
Non-Fatal    67357
Fatal(1)      6167
Fatal         5262
Fatal(2)      3711
Incident      2219
Name: count, dtype: int64
```

```
Registration.Number
NONE      344
UNREG     126
UNK        13
USAF        9
N20752      8
Name: count, dtype: int64
```

```
Make
Cessna    22227
Piper     12029
CESSNA     4922
Beech      4330
PIPER      2841
Name: count, dtype: int64
```

```
Amateur.Built
No     80312
Yes     8475
Name: count, dtype: int64
```

```
Publication.Date
25-09-2020    17019
26-09-2020     1769
03-11-2020     1155
31-03-1993      452
25-11-2003      396
Name: count, dtype: int64
```

```
Weather.Condition
VMC    77303
IMC     5976
UNK      856
Unk      262
Name: count, dtype: int64

---

Purpose.of.flight
Personal              49448
Instructional         10601
Unknown                6802
Aerial Application     4712
Business               4018
Name: count, dtype: int64

---
```

0.3 3. Data Preparation

[ ]:
```python
# check the no of missing values per column
df.isna().sum()
```

[ ]:
```
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
```

[ ]:

```python
#percentage of missing values per column
df.isna().mean()*100
```

[ ]:

```
Event.Id                   0.000000
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              43.601570
Airport.Name              40.708074
Injury.Severity            1.124999
Aircraft.damage            3.593246
Aircraft.Category         63.677170
Registration.Number        1.554748
Make                       0.070875
Model                      0.103500
Amateur.Built              0.114750
Number.of.Engines          6.844491
Engine.Type                7.982990
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              7.181991
Publication.Date          15.492356
dtype: float64
```

0.3.1 a. Dropping columns

[ ]:
```
#drop columns with more than 30% missing values
columns_to_drop_missing =['Latitude','Longitude', 'Airport.Code','Airport.
↪Name','Aircraft.Category','FAR.Description', 'Schedule','Air.carrier','Broad.
↪phase.of.flight']
df= df.drop(columns=columns_to_drop_missing)
```

[ ]:
```
# drop columns that are not necessary to the analysis
columns_to_drop_unuseful=['Accident.Number','Registration.Number','Report.
↪Status','Publication.Date']
df = df.drop(columns=columns_to_drop_unuseful)
```

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

[ ]:
```
Event.Id                      0
Investigation.Type            0
Event.Date                    0
Location                     52
Country                     226
Injury.Severity            1000
Aircraft.damage            3194
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7096
Purpose.of.flight          6192
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
dtype: int64
```

0.3.2 b. Renaming columns

[ ]:
```
# Rename the columns to easier names
new_column_names = {'Event.Id': 'ID', 'Investigation.Type': 'Type', 'Event.
↪Date':'Date', 'Injury.Severity':'Injury_Severity',
                    'Aircraft.damage':'Damage_type','Amateur.Built':
↪'Amateur_Built', 'Number.of.Engines':'Engines', 'Purpose.of.flight':
↪'Flight_Purpose',
                    'Total.Fatal.Injuries':'Fatal_Injuries','Engine.Type':
↪'Engine_Type', 'Total.Serious.Injuries':'Serious_Injuries',
                   'Total.Minor.Injuries':'Minor_Injuries', 'Total.Uninjured':
↪'Uninjured', 'Weather.Condition':'Weather',}
```

```
df = df.rename(columns=new_column_names)
```

[ ]:
```
#see the new columns names
df.columns
```

[ ]:
```
Index(['ID', 'Type', 'Date', 'Location', 'Country', 'Injury_Severity',
       'Damage_type', 'Make', 'Model', 'Amateur_Built', 'Engines',
       'Engine_Type', 'Flight_Purpose', 'Fatal_Injuries', 'Serious_Injuries',
       'Minor_Injuries', 'Uninjured', 'Weather'],
      dtype='object')
```

[ ]:
```
#Check for unique ID's
df['ID'].value_counts()
```

[ ]:
```
ID
20001212X19172    3
20001214X45071    3
20220730105623    2
20051213X01965    2
20001212X16765    2
                 ..
20001211X14216    1
20001211X14239    1
20001211X14207    1
20001211X14204    1
20221230106513    1
Name: count, Length: 87951, dtype: int64
```

[ ]:
```
#no of duplicated rows
df.duplicated().sum()
```

[ ]:
```
38
```

0.3.3 c. Droping duplicates

[ ]:
```
#drop the duplicated rows
df = df.drop_duplicates()
```

[ ]:
```
df.shape
```

[ ]:
```
(88851, 18)
```

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

[ ]:
```
ID                        0
Type                      0
Date                      0
Location                 52
```

```
Country               226
Injury_Severity       999
Damage_type          3191
Make                   63
Model                  92
Amateur_Built         102
Engines              6081
Engine_Type          7094
Flight_Purpose       6188
Fatal_Injuries      11396
Serious_Injuries    12500
Minor_Injuries      11923
Uninjured            5907
Weather              4491
dtype: int64
```

[ ]:
```
df.head()
```

[ ]:
```
              ID      Type        Date         Location        Country  \
0  20001218X45444  Accident  1948-10-24  MOOSE CREEK, ID  United States
1  20001218X45447  Accident  1962-07-19   BRIDGEPORT, CA  United States
2  20061025X01555  Accident  1974-08-30    Saltville, VA  United States
3  20001218X45448  Accident  1977-06-19       EUREKA, CA  United States
4  20041105X01764  Accident  1979-08-02       Canton, OH  United States

  Injury_Severity Damage_type      Make     Model Amateur_Built  Engines  \
0        Fatal(2)   Destroyed   Stinson     108-3            No      1.0
1        Fatal(4)   Destroyed     Piper  PA24-180            No      1.0
2        Fatal(3)   Destroyed    Cessna      172M            No      1.0
3        Fatal(2)   Destroyed  Rockwell       112            No      1.0
4        Fatal(1)   Destroyed    Cessna       501            No      NaN

     Engine_Type Flight_Purpose  Fatal_Injuries  Serious_Injuries  \
0  Reciprocating       Personal             2.0               0.0
1  Reciprocating       Personal             4.0               0.0
2  Reciprocating       Personal             3.0               NaN
3  Reciprocating       Personal             2.0               0.0
4            NaN       Personal             1.0               2.0

   Minor_Injuries  Uninjured Weather
0             0.0        0.0     UNK
1             0.0        0.0     UNK
2             NaN        NaN     IMC
3             0.0        0.0     IMC
4             NaN        0.0     VMC
```

0.3.4 d. Filling missing values per column

[ ]:
```
#Country
df['Country'].value_counts()/ len(df)*100
```

[ ]:
```
Country
United States                       92.529066
Brazil                               0.420929
Canada                               0.404047
Mexico                               0.402922
United Kingdom                       0.387165
                                       …
Seychelles                           0.001125
Palau                                0.001125
Libya                                0.001125
Saint Vincent and the Grenadines     0.001125
Turks and Caicos Islands             0.001125
Name: count, Length: 219, dtype: float64
```

[ ]:
```
#since Country data is categorical and the mode is at 92.5%, we replace missing␣
↪values with mode
df['Country'].fillna(df['Country'].mode()[0],inplace=True)
```

[ ]:
```
#To avoid bias, fill these columns with uknown 'UKN'
columns_to_fill_uknown␣
↪=['Location','Injury_Severity','Damage_type','Make','Model','Amateur_Built','Engine_Type','
for col in columns_to_fill_uknown:
  df[col].fillna("Unknown",inplace=True)
```

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

[ ]:
```
ID                      0
Type                    0
Date                    0
Location                0
Country                 0
Injury_Severity         0
Damage_type             0
Make                    0
Model                   0
Amateur_Built           0
Engines              6081
Engine_Type             0
Flight_Purpose          0
Fatal_Injuries      11396
Serious_Injuries    12500
Minor_Injuries      11923
Uninjured            5907
```

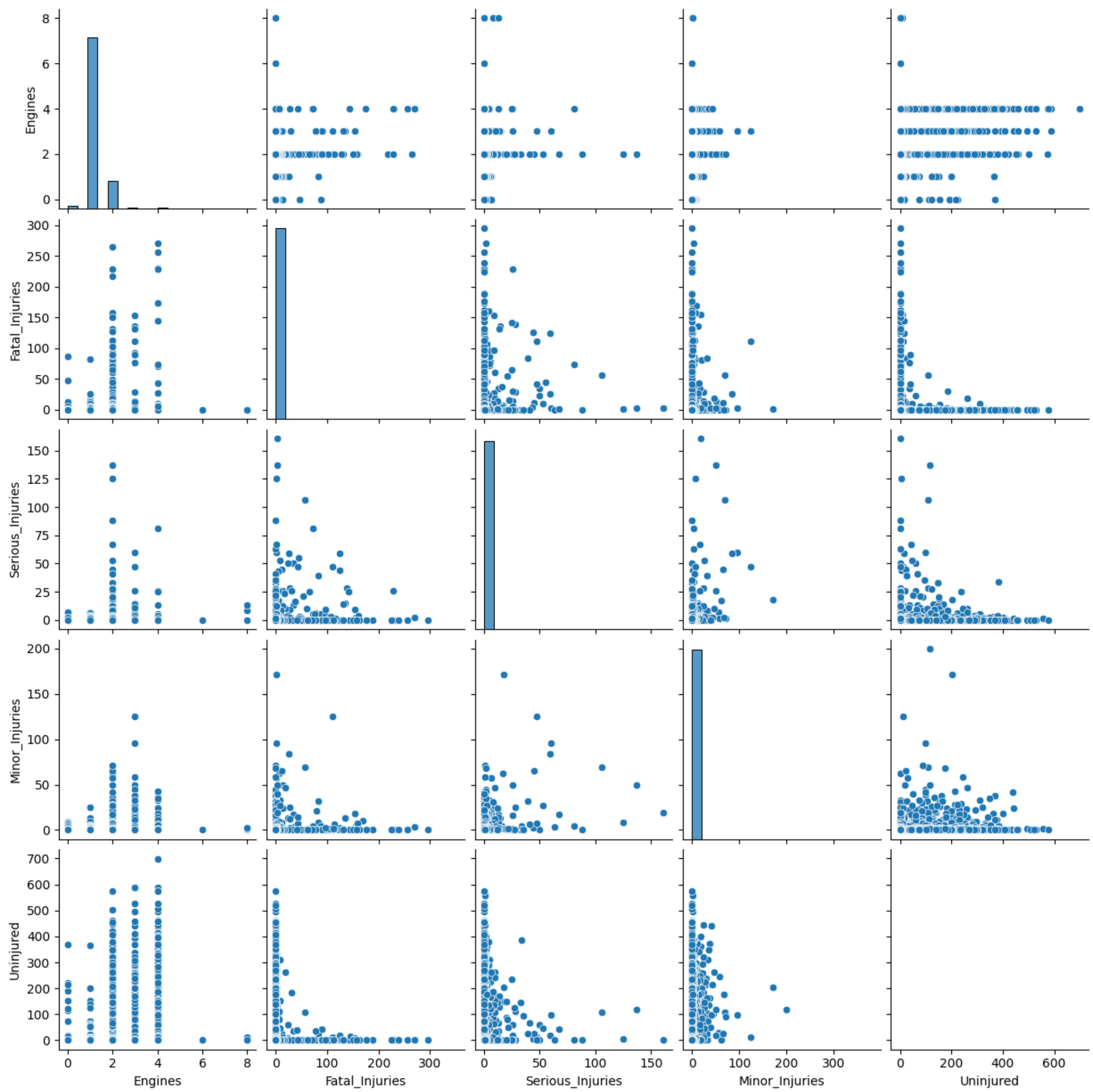
```
Weather                   0
dtype: int64
```

[ ]:
```
#the columns remaining columns have a numerical value, we have to fill them␣
↪with either mean, median, mode or drop rows
df.describe()
```

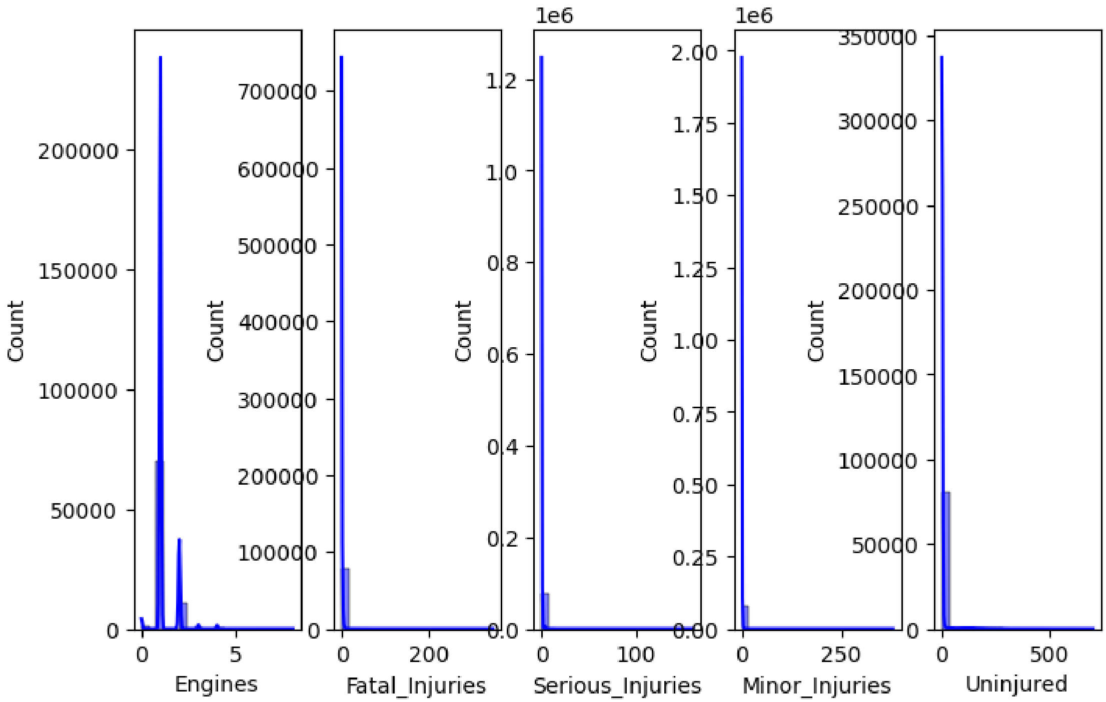
[ ]:
```
            Engines  Fatal_Injuries  Serious_Injuries  Minor_Injuries  \
count  82770.000000    77455.000000      76351.000000    76928.000000
mean       1.146478        0.647757          0.279892        0.357061
std        0.446379        5.487059          1.544285        2.235891
min        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

          Uninjured
count  82944.000000
mean       5.310004
std       27.841800
min        0.000000
25%        0.000000
50%        1.000000
75%        2.000000
max      699.000000
```

[ ]:
```
# pair plots
sns.pairplot(df)
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7897119f9a50>
```



[ ]:
```python
#subplots
numeric_columns=['Engines','Fatal_Injuries','Serious_Injuries','Minor_Injuries','Uninjured']
# Set up the subplots
fig, axes = plt.subplots(1, len(numeric_columns), figsize=(8, 5))
# Plot histograms for each column
for col, ax in zip(numeric_columns, axes):
    sns.histplot(data=df, x=col, bins=20, color='blue', ax=ax, kde=True)
```




```
[ ]: #From the subplots, all the columns are skewed and therefore will fill the
     missing values with median
     for col in numeric_columns:
         df[col].fillna(df[col].median(), inplace=True)
```

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

```
[ ]: ID                  0
     Type                0
     Date                0
     Location            0
     Country             0
     Injury_Severity     0
     Damage_type         0
     Make                0
     Model               0
     Amateur_Built       0
     Engines             0
     Engine_Type         0
     Flight_Purpose      0
     Fatal_Injuries      0
     Serious_Injuries    0
     Minor_Injuries      0
     Uninjured           0
```

```
Weather              0
dtype: int64
```

0.3.5 e. change place holders for some columns

```
[ ]: # Merge different capitalizations of 'Make' together
     df['Make'] = df['Make'].str.title()
     df['Make'].value_counts().head()
```

```
[ ]: Make
     Cessna     27143
     Piper      14869
     Beech       5372
     Boeing      2738
     Bell        2720
     Name: count, dtype: int64
```

```
[ ]: # convert Amateur Built into boolean
     df['Amateur_Built'].replace(to_replace = ['Yes', 'Y'], value = True, inplace =␣
      ↪True, regex = False)
     df['Amateur_Built'].replace(to_replace = ['No', 'N'], value = False, inplace =␣
      ↪True, regex = False)
     df['Amateur_Built'].value_counts()
```

```
[ ]: Amateur_Built
     False      80277
     True        8472
     Unknown      102
     Name: count, dtype: int64
```

```
[ ]: # Remove amount of injuries in 'Injury_Severity' as this is aleady in another␣
      ↪column
     df['Injury_Severity'] = df['Injury_Severity'].str.split('(').str[0]
     df['Injury_Severity'].value_counts()
```

```
[ ]: Injury_Severity
     Non-Fatal      67339
     Fatal          17814
     Incident        2213
     Unknown          999
     Minor            217
     Serious          173
     Unavailable       96
     Name: count, dtype: int64
```

```
[ ]: # Merge weather condition unknowns
```

```
df['Weather'].replace(to_replace = ['Unk', 'UNK'], value = 'Unknown', inplace =␣
↪True, regex = False)
df['Weather'].value_counts()
```

[ ]:
```
Weather
VMC        77268
IMC         5975
Unknown     5608
Name: count, dtype: int64
```

0.3.6 f. Convert date column to the appropriate format

[ ]:
```
# changing date type to the appropriate format
df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')
# create a new column Year
df['Year'] = df['Date'].dt.year
# create a new column Month
df['Month'] = df['Date'].dt.month
```

0.3.7 g. Introduce a column for seasons

[ ]:
```
#  creating a column for seasons using the US because it has the majority

seasons = {
    12: 'Winter', 1: 'Winter', 2: 'Winter',
    3: 'Spring', 4: 'Spring', 5: 'Spring',
    6: 'Summer', 7: 'Summer', 8: 'Summer',
    9: 'Fall', 10: 'Fall', 11: 'Fall'
}

df['Season'] = df['Month'].map(seasons)
```

[ ]:
```
df.head()
```

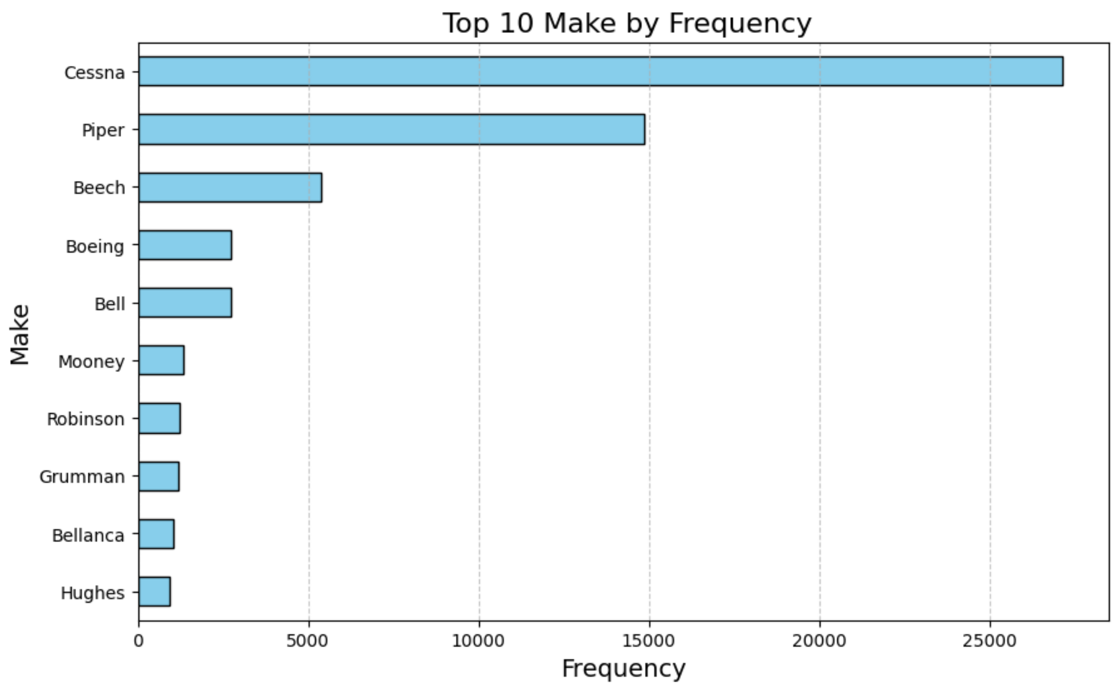
[ ]:
```
                ID      Type       Date          Location        Country  \
0  20001218X45444  Accident 1948-10-24  MOOSE CREEK, ID  United States
1  20001218X45447  Accident 1962-07-19   BRIDGEPORT, CA  United States
2  20061025X01555  Accident 1974-08-30    Saltville, VA  United States
3  20001218X45448  Accident 1977-06-19       EUREKA, CA  United States
4  20041105X01764  Accident 1979-08-02       Canton, OH  United States

  Injury_Severity Damage_type      Make     Model Amateur_Built  …  \
0           Fatal   Destroyed   Stinson     108-3         False  …
1           Fatal   Destroyed     Piper  PA24-180         False  …
2           Fatal   Destroyed    Cessna      172M         False  …
3           Fatal   Destroyed  Rockwell       112         False  …
4           Fatal   Destroyed    Cessna       501         False  …
```

```
     Engine_Type Flight_Purpose Fatal_Injuries  Serious_Injuries  \
0  Reciprocating       Personal            2.0               0.0
1  Reciprocating       Personal            4.0               0.0
2  Reciprocating       Personal            3.0               0.0
3  Reciprocating       Personal            2.0               0.0
4        Unknown       Personal            1.0               2.0

   Minor_Injuries  Uninjured  Weather  Year  Month  Season
0             0.0        0.0  Unknown  1948     10    Fall
1             0.0        0.0  Unknown  1962      7  Summer
2             0.0        1.0      IMC  1974      8  Summer
3             0.0        0.0      IMC  1977      6  Summer
4             0.0        0.0      VMC  1979      8  Summer

[5 rows x 21 columns]
```

0.4 3. Data Analysis

0.4.1 a. Frequency of Accidents and Incidents by Make

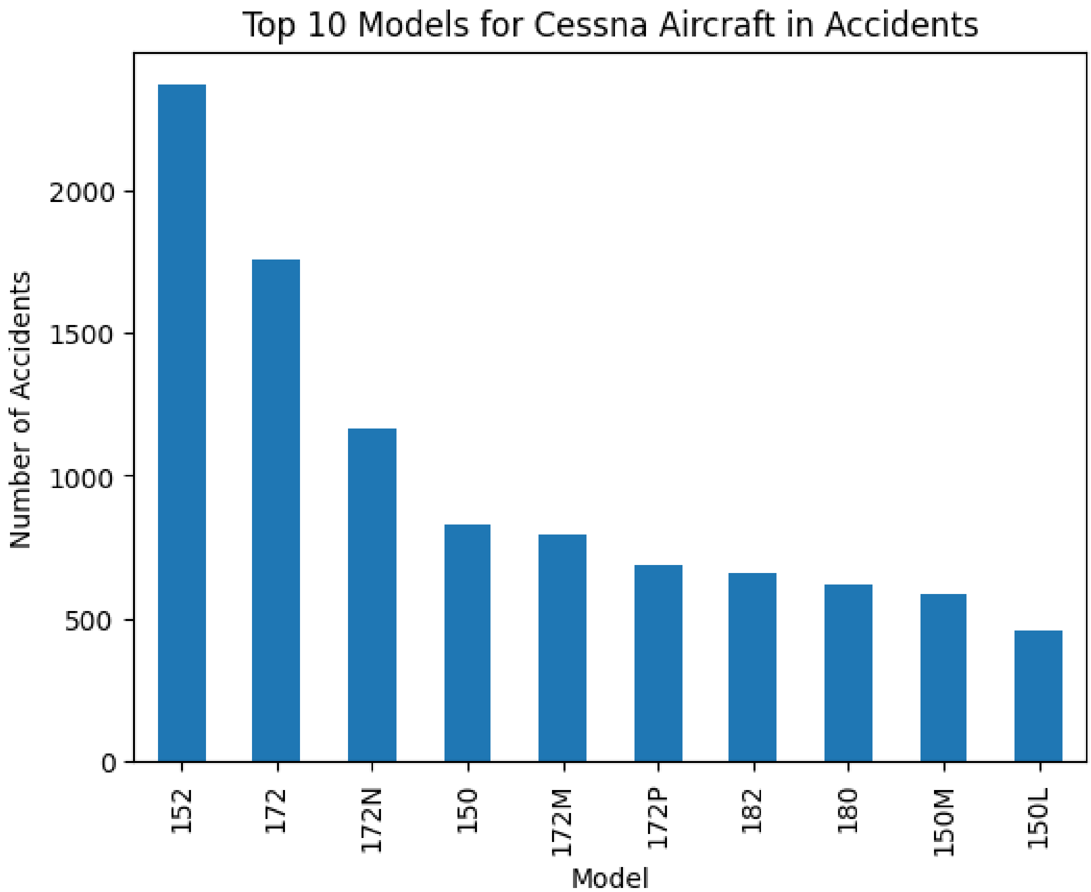
[ ]:
```
make_freq=df['Make'].value_counts().head(10)
# create a horizontal bar chart
plt.figure(figsize=(10, 6))
make_freq.plot(kind='barh', color='skyblue', edgecolor='black')

# Customize the plot
plt.title(f" {'Top 10 Make'} by Frequency", fontsize=16)
plt.xlabel("Frequency", fontsize=14)
plt.ylabel('Make', fontsize=14)
# Invert the y-axis to show the highest value at the top
plt.gca().invert_yaxis()
plt.grid(axis='x', linestyle='--', alpha=0.7)
# Save as PNG
plt.savefig('plot.png', dpi=300)
plt.show()
```

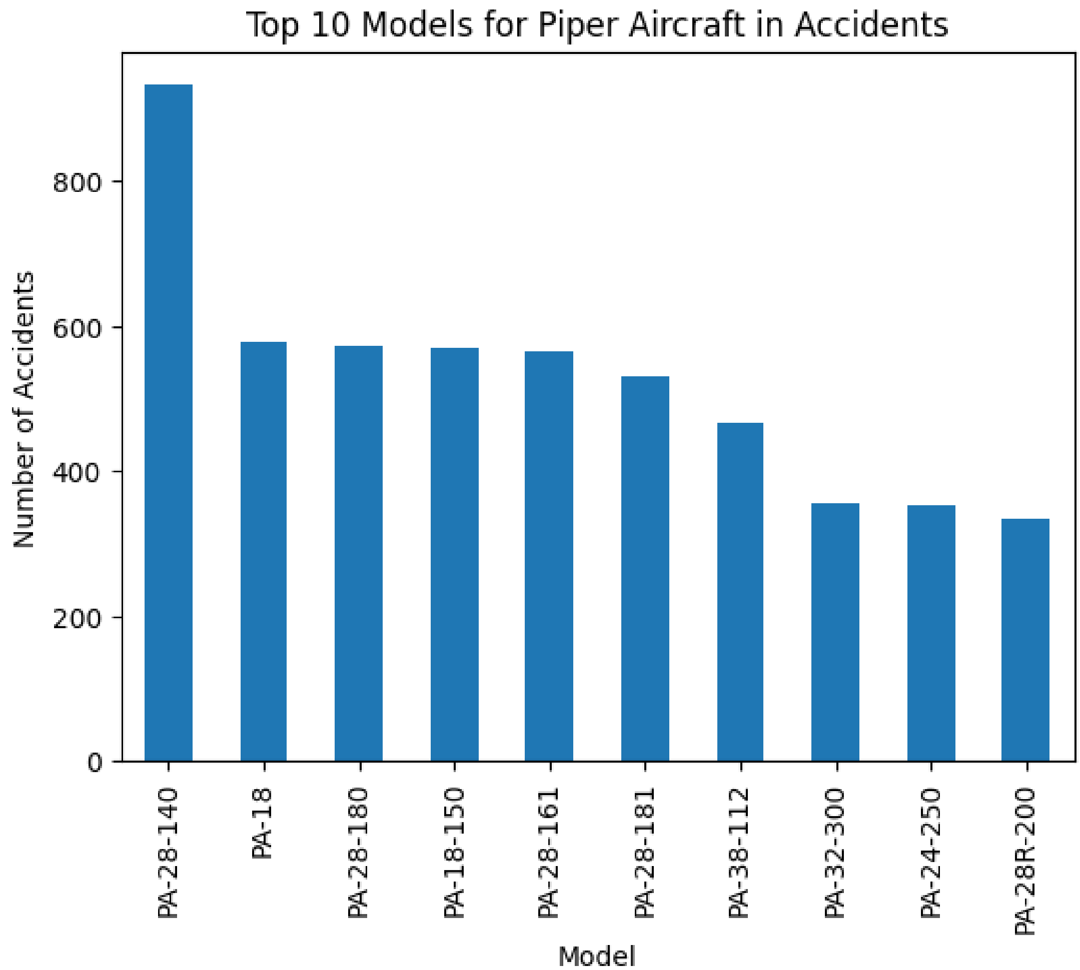


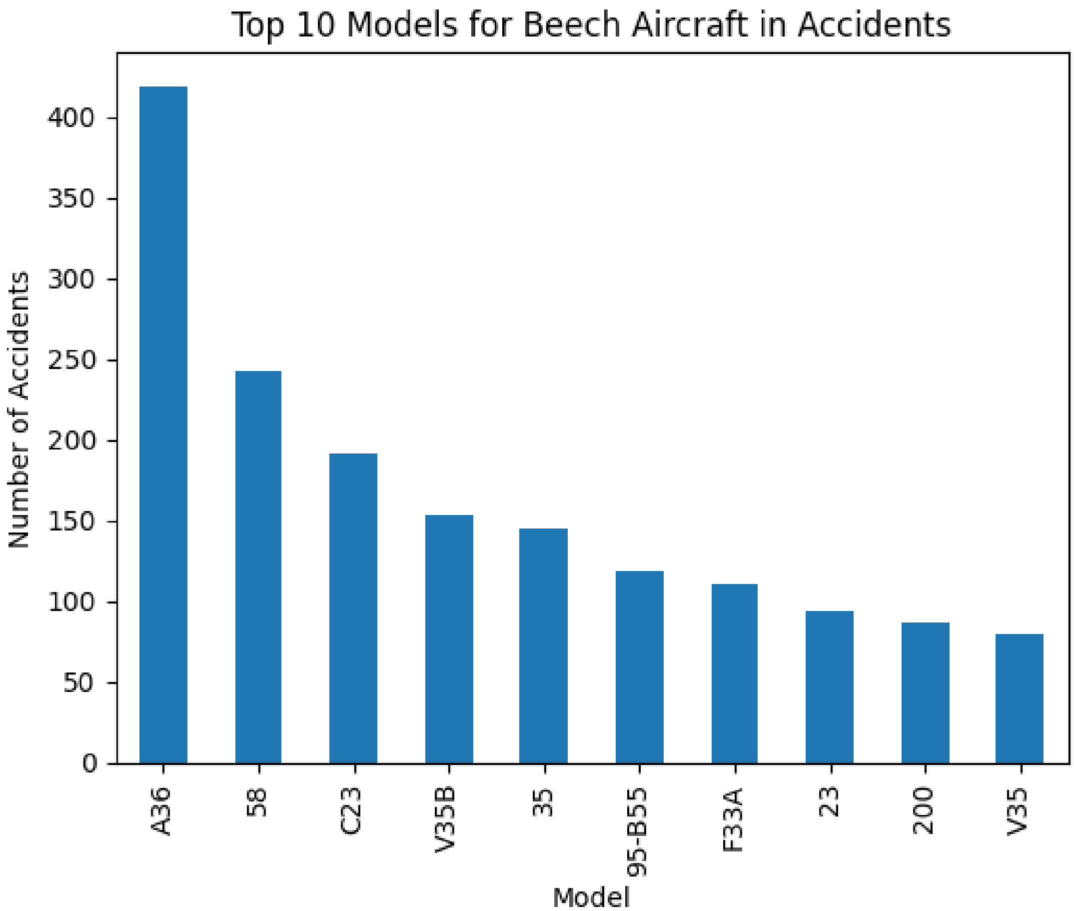
Observation The Cessna aircraft make tops with the number of accidents.

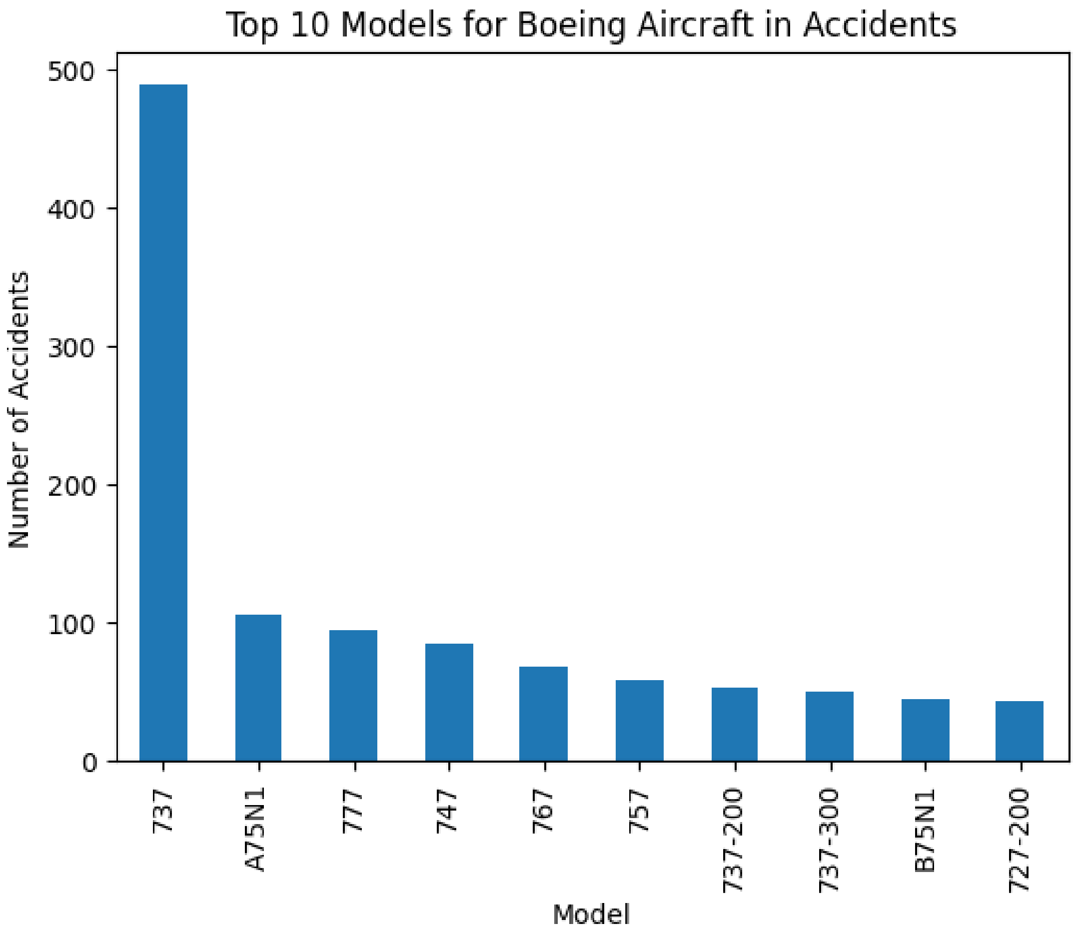
```
[ ]: # visualize the 10 makes by model with the function below
def visualize_top_10_models_by_make(self):
        top_10_makes = self['Make'].value_counts().head(10).index

        for make in top_10_makes:
            make_data = self[self['Make'] == make]
            top_10_models = make_data['Model'].value_counts().head(10)
            top_10_models.plot(kind='bar', title=f'Top 10 Models for {make}␣
 ↪Aircraft in Accidents')
            plt.xlabel('Model')
            plt.ylabel('Number of Accidents')
            plt.show()
        return top_10_makes
```

```
[ ]: # call the function with the parameter df
visualize_top_10_models_by_make(df)
```

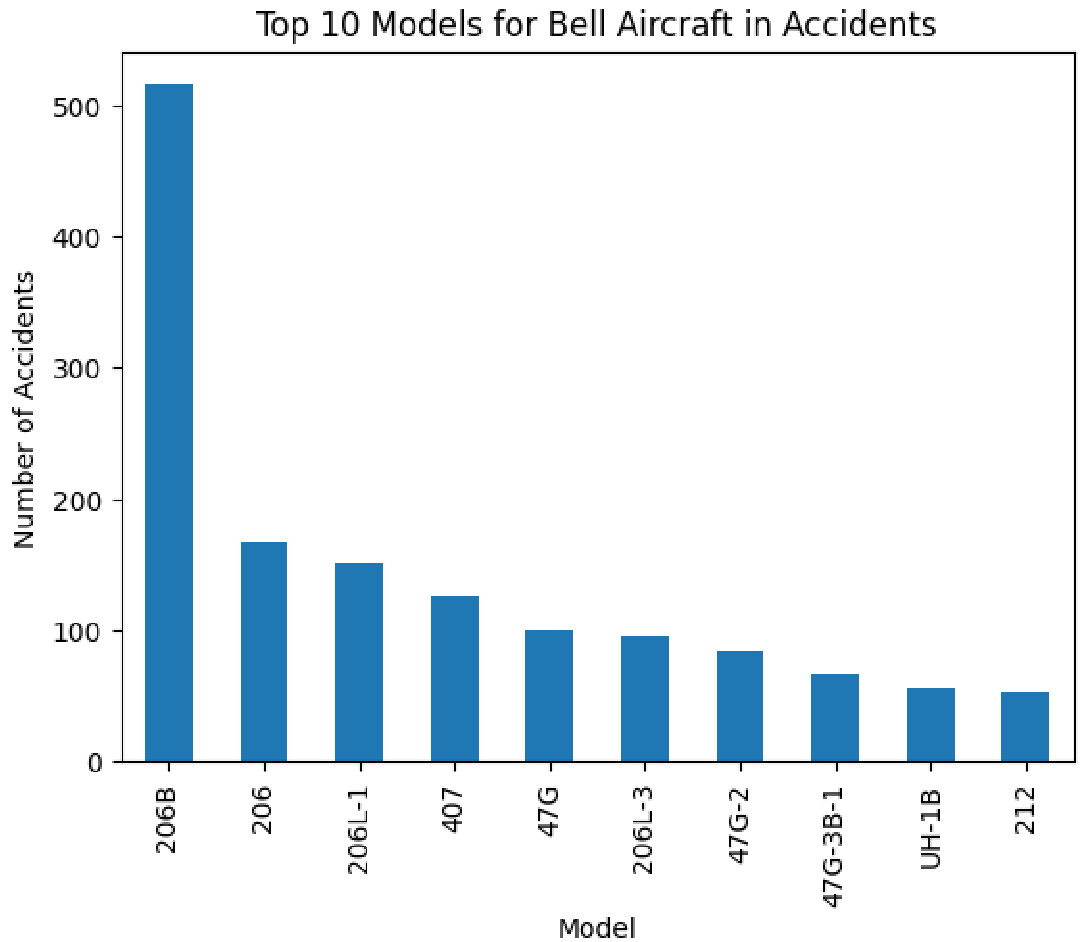

Top 10 Models for Cessna Aircraft in Accidents
Number of Accidents
0
500
1000
1500
2000
152
172
172N
150
172M
172P
182
180
150M
150L
Model

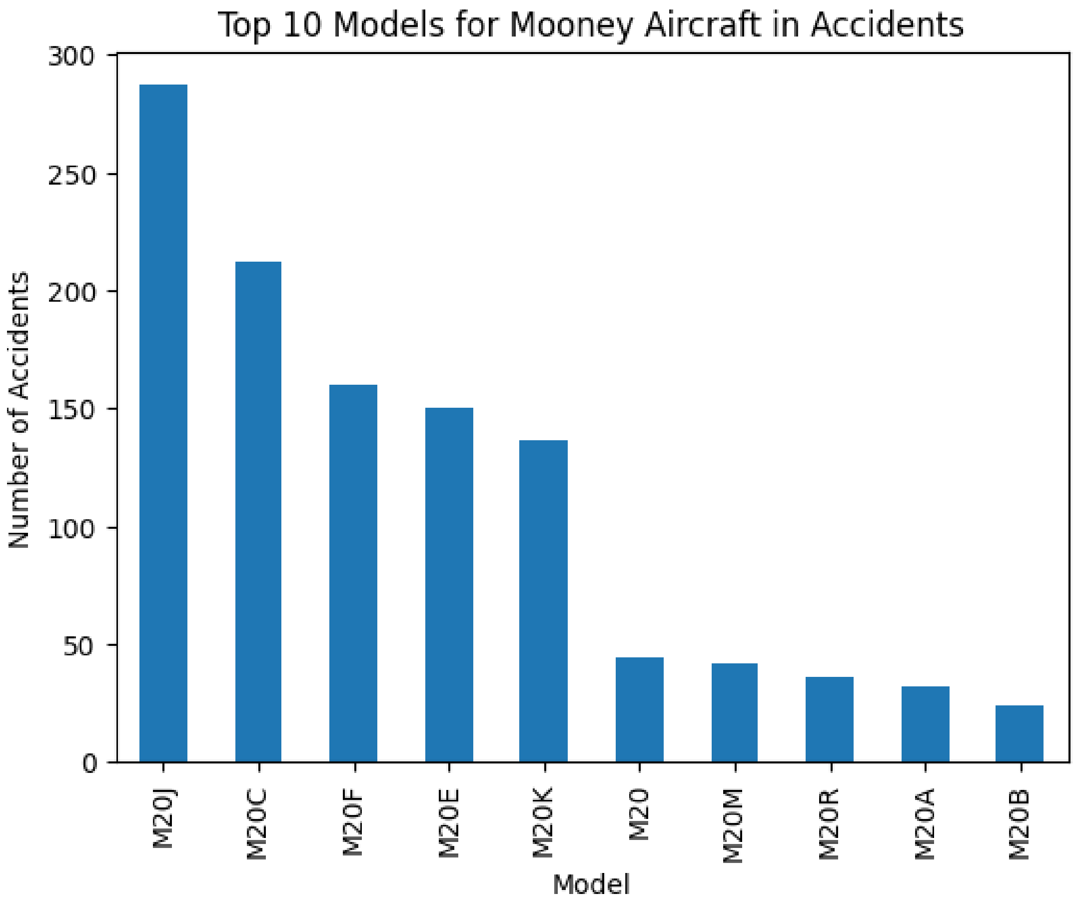

Top 10 Models for Piper Aircraft in Accidents
Number of Accidents
0
200
400
600
800
PA-28-140
PA-18
PA-28-180
PA-18-150
PA-28-161
PA-28-181
PA-38-112
PA-32-300
PA-24-250
PA-28R-200
Model

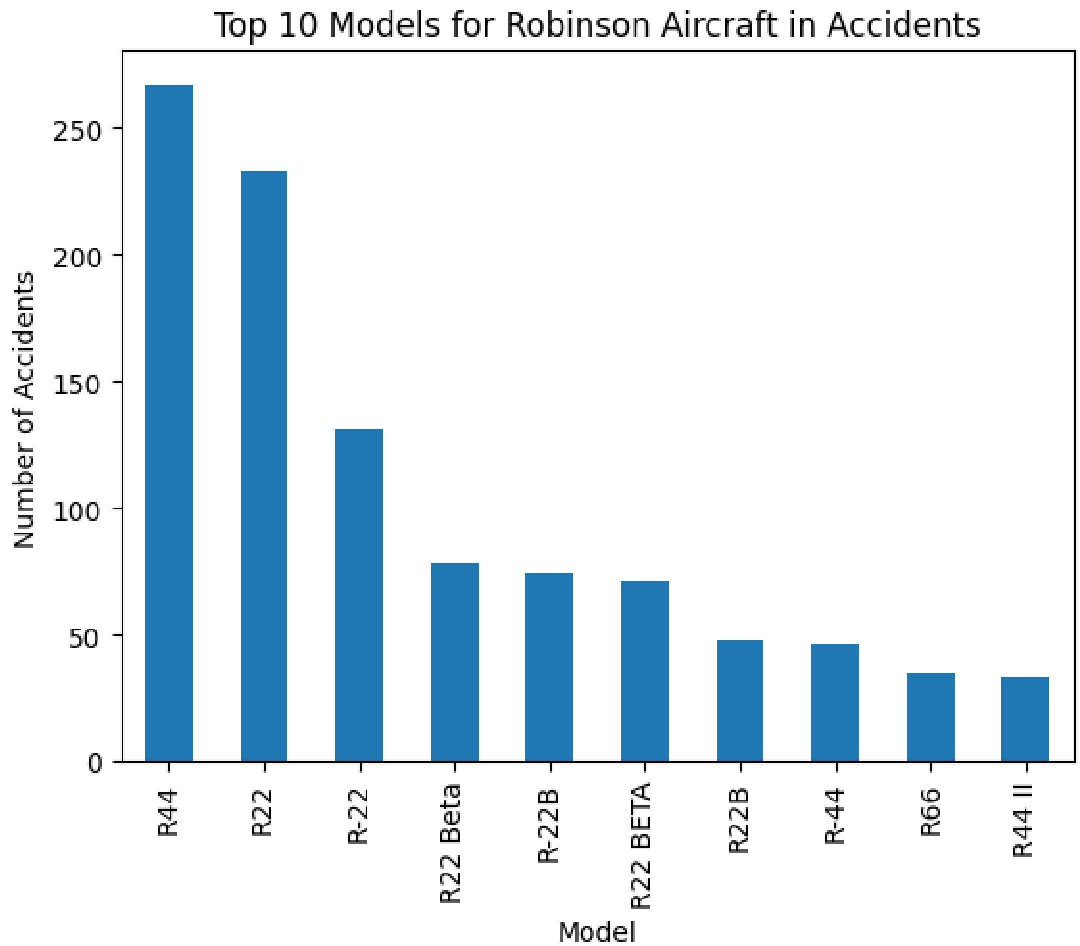

Top 10 Models for Beech Aircraft in Accidents
Number of Accidents
0
50
100
150
200
250
300
350
400
A36
58
C23
V35B
35
95-B55
F33A
23
200
V35
Model

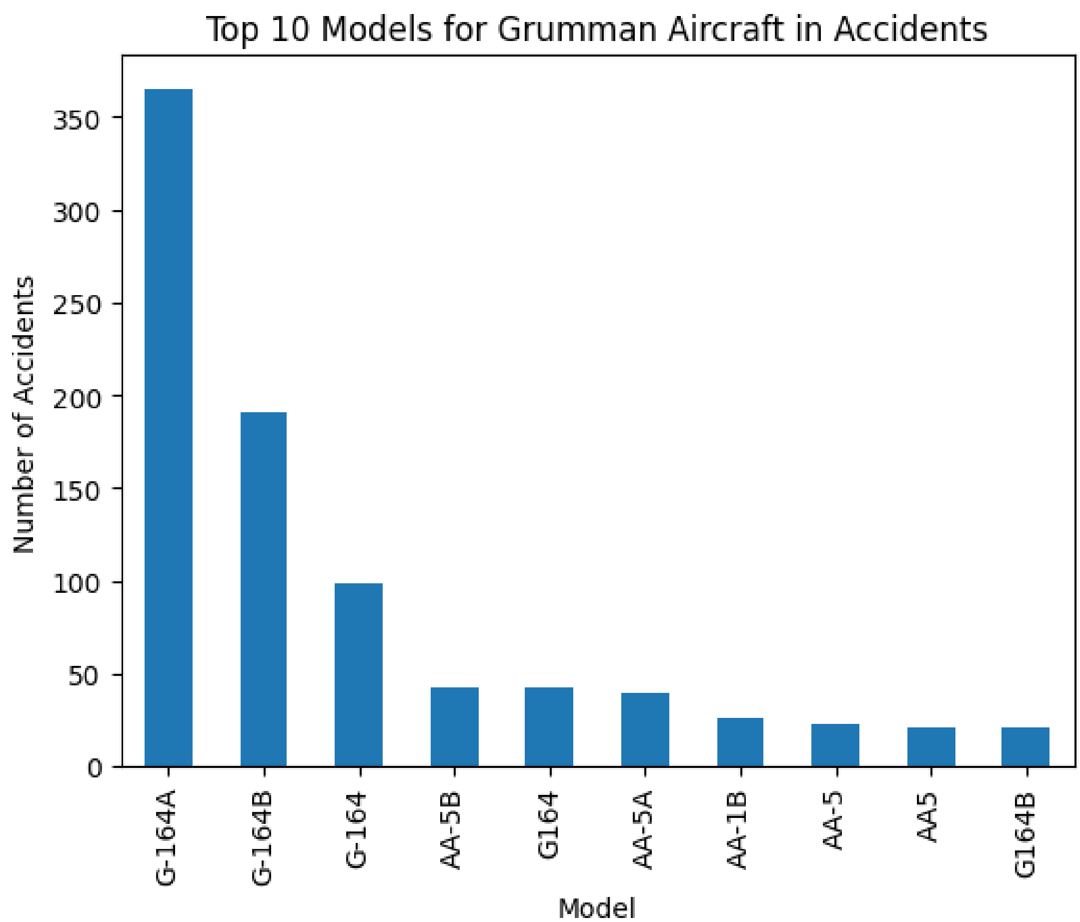


Top 10 Models for Boeing Aircraft in Accidents
Number of Accidents
0
100
200
300
400
500
737
A75N1
777
747
767
757
737-200
737-300
B75N1
727-200
Model

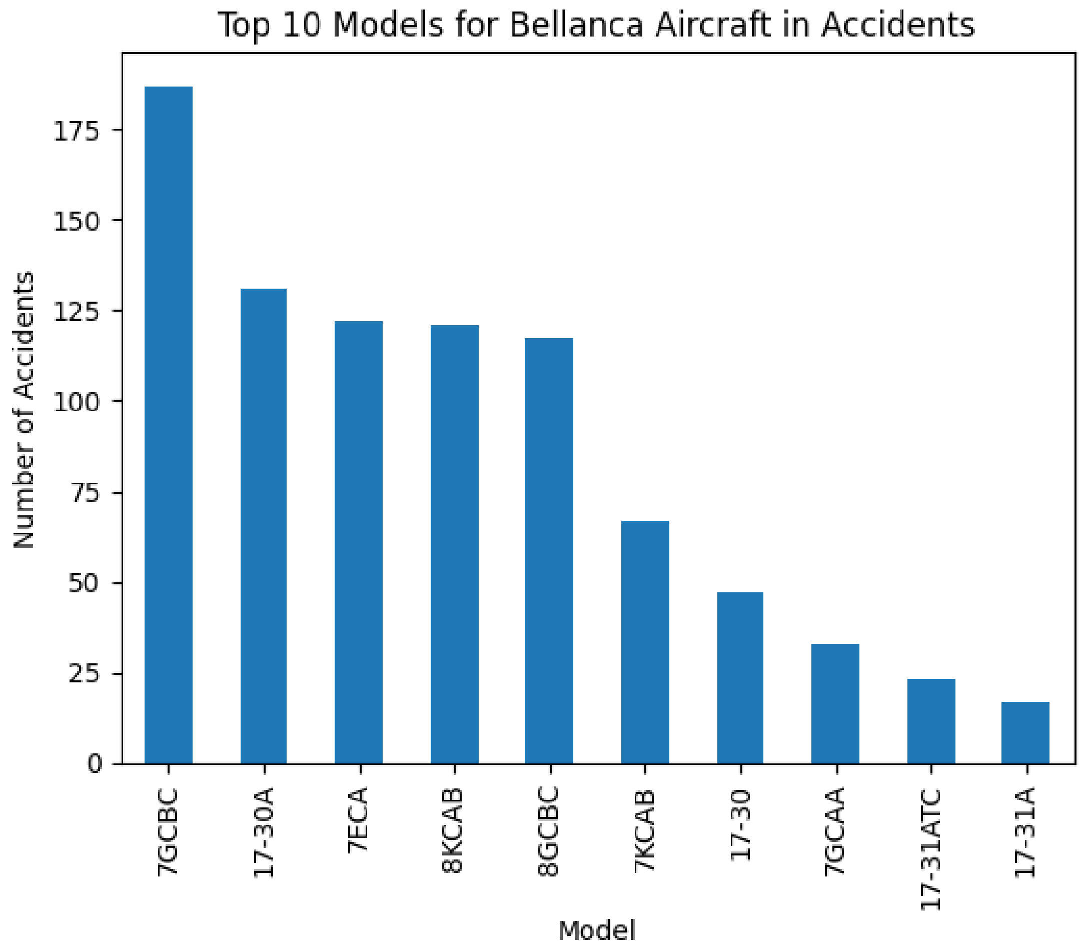

Top 10 Models for Bell Aircraft in Accidents
Number of Accidents
0
100
200
300
400
500
206B
206
206L-1
407
47G
206L-3
47G-2
47G-3B-1
UH-1B
212
Model

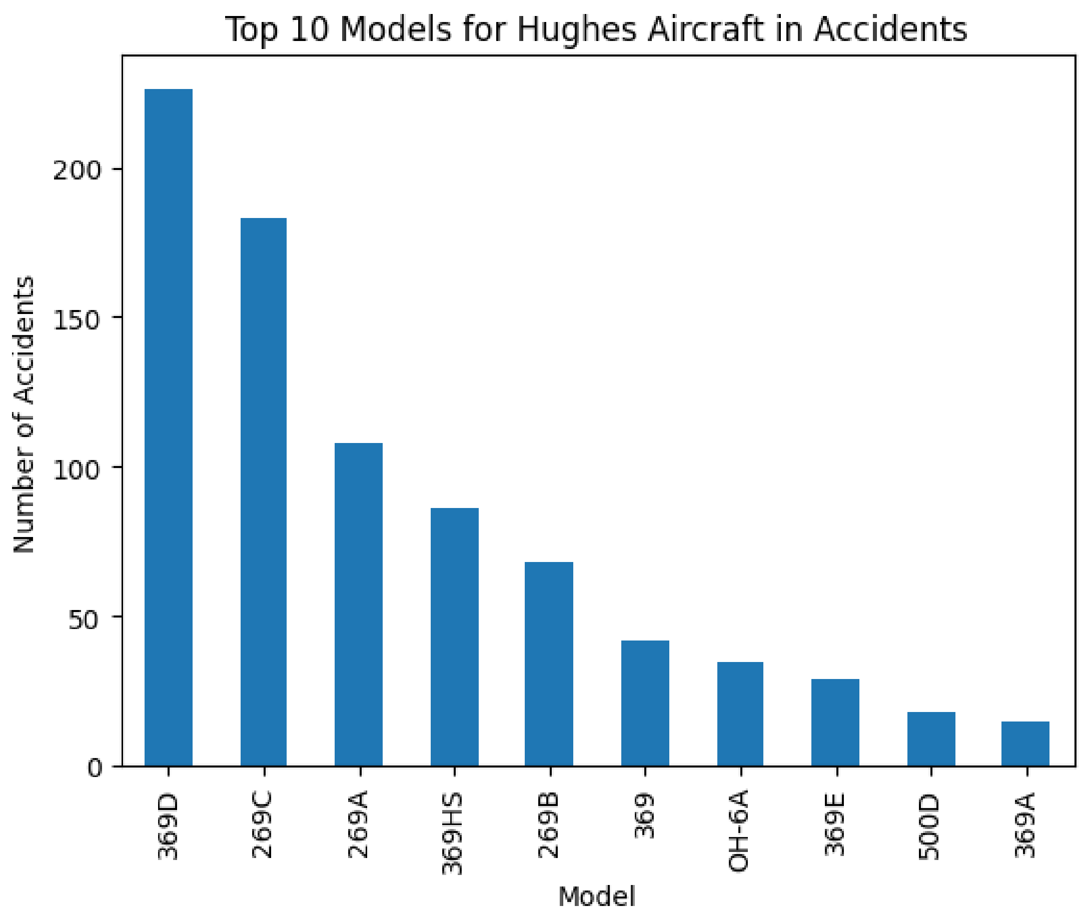

Top 10 Models for Mooney Aircraft in Accidents
Number of Accidents
300
250
200
150
100
50
0
M20J
M20C
M20F
M20E
M20K
M20
M20M
M20R
M20A
M20B
Model


Top 10 Models for Robinson Aircraft in Accidents
Number of Accidents
0
50
100
150
200
250
R44
R22
R-22
R22 Beta
R-22B
R22 BETA
R22B
R-44
R66
R44 II
Model



Top 10 Models for Grumman Aircraft in Accidents
Number of Accidents
0
50
100
150
200
250
300
350
G-164A
G-164B
G-164
AA-5B
G164
AA-5A
AA-1B
AA-5
AA5
G164B
Model


Top 10 Models for Bellanca Aircraft in Accidents
Number of Accidents
0
25
50
75
100
125
150
175
7GCBC
17-30A
7ECA
8KCAB
8GCBC
7KCAB
17-30
7GCAA
17-31ATC
17-31A
Model



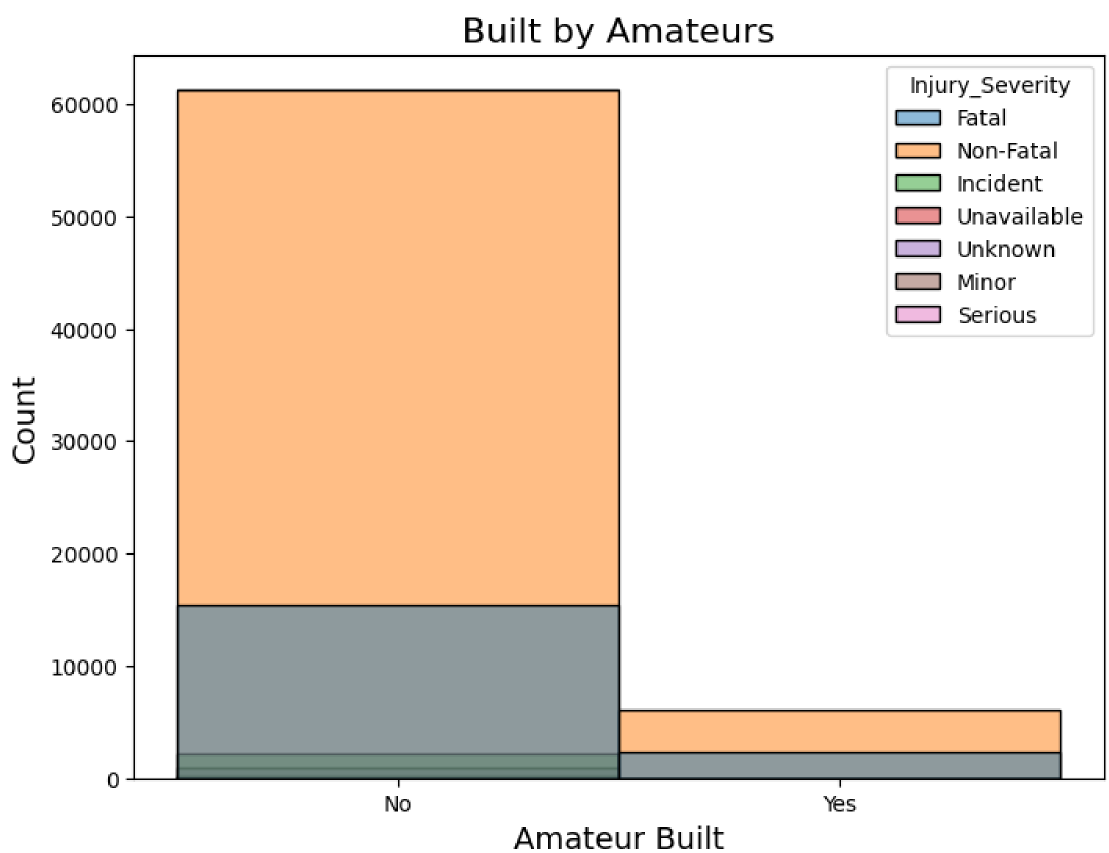

```
[ ]: Index(['Cessna', 'Piper', 'Beech', 'Boeing', 'Bell', 'Mooney', 'Robinson',
            'Grumman', 'Bellanca', 'Hughes'],
           dtype='object', name='Make')
```

Observations Breaking down the data, we identify the top Aircraft Makes with high accident involvement.Within these Makes, we then identify the primary models that consistently stand out in terms of accidents.

0.4.2 b. Built by amateurs

```
[ ]: # Histogram for Amateur_Built column
plt.figure(figsize=(8, 6))
# Remove rows where 'Amateur_Built' is 'Unknown'
df_Amateur = df[df['Amateur_Built'] != 'Unknown']
sns.histplot(data=df_Amateur, x='Amateur_Built', stat='count', bins=2,␣
 ↪hue='Injury_Severity', discrete=True)
# Add title and labels
plt.title('Built by Amateurs', fontsize=16)
```

```
plt.xlabel('Amateur Built', fontsize=14)
plt.ylabel('Count', fontsize=14)
# Set custom x-axis labels as True/False
plt.xticks(ticks=[0, 1], labels=['No', 'Yes'])
# Save as PNG
plt.savefig('plot_Amateur_built.png', dpi=300)
# Show the plot
plt.show()
```

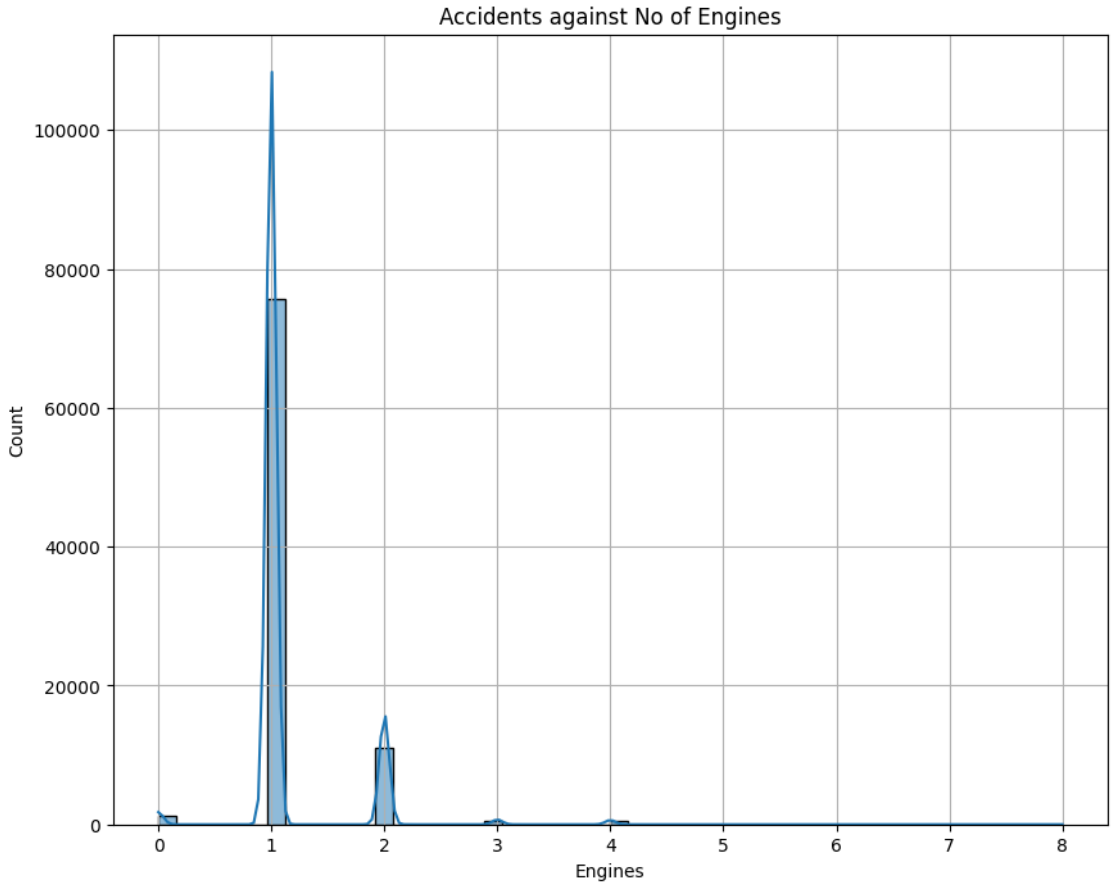


Observation Aircraft that were not Amateur built were involved in more accidents

0.4.3 c. Number of engines

[ ]:
```
# histgram
plt.figure(figsize=(10,8))
sns.histplot(data=df, x='Engines', bins=50,  kde=True, palette='bright')
plt.title('Accidents against No of Engines')
plt.grid()
# Save as PNG
plt.savefig('plot_Engine_numbers.png', dpi=300)
```

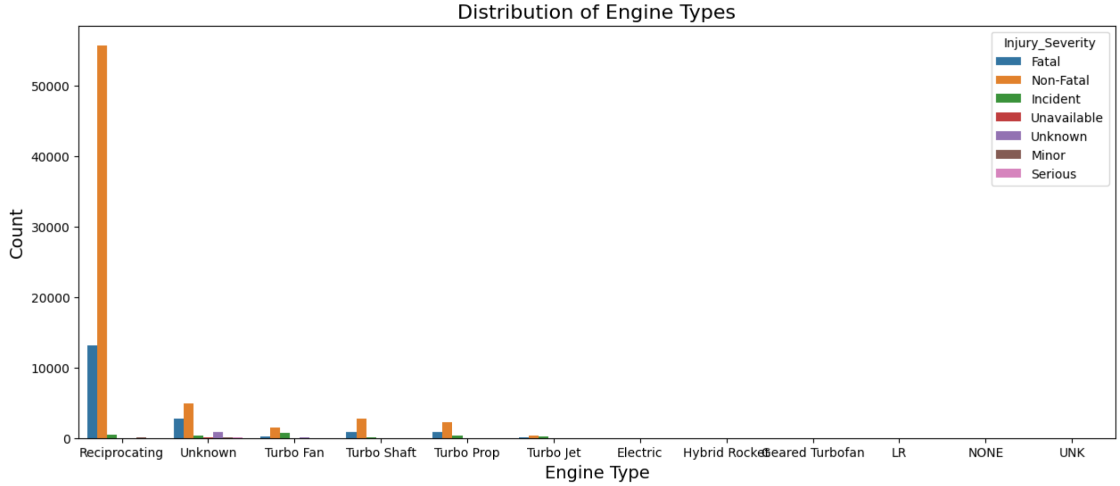
```
plt.show()
```



The plot shows that aircrafts with one engine were involved in majority of the accidents

0.4.4 e. Engine type

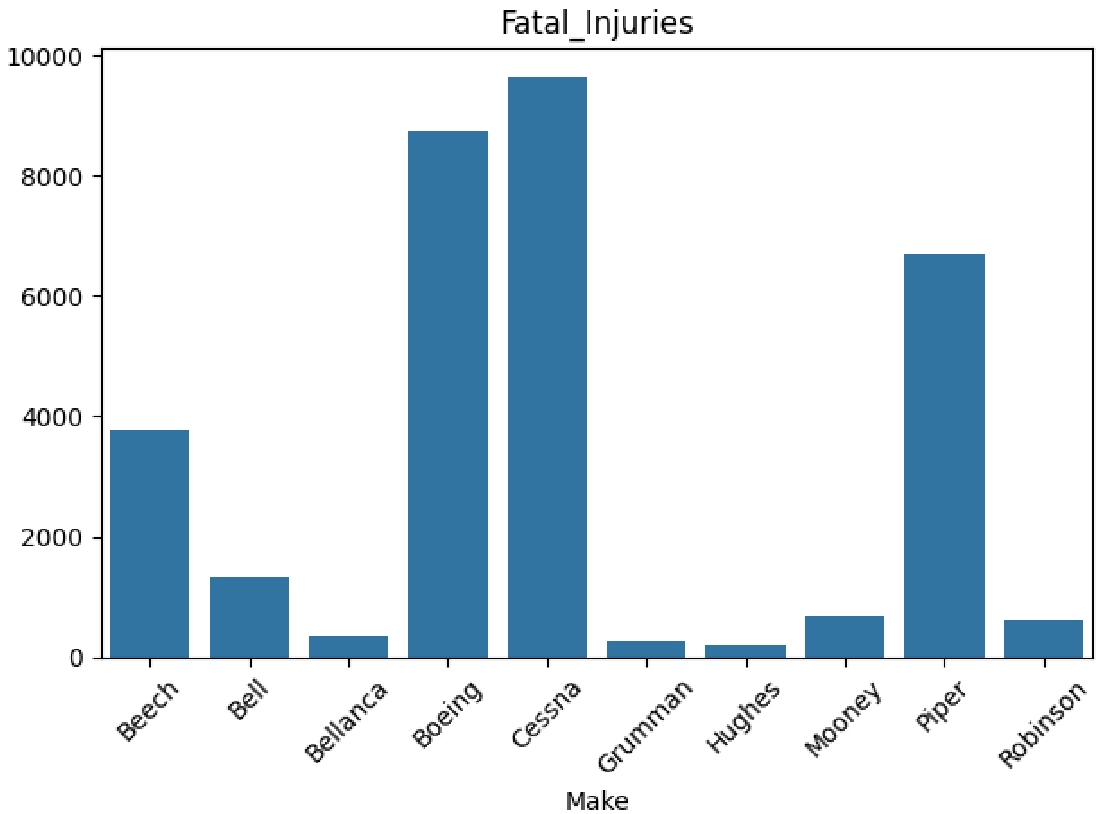
[ ]:
```
# Bar plot for engine types
plt.figure(figsize=(15, 6))
sns.countplot(data=df, x='Engine_Type', hue='Injury_Severity')
# title and labels
plt.title('Distribution of Engine Types', fontsize=16)
plt.xlabel('Engine Type', fontsize=14)
plt.ylabel('Count', fontsize=14)
# Save as PNG
plt.savefig('plot_Engine_type.png', dpi=300)
plt.show()
```



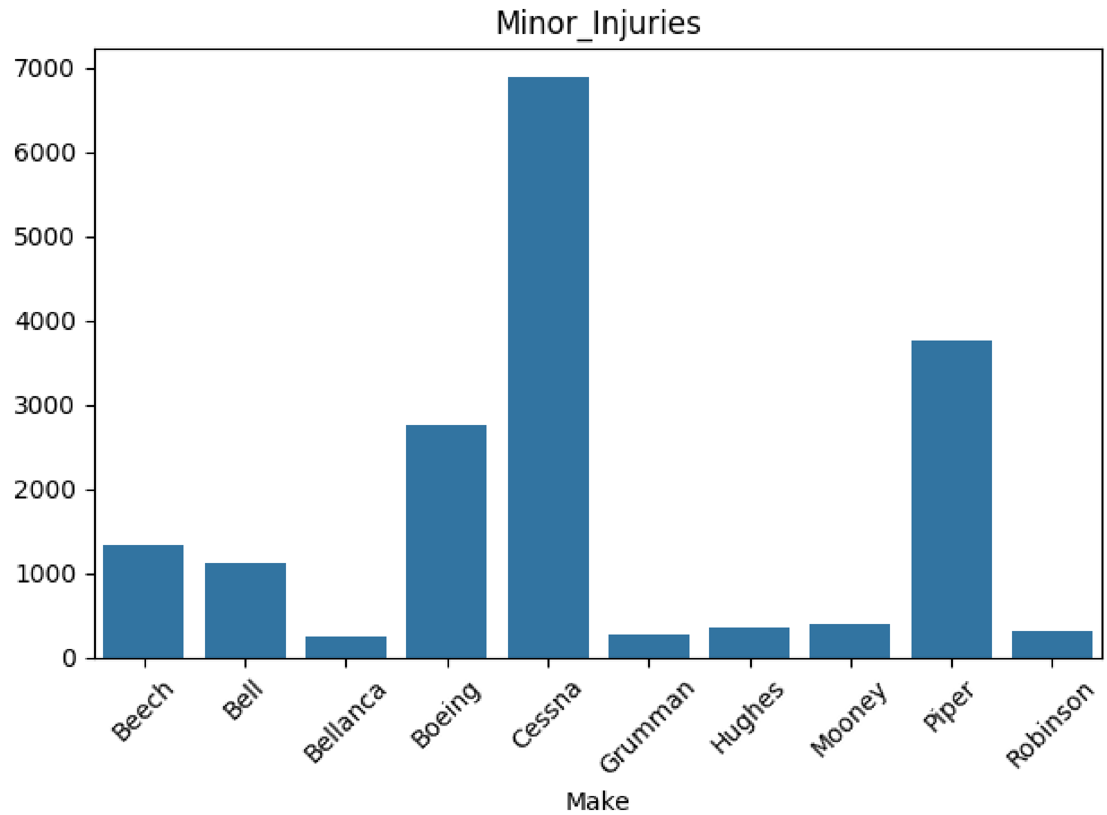
Observation The data shows that aircrafts with Reciprocating engines were involved in many accidents where the injury severity was mainly non fatal and quite a number being fatal.

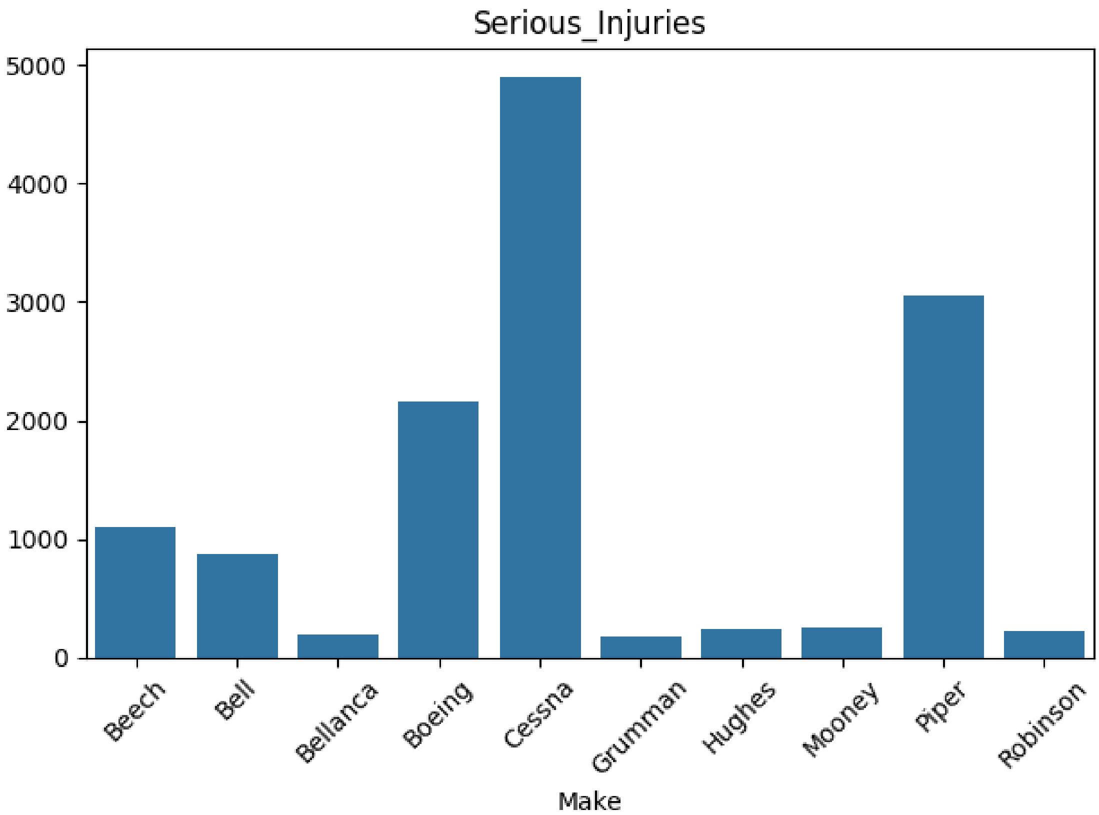
0.4.5 f. Injuries vs Make

[ ]:
```
top10_make=df["Make"].value_counts().head(10)
injuries=␣
↪['Fatal_Injuries',        'Serious_Injuries',        'Minor_Injuries',        'Uninjured']
df_injuries = df[['Make'] +␣
↪['Fatal_Injuries',        'Serious_Injuries',        'Minor_Injuries',        'Uninjured']]
top_10_make_injuries= df_injuries[df['Make'].isin(top10_make.index)]
inj_pivot= pd.
↪pivot_table(top_10_make_injuries,values=injuries,columns='Make',aggfunc='sum')
for i in range(0,len(inj_pivot)):
    sns.barplot(x=inj_pivot.iloc[i].index,y=inj_pivot.iloc[i].values)
    plt.title(inj_pivot.index[i])
    plt.xticks(rotation=45)
    # avoid layout issues
    plt.tight_layout()
    # Generate a unique filename for each plot
    filename = f"injuries_by_make_{i}.png".replace(" ", "_")
    # Save as Png
    plt.savefig(filename, dpi=300)
    plt.show()
```

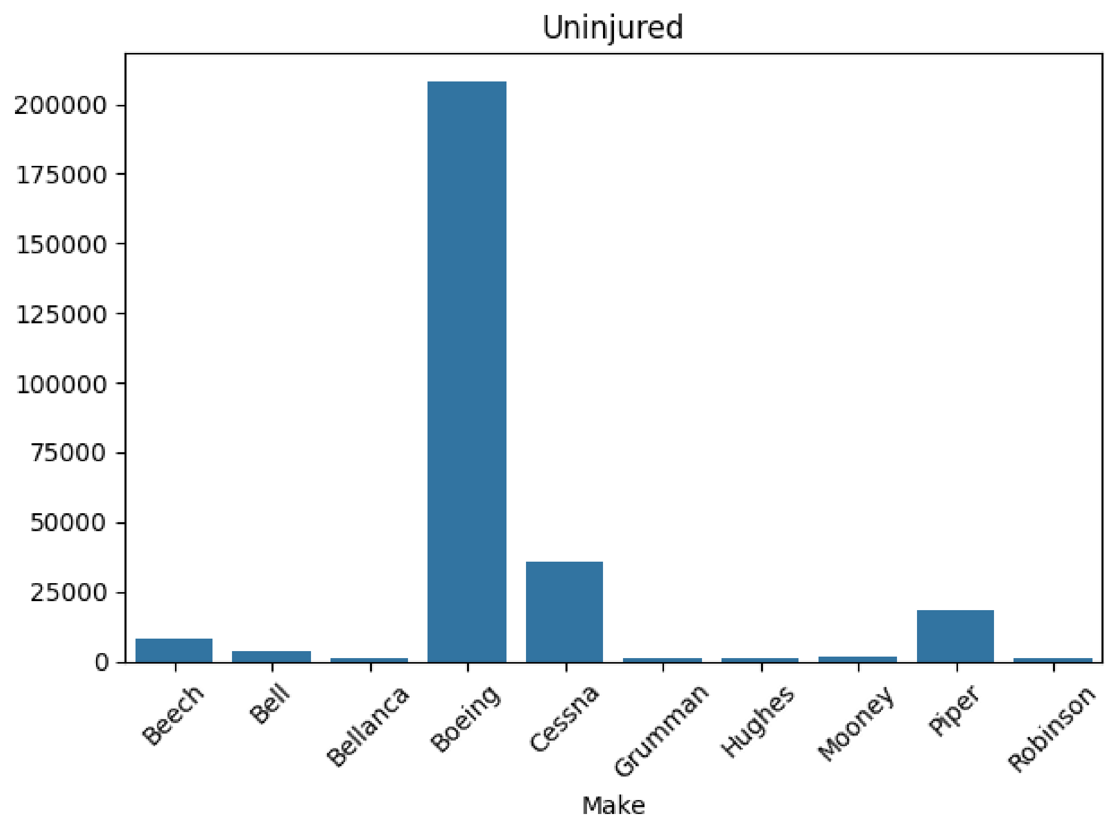


Fatal_Injuries
10000
8000
6000
4000
2000
0
Beech
Bell
Bellanca
Boeing
Cessna
Grumman
Hughes
Mooney
Piper
Robinson
Make


Minor_Injuries
7000
6000
5000
4000
3000
2000
1000
0
Beech
Bell
Bellanca
Boeing
Cessna
Grumman
Hughes
Mooney
Piper
Robinson
Make



Serious_Injuries
5000
4000
3000
2000
1000
0
Beech
Bell
Bellanca
Boeing
Cessna
Grumman
Hughes
Mooney
Piper
Robinson
Make



This data suggests that; 1. There are higher incidence of injuries in Cessna aircraft accidents. 2. Passengers aboard Boeing aircraft experienced a significantly higher likelihood of survival from the Uninjured plot.

0.5 4. Conclusion

From the data analysis, the following observations are made;

1. Cessna aircraft were involved in the highest number of accidents. This could be due to Cessna's large market share and therefore, warrants a closer inspection of specific models and their safety records.

2. Aircrafts with one engine were disproportionately involved in fatal accidents. it is implied that lack of engine redundancy increases risk during engine failure.

3. Amateur-built aircraft were involved in fewer accidents compared to professionally built ones. While fewer accidents may suggest better outcomes, the data may reflect the lower overall usage or different operational patterns of amateur-built aircraft.

4. Aircraft with reciprocating engines were involved in the most accidents. Reciprocating engines are usually used in general aviation aircraft, which may contribute to higher accident rates due to operational factors like outdated runways.

0.6 5. Recommendations

1. Conduct a thorough safety audit of Cessna models.Focus on newer models or those with a strong safety track record due to its large market share.
2. Prioritize acquiring aircraft with more than one engine for redundancy and safety in emergencies.
3. Amateur-built aircraft have fewer recorded accidents due to their limited use and operational scope. They should be avoided unless they are professionally inspected and meet rigorous safety standards.
4. Avoid reciprocating engine and shift focus to other engines, which are more reliable.

In general, for the company's aviation expansion, focus on professionally built, more than one engine and avoid reciprocating engine aircrafts. Conduct detailed assessments of manufacturers like Cessna to identify models with strong safety records. This strategy will minimize risk and align with the company's goal of safe and sustainable growth in the aviation industry.