

Business Understanding

- This Project analyzes the Aviation Accident Database from the National Transportation Safety Board which includes accident data from 1962-2023.
- The data can be found here: <https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses> (<https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses>)
- We are determining if the Personal and Public airline industry is a viable business option for expansion.
- Analysis shows that while all planes crash, certain Models are safer and therefore reduce liability and risk. Analysis of Weather, severity of damage to the plane, phase of flight, and total fatal injuries can be used to determine the best options for the business.

Data Understanding

- This Project analyzes the Aviation Accident Database from the National Transportation Safety Board which includes accident data from 1962-2023.
- The CSV file was downloaded from Kaggle. This is a public website with various datasets. This is effectively all US crash data since the 60's.
- There are 88889 entries with 30 different attributes for each entry. Attributes include Make and Model of plane, purpose of flight, aircraft damage, location of crash, and aircraft damage to name a few.

Data Preperation

- Start by importing libraries for analysis and data cleaning
- Checking for whitespaces and empty cells
- Removing NaN values and duplicates from df
- Creating new df 'df_1985' for analysis

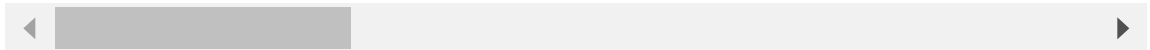
```
In [1]: 1 # import libraries for data analysis
        2 import pandas as pd
        3 import numpy as np
        4 import seaborn as sns
        5 import matplotlib.pyplot as plt
        6
        7 %matplotlib inline
```

```
In [2]: 1 #loading the AviationData.csv file
        2 # using encoding 'Latin-1' to handle values
        3 df = pd.read_csv('data/AviationData.csv',
        4                   encoding='Latin-1',
        5                   low_memory=False,
        6                   dtype=str)
        7 df.head()
```

Out[2]:

| | Event.Id | Investigation.Type | Accident.Number | Event.Date | Location | Countr |
|---|----------------|--------------------|-----------------|------------|-----------------|-------------|
| 0 | 20001218X45444 | Accident | SEA87LA080 | 1948-10-24 | MOOSE CREEK, ID | Unite State |
| 1 | 20001218X45447 | Accident | LAX94LA336 | 1962-07-19 | BRIDGEPORT, CA | Unite State |
| 2 | 20061025X01555 | Accident | NYC07LA005 | 1974-08-30 | Saltville, VA | Unite State |
| 3 | 20001218X45448 | Accident | LAX96LA321 | 1977-06-19 | EUREKA, CA | Unite State |
| 4 | 20041105X01764 | Accident | CHI79FA064 | 1979-08-02 | Canton, OH | Unite State |

5 rows × 31 columns



```
In [3]: 1 #determining shape of dataset
        2 df.shape
```

Out[3]: (88889, 31)

In [4]:

```
1 #examinig null values
2 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                       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  object
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  object
24  Total.Serious.Injuries              76379 non-null  object
25  Total.Minor.Injuries                76956 non-null  object
26  Total.Uninjured                     82977 non-null  object
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: object(31)
memory usage: 21.0+ MB
```

In [5]:

```
1 #setting Event.Id as index
2 df = pd.read_csv('data/AviationData.csv', index_col = 0,
3                 encoding='Latin-1',
4                 low_memory=False,
5                 dtype=str)
6 df
```

Out[5]:

| | Investigation.Type | Accident.Number | Event.Date | Location | Country |
|----------------|--------------------|-----------------|------------|-----------------|---------------|
| Event.Id | | | | | |
| 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 |
| 20001218X45448 | Accident | LAX96LA321 | 1977-06-19 | EUREKA, CA | United States |
| 20041105X01764 | Accident | CHI79FA064 | 1979-08-02 | Canton, OH | United States |
| ... | ... | ... | ... | ... | ... |
| 20221227106491 | Accident | ERA23LA093 | 2022-12-26 | Annapolis, MD | United States |
| 20221227106494 | Accident | ERA23LA095 | 2022-12-26 | Hampton, NH | United States |
| 20221227106497 | Accident | WPR23LA075 | 2022-12-26 | Payson, AZ | United States |
| 20221227106498 | Accident | WPR23LA076 | 2022-12-26 | Morgan, UT | United States |
| 20221230106513 | Accident | ERA23LA097 | 2022-12-29 | Athens, GA | United States |

88889 rows × 30 columns

```
In [6]: 1 # Calculating NaN values sum for each column
        2 df.isna().sum()
```

```
Out[6]: 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 [7]: 1 #Removing whitespaces from column names
        2 df.columns = df.columns.str.strip()
```

```
In [8]: 1 #checking how many aircrafts are Airplanes
        2 (df['Aircraft.Category'] == 'Airplane').value_counts()
```

```
Out[8]: False      61272
        True       27617
        Name: Aircraft.Category, dtype: int64
```

```
In [9]: 1 # Keeping only rows where the Aircraft is an Airplane.  
2 # Creating a new df where Aircraft.Category is equal to Airplane and  
3 df= df[df['Aircraft.Category'] == 'Airplane']  
4 # Pulling series to briefly spot check Aircraft.Category column  
5 df['Aircraft.Category']
```

```
Out[9]: Event.Id  
20170710X52551    Airplane  
20020909X01562    Airplane  
20020909X01561    Airplane  
20020917X02148    Airplane  
20020917X02134    Airplane  
...  
20221213106455    Airplane  
20221215106463    Airplane  
20221219106475    Airplane  
20221219106470    Airplane  
20221227106497    Airplane  
Name: Aircraft.Category, Length: 27617, dtype: object
```

```
In [10]: 1 #keeping rows where the Airplanes are NOT built by amateurs  
2 df = df[df['Amateur.Built'] == 'No']  
3  
4 df['Amateur.Built']
```

```
Out[10]: Event.Id  
20170710X52551    No  
20020909X01562    No  
20020909X01561    No  
20020917X02148    No  
20020917X02134    No  
..  
20221213106455    No  
20221215106463    No  
20221219106475    No  
20221219106470    No  
20221227106497    No  
Name: Amateur.Built, Length: 24417, dtype: object
```

```
In [11]: 1 # Determining value type  
2 type(df['Total.Fatal.Injuries'][0])
```

```
Out[11]: float
```

In [12]:

```
1 # Quick spot check to identify NaN value
2 df['Total.Fatal.Injuries']
```

Out[12]:

```
Event.Id
20170710X52551    NaN
20020909X01562      0
20020909X01561      0
20020917X02148      0
20020917X02134      1
...
20221213106455      0
20221215106463      0
20221219106475      0
20221219106470      0
20221227106497      0
Name: Total.Fatal.Injuries, Length: 24417, dtype: object
```

In [13]:

```
1 # Imputing Fatality values from Injury Severity Column
2 # Using a for loop to iterate over each row to make 'Total.Fatal Inju
3 for index, row in df.iterrows():
4     if row['Injury.Severity'] == 'Non-Fatal' and pd.isnull(row['Total
5         df.at[index, 'Total.Fatal.Injuries'] = 0
```

In [14]:

```
1 # Spot Check
2 df.head()
```

Out[14]:

| Event.Id | Investigation.Type | Accident.Number | Event.Date | Location | Country | L |
|----------------|--------------------|-----------------|------------|------------------|---------------|------|
| 20170710X52551 | Accident | NYC79AA106 | 1979-09-17 | BOSTON, MA | United States | 42.4 |
| 20020909X01562 | Accident | SEA82DA022 | 1982-01-01 | PULLMAN, WA | United States | |
| 20020909X01561 | Accident | NYC82DA015 | 1982-01-01 | EAST HANOVER, NJ | United States | |
| 20020917X02148 | Accident | FTW82FRJ07 | 1982-01-02 | HOMER, LA | United States | |
| 20020917X02134 | Accident | FTW82FRA14 | 1982-01-02 | HEARNE, TX | United States | |

5 rows × 30 columns

```
In [15]: 1 # Checking NaN values by counting sum of isna() for each column
2 nan_values = df.isna()
3
4 nan_counts = nan_values.sum()
5
6 nan_counts
```

```
Out[15]: Investigation.Type      0
Accident.Number      0
Event.Date          0
Location            6
Country             7
Latitude           5246
Longitude           5252
Airport.Code        8843
Airport.Name        8405
Injury.Severity      813
Aircraft.damage     1270
Aircraft.Category    0
Registration.Number  203
Make                3
Model              18
Amateur.Built        0
Number.of.Engines   2551
Engine.Type         3956
FAR.Description     480
Schedule           21481
Purpose.of.flight   3683
Air.carrier         14061
Total.Fatal.Injuries  126
Total.Serious.Injuries 2846
Total.Minor.Injuries 2562
Total.Uninjured      718
Weather.Condition   2979
Broad.phase.of.flight 18621
Report.Status       4662
Publication.Date     957
dtype: int64
```



```
In [16]: 1 # Create a boolean mask for rows with NaN values in 'Total.Fatal.Inju
2 mask_nan_values = df['Total.Fatal.Injuries'].isna()
3
4 # Use the boolean mask to select the rows with NaN values
5 rows_with_nan = df[mask_nan_values]
6
7 # Display the rows with NaN values
8 rows_with_nan
9
```

Out[16]:

| Event.Id | Investigation.Type | Accident.Number | Event.Date | Location | Count |
|----------------|--------------------|-----------------|------------|---------------------|-----------|
| 20020917X02151 | Incident | FTW82IA062 | 1982-02-19 | HARLINGEN, TX | Unit Stat |
| 20020917X03106 | Incident | DCA82IA021 | 1982-05-12 | FT. MYER, FL | Unit Stat |
| 20020917X04065 | Incident | NYC82IA137 | 1982-07-05 | FLUSHING, NY | Unit Stat |
| 20020917X04216 | Incident | ANC82IA095 | 1982-09-21 | NEAR DILLINGHAM, AK | Unit Stat |
| 20001214X42478 | Incident | LAX83IA149B | 1983-03-18 | LOS ANGELES, CA | Unit Stat |

```
In [17]: 1 # Spot Check NaN values are removed
2 df['Total.Fatal.Injuries']
```

```
Out[17]: Event.Id
20170710X52551    0
20020909X01562    0
20020909X01561    0
20020917X02148    0
20020917X02134    1
..
20221213106455    0
20221215106463    0
20221219106475    0
20221219106470    0
20221227106497    0
Name: Total.Fatal.Injuries, Length: 24417, dtype: object
```

```
In [18]: 1 df.shape
```

Out[18]: (24417, 30)

In [19]:

```
1 # null value check
2 # df.isna().sum()
```

In [20]:

```
1 # Removing all accidents that happened prior to 1985 since avg Lifesp
2 # Converting 'Event.Date' to datetime format for easy manipulation
3 # Using .loc to avoid SettingwithCopyWarning
4 df['Event.Date'] = pd.to_datetime(df.loc[:, 'Event.Date'])
5 mask = df['Event.Date'].dt.year >=1985
6 df_current = df.loc[mask]
7 df_current
```

Out[20]:

| | Investigation.Type | Accident.Number | Event.Date | Location | Country | Lat |
|-------------------------|--------------------|-----------------|-------------|------------|-----------------|-------------------|
| | Event.Id | | | | | |
| | 20001214X35509 | Accident | DEN85LA064 | 1985-01-14 | WAPITI, WY | United States |
| | 20001214X36510 | Accident | LAX85LA257 | 1985-05-13 | MESA, AZ | United States |
| | 20001214X36887 | Accident | NYC85FA145B | 1985-06-11 | BELMAR, NJ | United States |
| | 20001214X37274 | Accident | NYC85LA188 | 1985-07-21 | SIDNEY, ME | United States |
| | 20001214X37356 | Incident | ATL85IA251 | 1985-08-16 | HILTON HEAD, SC | United States |
| | ... | ... | ... | ... | ... | ... |
| | 20221213106455 | Accident | WPR23LA065 | 2022-12-13 | Lewistown, MT | United States 047 |
| | 20221215106463 | Accident | ERA23LA090 | 2022-12-14 | San Juan, PR | United States 182 |
| | 20221219106475 | Accident | WPR23LA069 | 2022-12-15 | Wichita, KS | United States 373 |
| | 20221219106470 | Accident | ERA23LA091 | 2022-12-16 | Brooksville, FL | United States 282 |
| | 20221227106497 | Accident | WPR23LA075 | 2022-12-26 | Payson, AZ | United States 341 |
| 21430 rows × 30 columns | | | | | | |

```
In [21]: 1 # Checking my code
2 df_1985 = df_current.copy()
3 df_1985['Event.Date'] = pd.to_datetime(df_1985.loc[:, 'Event.Date'])
4 num_rows_before_1985 = (df_1985['Event.Date'].dt.year < 1985).sum()
5 print("Number of rows with a date before 1985:", num_rows_before_1985)
6
```

Number of rows with a date before 1985: 0

```
In [22]: 1 # Confirming 'Event.Date' is in Datetime
2 type(df_1985['Event.Date'][0])
```

Out[22]: pandas._libs.tslibs.timestamps.Timestamp

```
In [23]: 1 # Determining all unique values for 'Aircraft.Damage'
2 df_1985['Aircraft.damage'].unique()
```

Out[23]: array(['Destroyed', 'Substantial', nan, 'Minor', 'Unknown'], dtype=object)

```
In [24]: 1 # Imputing 'Aircraft.Damage' value
2 df_1985['Total.Uninjured'] = pd.to_numeric(df_1985['Total.Uninjured'])
3 condition = df_1985['Total.Uninjured'] >= 1
4 df_1985.loc[condition, 'Aircraft.damage'] = 'Minor'
```

```
In [25]: 1 # Removes all rows where 'Total.Fatal.Injuries' is NaN
2 df_1985 = df_1985.dropna(subset=['Total.Fatal.Injuries'], axis=0)
3 # Ensuring all values in the 'Total.Fatal.Injuries' column are integers
4 df_1985['Total.Fatal.Injuries'] = df_1985['Total.Fatal.Injuries'].astype(int)
```

```
In [26]: 1 # Calculating the mean of 'Total.Fatal.Injuries' for each accident'
2 df_1985['Total.Fatal.Injuries'].mean()
```

Out[26]: 0.6491812508797448

```
In [27]: 1 # Removing all rows with NaN in the 'Make' and 'Model' columns'
2 df_1985 = df_1985.dropna(subset=['Make', 'Model'])
```

```
In [28]: 1 # Checking how many unique 'Model' values there are
2 df_1985['Model'].nunique()
```

Out[28]: 3514

```
In [29]: 1 #df_1985.shape
```


In [30]:

```

1 # Cleaning up 'Model' values to reduce redundancy
2 df_1985.loc[df_1985['Model'] == '108 3', 'Model'] = '108-3'
3 df_1985.loc[df_1985['Model'] == '100 180', 'Model'] = '100-180'
4 df_1985.loc[df_1985['Model'] == '108 1', 'Model'] = '108-1'
5 df_1985.loc[df_1985['Model'] == '114 B', 'Model'] = '114-B'
6 df_1985.loc[df_1985['Model'] == '114B', 'Model'] = '114-B'
7 df_1985.loc[df_1985['Model'] == '14 19', 'Model'] = '14-19'
8 df_1985.loc[df_1985['Model'] == '150 - F', 'Model'] = '150-F'
9 df_1985.loc[df_1985['Model'] == '150 - G', 'Model'] = '150-G'
10 df_1985.loc[df_1985['Model'] == '164B', 'Model'] = '164-B'
11 df_1985.loc[df_1985['Model'] == '17 30', 'Model'] = '17-30'
12 df_1985.loc[df_1985['Model'] == '17 30A', 'Model'] = '17-30A'
13 df_1985.loc[df_1985['Model'] == '17 31A', 'Model'] = '17-31A'
14 df_1985.loc[df_1985['Model'] == '17 30', 'Model'] = '17-30'
15 df_1985.loc[df_1985['Model'] == '170 B', 'Model'] = '170-B'
16 df_1985.loc[df_1985['Model'] == '170 - B', 'Model'] = '170-B'
17 df_1985.loc[df_1985['Model'] == '170B', 'Model'] = '170-B'
18 df_1985.loc[df_1985['Model'] == '172 M', 'Model'] = '172-M'
19 df_1985.loc[df_1985['Model'] == '172 - M', 'Model'] = '172-M'
20 df_1985.loc[df_1985['Model'] == '172M', 'Model'] = '172-M'
21 df_1985.loc[df_1985['Model'] == '172P', 'Model'] = '172-P'
22 df_1985.loc[df_1985['Model'] == '172 P', 'Model'] = '172-P'
23 df_1985.loc[df_1985['Model'] == '172N', 'Model'] = '172-N'
24 df_1985.loc[df_1985['Model'] == '172 N', 'Model'] = '172-N'
25 df_1985.loc[df_1985['Model'] == '172A', 'Model'] = '172-A'
26 df_1985.loc[df_1985['Model'] == '172 A', 'Model'] = '172-A'
27 df_1985.loc[df_1985['Model'] == '172S', 'Model'] = '172-S'
28 df_1985.loc[df_1985['Model'] == '172 S', 'Model'] = '172-S'
29 df_1985.loc[df_1985['Model'] == '172F', 'Model'] = '172-F'
30 df_1985.loc[df_1985['Model'] == '172 - F', 'Model'] = '172-F'
31 df_1985.loc[df_1985['Model'] == '172 - H', 'Model'] = '172-H'
32 df_1985.loc[df_1985['Model'] == '172H', 'Model'] = '172-H'
33 df_1985.loc[df_1985['Model'] == '172 - S', 'Model'] = '172-S'
34 df_1985.loc[df_1985['Model'] == '172 - P', 'Model'] = '172-P'
35 df_1985.loc[df_1985['Model'] == '172 - N', 'Model'] = '172-N'
36 df_1985.loc[df_1985['Model'] == '172 K', 'Model'] = '172-K'
37 df_1985.loc[df_1985['Model'] == '172K', 'Model'] = '172-K'
38 df_1985.loc[df_1985['Model'] == '172 - R', 'Model'] = '172-R'
39 df_1985.loc[df_1985['Model'] == '172R', 'Model'] = '172-R'
40 df_1985.loc[df_1985['Model'] == '172RG', 'Model'] = '172-RG'
41 df_1985.loc[df_1985['Model'] == '172 RG', 'Model'] = '172-RG'
42 df_1985.loc[df_1985['Model'] == '172SP', 'Model'] = '172-SP'
43 df_1985.loc[df_1985['Model'] == '172 SP', 'Model'] = '172-SP'
44 df_1985.loc[df_1985['Model'] == '1730A', 'Model'] = '1730-A'
45 df_1985.loc[df_1985['Model'] == '1730 - A', 'Model'] = '1730-A'
46 df_1985.loc[df_1985['Model'] == '172B', 'Model'] = '172-B'
47 df_1985.loc[df_1985['Model'] == '172C', 'Model'] = '172-C'
48 df_1985.loc[df_1985['Model'] == '172D', 'Model'] = '172-D'
49 df_1985.loc[df_1985['Model'] == '172E', 'Model'] = '172-E'
50 df_1985.loc[df_1985['Model'] == '172G', 'Model'] = '172-G'
51 df_1985.loc[df_1985['Model'] == '172I', 'Model'] = '172-I'
52 df_1985.loc[df_1985['Model'] == '172L', 'Model'] = '172-L'
53 df_1985.loc[df_1985['Model'] == '172Q', 'Model'] = '172-Q'
54 df_1985.loc[df_1985['Model'] == '172XP', 'Model'] = '172-XP'
55 df_1985.loc[df_1985['Model'] == '175A', 'Model'] = '175-A'
56 df_1985.loc[df_1985['Model'] == '175B', 'Model'] = '175-B'
57 df_1985.loc[df_1985['Model'] == '175C', 'Model'] = '175-C'

```

```

58 df_1985.loc[df_1985['Model'] == '177 RG', 'Model'] = '177-RG'
59 df_1985.loc[df_1985['Model'] == '177RG', 'Model'] = '177-RG'
60 df_1985.loc[df_1985['Model'] == '177A', 'Model'] = '177-A'
61 df_1985.loc[df_1985['Model'] == '177B', 'Model'] = '177-B'
62 df_1985.loc[df_1985['Model'] == '180 - B', 'Model'] = '180-B'
63 df_1985.loc[df_1985['Model'] == '180 H', 'Model'] = '180-H'
64 df_1985.loc[df_1985['Model'] == '180A', 'Model'] = '180-A'
65 df_1985.loc[df_1985['Model'] == '180 H', 'Model'] = '180-H'
66 df_1985.loc[df_1985['Model'] == '180A', 'Model'] = '180-A'
67 df_1985.loc[df_1985['Model'] == '180B', 'Model'] = '180-B'
68 df_1985.loc[df_1985['Model'] == '180C', 'Model'] = '180-C'
69 df_1985.loc[df_1985['Model'] == '180D', 'Model'] = '180-D'
70 df_1985.loc[df_1985['Model'] == '180E', 'Model'] = '180-E'
71 df_1985.loc[df_1985['Model'] == '180F', 'Model'] = '180-F'
72 df_1985.loc[df_1985['Model'] == '180G', 'Model'] = '180-G'
73 df_1985.loc[df_1985['Model'] == '180J', 'Model'] = '180-J'
74 df_1985.loc[df_1985['Model'] == '180H', 'Model'] = '180-H'
75 df_1985.loc[df_1985['Model'] == '180J', 'Model'] = '180-J'
76 df_1985.loc[df_1985['Model'] == '180K', 'Model'] = '180-K'
77 df_1985.loc[df_1985['Model'] == '180M', 'Model'] = '180-M'
78 df_1985.loc[df_1985['Model'] == '185E', 'Model'] = '185-E'
79 df_1985.loc[df_1985['Model'] == '185 - E', 'Model'] = '180-E'
80 df_1985.loc[df_1985['Model'] == '206 - H', 'Model'] = '206-H'
81 df_1985.loc[df_1985['Model'] == '206H', 'Model'] = '206-H'
82 df_1985.loc[df_1985['Model'] == '210 5', 'Model'] = '210-5'
83 df_1985.loc[df_1985['Model'] == '210 5(205)', 'Model'] = '210-5(205)'
84 df_1985.loc[df_1985['Model'] == '210N', 'Model'] = '210-N'
85 df_1985.loc[df_1985['Model'] == '210 - N', 'Model'] = '210-N'
86 df_1985.loc[df_1985['Model'] == '210D', 'Model'] = '210-D'
87 df_1985.loc[df_1985['Model'] == '210 D', 'Model'] = '210-D'
88 df_1985.loc[df_1985['Model'] == '210A', 'Model'] = '210-A'
89 df_1985.loc[df_1985['Model'] == '210B', 'Model'] = '210-B'
90 df_1985.loc[df_1985['Model'] == '210C', 'Model'] = '210-C'
91 df_1985.loc[df_1985['Model'] == '210E', 'Model'] = '210-E'
92 df_1985.loc[df_1985['Model'] == '210F', 'Model'] = '210-F'
93 df_1985.loc[df_1985['Model'] == '210G', 'Model'] = '210-G'
94 df_1985.loc[df_1985['Model'] == '210H', 'Model'] = '210-H'
95 df_1985.loc[df_1985['Model'] == '210J', 'Model'] = '210-J'
96 df_1985.loc[df_1985['Model'] == '210K', 'Model'] = '210-K'
97 df_1985.loc[df_1985['Model'] == '210L', 'Model'] = '210-L'
98 df_1985.loc[df_1985['Model'] == '210M', 'Model'] = '210-M'
99 df_1985.loc[df_1985['Model'] == '2T 1A', 'Model'] = '2T-1A'
100 df_1985.loc[df_1985['Model'] == '2T1A', 'Model'] = '2T-1A'
101 df_1985.loc[df_1985['Model'] == '2T 1A 2', 'Model'] = '2T-1A-2'
102 df_1985.loc[df_1985['Model'] == '305 A', 'Model'] = '305-A'
103 df_1985.loc[df_1985['Model'] == '305A', 'Model'] = '305-A'
104 df_1985.loc[df_1985['Model'] == '305C', 'Model'] = '305-C'
105 df_1985.loc[df_1985['Model'] == '35 33', 'Model'] = '35-33'
106 df_1985.loc[df_1985['Model'] == '35 A33', 'Model'] = '35-A33'
107 df_1985.loc[df_1985['Model'] == '35A33', 'Model'] = '35-A33'
108 df_1985.loc[df_1985['Model'] == '35 B33', 'Model'] = '35-B33'
109 df_1985.loc[df_1985['Model'] == '35B33', 'Model'] = '35-B33'
110 df_1985.loc[df_1985['Model'] == '35C33', 'Model'] = '35-C33'
111 df_1985.loc[df_1985['Model'] == '35 C33', 'Model'] = '35-C33'
112 df_1985.loc[df_1985['Model'] == '35 - A', 'Model'] = '35-A'
113 df_1985.loc[df_1985['Model'] == '35A', 'Model'] = '35-A'
114 df_1985.loc[df_1985['Model'] == '415 C', 'Model'] = '415-C'

```

```

115 df_1985.loc[df_1985['Model'] == '415C', 'Model'] = '415-C'
116 df_1985.loc[df_1985['Model'] == '415 CD', 'Model'] = '415-CD'
117 df_1985.loc[df_1985['Model'] == '415 C/D', 'Model'] = '415-CD'
118 df_1985.loc[df_1985['Model'] == '415-C/D', 'Model'] = '415-CD'
119 df_1985.loc[df_1985['Model'] == '415 D', 'Model'] = '415-D'
120 df_1985.loc[df_1985['Model'] == '415D', 'Model'] = '415-D'
121 df_1985.loc[df_1985['Model'] == '415G', 'Model'] = '415-G'
122 df_1985.loc[df_1985['Model'] == '421 - C', 'Model'] = '421-C'
123 df_1985.loc[df_1985['Model'] == '421C', 'Model'] = '421-C'
124 df_1985.loc[df_1985['Model'] == '500 - B', 'Model'] = '500-B'
125 df_1985.loc[df_1985['Model'] == '500 B', 'Model'] = '500-B'
126 df_1985.loc[df_1985['Model'] == '500B', 'Model'] = '500-B'
127 df_1985.loc[df_1985['Model'] == '500 S', 'Model'] = '500-S'
128 df_1985.loc[df_1985['Model'] == '560 - XL', 'Model'] = '560-XL'
129 df_1985.loc[df_1985['Model'] == '560XL', 'Model'] = '560-XL'
130 df_1985.loc[df_1985['Model'] == '402A', 'Model'] = 'AT-402A'
131 df_1985.loc[df_1985['Model'] == '402B', 'Model'] = 'AT-402B'
132 df_1985.loc[df_1985['Model'] == 'S2-R', 'Model'] = 'S2R'
133 df_1985.loc[df_1985['Model'] == 'S-2R', 'Model'] = 'S2R'
134 df_1985.loc[df_1985['Model'] == 'SR2', 'Model'] = 'S2R'
135 df_1985.loc[df_1985['Model'] == 'CH2000', 'Model'] = 'CH-2000'
136 df_1985.loc[df_1985['Model'] == 'CH 2000', 'Model'] = 'CH-2000'
137 df_1985.loc[df_1985['Model'] == 'A1A', 'Model'] = 'A-1A'
138 df_1985.loc[df_1985['Model'] == 'A1-A', 'Model'] = 'A-1A'
139 df_1985.loc[df_1985['Model'] == 'Husky A1-B', 'Model'] = 'Husky A-1B'
140 df_1985.loc[df_1985['Model'] == 'SR22', 'Model'] = 'SR-22'
141 df_1985.loc[df_1985['Model'] == 'YMF 5C', 'Model'] = 'YMF-5C'
142 df_1985.loc[df_1985['Model'] == 'Gulfstream AM G-164B', 'Model'] = 'G'
143 model_counts = df_1985['Model'].value_counts().sort_index()
144 model_counts

```

```

Out[30]: 0-1A          1
         0-58A       1
         0-58B       2
         01          1
         1           1
         ..
         Zlin 526F    1
         Zodiac 601 XL 1
         Zodiac 601XL 1
         lebed        1
         sportstar    1
         Name: Model, Length: 3436, dtype: int64

```

```

In [31]: 1 # Removed 78 redundant 'Model' values
         2 df_1985['Model'].nunique()

```

```
Out[31]: 3436
```

```

In [32]: 1 df_1985['Make'].nunique()

```

```
Out[32]: 1329
```


In [33]:

```

1 # Consolidating 'Make' values
2 df_1985['Make'].fillna('', inplace=True)
3 df_1985.loc[df_1985['Make'].str.contains('Cessna', case=False), 'Make']
4 df_1985.loc[df_1985['Make'].str.contains('Piper', case=False), 'Make']
5 df_1985.loc[df_1985['Make'].str.contains('Beechcraft|Beech', case=False), 'Make']
6 df_1985.loc[df_1985['Make'].str.contains('Boeing|Boeing Stearman', case=False), 'Make']
7 df_1985.loc[df_1985['Make'].str.contains('Air tractor', case=False), 'Make']
8 df_1985.loc[df_1985['Make'].str.contains('Mooney', case=False), 'Make']
9 df_1985.loc[df_1985['Make'].str.contains('CIRRUS', case=False), 'Make']
10 df_1985.loc[df_1985['Make'].str.contains('American', case=False), 'Make']
11 df_1985.loc[df_1985['Make'].str.contains('Airbus', case=False), 'Make']
12 df_1985.loc[df_1985['Make'].str.contains('Grumman', case=False), 'Make']
13 df_1985.loc[df_1985['Make'].str.contains('Bellanca', case=False), 'Make']
14 df_1985.loc[df_1985['Make'].str.contains('Maule', case=False), 'Make']
15 df_1985.loc[df_1985['Make'].str.contains('Aeronca', case=False), 'Make']
16 df_1985.loc[df_1985['Make'].str.contains('Embraer', case=False), 'Make']
17 df_1985.loc[df_1985['Make'].str.contains('Champion', case=False), 'Make']
18 df_1985.loc[df_1985['Make'].str.contains('Luscombe', case=False), 'Make']
19 df_1985.loc[df_1985['Make'].str.contains('Stinson', case=False), 'Make']
20 df_1985.loc[df_1985['Make'].str.contains('TaylorCraft', case=False), 'Make']
21 df_1985.loc[df_1985['Make'].str.contains('Dehavilland|De havilland', case=False), 'Make']
22 df_1985.loc[df_1985['Make'].str.contains('Ayres', case=False), 'Make']
23 df_1985.loc[df_1985['Make'].str.contains('Raytheon', case=False), 'Make']
24 df_1985.loc[df_1985['Make'].str.contains('Diamond Aircraft', case=False), 'Make']
25 df_1985.loc[df_1985['Make'].str.contains('Grumman Schweizer', case=False), 'Make']
26 df_1985.loc[df_1985['Make'].str.contains('Gulfstream-Schweizer|Gulfstream', case=False), 'Make']
27 df_1985.loc[df_1985['Make'].str.contains('Gulfstream American', case=False), 'Make']
28 df_1985.loc[df_1985['Make'].str.contains('Ted Aerostar', case=False), 'Make']
29 df_1985.loc[df_1985['Make'].str.contains('Lockheed', case=False), 'Make']
30 df_1985.loc[df_1985['Make'].str.contains('Gulfstream Aerospace|Gulfstream', case=False), 'Make']
31 df_1985.loc[df_1985['Make'].str.contains('Northrop', case=False), 'Make']
32 df_1985.loc[df_1985['Make'].str.contains('Helio', case=False), 'Make']
33 df_1985.loc[df_1985['Make'].str.contains('Canadair', case=False), 'Make']
34 df_1985.loc[df_1985['Make'].str.contains('Learjet', case=False), 'Make']
35 df_1985.loc[df_1985['Make'].str.contains('Volmer', case=False), 'Make']
36 df_1985.loc[df_1985['Make'].str.contains('Ryan', case=False), 'Make']
37 df_1985.loc[df_1985['Make'].str.contains('Britten', case=False), 'Make']
38 df_1985.loc[df_1985['Make'].str.contains('Howard', case=False), 'Make']
39 df_1985.loc[df_1985['Make'].str.contains('British Airways', case=False), 'Make']
40 df_1985.loc[df_1985['Make'].str.contains('British Aerospace', case=False), 'Make']
41 df_1985.loc[df_1985['Make'].str.contains('Textron', case=False), 'Make']
42 df_1985.loc[df_1985['Make'].str.contains('Aviat', case=False), 'Make']
43 df_1985.loc[df_1985['Make'].str.contains('Waco', case=False), 'Make']
44 df_1985.loc[df_1985['Make'].str.contains('Quicksilver', case=False), 'Make']
45 df_1985.loc[df_1985['Make'].str.contains('Cub', case=False), 'Make']
46 df_1985.loc[df_1985['Make'].str.contains('Bombardier', case=False), 'Make']
47 df_1985.loc[df_1985['Make'].str.contains('Dassault', case=False), 'Make']
48 df_1985.loc[df_1985['Make'].str.contains('Found A', case=False), 'Make']
49 df_1985.loc[df_1985['Make'].str.contains('Ercoupe', case=False), 'Make']
50 df_1985.loc[df_1985['Make'].str.contains('Extra Flugzeugbau', case=False), 'Make']
51 df_1985.loc[df_1985['Make'].str.contains('Evektor', case=False), 'Make']
52 df_1985.loc[df_1985['Make'].str.contains('Flight Design', case=False), 'Make']

```



```
In [34]: 1 # Fill missing values in 'Engine.Type' with a placeholder value (e.g.
2 df_1985.loc[:, 'Engine.Type'].fillna('Unknown', inplace=True)
3
4 # Replace 'Engine.Type' values containing 'Turbo' (case-insensitive)
5 df_1985.loc[df_1985['Engine.Type'].str.contains('Turbo', case=False),
6
```

```
In [35]: 1 # Viewing top 35 most common 'Model' values
2 model_frequency = df_1985['Model'].value_counts()
3 model_frequency[:35]
```

```
Out[35]: 172          772
737          402
152          317
182          304
172-S        283
SR-22        281
PA28         273
172-N        253
180          213
A36          189
172-M        184
PA-18-150    180
150          179
PA-28-140    168
172-P        146
140          117
172-R        113
170-B        110
PA-28-161    106
PA-28-100    105
```

```
In [36]: 1 # Viewing top 35 most common 'Make' values
2 make_frequency = df_1985['Make'].value_counts()
3 make_frequency[:35]
```

```
Out[36]: CESSNA          7190
PIPER          4050
BEECH          1496
BOEING          1241
AIR TRACTOR      432
MOONEY          410
CIRRUS          400
AMERICAN        359
AIRBUS          274
GRUMMAN         231
BELLANCA        219
MAULE           216
AVIAT           205
AERONCA         201
EMBAER          170
DEHAVILLAND     166
CHAMPION        158
LUSCOMBE        142
STINSON         136
BOMBARDIER      133
```

```
In [37]: 1 #removing outliers
        2 df_1985 = df_1985.drop(df_1985[df_1985['Model'].isin(['747-300', '767
```

```
In [38]: 1 # Removed 332 redundant 'Make' values
        2 df_1985['Make'].nunique()
```

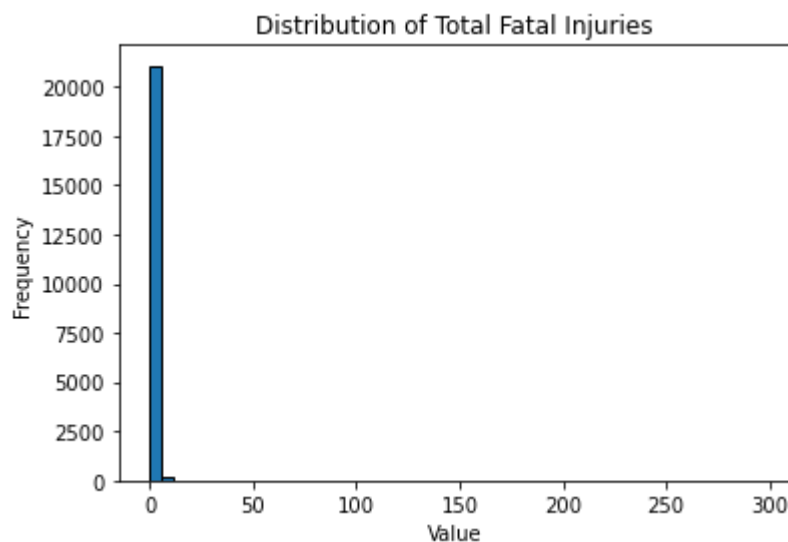
Out[38]: 997

```
In [39]: 1 # New shape. Removed 64,472 rows with NaN values
        2 #df_1985.shape
```

Data Analysis

- Checking for trends in data
- Creating new column 'Make_Model' for enhanced readability and analysis
- Running groupby to determine any aggregate values
- Specifically targeting frequency of planes in df_1985, 'Total.Fatal.Injuries,' 'Total.Uninjured,' 'Engine.Type,' and 'Number.of.Engines'

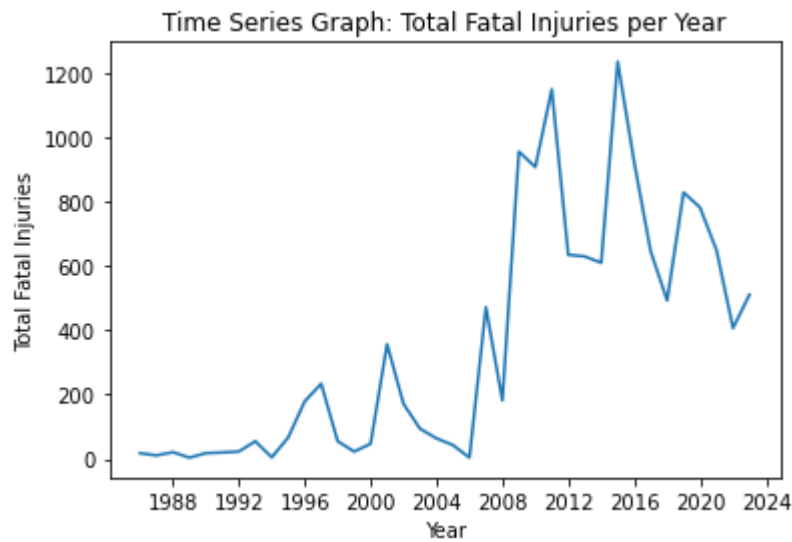
```
In [40]: 1 # Viewing Distrobution of 'Total.Fatal.Injuries' values
        2 plt.hist(df_1985['Total.Fatal.Injuries'], bins=50, edgecolor='black')
        3
        4 plt.xlabel('Value')
        5 plt.ylabel('Frequency')
        6 plt.title('Distribution of Total Fatal Injuries')
        7
        8 plt.show()
```



```
In [41]: 1 #df_1985.shape
```

In [42]:

```
1 # Plotting 'Total.Fatal.Injuries' over time
2 df_1985['Total.Fatal.Injuries'] = pd.to_numeric(df_1985['Total.Fatal.
3
4 df_resampled = df_1985.resample('Y', on='Event.Date')['Total.Fatal.In
5
6 plt.plot(df_resampled.index, df_resampled)
7 plt.xlabel('Year')
8 plt.ylabel('Total Fatal Injuries')
9 plt.title('Time Series Graph: Total Fatal Injuries per Year')
10 plt.show()
```



In [43]:

```
1 #df_1985.shape
```

In [44]:

```
1 df_1985 = df_1985[df_1985["Publication.Date"].notna()]
2 df_1985
```

Out[44]:

| | Investigation.Type | Accident.Number | Event.Date | Location | Coun |
|----------------|--------------------|-----------------|------------|--------------------|-----------|
| Event.Id | | | | | |
| 20001214X35509 | Accident | DEN85LA064 | 1985-01-14 | WAPITI, WY | Un Sta |
| 20001214X36510 | Accident | LAX85LA257 | 1985-05-13 | MESA, AZ | Un Sta |
| 20001214X36887 | Accident | NYC85FA145B | 1985-06-11 | BELMAR, NJ | Un Sta |
| 20001214X37274 | Accident | NYC85LA188 | 1985-07-21 | SIDNEY, ME | Un Sta |
| 20001214X37718 | Accident | ATL85MA286 | 1985-09-29 | JENKINSBURG, GA | Un Sta |

In [45]: 1 `#df_1985.shape`

In [46]: 1 `# Loading in State Abbreviations`
 2 `dfa = pd.read_csv('data/USState_Codes.csv',`
 3 `encoding='Latin-1',`
 4 `low_memory=False,`
 5 `dtype=str)`
 6 `dfa.head()`

Out[46]:

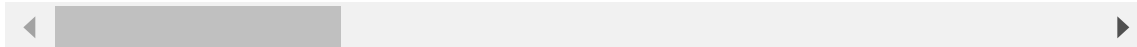
| | US_State | Abbreviation |
|---|------------|--------------|
| 0 | Alabama | AL |
| 1 | Alaska | AK |
| 2 | Arizona | AZ |
| 3 | Arkansas | AR |
| 4 | California | CA |

In [47]: 1 `df_1985.loc[:, 'Total.Fatal.Injuries'] = pd.to_numeric(df_1985['Total`
 2 `dfi_1985 = df_1985[df_1985['Total.Fatal.Injuries'] >= 1]`
 3 `dfi_1985.head()`

Out[47]:

| | Investigation.Type | Accident.Number | Event.Date | Location | Country |
|----------------|--------------------|-----------------|------------|-----------------|---------------|
| Event.Id | | | | | |
| 20001214X36887 | Accident | NYC85FA145B | 1985-06-11 | BELMAR, NJ | United States |
| 20001214X37718 | Accident | ATL85MA286 | 1985-09-29 | JENKINSBURG, GA | United States |
| 20001213X33054 | Accident | FTW86FA050 | 1986-03-29 | SEAGOVILLE, TX | United States |
| 20001213X33276 | Accident | FTW86FA066B | 1986-04-15 | HANKAMER, TX | United States |
| 20001213X33276 | Accident | FTW86FA066A | 1986-04-15 | HANKAMER, TX | United States |

5 rows × 30 columns



In [48]: 1 `#df_1985.shape`

```
In [49]: 1 make_frequency = df_1985['Make'].value_counts()
        2 make_frequency
```

```
Out[49]: CESSNA                6922
         PIPER                 3921
         BEECH                 1450
         BOEING                1089
         AIR TRACTOR           419
         ...
         Colyaer SL            1
         PACIFIC AEROSPACE LTD  1
         BUCKER JUNGMEISTER     1
         Sherman John, H       1
         D-Fly                 1
         Name: Make, Length: 952, dtype: int64
```

```
In [50]: 1 # Checking 'Total.Fatal.Injuries' for a specific 'Model'
        2 specific_model = '152'
        3 fatalities_with_specific_model = df_1985[df_1985['Model'] == specific
        4
        5 fatalities_with_specific_model
```

```
Out[50]: 50
```

```
In [51]: 1 # Viewing 'Total.Fatal.Injuries' by 'Model'
        2 fatalities_by_model = df_1985.groupby('Model')['Total.Fatal.Injuries']
        3 fatalities_by_model = fatalities_by_model.sort_values(ascending=False)
        4 fatalities_by_model
```

```
Out[51]: Model
         737                1279
         777 - 206           534
         A321                381
         A330                331
         172                 197
         ...
         J3F-65                0
         J3F-60                0
         J3F-50                0
         J3F 65                0
         0-1A                  0
         Name: Total.Fatal.Injuries, Length: 3303, dtype: int32
```

```
In [52]: 1 # Viewing 'Total.Fatal.Injuries' by 'Make'
2 fatalities_by_make = df_1985.groupby('Make')['Total.Fatal.Injuries'].
3 fatalities_by_make = fatalities_by_make.sort_values(ascending=False)
4 fatalities_by_make
```

```
Out[52]: Make
BOEING                2404
CESSNA                2371
PIPER                1522
AIRBUS                1236
BEECH                1063
...
MARY ALEXANDER         0
MARTIN CHARLES A       0
MAGNAGHI AERONAUTICA SPA 0
M-SQUARED AIRCRAFT     0
177MF LLC              0
Name: Total.Fatal.Injuries, Length: 952, dtype: int32
```

```
In [53]: 1 Engine_numbers = df_1985['Number.ofEngines'].value_counts()
2 Engine_numbers
```

```
Out[53]: 1    15458
2     2864
4       78
3       31
0        7
6         1
8         1
Name: Number.ofEngines, dtype: int64
```

```
In [54]: 1 FAR_description = df_1985['FAR.Description'].value_counts()
2 FAR_description
```

```
Out[54]: 091          13203
Part 91: General Aviation    2478
NUSN          884
137           736
NUSC          674
121           602
135           533
UNK           181
129           155
Part 137: Agricultural    150
PUBU          125
Part 121: Air Carrier      78
Part 135: Air Taxi & Commuter 74
Non-U.S., Non-Commercial   14
091K           13
Public Use         11
125              5
Non-U.S., Commercial      4
ARMF             4
Unknown          4
Part 129: Foreign        4
107              1
Public Aircraft         1
Part 125: 20+ Pax,6000+ lbs 1
Part 91 Subpart K: Fractional 1
Name: FAR.Description, dtype: int64
```

```
In [55]: 1 weather_condition = df_1985['Weather.Condition'].value_counts()
2 weather_condition
```

```
Out[55]: VMC      16822
IMC       1023
Unk        206
UNK         13
Name: Weather.Condition, dtype: int64
```

```
In [56]: 1 weather_impact = df_1985.groupby('Weather.Condition')['Total.Fatal.In
2 weather_impact = weather_impact.sort_values(ascending=False)
3 weather_impact
```

```
Out[56]: Weather.Condition
VMC      4167
IMC      1813
Unk       206
UNK         9
Name: Total.Fatal.Injuries, dtype: int32
```

```
In [57]: 1 # Where do crashes happen most frequently
2 #df_1985['Broad.phase.of.flight'].value_counts()
```

```
In [58]: 1 #Showing which Broad.phase.of.flight has the most fatal injuries
2 grouped_df = df_1985.groupby('Broad.phase.of.flight')['Total.Fatal.In
3 fatalities_by_phase = grouped_df.sort_values(ascending=False)
4 fatalities_by_phase[:12]
```

```
Out[58]: Broad.phase.of.flight
Takeoff      184
Maneuvering   99
Cruise       90
Approach      84
Climb         46
Descent       32
Go-around    13
Unknown       9
Landing       9
Standing      5
Taxi          1
Other         0
Name: Total.Fatal.Injuries, dtype: int32
```

```
In [59]: 1 # Viewing how often each 'Make' experiences a specific level of 'Airc
2 specific_damage_counts = df_1985.groupby(['Make', 'Aircraft.damage'])
3 specific_damage_counts_sorted = specific_damage_counts.sort_values(as
4 specific_damage_counts_sorted[:25])
```

```
Out[59]: Make      Aircraft.damage
CESSNA      Minor      4710
PIPER       Minor      2504
CESSNA      Substantial 1613
PIPER       Substantial 1001
BEECH       Minor       804
BOEING      Minor       785
CESSNA      Destroyed   561
PIPER       Destroyed   402
BEECH       Substantial 395
            Destroyed   246
AIR TRACTOR Minor      241
MOONEY      Minor      207
AMERICAN    Minor      207
CIRRUS      Minor      203
MAULE       Minor      165
AIRBUS      Minor      155
BELLANCA    Minor      145
MOONEY      Substantial 142
AVIAT       Minor      138
AERONCA     Minor      135
AIR TRACTOR Substantial 130
GRUMMAN     Minor      126
EMBAER      Minor      117
CIRRUS      Substantial 112
DEHAVILLAND Minor      111
dtype: int64
```



```
In [60]: 1 #Total.Fatal.Injuries by Aircraft.Damage
2 Damage_Fatal = df_1985.groupby('Aircraft.damage')['Total.Fatal.Injuri
3 fatalities_by_Damage = Damage_Fatal.sort_values(ascending=False)
4 fatalities_by_Damage
```

```
Out[60]: Aircraft.damage
Destroyed      7623
Substantial    3268
Minor          787
Unknown        30
Name: Total.Fatal.Injuries, dtype: int32
```

```
In [61]: 1 #Total.Fatal.Injuries by Weather.Condition
2 Weather_Fatal = df_1985.groupby('Weather.Condition')['Total.Fatal.Inj
3 fatalities_by_Weather = Weather_Fatal.sort_values(ascending=False)
4 fatalities_by_Weather
```

```
Out[61]: Weather.Condition
VMC      4167
IMC      1813
Unk       206
UNK         9
Name: Total.Fatal.Injuries, dtype: int32
```

```
In [62]: 1 #Total.Fatal.Injuries by Make
2 Make_Fatal = df_1985.groupby('Make')['Total.Fatal.Injuries'].sum()
3 fatalities_by_Make = Make_Fatal.sort_values(ascending=False)
4 fatalities_by_Make[:25]
```

```
Out[62]: Make
BOEING      2404
CESSNA      2371
PIPER       1522
AIRBUS      1236
BEECH       1063
EMBAER       329
CIRRUS       233
MOONEY       188
DEHAVILLAND  150
AMERICAN     118
ATR          101
RAYTHEON      90
TUPOLEV       89
SUKHOI        86
AIR TRACTOR   85
BOMBARDIER    78
ANTONOV       71
SOCATA        56
CRJ          54
```

```
In [63]: 1 #Total.Fatal.Injuries by 'Make_Model'
2 df_1985.loc[:, 'Make_Model'] = df_1985.loc[:, 'Make'] + '_' + df_1985
3 Model_Fatal = df_1985.groupby('Make_Model')['Total.Fatal.Injuries'].s
4 fatalities_by_Model = Model_Fatal.sort_values(ascending=False)
5 fatalities_by_Model[:50]
```

```
Out[63]: Make_Model
BOEING_737                1279
BOEING_777 - 206           534
AIRBUS_A321                381
AIRBUS_A330                331
CESSNA_172                 197
AIRBUS_A320                170
CIRRUS_SR-22              168
AIRBUS_A320 - 216          162
BOEING_737-800             154
EMBAER_E135 Legacy         154
BOEING_MD-82               154
AIRBUS_A310                152
CESSNA_182                 124
BEECH_A36                  121
CESSNA_208                 111
PIPER_PA28                 105
TUPOLEV_TU154              89
BOEING_737-500             88
CESSNA_441                 85
```

```
In [64]: 1 grouped_by_make = df_1985.groupby('Make')['Model'].unique()
2
3 for make, models in grouped_by_make.items():
4     print(f"Make: {make}")
5     print("Models:")
6     for model in models:
7         print(f"    - {model}")
8     print()
```

```
Make: 177MF LLC
Models:
    - PITTS MODEL 12
```

```
Make: 2021FX3 LLC
Models:
    - CCX-2000
```

```
Make: 3XTRIM
Models:
    - 450 ULTRA
```

```
Make: 737
Models:
    - 800
```

```
Make: AAA AIRCRAFT LLC
Models:
    - CCX-2000
```

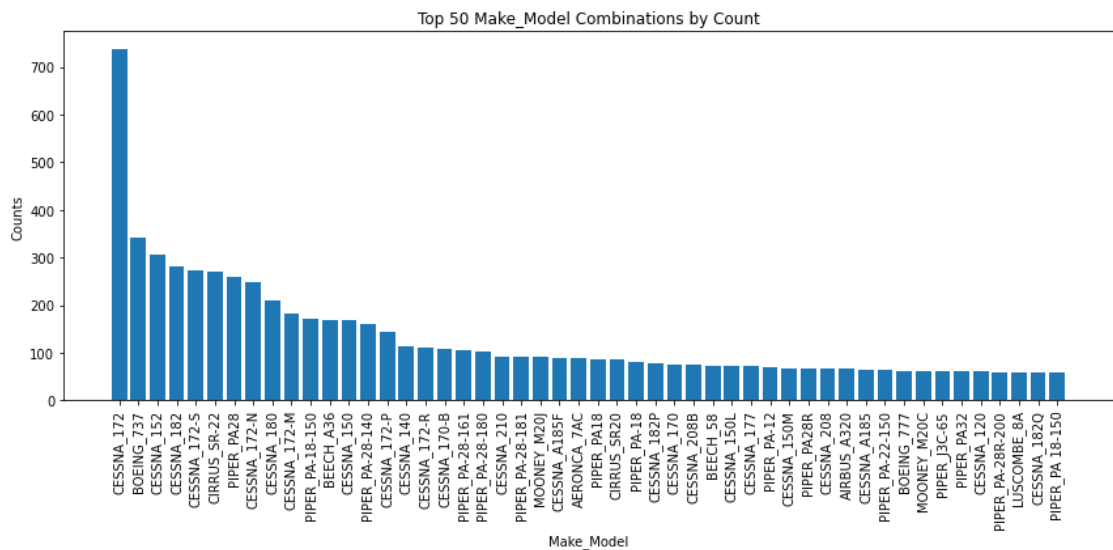
```
In [65]: 1 df_1985['Make_Model'].value_counts()[:25]
```

```
Out[65]: CESSNA_172          738
          BOEING_737         343
          CESSNA_152         307
          CESSNA_182         281
          CESSNA_172-S       274
          CIRRUS_SR-22       271
          PIPER_PA28         260
          CESSNA_172-N       247
          CESSNA_180         211
          CESSNA_172-M       181
          PIPER_PA-18-150    172
          BEECH_A36         168
          CESSNA_150         168
          PIPER_PA-28-140    161
          CESSNA_172-P       144
          CESSNA_140         114
          CESSNA_172-R       110
          CESSNA_170-B       108
          PIPER_PA-28-161    106
          PIPER_PA-28-180    102
          CESSNA_210         92
          PIPER_PA-28-181     92
          MOONEY_M20J        91
          CESSNA_A185F       90
          AERONCA_7AC        89
          Name: Make_Model, dtype: int64
```

```

In [66]: 1 # Count the occurrences of each 'Make_Model' combination
2 make_model_counts = df_1985['Make_Model'].value_counts()
3
4 # Sort the values in descending order and select the top 50 entries (
5 top_50_counts = make_model_counts.sort_values(ascending=False)[:50]
6
7 # Set up the figure and axes
8 fig, ax = plt.subplots(figsize=(12, 6))
9
10 # Create the bar plot
11 ax.bar(top_50_counts.index, top_50_counts.values)
12
13 # Rotate the x-axis labels for better visibility
14 plt.xticks(rotation=90)
15
16 # Set labels and title
17 ax.set_xlabel('Make_Model')
18 ax.set_ylabel('Counts')
19 ax.set_title('Top 50 Make_Model Combinations by Count')
20
21 # Show the plot
22 plt.tight_layout()
23 plt.show()
24
25

```



```

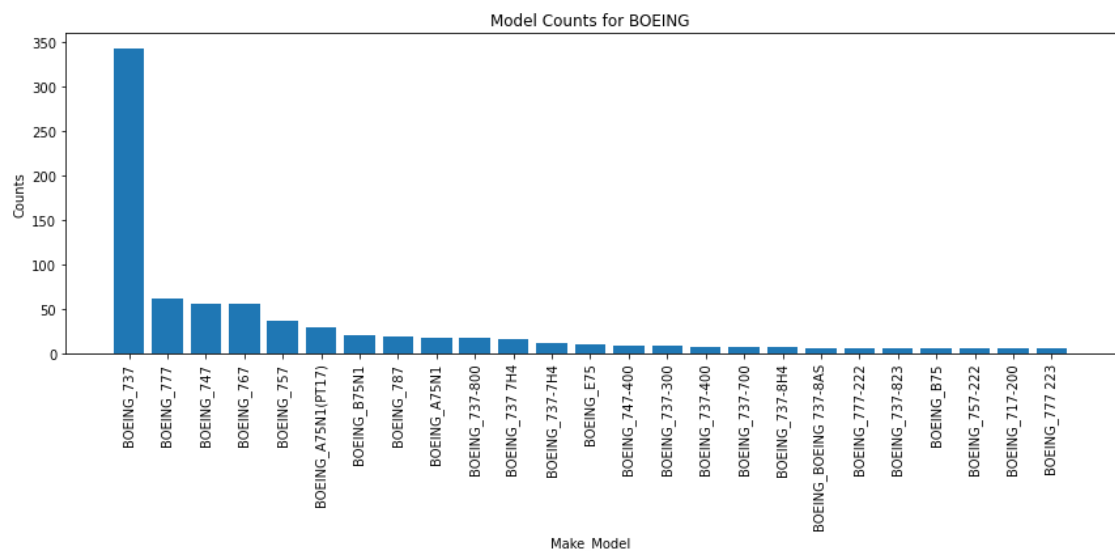
In [67]: 1 #df_1985.shape

```

```

In [68]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'BOEING'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Combine 'Make' and 'Model' columns to create a new column represent
6 chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']
7
8 # Count the occurrences of each 'Make_Model' combination
9 make_model_counts = chart_df['Make_Model'].value_counts()
10
11 # Sort the values in descending order and select all entries
12 all_make_model_counts = make_model_counts.sort_values(ascending=False)
13
14 # Set up the figure and axes
15 fig, ax = plt.subplots(figsize=(12, 6))
16
17 # Create the bar plot
18 ax.bar(all_make_model_counts.index, all_make_model_counts.values)
19
20 # Rotate the x-axis labels for better visibility
21 plt.xticks(rotation=90)
22
23 # Set labels and title
24 ax.set_xlabel('Make_Model')
25 ax.set_ylabel('Counts')
26 ax.set_title(f'Model Counts for {specific_make}')
27
28 # Show the plot
29 plt.tight_layout()
30 plt.show()
31
32

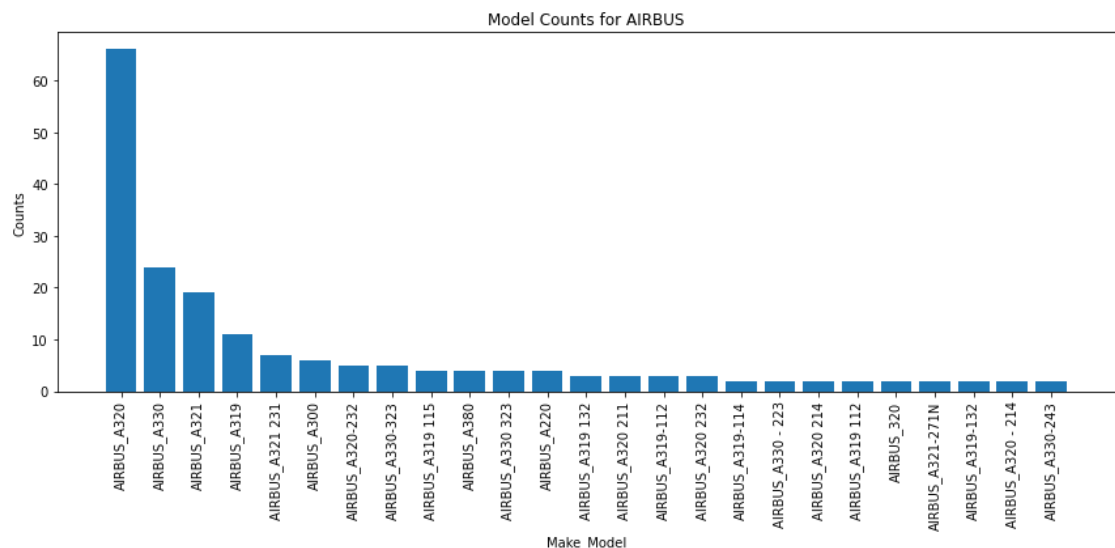
```



```

In [69]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'AIRBUS'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Combine 'Make' and 'Model' columns to create a new column represent
6 chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']
7
8 # Count the occurrences of each 'Make_Model' combination
9 make_model_counts = chart_df['Make_Model'].value_counts()
10
11 # Sort the values in descending order and select all entries
12 all_make_model_counts = make_model_counts.sort_values(ascending=False)
13
14 # Set up the figure and axes
15 fig, ax = plt.subplots(figsize=(12, 6))
16
17 # Create the bar plot
18 ax.bar(all_make_model_counts.index, all_make_model_counts.values)
19
20 # Rotate the x-axis labels for better visibility
21 plt.xticks(rotation=90)
22
23 # Set labels and title
24 ax.set_xlabel('Make_Model')
25 ax.set_ylabel('Counts')
26 ax.set_title(f'Model Counts for {specific_make}')
27
28 # Show the plot
29 plt.tight_layout()
30 plt.show()
31

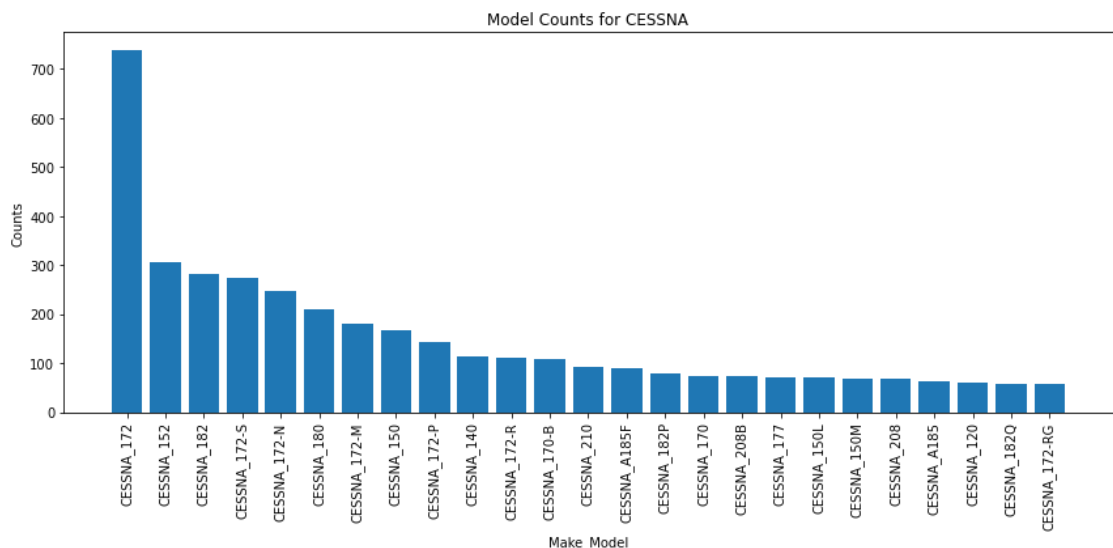
```



```

In [70]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'CESSNA'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Combine 'Make' and 'Model' columns to create a new column represent
6 chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']
7
8 # Count the occurrences of each 'Make_Model' combination
9 make_model_counts = chart_df['Make_Model'].value_counts()
10
11 # Sort the values in descending order and select all entries
12 all_make_model_counts = make_model_counts.sort_values(ascending=False)
13
14 # Set up the figure and axes
15 fig, ax = plt.subplots(figsize=(12, 6))
16
17 # Create the bar plot
18 ax.bar(all_make_model_counts.index, all_make_model_counts.values)
19
20 # Rotate the x-axis labels for better visibility
21 plt.xticks(rotation=90)
22
23 # Set labels and title
24 ax.set_xlabel('Make_Model')
25 ax.set_ylabel('Counts')
26 ax.set_title(f'Model Counts for {specific_make}')
27
28 # Show the plot
29 plt.tight_layout()
30 plt.show()
31

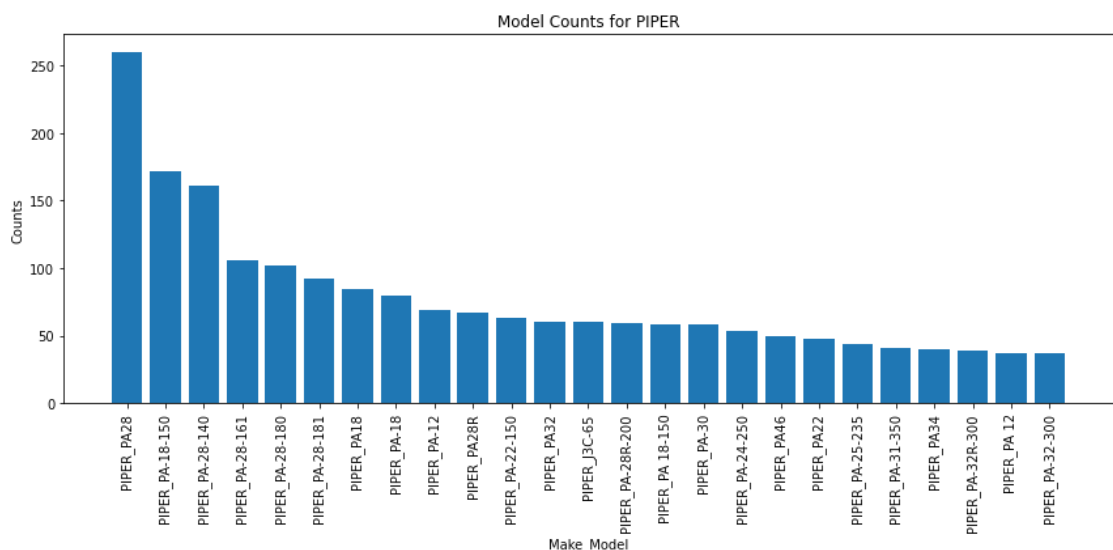
```



```

In [71]: 1 specific_make = 'PIPER'
2 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
3
4 # Combine 'Make' and 'Model' columns to create a new column represent
5 chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']
6
7 # Count the occurrences of each 'Make_Model' combination
8 make_model_counts = chart_df['Make_Model'].value_counts()
9
10 # Sort the values in descending order and select all entries
11 all_make_model_counts = make_model_counts.sort_values(ascending=False)
12
13 # Set up the figure and axes
14 fig, ax = plt.subplots(figsize=(12, 6))
15
16 # Create the bar plot
17 ax.bar(all_make_model_counts.index, all_make_model_counts.values)
18
19 # Rotate the x-axis labels for better visibility
20 plt.xticks(rotation=90)
21
22 # Set labels and title
23 ax.set_xlabel('Make_Model')
24 ax.set_ylabel('Counts')
25 ax.set_title(f'Model Counts for {specific_make}')
26
27 # Show the plot
28 plt.tight_layout()
29 plt.show()

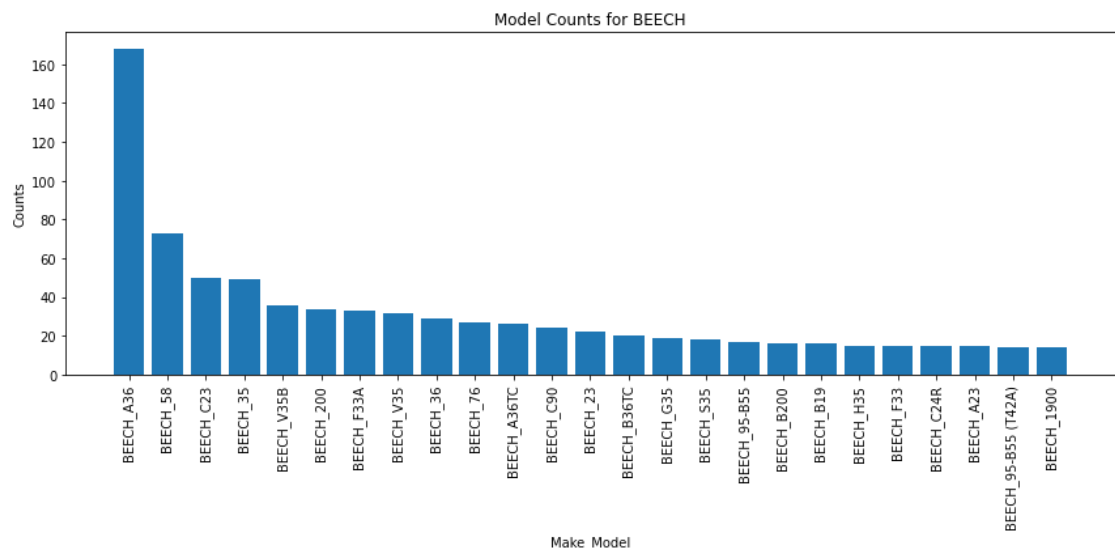
```




```

In [72]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'BEECH'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Combine 'Make' and 'Model' columns to create a new column represent
6 chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']
7
8 # Count the occurrences of each 'Make_Model' combination
9 make_model_counts = chart_df['Make_Model'].value_counts()
10
11 # Sort the values in descending order and select all entries
12 all_make_model_counts = make_model_counts.sort_values(ascending=False)
13
14 # Set up the figure and axes
15 fig, ax = plt.subplots(figsize=(12, 6))
16
17 # Create the bar plot
18 ax.bar(all_make_model_counts.index, all_make_model_counts.values)
19
20 # Rotate the x-axis labels for better visibility
21 plt.xticks(rotation=90)
22
23 # Set labels and title
24 ax.set_xlabel('Make_Model')
25 ax.set_ylabel('Counts')
26 ax.set_title(f'Model Counts for {specific_make}')
27
28 # Show the plot
29 plt.tight_layout()
30 plt.show()

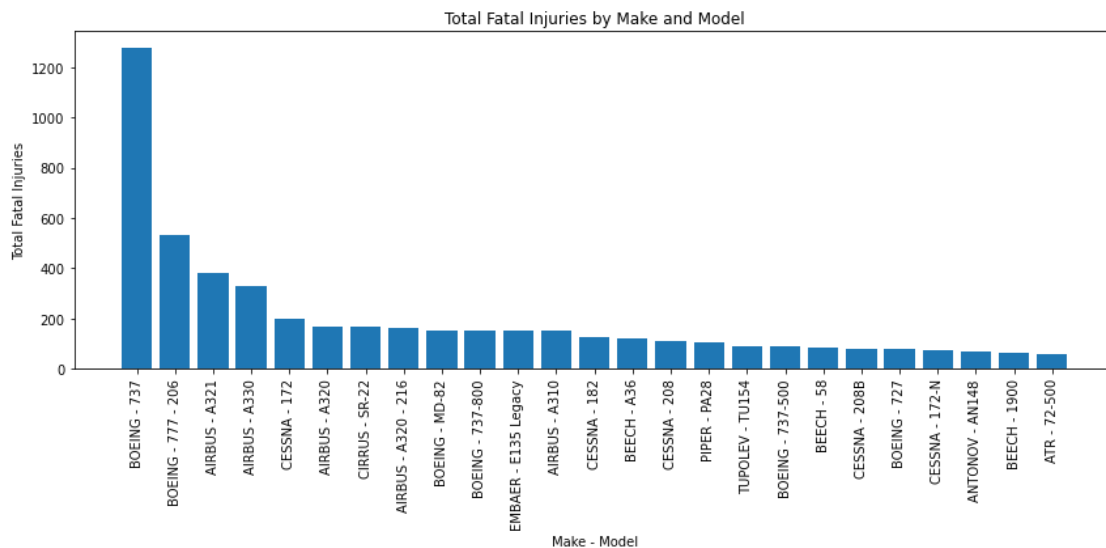
```



```

In [73]: 1 # Group the DataFrame by 'Make' and 'Model' and calculate the sum of
2 make_model_fatal_injuries = df_1985.groupby(['Make', 'Model'])['Total
3 make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val
4
5 # Create a new column 'Make_Model' by combining 'Make' and 'Model'
6 make_model_fatal_injuries_sorted.index = make_model_fatal_injuries_so
7 fig, ax = plt.subplots(figsize=(12, 6))
8 ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
9 plt.xticks(rotation=90)
10
11 ax.set_xlabel('Make - Model')
12 ax.set_ylabel('Total Fatal Injuries')
13 ax.set_title('Total Fatal Injuries by Make and Model')
14
15 plt.tight_layout()
16 plt.show()
17
18

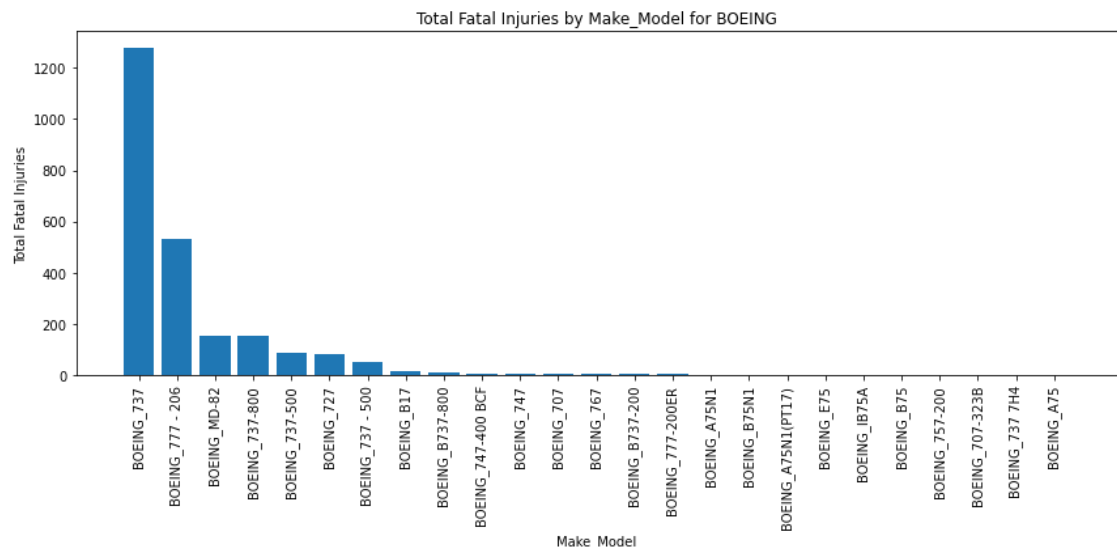
```



```

In [74]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'BOEING'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
6 make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat
7
8 # Sort the values in descending order
9 make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Total Fatal Injuries')
23 ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}')
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

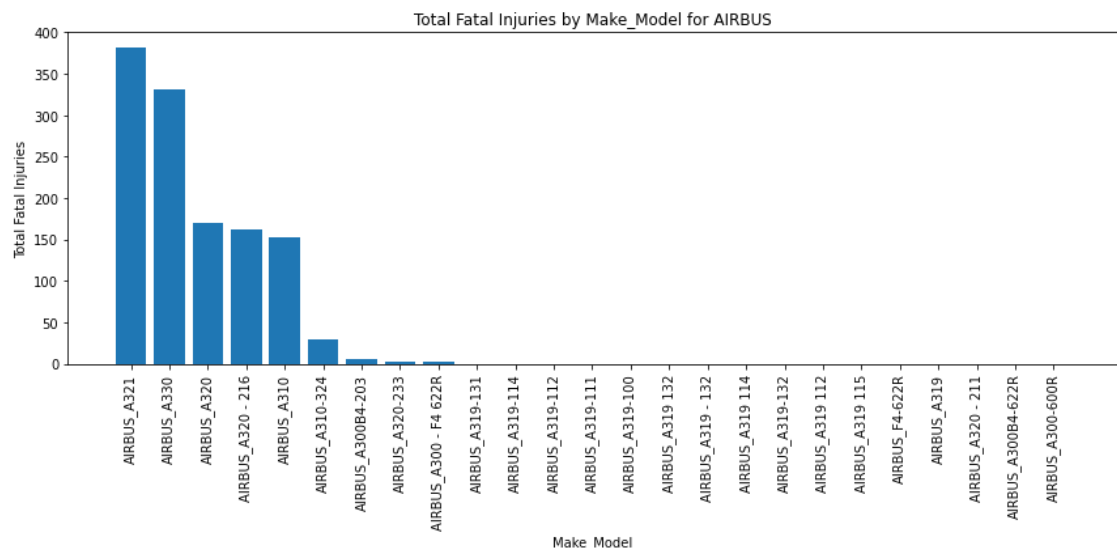
```



```

In [75]: 1 specific_make = 'AIRBUS'
2 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
3
4 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
5 make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat
6
7 # Sort the values in descending order
8 make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val
9
10 # Set up the figure and axes
11 fig, ax = plt.subplots(figsize=(12, 6))
12
13 # Create the bar plot
14 ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
15
16 # Rotate the x-axis labels for better visibility
17 plt.xticks(rotation=90)
18
19 # Set labels and title
20 ax.set_xlabel('Make_Model')
21 ax.set_ylabel('Total Fatal Injuries')
22 ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}')
23
24 # Show the plot
25 plt.tight_layout()
26 plt.show()
27

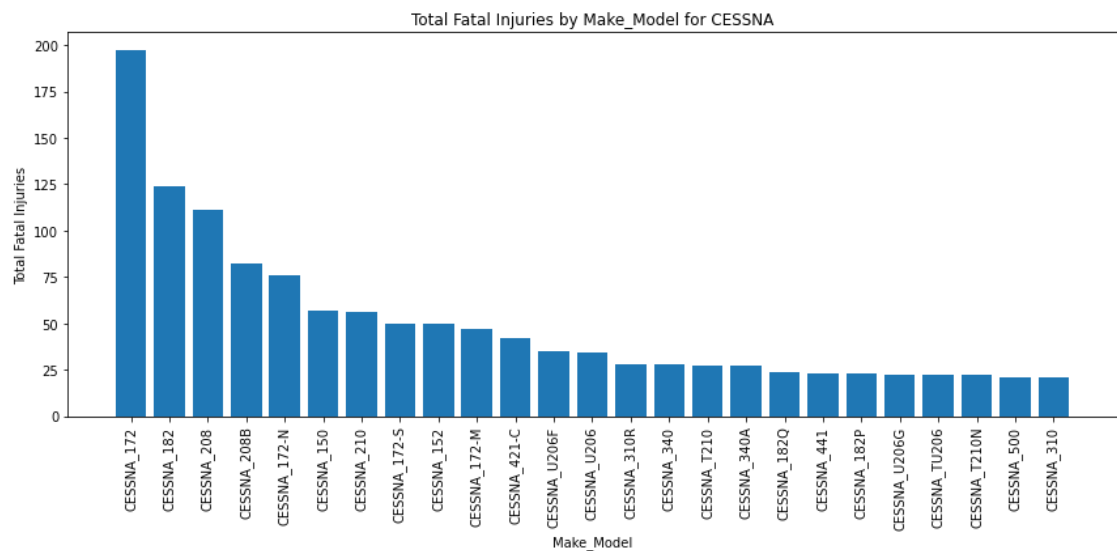
```



```

In [76]: 1 specific_make = 'CESSNA'
2 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
3
4 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
5 make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat
6
7 # Sort the values in descending order
8 make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val
9
10 # Set up the figure and axes
11 fig, ax = plt.subplots(figsize=(12, 6))
12
13 # Create the bar plot
14 ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
15
16 # Rotate the x-axis labels for better visibility
17 plt.xticks(rotation=90)
18
19 # Set labels and title
20 ax.set_xlabel('Make_Model')
21 ax.set_ylabel('Total Fatal Injuries')
22 ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}')
23
24 # Show the plot
25 plt.tight_layout()
26 plt.show()
27

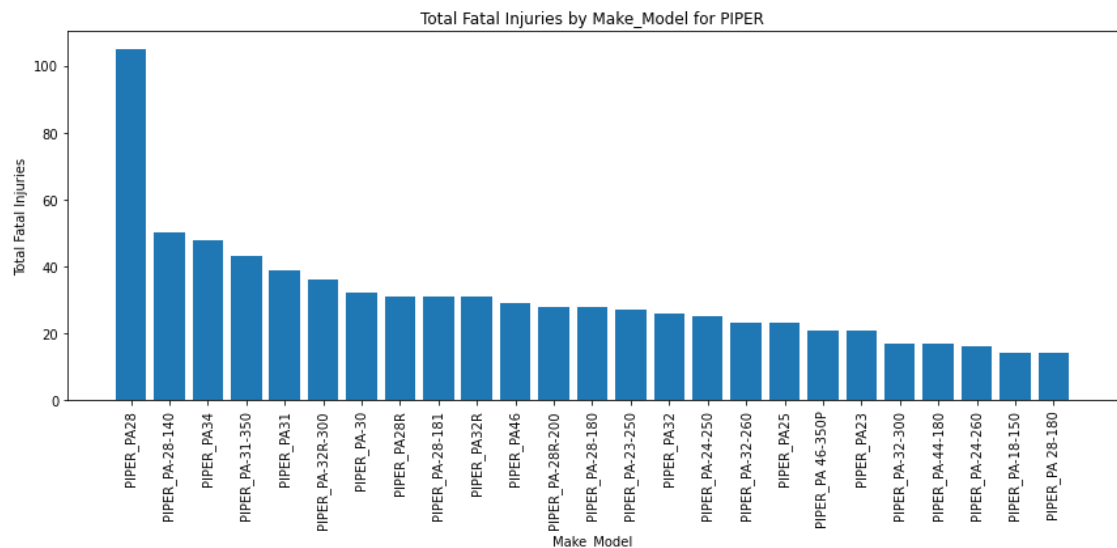
```



```

In [77]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'PIPER'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
6 make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat
7
8 # Sort the values in descending order
9 make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Total Fatal Injuries')
23 ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}')
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

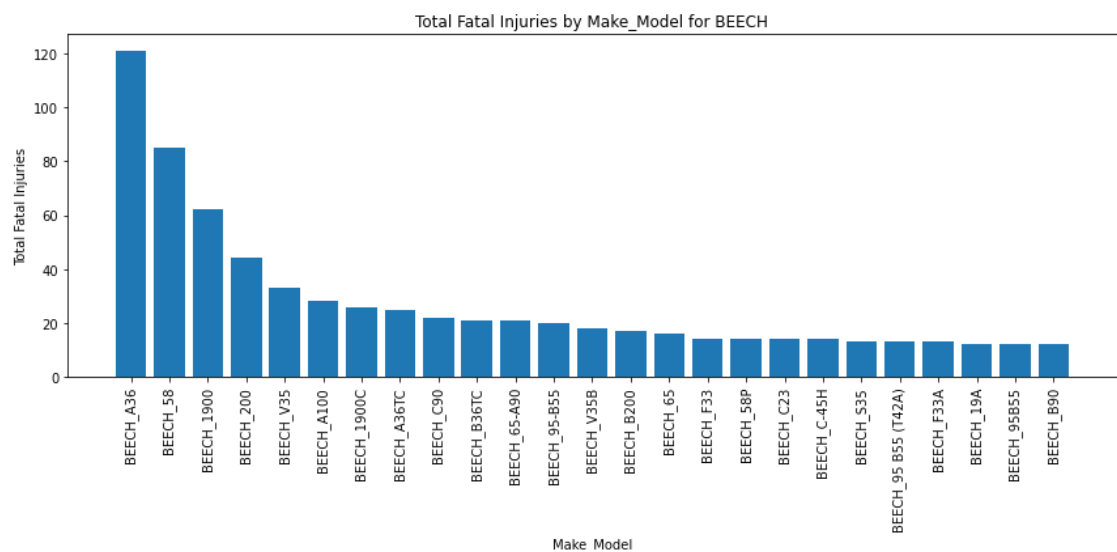
```



```

In [78]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'BEECH'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
6 make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat
7
8 # Sort the values in descending order
9 make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Total Fatal Injuries')
23 ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}')
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28
29

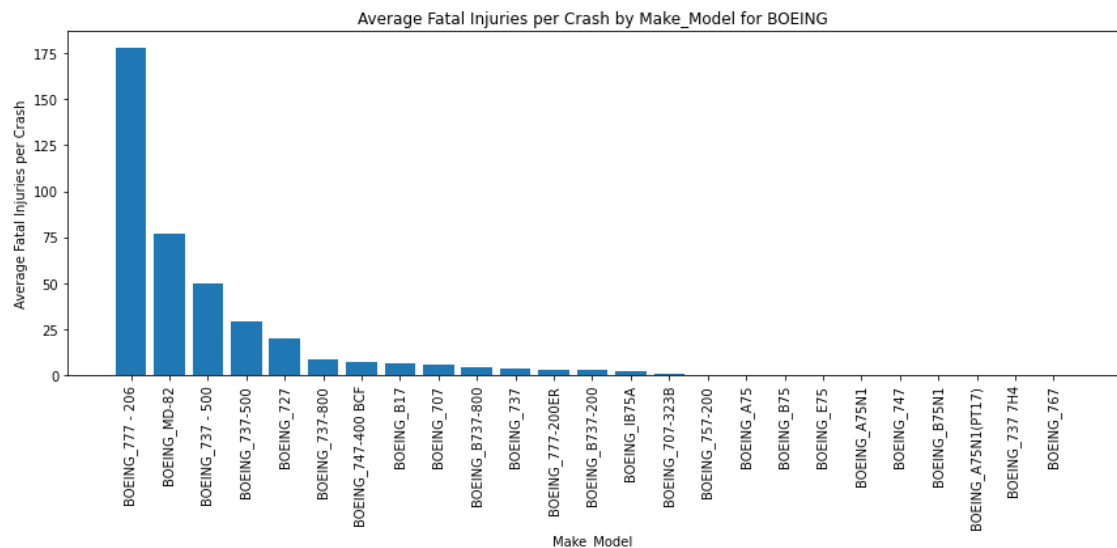
```



```

In [79]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'BOEING'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the average of 'Total
6 make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total
7
8 # Sort the values in descending order
9 make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fatal_injuries_sorted.values)
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Average Fatal Injuries per Crash')
23 ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {specific_make}')
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

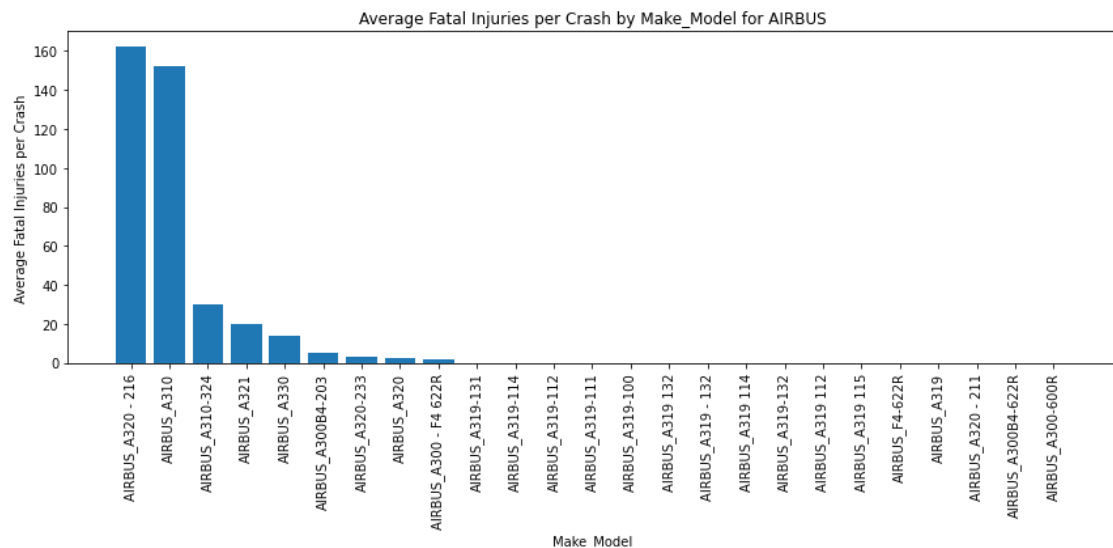
```




```

In [80]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'AIRBUS'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the average of 'Total
6 make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total
7
8 # Sort the values in descending order
9 make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Average Fatal Injuries per Crash')
23 ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

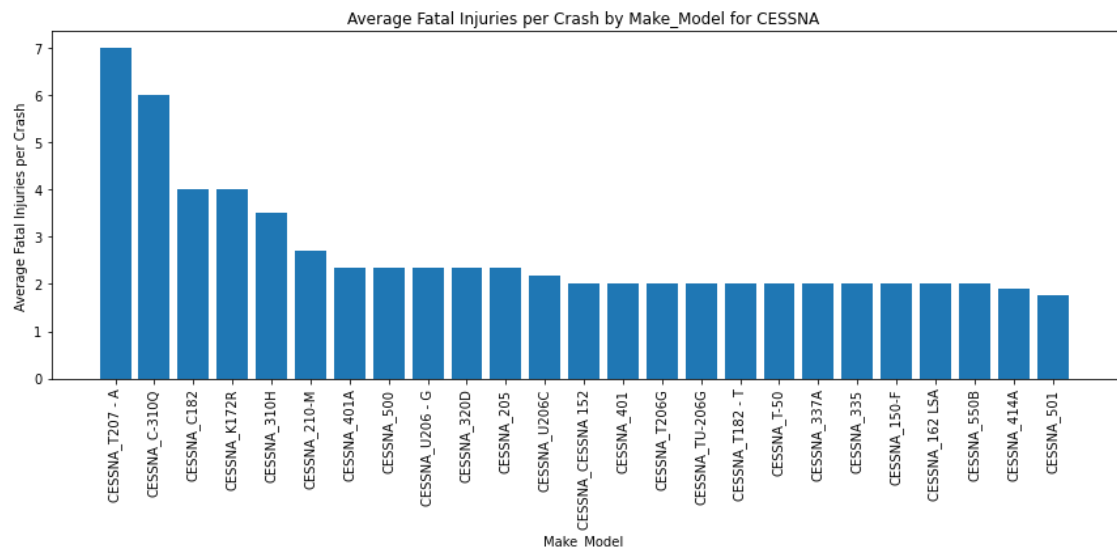
```



```

In [81]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'CESSNA'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the average of 'Total
6 make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total
7
8 # Sort the values in descending order
9 make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fatal_injuries_sorted.values)
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Average Fatal Injuries per Crash')
23 ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {specific_make}')
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

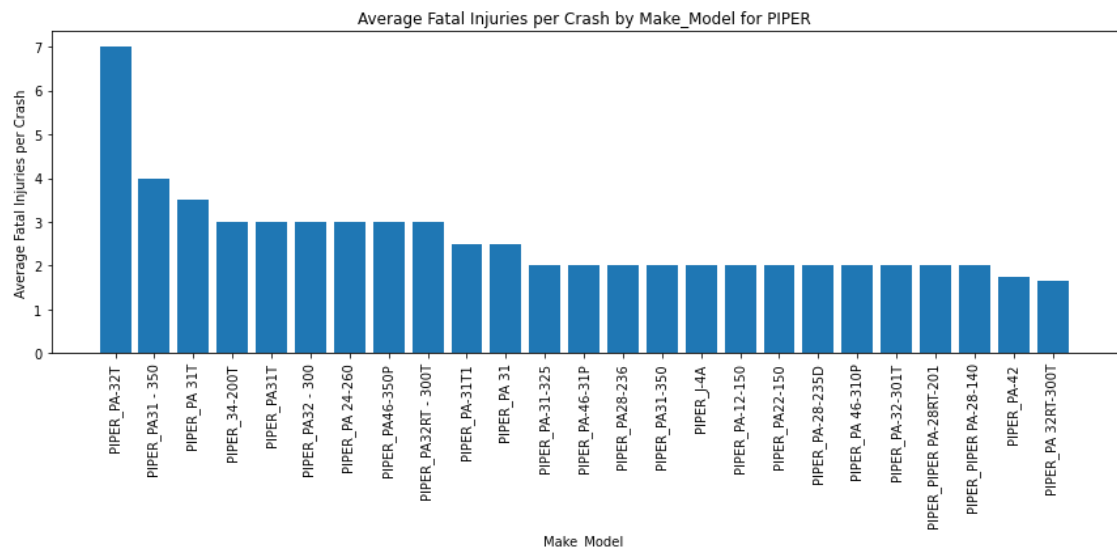
```



```

In [82]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'PIPER'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the average of 'Total
6 make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total
7
8 # Sort the values in descending order
9 make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set Labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Average Fatal Injuries per Crash')
23 ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

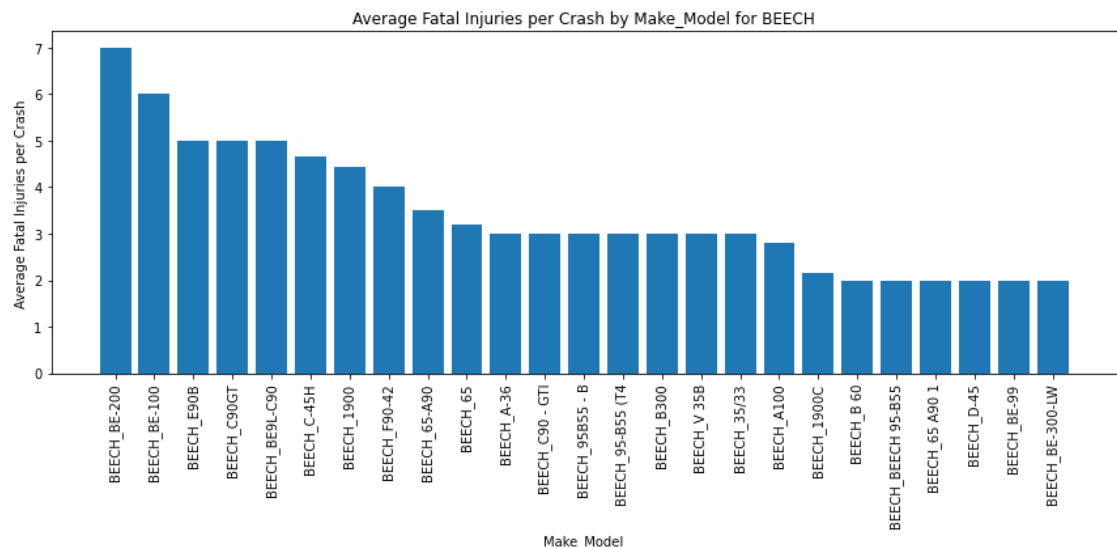
```



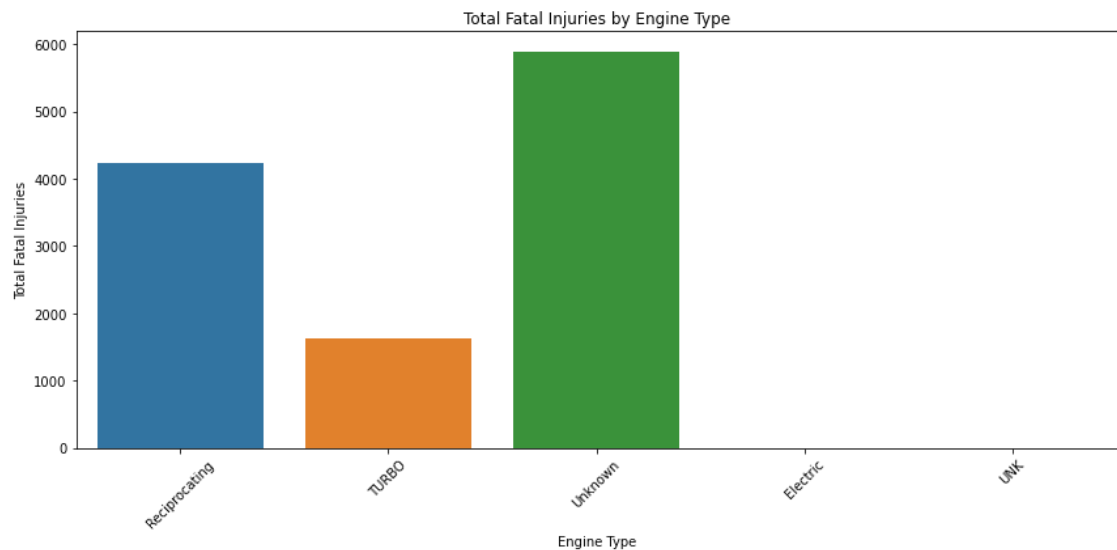
```

In [83]: 1 # Filter the DataFrame to include only rows with the desired 'Make'
2 specific_make = 'BEECH'
3 chart_df = df_1985[df_1985['Make'] == specific_make].copy() # Make a
4
5 # Group the DataFrame by 'Make_Model' and calculate the average of 'Total
6 make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total
7
8 # Sort the values in descending order
9 make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.
10
11 # Set up the figure and axes
12 fig, ax = plt.subplots(figsize=(12, 6))
13
14 # Create the bar plot
15 ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat
16
17 # Rotate the x-axis labels for better visibility
18 plt.xticks(rotation=90)
19
20 # Set labels and title
21 ax.set_xlabel('Make_Model')
22 ax.set_ylabel('Average Fatal Injuries per Crash')
23 ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp
24
25 # Show the plot
26 plt.tight_layout()
27 plt.show()
28

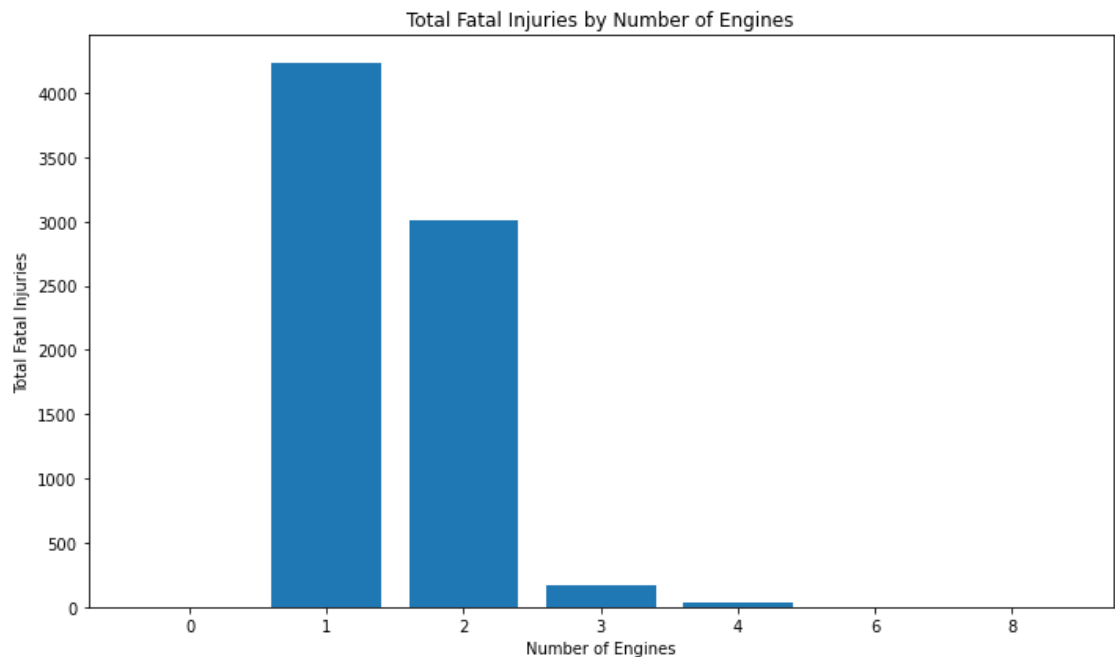
```



```
In [84]: 1 # Set up the figure and axes
2 fig, ax = plt.subplots(figsize=(12, 6))
3
4 # Use seaborn's bar plot to compare 'Engine.Type' and 'Total.Fatal.In
5 sns.barplot(x='Engine.Type', y='Total.Fatal.Injuries', data=df_1985,
6
7 # Set labels and title
8 ax.set_xlabel('Engine Type')
9 ax.set_ylabel('Total Fatal Injuries')
10 ax.set_title('Total Fatal Injuries by Engine Type')
11
12 # Rotate the x-axis labels for better visibility
13 plt.xticks(rotation=45)
14
15 # Show the plot
16 plt.tight_layout()
17 plt.show()
18
```



```
In [85]: ▶ 1 # Group the DataFrame by 'Number.ofEngines' and calculate the sum of
2 engine_injuries_sum = df_1985.groupby('Number.ofEngines')['Total.Fat
3
4 # Set up the figure and axes
5 fig, ax = plt.subplots(figsize=(10, 6))
6
7 # Create the bar plot
8 ax.bar(engine_injuries_sum.index, engine_injuries_sum.values)
9
10 # Set labels and title
11 ax.set_xlabel('Number of Engines')
12 ax.set_ylabel('Total Fatal Injuries')
13 ax.set_title('Total Fatal Injuries by Number of Engines')
14
15 # Show the plot
16 plt.tight_layout()
17 plt.show()
18
```



```
In [86]: ▶ 1 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
2 make_model_fatal_injuries = df_1985.groupby('Make_Model')['Total.Fata
3
4 # Get the value counts of 'Make_Model' to represent occurrences
5 make_model_value_counts = df_1985['Make_Model'].value_counts()
6
7 # Combine the two Series into a DataFrame
8 result_df = pd.DataFrame({'Occurrences': make_model_value_counts, 'To
9
10 # Calculate the average of 'Total.Fatal.Injuries' for each 'Make_Mode
11 result_df['Average.Fatal.Injuries'] = result_df['Total.Fatal.Injuries
12
13 # Sort the DataFrame by average of 'Total.Fatal.Injuries' in ascendin
14 result_df = result_df.sort_values(by='Average.Fatal.Injuries', ascend
15
16 # Display the resulting DataFrame
17 result_df[:50]
18
```

Out[86]:

| | Occurrences | Total.Fatal.Injuries | Average.Fatal.Injuries |
|--------------------------------|-------------|----------------------|------------------------|
| BOEING_777 - 206 | 3 | 534 | 178.000000 |
| AIRBUS_A320 - 216 | 1 | 162 | 162.000000 |
| EMBAER_E135 Legacy | 1 | 154 | 154.000000 |
| AIRBUS_A310 | 1 | 152 | 152.000000 |
| TUPOLEV_TU154 | 1 | 89 | 89.000000 |
| BOEING_MD-82 | 2 | 154 | 77.000000 |
| ANTONOV_AN148 | 1 | 71 | 71.000000 |
| ATR_72-500 | 1 | 58 | 58.000000 |
| BOEING_737 - 500 | 1 | 50 | 50.000000 |
| SUKHOI_SJ100 | 1 | 44 | 44.000000 |
| ATR_72 | 1 | 43 | 43.000000 |
| SUKHOI_RRJ95 | 1 | 41 | 41.000000 |
| AIRBUS_A310-324 | 1 | 30 | 30.000000 |
| BOEING_737-500 | 3 | 88 | 29.333333 |
| Fokker_28-4000 | 1 | 27 | 27.000000 |
| AIRBUS_A321 | 19 | 381 | 20.052632 |
| BOEING_727 | 4 | 80 | 20.000000 |
| Aviocar CASA_C212 | 1 | 18 | 18.000000 |
| EMBAER_ERJ190 - UNDESIGNAT | 2 | 33 | 16.500000 |
| RAYTHEON_B-350 | 1 | 14 | 14.000000 |
| AIRBUS_A330 | 24 | 331 | 13.791667 |
| BRITISH AEROSPACE_Jetstream 32 | 1 | 13 | 13.000000 |
| EMBAER_EMB110 | 2 | 24 | 12.000000 |
| BRITTEN-NORMAN_BN2 | 1 | 12 | 12.000000 |
| TEXTRON_B-300 | 1 | 10 | 10.000000 |
| EMBAER_EMB-820C | 1 | 10 | 10.000000 |
| Swearingen_SA-226TC | 1 | 10 | 10.000000 |
| MOONEY_M-20C | 1 | 10 | 10.000000 |
| RAYTHEON_99A | 1 | 9 | 9.000000 |
| GULFSTREAM_G1159A | 1 | 9 | 9.000000 |
| AIRvan_GA8 | 1 | 9 | 9.000000 |
| BOEING_737-800 | 18 | 154 | 8.555556 |
| BRITISH AEROSPACE_HS 125 700A | 2 | 17 | 8.500000 |
| DORNIER_228 | 4 | 33 | 8.250000 |
| BOMBARDIER_DHC-8-402 | 6 | 49 | 8.166667 |

| | Occurrences | Total.Fatal.Injuries | Average.Fatal.Injuries |
|----------------------------------|-------------|----------------------|------------------------|
| LOCKHEED_L-100 | 1 | 8 | 8.000000 |
| DEHAVILLAND_DHC-6-200 | 2 | 16 | 8.000000 |
| Israel Aircraft Industries_1124A | 1 | 8 | 8.000000 |
| M7Aero_SW3 | 1 | 8 | 8.000000 |
| RAYTHEON_BAE 125-800A | 1 | 8 | 8.000000 |
| EMBAER_EMB820 | 1 | 8 | 8.000000 |
| CANADAIR_CL 600 2B16 | 2 | 15 | 7.500000 |
| EMBAER_EMB-810C | 1 | 7 | 7.000000 |
| BEECH_BE-200 | 1 | 7 | 7.000000 |
| BRITTEN-NORMAN_BN-2A-27 | 1 | 7 | 7.000000 |
| CESSNA_T207 - A | 1 | 7 | 7.000000 |
| BOEING_747-400 BCF | 1 | 7 | 7.000000 |
| PIPER_PA-32T | 1 | 7 | 7.000000 |
| BOEING_B17 | 2 | 13 | 6.500000 |
| PILATUS_PC-12/45 | 3 | 19 | 6.333333 |

```

In [87]: 1 # Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
2 make_model_fatal_injuries = df_1985.groupby('Make_Model')['Total.Fatal
3
4 # Get the value counts of 'Make_Model' to represent occurrences
5 make_model_value_counts = df_1985['Make_Model'].value_counts()
6
7 # Combine the two Series into a DataFrame
8 result_df = pd.DataFrame({'Crashes': make_model_value_counts, 'Total.
9
10 # Calculate the average of 'Total.Fatal.Injuries' for each 'Make_Mode
11 result_df['Average.Fatal.Injuries'] = result_df['Total.Fatal.Injuries
12 result_df = result_df.sort_values(by=['Crashes', 'Average.Fatal.Injur
13
14 result_df[:25]

```

Out[87]:

| | Crashes | Total.Fatal.Injuries | Average.Fatal.Injuries |
|-----------------|---------|----------------------|------------------------|
| CESSNA_172 | 738 | 197 | 0.266938 |
| BOEING_737 | 343 | 1279 | 3.728863 |
| CESSNA_152 | 307 | 50 | 0.162866 |
| CESSNA_182 | 281 | 124 | 0.441281 |
| CESSNA_172-S | 274 | 50 | 0.182482 |
| CIRRUS_SR-22 | 271 | 168 | 0.619926 |
| PIPER_PA28 | 260 | 105 | 0.403846 |
| CESSNA_172-N | 247 | 76 | 0.307692 |
| CESSNA_180 | 211 | 15 | 0.071090 |
| CESSNA_172-M | 181 | 47 | 0.259669 |
| PIPER_PA-18-150 | 172 | 14 | 0.081395 |
| BEECH_A36 | 168 | 121 | 0.720238 |
| CESSNA_150 | 168 | 57 | 0.339286 |
| PIPER_PA-28-140 | 161 | 50 | 0.310559 |
| CESSNA_172-P | 144 | 12 | 0.083333 |
| CESSNA_140 | 114 | 17 | 0.149123 |
| CESSNA_172-R | 110 | 19 | 0.172727 |
| CESSNA_170-B | 108 | 10 | 0.092593 |
| PIPER_PA-28-161 | 106 | 13 | 0.122642 |
| PIPER_PA-28-180 | 102 | 28 | 0.274510 |
| CESSNA_210 | 92 | 56 | 0.608696 |
| PIPER_PA-28-181 | 92 | 31 | 0.336957 |
| MOONEY_M20J | 91 | 45 | 0.494505 |
| CESSNA_A185F | 90 | 18 | 0.200000 |
| AERONCA_7AC | 89 | 15 | 0.168539 |

```
In [88]: 1 # Crash/Fatal Injury data on top
2 top_25_occurrences = result_df.nlargest(25, 'Crashes')
3 lowest_avg_fatal_injuries = top_25_occurrences.nsmallest(25, 'Average
4
5 lowest_avg_fatal_injuries[:25]
```

Out[88]:

| | Crashes | Total.Fatal.Injuries | Average.Fatal.Injuries |
|-----------------|---------|----------------------|------------------------|
| CESSNA_180 | 211 | 15 | 0.071090 |
| PIPER_PA-18-150 | 172 | 14 | 0.081395 |
| CESSNA_172-P | 144 | 12 | 0.083333 |
| CESSNA_170-B | 108 | 10 | 0.092593 |
| PIPER_PA-28-161 | 106 | 13 | 0.122642 |
| CESSNA_140 | 114 | 17 | 0.149123 |
| CESSNA_152 | 307 | 50 | 0.162866 |
| AERONCA_7AC | 89 | 15 | 0.168539 |
| CESSNA_172-R | 110 | 19 | 0.172727 |
| CESSNA_172-S | 274 | 50 | 0.182482 |
| CESSNA_A185F | 90 | 18 | 0.200000 |
| CESSNA_172-M | 181 | 47 | 0.259669 |
| CESSNA_172 | 738 | 197 | 0.266938 |
| PIPER_PA-28-180 | 102 | 28 | 0.274510 |
| CESSNA_172-N | 247 | 76 | 0.307692 |
| PIPER_PA-28-140 | 161 | 50 | 0.310559 |
| PIPER_PA-28-181 | 92 | 31 | 0.336957 |
| CESSNA_150 | 168 | 57 | 0.339286 |
| PIPER_PA28 | 260 | 105 | 0.403846 |
| CESSNA_182 | 281 | 124 | 0.441281 |
| MOONEY_M20J | 91 | 45 | 0.494505 |
| CESSNA_210 | 92 | 56 | 0.608696 |
| CIRRUS_SR-22 | 271 | 168 | 0.619926 |
| BEECH_A36 | 168 | 121 | 0.720238 |
| BOEING_737 | 343 | 1279 | 3.728863 |

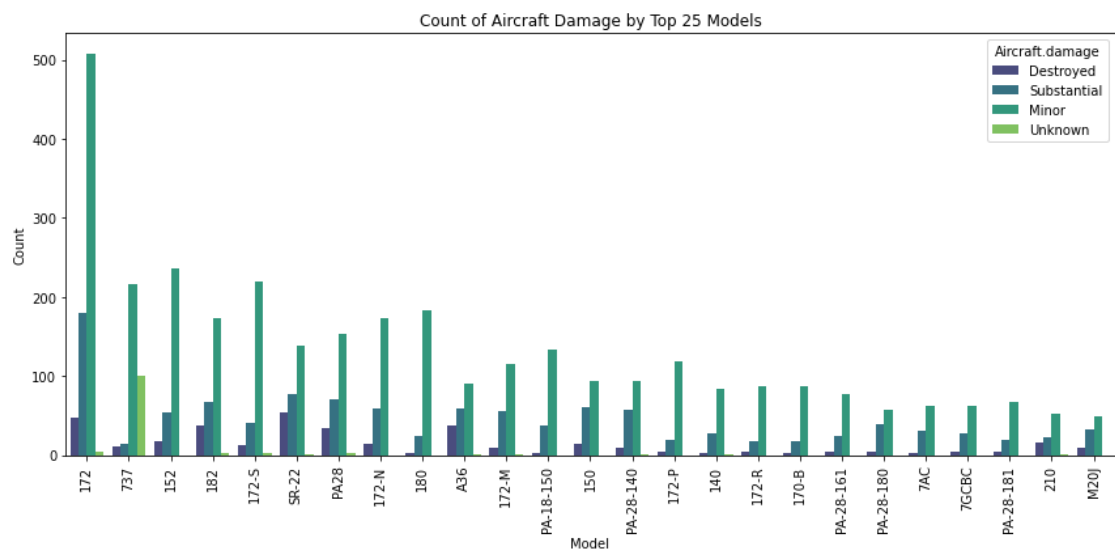
```
In [89]: 1 # Group the DataFrame by 'Model' and calculate the sum of 'Total.Uninjured'
2 model_uninjured = df_1985.groupby('Make_Model')['Total.Uninjured'].sum()
3 most_uninjured_models = model_uninjured.sort_values(ascending=False)
4
5 most_uninjured_models[:25]
6
```

```
Out[89]: Make_Model
BOEING_737          18988.0
BOEING_777           7743.0
BOEING_767           4889.0
BOEING_757           3039.0
AIRBUS_A320           2788.0
AIRBUS_A330           2770.0
BOEING_747           2320.0
BOEING_787           2290.0
BOEING_747-400        1712.0
BOEING_737_7H4         1704.0
BOEING_737-7H4         1584.0
BOEING_777-222         1523.0
AIRBUS_A330-323         1395.0
AIRBUS_A321           1349.0
BOEING_737-800         1229.0
AIRBUS_A380           1097.0
EMBAER_EMB-145LR         955.0
BOEING_757-222           908.0
BOEING_767_332           898.0
MCDONNELL DOUGLAS_MD80   894.0
AIRBUS_A321_231          848.0
AIRBUS_A330_323          847.0
BOEING_737-300          846.0
BOEING_747-422          846.0
CESSNA_172             815.0
Name: Total.Uninjured, dtype: float64
```

```

In [90]: 1 # Convert 'Model' column to string to avoid potential issues with plo
2 df_1985['Model'] = df_1985['Model'].astype(str)
3
4 # Fill missing values in 'Aircraft.damage' column with 'Unknown'
5 df_1985['Aircraft.damage'].fillna('Unknown', inplace=True)
6
7 # Get the top 25 unique 'Model' values based on their counts
8 top_25_models = df_1985['Model'].value_counts().nlargest(25).index
9
10 # Filter the DataFrame to include only the top 25 'Model' values
11 df_top_25_models = df_1985[df_1985['Model'].isin(top_25_models)]
12
13 # Set up the figure and axes
14 fig, ax = plt.subplots(figsize=(12, 6))
15
16 # Use seaborn's countplot to create the bar chart
17 sns.countplot(x='Model', hue='Aircraft.damage', data=df_top_25_models
18               order=top_25_models) # Specify the order of x-axis lab
19
20 # Rotate the x-axis labels for better visibility
21 plt.xticks(rotation=90)
22
23 # Set labels and title
24 ax.set_xlabel('Model')
25 ax.set_ylabel('Count')
26 ax.set_title('Count of Aircraft Damage by Top 25 Models')
27
28 # Show the plot
29 plt.tight_layout()
30 plt.show()
31
32

```



Conclusions

Recommendations

- 1-We should target Models with more engines as the more engines on a plane, the less likely Fatal Injuries are to occur. More Engines means the plane is larger. We can tell by the frequency of crashes with the BOEING 737 and the amount of 'TotalUninjured' and 'Total.Fatal.Injuries' that the **BOEING 737 is one of the largest planes in our dataset.**
- 2-There is a **negative correlation between Turbo engines and Fatal Injuries** as well, indicating that Turbo Engines are much safer than Reciprocating. BOEING and AIRBUS largely use Turbo fans and CESSNA uses Reciprocating.
- 3-Based on the data we have present and a simple review of popularity of Models, the **Cessna 172 has one of the best safety record of planes in this dataset.** It ranks 25th in most uninjured passengers of all planes.

Limitations

- This is only crash data, so if a plane theoretically has never crashed before, it would not be in this dataset.
- The majority of the rows were not used in final calculations due to so many NaN/missing values.
- This data is US based which limits international analysis.

Next Steps

- Make a deeper analysis of Boeing 737 and Cessna 172.
- Analyze cost of maintenance for these 2 planes

```
In [91]: 1 #df_1985.shape
```

```
In [92]: 1 df_1985.to_csv('edited_aviation_dataset.csv', index=False)
```