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)](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 donwloaded 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 arcraft 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]:

```
# import libraries for data analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

In [2]:

```
#loading the AviationData.csv file
# using encoding 'Latin-1' to handle values
df = pd.read_csv('data/AviationData.csv',
                 encoding='Latin-1',
                 low_memory=False,
                 dtype=str)
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]:

```
#determining shape of dataset
df.shape
```

Out[3]: (88889, 31)

In [4]:

```
#examinig null values
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]:

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

Out[5]:

Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States
20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States
20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States
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]:
```
# Calculating NaN values sum for each column
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]:
```
#Removing whitespaces from column names
df.columns = df.columns.str.strip()
```

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

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

In [9]:

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

```
#keeping rows where the Airplanes are NOT built by amateurs
df = df[df['Amateur.Built'] =='No']

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]:

```
# Determining value type
type(df['Total.Fatal.Injuries'][0])
```

Out[11]:

```
float
```

In [12]:

```
# Quick spot check to identify NaN value
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]:

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

In [14]:

```
# Spot Check
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.
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]:

```
# Checking NaN values by counting sum of isna() for each column
nan_values = df.isna()

nan_counts = nan_values.sum()

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]:

```
# Create a boolean mask for rows with NaN values in 'Total.Fatal.Inju
mask_nan_values = df['Total.Fatal.Injuries'].isna()

# Use the boolean mask to select the rows with NaN values
rows_with_nan = df[mask_nan_values]

# Display the rows with NaN values
rows_with_nan

```

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]:

```
# Spot Check NaN values are removed
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]:

```
df.shape
```

Out[18]:

```
(24417, 30)
```

In [19]:

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

In [20]:

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

Out[20]:

Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Lat
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]:
```
# Checking my code
df_1985 = df_current.copy()
df_1985['Event.Date'] = pd.to_datetime(df_1985.loc[:, 'Event.Date'])
num_rows_before_1985 = (df_1985['Event.Date'].dt.year < 1985).sum()
print("Number of rows with a date before 1985:", num_rows_before_1985

```

```
Number of rows with a date before 1985: 0
```

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

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

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

Out[23]:
```
array(['Destroyed', 'Substantial', nan, 'Minor', 'Unknown'], dtype=objec
t)
```

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

In [25]:
```
# Removes all rows where 'Total.Fatal.Injuries' is NaN
df_1985 = df_1985.dropna(subset=['Total.Fatal.Injuries'], axis=0)
# Ensuring all values in the 'Total.Fatal.Injuries' column are intege
df_1985['Total.Fatal.Injuries'] = df_1985['Total.Fatal.Injuries'].ast
```

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

Out[26]:
```
0.6491812508797448
```

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

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

Out[28]:
```
3514
```

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

In [30]:

```
# Cleaning up 'Model' values to reduce redundancy
df_1985.loc[df_1985['Model'] == '108 3', 'Model'] = '108-3'
df_1985.loc[df_1985['Model'] == '100 180', 'Model'] = '100-180'
df_1985.loc[df_1985['Model'] == '108 1', 'Model'] = '108-1'
df_1985.loc[df_1985['Model'] == '114 B', 'Model'] = '114-B'
df_1985.loc[df_1985['Model'] == '114B', 'Model'] = '114-B'
df_1985.loc[df_1985['Model'] == '14 19', 'Model'] = '14-19'
df_1985.loc[df_1985['Model'] == '150 - F', 'Model'] = '150-F'
df_1985.loc[df_1985['Model'] == '150 - G', 'Model'] = '150-G'
df_1985.loc[df_1985['Model'] == '164B', 'Model'] = '164-B'
df_1985.loc[df_1985['Model'] == '17 30', 'Model'] = '17-30'
df_1985.loc[df_1985['Model'] == '17 30A', 'Model'] = '17-30A'
df_1985.loc[df_1985['Model'] == '17 31A', 'Model'] = '17-31A'
df_1985.loc[df_1985['Model'] == '17 30', 'Model'] = '17-30'
df_1985.loc[df_1985['Model'] == '170 B', 'Model'] = '170-B'
df_1985.loc[df_1985['Model'] == '170 - B', 'Model'] = '170-B'
df_1985.loc[df_1985['Model'] == '170B', 'Model'] = '170-B'
df_1985.loc[df_1985['Model'] == '172 M', 'Model'] = '172-M'
df_1985.loc[df_1985['Model'] == '172 - M', 'Model'] = '172-M'
df_1985.loc[df_1985['Model'] == '172M', 'Model'] = '172-M'
df_1985.loc[df_1985['Model'] == '172P', 'Model'] = '172-P'
df_1985.loc[df_1985['Model'] == '172 P', 'Model'] = '172-P'
df_1985.loc[df_1985['Model'] == '172N', 'Model'] = '172-N'
df_1985.loc[df_1985['Model'] == '172 N', 'Model'] = '172-N'
df_1985.loc[df_1985['Model'] == '172A', 'Model'] = '172-A'
df_1985.loc[df_1985['Model'] == '172 A', 'Model'] = '172-A'
df_1985.loc[df_1985['Model'] == '172S', 'Model'] = '172-S'
df_1985.loc[df_1985['Model'] == '172 S', 'Model'] = '172-S'
df_1985.loc[df_1985['Model'] == '172F', 'Model'] = '172-F'
df_1985.loc[df_1985['Model'] == '172 - F', 'Model'] = '172-F'
df_1985.loc[df_1985['Model'] == '172 - H', 'Model'] = '172-H'
df_1985.loc[df_1985['Model'] == '172H', 'Model'] = '172-H'
df_1985.loc[df_1985['Model'] == '172 - S', 'Model'] = '172-S'
df_1985.loc[df_1985['Model'] == '172 - P', 'Model'] = '172-P'
df_1985.loc[df_1985['Model'] == '172 - N', 'Model'] = '172-N'
df_1985.loc[df_1985['Model'] == '172 K', 'Model'] = '172-K'
df_1985.loc[df_1985['Model'] == '172K', 'Model'] = '172-K'
df_1985.loc[df_1985['Model'] == '172 - R', 'Model'] = '172-R'
df_1985.loc[df_1985['Model'] == '172R', 'Model'] = '172-R'
df_1985.loc[df_1985['Model'] == '172RG', 'Model'] = '172-RG'
df_1985.loc[df_1985['Model'] == '172 RG', 'Model'] = '172-RG'
df_1985.loc[df_1985['Model'] == '172SP', 'Model'] = '172-SP'
df_1985.loc[df_1985['Model'] == '172 SP', 'Model'] = '172-SP'
df_1985.loc[df_1985['Model'] == '1730A', 'Model'] = '1730-A'
df_1985.loc[df_1985['Model'] == '1730 - A', 'Model'] = '1730-A'
df_1985.loc[df_1985['Model'] == '172B', 'Model'] = '172-B'
df_1985.loc[df_1985['Model'] == '172C', 'Model'] = '172-C'
df_1985.loc[df_1985['Model'] == '172D', 'Model'] = '172-D'
df_1985.loc[df_1985['Model'] == '172E', 'Model'] = '172-E'
df_1985.loc[df_1985['Model'] == '172G', 'Model'] = '172-G'
df_1985.loc[df_1985['Model'] == '172I', 'Model'] = '172-I'
df_1985.loc[df_1985['Model'] == '172L', 'Model'] = '172-L'
df_1985.loc[df_1985['Model'] == '172Q', 'Model'] = '172-Q'
df_1985.loc[df_1985['Model'] == '172XP', 'Model'] = '172-XP'
df_1985.loc[df_1985['Model'] == '175A', 'Model'] = '175-A'
df_1985.loc[df_1985['Model'] == '175B', 'Model'] = '175-B'
df_1985.loc[df_1985['Model'] == '175C', 'Model'] = '175-C'
```

```
df_1985.loc[df_1985['Model'] == '177 RG', 'Model'] = '177-RG'
df_1985.loc[df_1985['Model'] == '177RG', 'Model'] = '177-RG'
df_1985.loc[df_1985['Model'] == '177A', 'Model'] = '177-A'
df_1985.loc[df_1985['Model'] == '177B', 'Model'] = '177-B'
df_1985.loc[df_1985['Model'] == '180 - B', 'Model'] = '180-B'
df_1985.loc[df_1985['Model'] == '180 H', 'Model'] = '180-H'
df_1985.loc[df_1985['Model'] == '180A', 'Model'] = '180-A'
df_1985.loc[df_1985['Model'] == '180 H', 'Model'] = '180-H'
df_1985.loc[df_1985['Model'] == '180A', 'Model'] = '180-A'
df_1985.loc[df_1985['Model'] == '180B', 'Model'] = '180-B'
df_1985.loc[df_1985['Model'] == '180C', 'Model'] = '180-C'
df_1985.loc[df_1985['Model'] == '180D', 'Model'] = '180-D'
df_1985.loc[df_1985['Model'] == '180E', 'Model'] = '180-E'
df_1985.loc[df_1985['Model'] == '180F', 'Model'] = '180-F'
df_1985.loc[df_1985['Model'] == '180G', 'Model'] = '180-G'
df_1985.loc[df_1985['Model'] == '180J', 'Model'] = '180-J'
df_1985.loc[df_1985['Model'] == '180H', 'Model'] = '180-H'
df_1985.loc[df_1985['Model'] == '180J', 'Model'] = '180-J'
df_1985.loc[df_1985['Model'] == '180K', 'Model'] = '180-K'
df_1985.loc[df_1985['Model'] == '180M', 'Model'] = '180-M'
df_1985.loc[df_1985['Model'] == '185E', 'Model'] = '185-E'
df_1985.loc[df_1985['Model'] == '185 - E', 'Model'] = '180-E'
df_1985.loc[df_1985['Model'] == '206 - H', 'Model'] = '206-H'
df_1985.loc[df_1985['Model'] == '206H', 'Model'] = '206-H'
df_1985.loc[df_1985['Model'] == '210 5', 'Model'] = '210-5'
df_1985.loc[df_1985['Model'] == '210 5(205)', 'Model'] = '210-5(205)'
df_1985.loc[df_1985['Model'] == '210N', 'Model'] = '210-N'
df_1985.loc[df_1985['Model'] == '210 - N', 'Model'] = '210-N'
df_1985.loc[df_1985['Model'] == '210D', 'Model'] = '210-D'
df_1985.loc[df_1985['Model'] == '210 D', 'Model'] = '210-D'
df_1985.loc[df_1985['Model'] == '210A', 'Model'] = '210-A'
df_1985.loc[df_1985['Model'] == '210B', 'Model'] = '210-B'
df_1985.loc[df_1985['Model'] == '210C', 'Model'] = '210-C'
df_1985.loc[df_1985['Model'] == '210E', 'Model'] = '210-E'
df_1985.loc[df_1985['Model'] == '210F', 'Model'] = '210-F'
df_1985.loc[df_1985['Model'] == '210G', 'Model'] = '210-G'
df_1985.loc[df_1985['Model'] == '210H', 'Model'] = '210-H'
df_1985.loc[df_1985['Model'] == '210J', 'Model'] = '210-J'
df_1985.loc[df_1985['Model'] == '210K', 'Model'] = '210-K'
df_1985.loc[df_1985['Model'] == '210L', 'Model'] = '210-L'
df_1985.loc[df_1985['Model'] == '210M', 'Model'] = '210-M'
df_1985.loc[df_1985['Model'] == '2T 1A', 'Model'] = '2T-1A'
df_1985.loc[df_1985['Model'] == '2T1A', 'Model'] = '2T-1A'
df_1985.loc[df_1985['Model'] == '2T 1A 2', 'Model'] = '2T-1A-2'
df_1985.loc[df_1985['Model'] == '305 A', 'Model'] = '305-A'
df_1985.loc[df_1985['Model'] == '305A', 'Model'] = '305-A'
df_1985.loc[df_1985['Model'] == '305C', 'Model'] = '305-C'
df_1985.loc[df_1985['Model'] == '35 33', 'Model'] = '35-33'
df_1985.loc[df_1985['Model'] == '35 A33', 'Model'] = '35-A33'
df_1985.loc[df_1985['Model'] == '35A33', 'Model'] = '35-A33'
df_1985.loc[df_1985['Model'] == '35 B33', 'Model'] = '35-B33'
df_1985.loc[df_1985['Model'] == '35B33', 'Model'] = '35-B33'
df_1985.loc[df_1985['Model'] == '35C33', 'Model'] = '35-C33'
df_1985.loc[df_1985['Model'] == '35 C33', 'Model'] = '35-C33'
df_1985.loc[df_1985['Model'] == '35 - A', 'Model'] = '35-A'
df_1985.loc[df_1985['Model'] == '35A', 'Model'] = '35-A'
df_1985.loc[df_1985['Model'] == '415 C', 'Model'] = '415-C'
```

```
df_1985.loc[df_1985['Model'] == '415C', 'Model'] = '415-C'
df_1985.loc[df_1985['Model'] == '415 CD', 'Model'] = '415-CD'
df_1985.loc[df_1985['Model'] == '415 C/D', 'Model'] = '415-CD'
df_1985.loc[df_1985['Model'] == '415-C/D', 'Model'] = '415-CD'
df_1985.loc[df_1985['Model'] == '415 D', 'Model'] = '415-D'
df_1985.loc[df_1985['Model'] == '415D', 'Model'] = '415-D'
df_1985.loc[df_1985['Model'] == '415G', 'Model'] = '415-G'
df_1985.loc[df_1985['Model'] == '421 - C', 'Model'] = '421-C'
df_1985.loc[df_1985['Model'] == '421C', 'Model'] = '421-C'
df_1985.loc[df_1985['Model'] == '500 - B', 'Model'] = '500-B'
df_1985.loc[df_1985['Model'] == '500 B', 'Model'] = '500-B'
df_1985.loc[df_1985['Model'] == '500B', 'Model'] = '500-B'
df_1985.loc[df_1985['Model'] == '500 S', 'Model'] = '500-S'
df_1985.loc[df_1985['Model'] == '560 - XL', 'Model'] = '560-XL'
df_1985.loc[df_1985['Model'] == '560XL', 'Model'] = '560-XL'
df_1985.loc[df_1985['Model'] == '402A', 'Model'] = 'AT-402A'
df_1985.loc[df_1985['Model'] == '402B', 'Model'] = 'AT-402B'
df_1985.loc[df_1985['Model'] == 'S2-R', 'Model'] = 'S2R'
df_1985.loc[df_1985['Model'] == 'S-2R', 'Model'] = 'S2R'
df_1985.loc[df_1985['Model'] == 'SR2', 'Model'] = 'S2R'
df_1985.loc[df_1985['Model'] == 'CH2000', 'Model'] = 'CH-2000'
df_1985.loc[df_1985['Model'] == 'CH 2000', 'Model'] = 'CH-2000'
df_1985.loc[df_1985['Model'] == 'A1A', 'Model'] = 'A-1A'
df_1985.loc[df_1985['Model'] == 'A1-A', 'Model'] = 'A-1A'
df_1985.loc[df_1985['Model'] == 'Husky A1-B', 'Model'] = 'Husky A-1B'
df_1985.loc[df_1985['Model'] == 'SR22', 'Model'] = 'SR-22'
df_1985.loc[df_1985['Model'] == 'YMF 5C', 'Model'] = 'YMF-5C'
df_1985.loc[df_1985['Model'] == 'Gulfstream AM G-164B', 'Model'] = 'G
model_counts = df_1985['Model'].value_counts().sort_index()
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]:

```
# Removed 78 redundant 'Model' values
df_1985['Model'].nunique()
```

Out[31]: 3436

In [32]:

```
df_1985['Make'].nunique()
```

Out[32]: 1329

In [33]:

```
# Consolidating 'Make' values
df_1985['Make'].fillna('', inplace=True)
df_1985.loc[df_1985['Make'].str.contains('Cessna', case=False), 'Make
df_1985.loc[df_1985['Make'].str.contains('Piper', case=False), 'Make'
df_1985.loc[df_1985['Make'].str.contains('Beechcraft|Beech', case=Fal
df_1985.loc[df_1985['Make'].str.contains('Boeing|Boeing Stearman', ca
df_1985.loc[df_1985['Make'].str.contains('Air tractor', case=False),
df_1985.loc[df_1985['Make'].str.contains('Mooney', case=False), 'Make
df_1985.loc[df_1985['Make'].str.contains('CIRRUS', case=False), 'Make
df_1985.loc[df_1985['Make'].str.contains('American', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Airbus', case=False), 'Make
df_1985.loc[df_1985['Make'].str.contains('Grumman', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Bellanca', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Maule', case=False), 'Make'
df_1985.loc[df_1985['Make'].str.contains('Aeronca', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Embraer', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Champion', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Luscombe', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Stinson', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('TaylorCraft', case=False),
df_1985.loc[df_1985['Make'].str.contains('Dehavilland|De havilland',
df_1985.loc[df_1985['Make'].str.contains('Ayres', case=False), 'Make'
df_1985.loc[df_1985['Make'].str.contains('Raytheon', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Diamond Aircraft', case=Fal
df_1985.loc[df_1985['Make'].str.contains('Grumman Schweizer', case=Fa
df_1985.loc[df_1985['Make'].str.contains('Gulfstream-Schweizer|Gulfst
df_1985.loc[df_1985['Make'].str.contains('Gulfstream American', case=
df_1985.loc[df_1985['Make'].str.contains('Ted Aerostar', case=False),
df_1985.loc[df_1985['Make'].str.contains('Lockheed', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Gulfstream Aerospace|Gulfst
df_1985.loc[df_1985['Make'].str.contains('Northrop', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Helio', case=False), 'Make'
df_1985.loc[df_1985['Make'].str.contains('Canadair', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Learjet', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Volmer', case=False), 'Make
df_1985.loc[df_1985['Make'].str.contains('Ryan', case=False), 'Make']
df_1985.loc[df_1985['Make'].str.contains('Britten', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Howard', case=False), 'Make
df_1985.loc[df_1985['Make'].str.contains('British Airways', case=Fals
df_1985.loc[df_1985['Make'].str.contains('British Aerospace', case=Fa
df_1985.loc[df_1985['Make'].str.contains('Textron', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Aviat', case=False), 'Make'
df_1985.loc[df_1985['Make'].str.contains('Waco', case=False), 'Make']
df_1985.loc[df_1985['Make'].str.contains('Quicksilver', case=False),
df_1985.loc[df_1985['Make'].str.contains('Cub', case=False), 'Make']
df_1985.loc[df_1985['Make'].str.contains('Bombardier', case=False), '
df_1985.loc[df_1985['Make'].str.contains('Dassault', case=False), 'Ma
df_1985.loc[df_1985['Make'].str.contains('Found A', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Ercoupe', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Extra Flugzeugbau', case=Fa
df_1985.loc[df_1985['Make'].str.contains('Evektor', case=False), 'Mak
df_1985.loc[df_1985['Make'].str.contains('Flight Design', case=False)
```

In [34]:

```
# Fill missing values in 'Engine.Type' with a placeholder value (e.g.
df_1985.loc[:, 'Engine.Type'].fillna('Unknown', inplace=True)

# Replace 'Engine.Type' values containing 'Turbo' (case-insensitive)
df_1985.loc[df_1985['Engine.Type'].str.contains('Turbo', case=False),

```

In [35]:

```
# Viewing top 35 most common 'Model' values
model_frequency = df_1985['Model'].value_counts()
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
```

In [36]:

```
# Viewing top 35 most common 'Make' values
make_frequency = df_1985['Make'].value_counts()
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
```

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

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

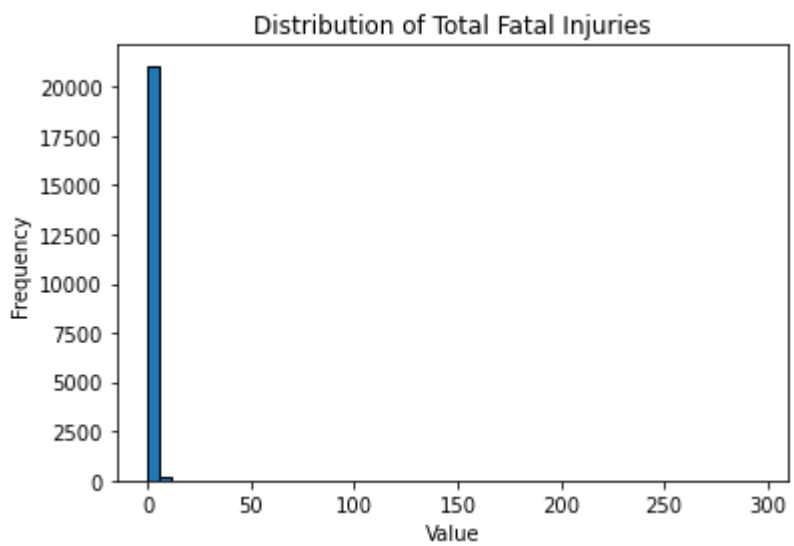
Out[38]: 997

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

Data Analysis

- Checking for trends in data
- Creating new volumn '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'

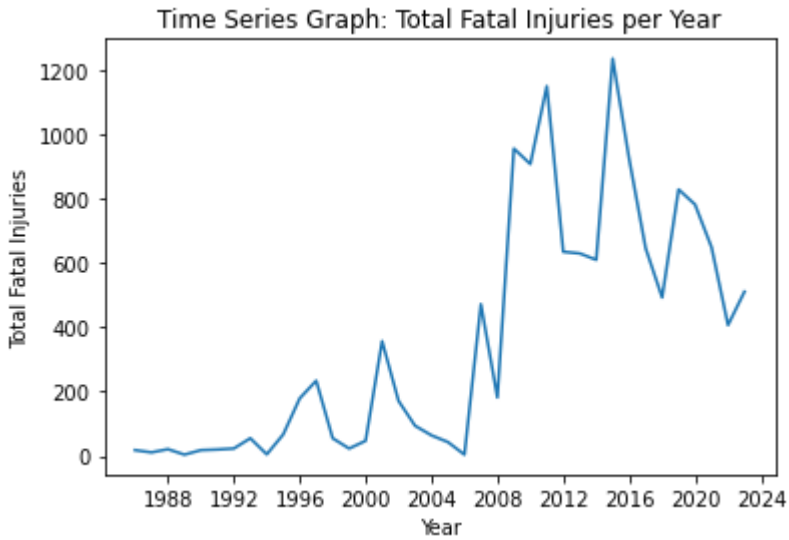
In [40]:
```
# Viewing Distrobution of 'Total.Fatal.Injuries' values
plt.hist(df_1985['Total.Fatal.Injuries'], bins=50, edgecolor='black')

plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Distribution of Total Fatal Injuries')

plt.show()
```



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

In [42]:

```
# Plotting 'Total.Fatal.Injuries' over time
df_1985['Total.Fatal.Injuries'] = pd.to_numeric(df_1985['Total.Fatal.

df_resampled = df_1985.resample('Y', on='Event.Date')['Total.Fatal.In

plt.plot(df_resampled.index, df_resampled)
plt.xlabel('Year')
plt.ylabel('Total Fatal Injuries')
plt.title('Time Series Graph: Total Fatal Injuries per Year')
plt.show()
```



In [43]:

```
#df_1985.shape
```

In [44]:

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

Out[44]:

Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Cou
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]:
```
#df_1985.shape
```

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

Out[46]:

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

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

Out[47]:

Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
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]:
```
#df_1985.shape
```

- Investigating frequency of Make and Model when using groupby with Total.Fatal.Injuries

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

Out[49]:
```
CESSNA                            6922
PIPER                             3921
BEECH                             1450
BOEING                            1089
AIR TRACTOR                        419
                                  ...
Believer                             1
Scottish                             1
BLUE SIDE UP INC                     1
AIRCRAFT MFG & DESIGN LLC            1
Remos Aircraft GMBH Flugzeugba       1
Name: Make, Length: 952, dtype: int64
```

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

fatalities_with_specific_model
```

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

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

- Looking at distrobution of Number.of.Engines, FAR_Description, Weather.Condition

In [53]:
```
Engine_numbers = df_1985['Number.of.Engines'].value_counts()
Engine_numbers
```

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

In [54]:
```
FAR_description = df_1985['FAR.Description'].value_counts()
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
Part 129: Foreign                      4
ARMF                                   4
Unknown                                4
Non-U.S., Commercial                   4
Part 91 Subpart K: Fractional          1
107                                    1
Part 125: 20+ Pax,6000+ lbs            1
Public Aircraft                        1
Name: FAR.Description, dtype: int64
```

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

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

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

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

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

In [58]:

```
#Showing which Broad.phase.of.flight has the most fatal injuries
grouped_df = df_1985.groupby('Broad.phase.of.flight')['Total.Fatal.In
fatalaties_by_phase = grouped_df.sort_values(ascending=False)
fatalaties_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]:

```
# Viewing how often each 'Make' experiences a specific level of 'Airc
specific_damage_counts = df_1985.groupby(['Make', 'Aircraft.damage'])
specific_damage_counts_sorted = specific_damage_counts.sort_values(as
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]:
```
#Total.Fatal.Injuries by Aircraft.Damage
Damage_Fatal = df_1985.groupby('Aircraft.damage')['Total.Fatal.Injuri
fatalaties_by_Damage = Damage_Fatal.sort_values(ascending=False)
fatalaties_by_Damage
```

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

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

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

In [62]:
```
#Total.Fatal.Injuries by Make
Make_Fatal = df_1985.groupby('Make')['Total.Fatal.Injuries'].sum()
fatalaties_by_Make = Make_Fatal.sort_values(ascending=False)
fatalaties_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
```

In [63]:

```
#Total.Fatal.Injuries by 'Make_Model'
df_1985.loc[:, 'Make_Model'] = df_1985.loc[:, 'Make'] + '_' + df_1985
Model_Fatal = df_1985.groupby('Make_Model')['Total.Fatal.Injuries'].s
fatalaties_by_Model = Model_Fatal.sort_values(ascending=False)
fatalaties_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
BEECH_58                        85
CESSNA_208B                     82
BOEING_727                      80
CESSNA_172-N                    76
ANTONOV_AN148                   71
BEECH_1900                      62
ATR_72-500                      58
CESSNA_150                      57
CESSNA_210                      56
CIRRUS_SR20                     53
CESSNA_152                      50
PIPER_PA-28-140                 50
BOEING_737 - 500                50
CESSNA_172-S                    50
BOMBARDIER_DHC-8-402            49
PIPER_PA34                      48
CESSNA_172-M                    47
MOONEY_M20J                     45
SUKHOI_SJ100                    44
BEECH_200                       44
PIPER_PA-31-350                 43
ATR_72                          43
CESSNA_421-C                    42
EMBAER_ERJ190                   42
SUKHOI_RRJ95                    41
PIPER_PA31                      39
DEHAVILLAND_DHC-2               36
PIPER_PA-32R-300                36
CESSNA_U206F                    35
CESSNA_U206                     34
DEHAVILLAND_DHC-3               34
EMBAER_ERJ190 - UNDESIGNAT      33
Name: Total.Fatal.Injuries, dtype: int32
```

Pulling all make an models to spot check fro duplicates.

In [64]:

```
#grouped_by_make = df_1985.groupby('Make')['Model'].unique()

#for make, models in grouped_by_make.items():
#    print(f"Make: {make}")
#    print("Models:")
#    for model in models:
#        print(f"  - {model}")
#    print()
```

In [65]:

```
#Checking value counts for top 25 models.  Using the new 'Make_Model'
df_1985['Make_Model'].value_counts()[:25]
```

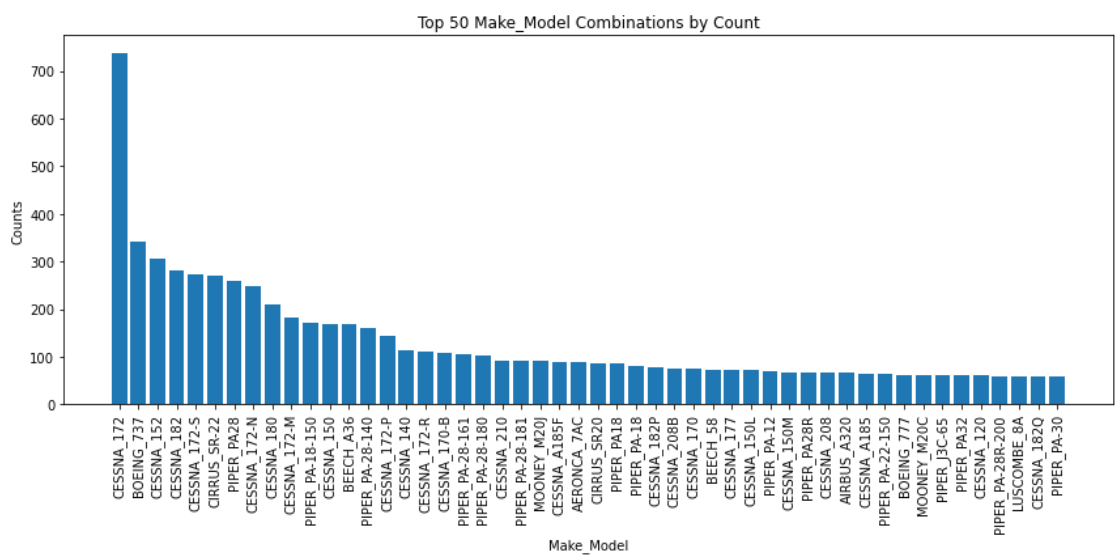
Out[65]:

```
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
CESSNA_150         168
BEECH_A36          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
```

- Now we're going to create some visuals to better understand trends within specific 'Make" values.
- But we've created the 'Make_Model' column so it's easier to read!

In [66]:

```
# Count the occurrences of each 'Make_Model' combination
make_model_counts = df_1985['Make_Model'].value_counts()

# Sort the values in descending order and select the top 50 entries (
top_50_counts = make_model_counts.sort_values(ascending=False)[:50]

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(top_50_counts.index, top_50_counts.values)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Counts')
ax.set_title('Top 50 Make_Model Combinations by Count')

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


```

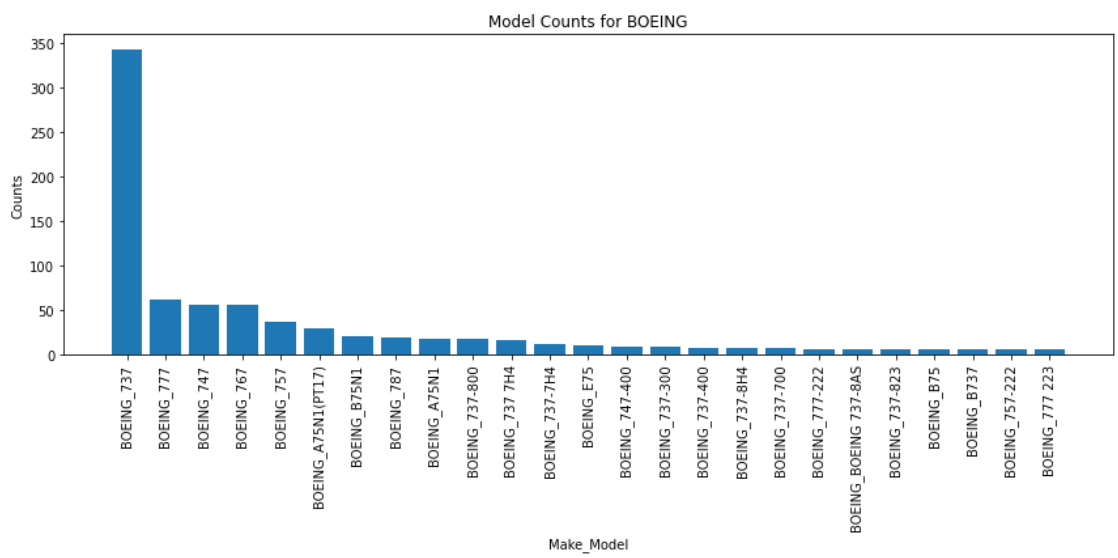


In [67]:

```
#df_1985.shape
```

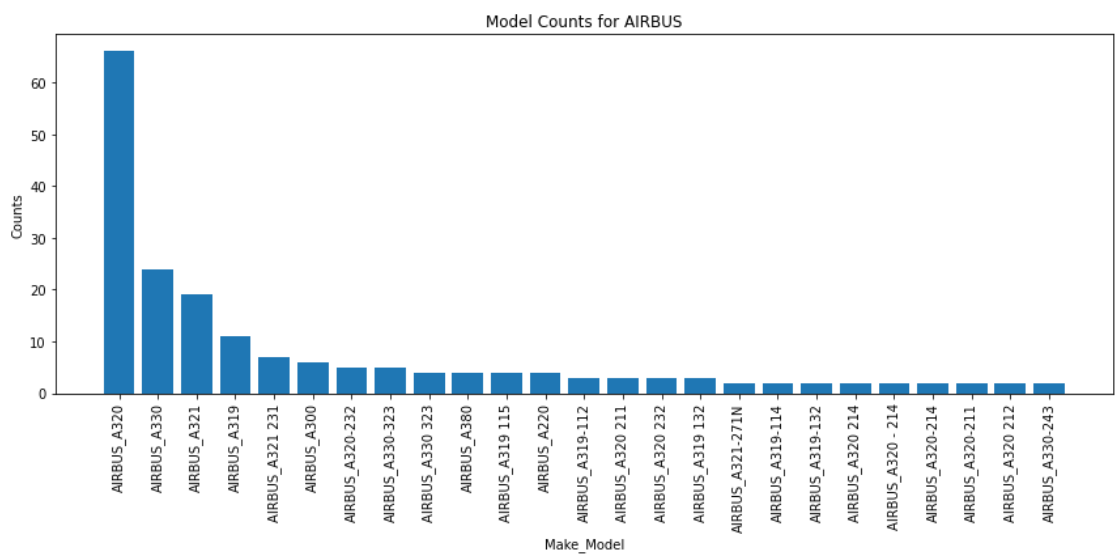
In [68]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'BOEING'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Combine 'Make' and 'Model' columns to create a new column represent
chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']

# Count the occurrences of each 'Make_Model' combination
make_model_counts = chart_df['Make_Model'].value_counts()

# Sort the values in descending order and select all entries
all_make_model_counts = make_model_counts.sort_values(ascending=False

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(all_make_model_counts.index, all_make_model_counts.values)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Counts')
ax.set_title(f'Model Counts for {specific_make}')

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



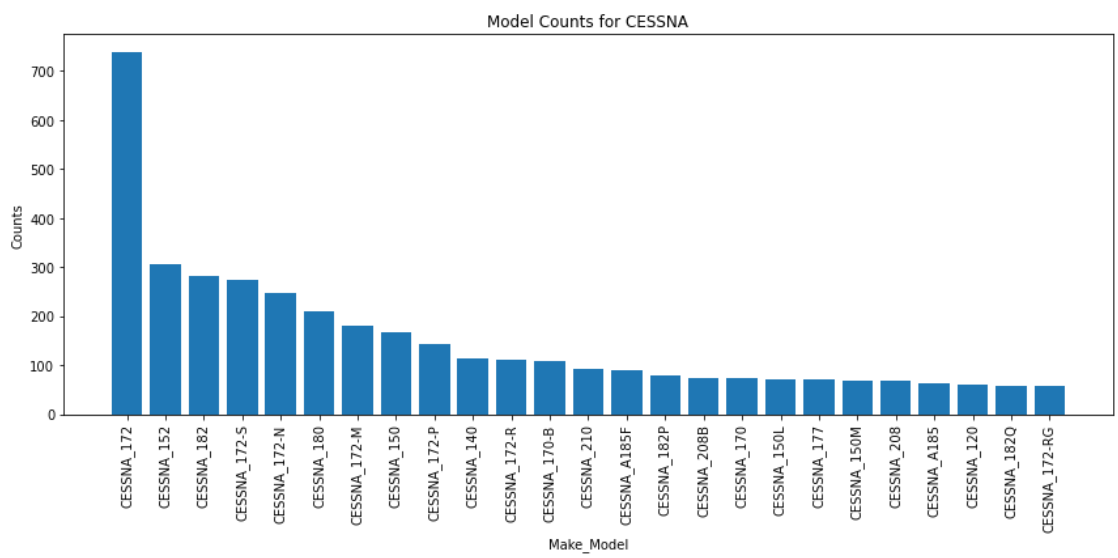
In [69]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'AIRBUS'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Combine 'Make' and 'Model' columns to create a new column represent
chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']

# Count the occurrences of each 'Make_Model' combination
make_model_counts = chart_df['Make_Model'].value_counts()

# Sort the values in descending order and select all entries
all_make_model_counts = make_model_counts.sort_values(ascending=False

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(all_make_model_counts.index, all_make_model_counts.values)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Counts')
ax.set_title(f'Model Counts for {specific_make}')

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

```



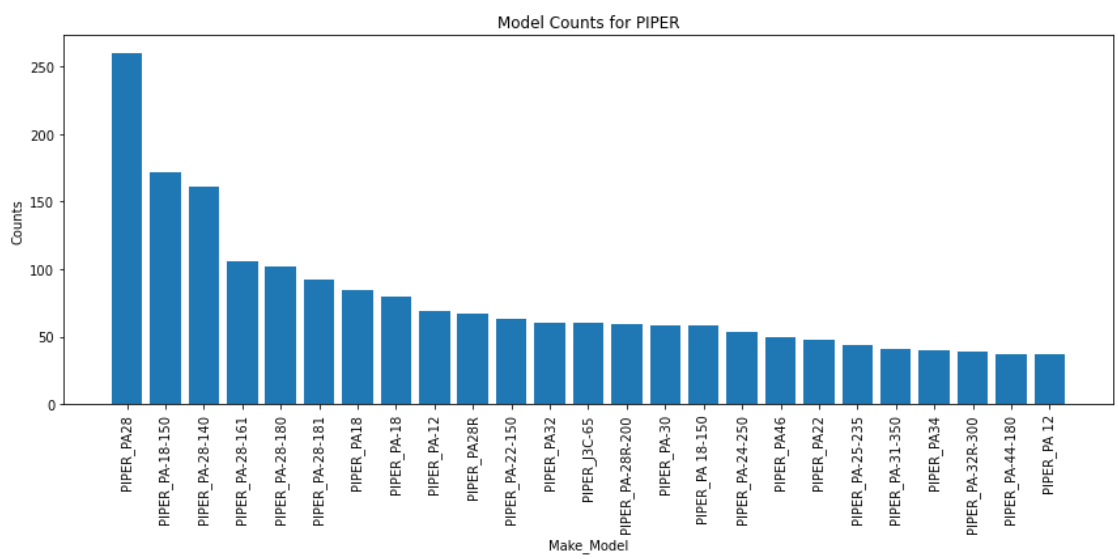
In [70]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'CESSNA'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Combine 'Make' and 'Model' columns to create a new column represent
chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']

# Count the occurrences of each 'Make_Model' combination
make_model_counts = chart_df['Make_Model'].value_counts()

# Sort the values in descending order and select all entries
all_make_model_counts = make_model_counts.sort_values(ascending=False

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(all_make_model_counts.index, all_make_model_counts.values)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Counts')
ax.set_title(f'Model Counts for {specific_make}')

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

```



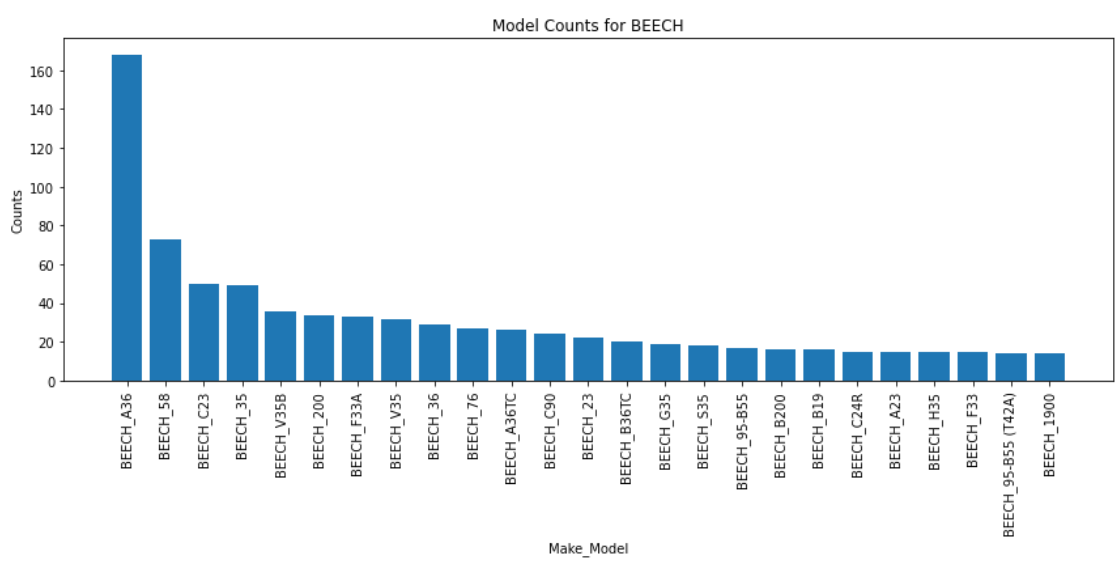
In [71]:

```
specific_make = 'PIPER'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Combine 'Make' and 'Model' columns to create a new column represent
chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']

# Count the occurrences of each 'Make_Model' combination
make_model_counts = chart_df['Make_Model'].value_counts()

# Sort the values in descending order and select all entries
all_make_model_counts = make_model_counts.sort_values(ascending=False

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(all_make_model_counts.index, all_make_model_counts.values)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Counts')
ax.set_title(f'Model Counts for {specific_make}')

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



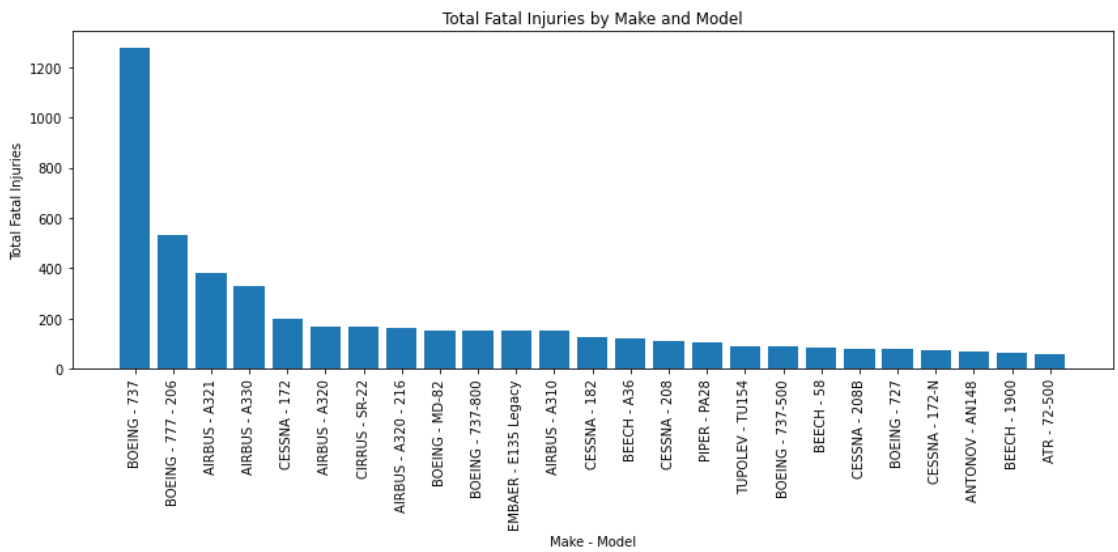
In [72]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'BEECH'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Combine 'Make' and 'Model' columns to create a new column represent
chart_df['Make_Model'] = chart_df['Make'] + '_' + chart_df['Model']

# Count the occurrences of each 'Make_Model' combination
make_model_counts = chart_df['Make_Model'].value_counts()

# Sort the values in descending order and select all entries
all_make_model_counts = make_model_counts.sort_values(ascending=False

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(all_make_model_counts.index, all_make_model_counts.values)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Counts')
ax.set_title(f'Model Counts for {specific_make}')

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



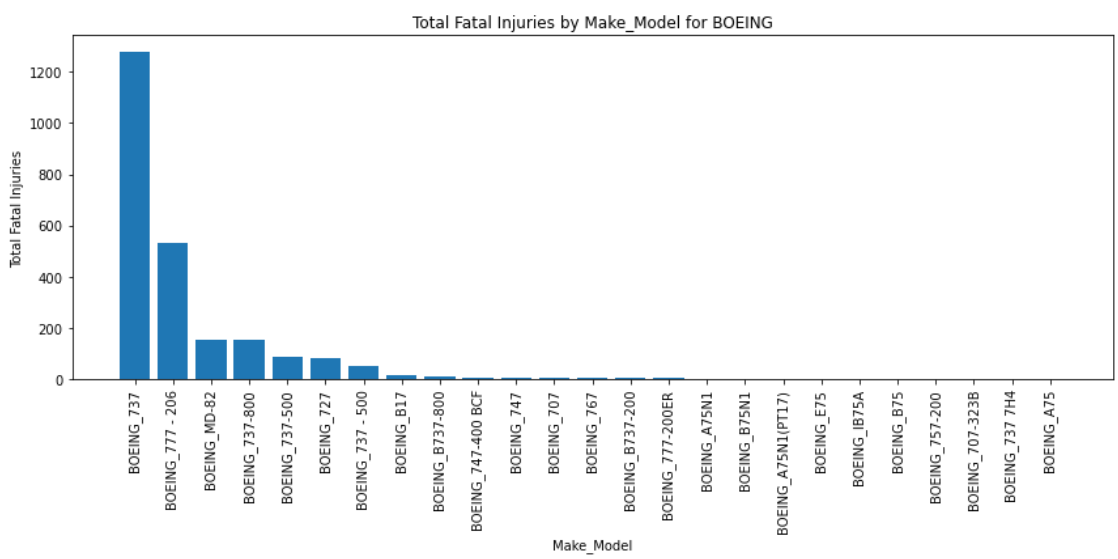
In [73]:

```
# Group the DataFrame by 'Make' and 'Model' and calculate the sum of
make_model_fatal_injuries = df_1985.groupby(['Make', 'Model'])['Total
make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val

# Create a new column 'Make_Model' by combining 'Make' and 'Model'
make_model_fatal_injuries_sorted.index = make_model_fatal_injuries_so
fig, ax = plt.subplots(figsize=(12, 6))
ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur
plt.xticks(rotation=90)

ax.set_xlabel('Make - Model')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title('Total Fatal Injuries by Make and Model')

plt.tight_layout()
plt.show()


```



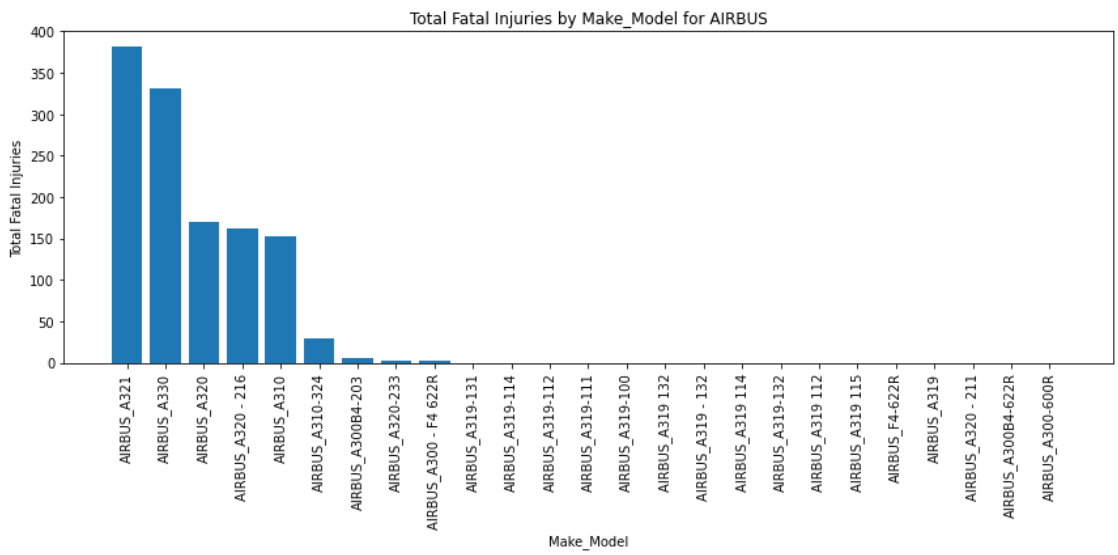
In [74]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'BOEING'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat

# Sort the values in descending order
make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}

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

```



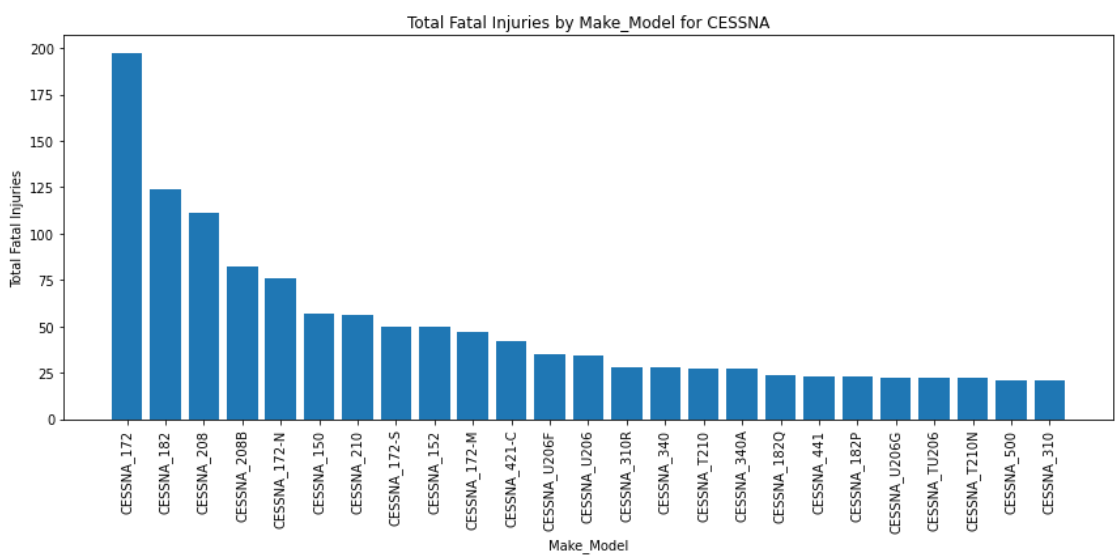
In [75]:

```
specific_make = 'AIRBUS'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat

# Sort the values in descending order
make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}

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

```



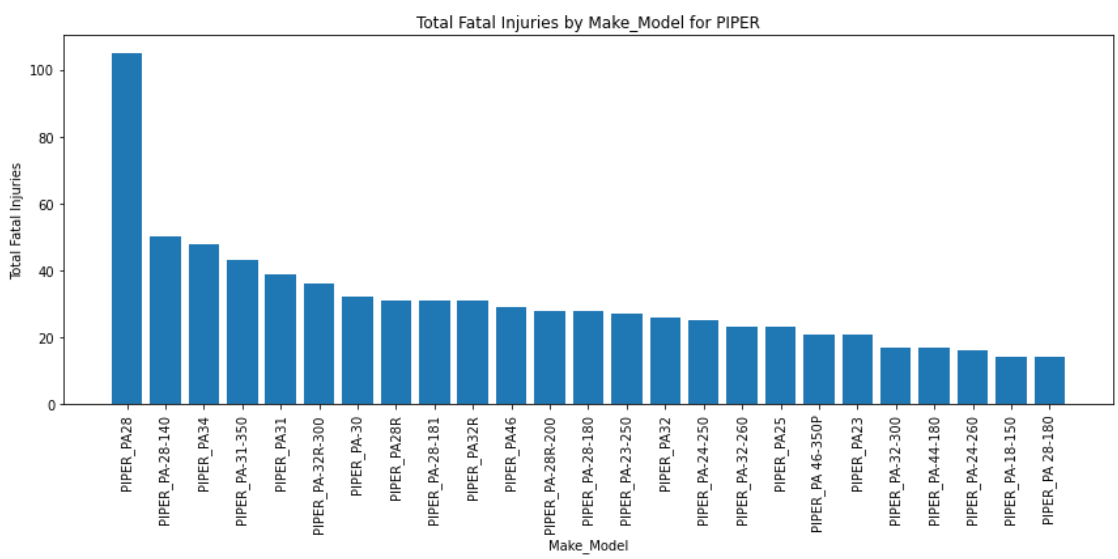
In [76]:

```
specific_make = 'CESSNA'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat

# Sort the values in descending order
make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}

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

```



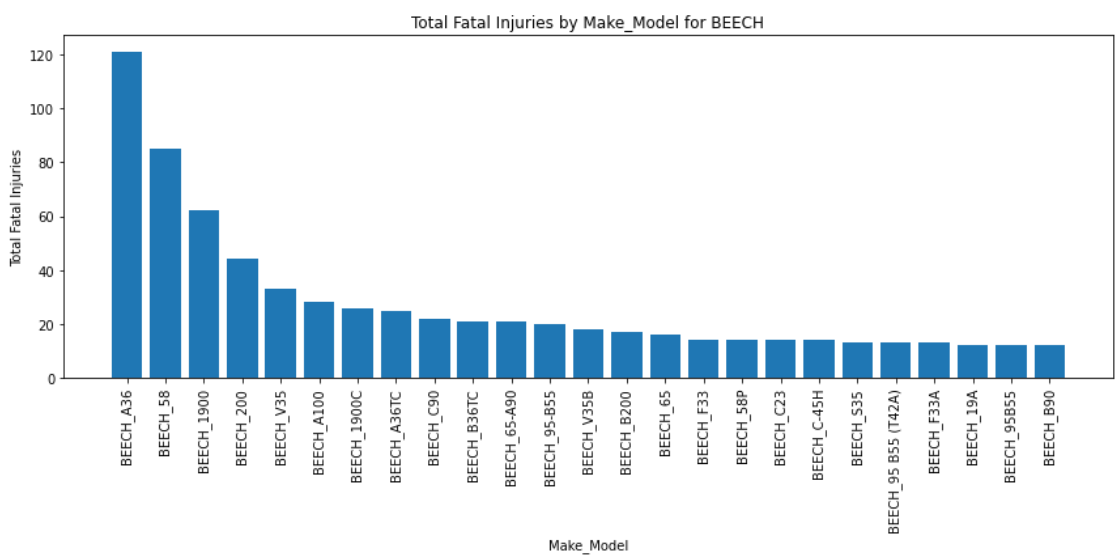
In [77]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'PIPER'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat

# Sort the values in descending order
make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}

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

```



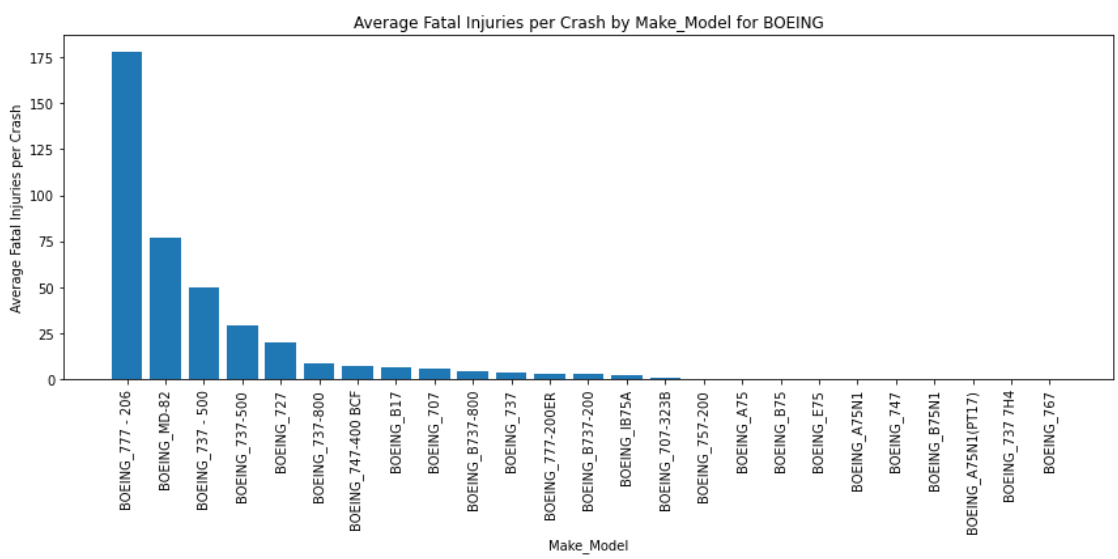
In [78]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'BEECH'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = chart_df.groupby('Make_Model')['Total.Fat

# Sort the values in descending order
make_model_fatal_injuries_sorted = make_model_fatal_injuries.sort_val

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_fatal_injuries_sorted.index, make_model_fatal_injur

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title(f'Total Fatal Injuries by Make_Model for {specific_make}

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



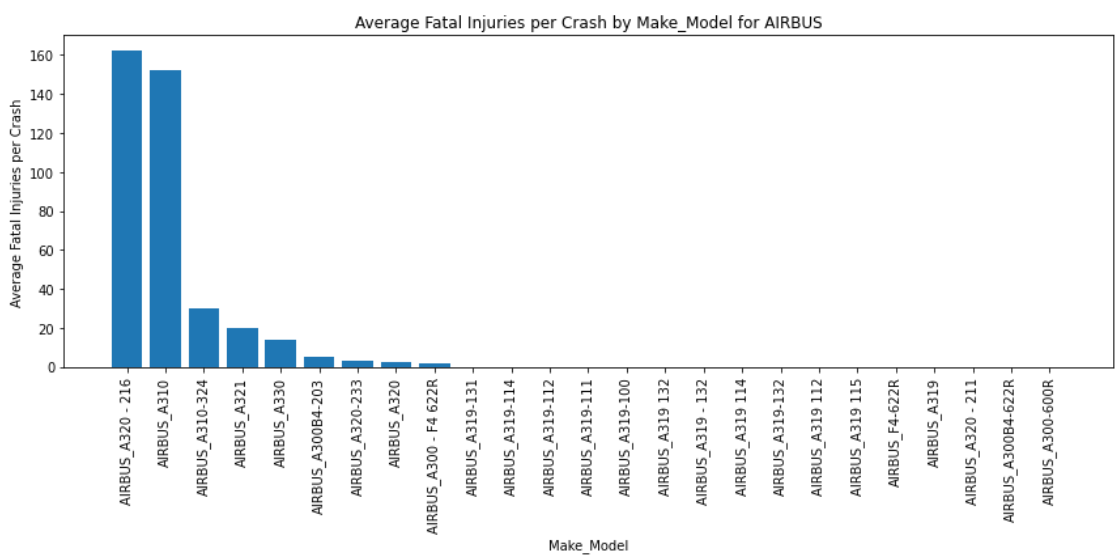
In [79]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'BOEING'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the average of 'T
make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total

# Sort the values in descending order
make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Average Fatal Injuries per Crash')
ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp

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

```



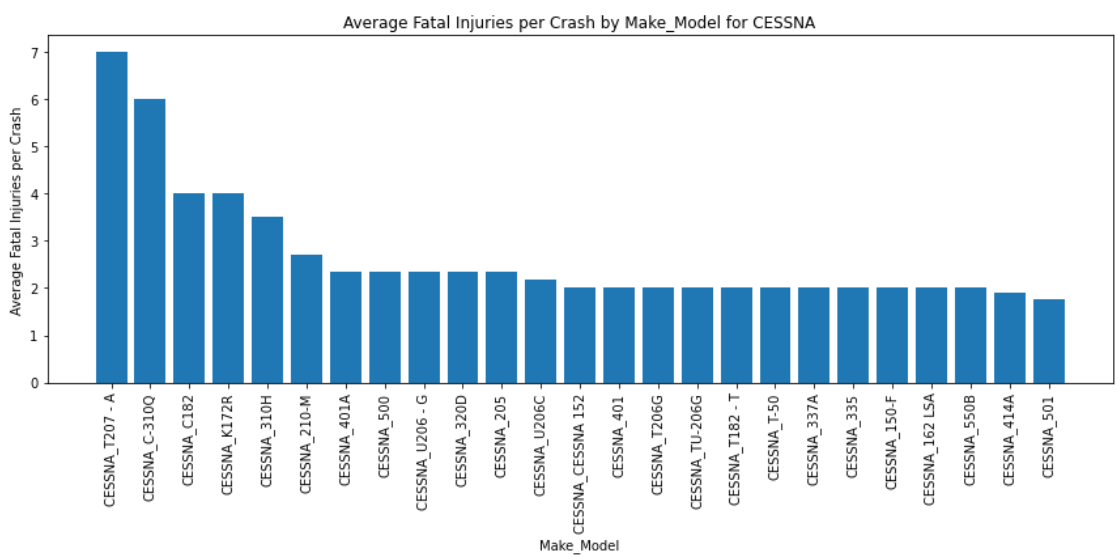
In [80]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'AIRBUS'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the average of 'T
make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total

# Sort the values in descending order
make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Average Fatal Injuries per Crash')
ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp

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

```



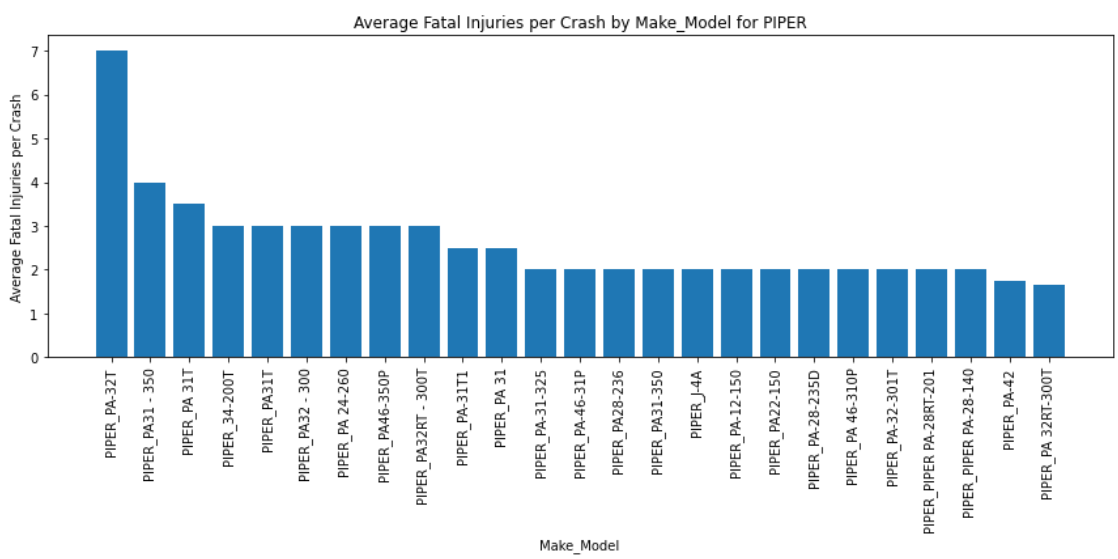
In [81]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'CESSNA'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the average of 'T
make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total

# Sort the values in descending order
make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Average Fatal Injuries per Crash')
ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp

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

```



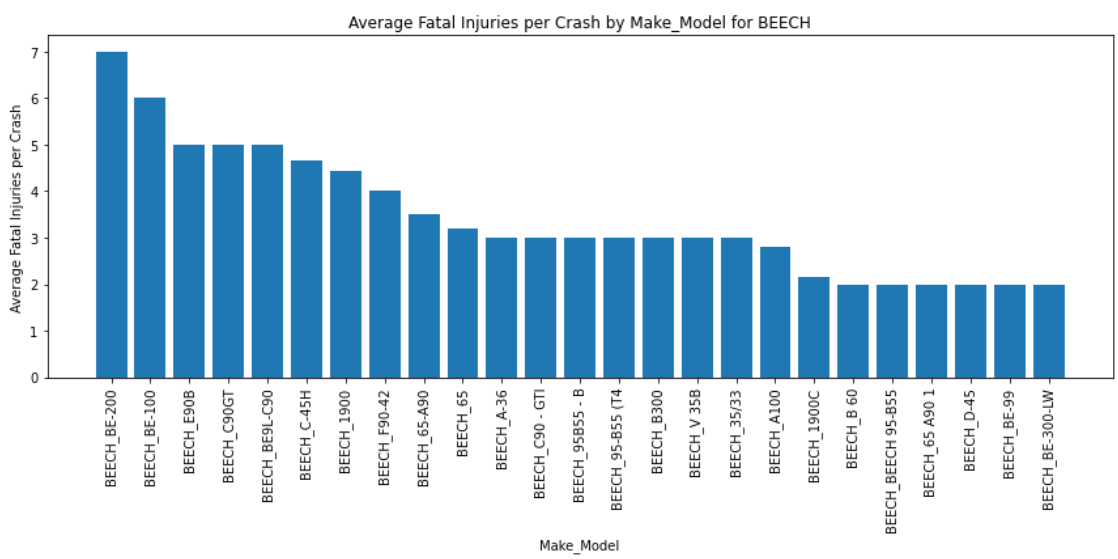
In [82]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'PIPER'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the average of 'T
make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total

# Sort the values in descending order
make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Average Fatal Injuries per Crash')
ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp

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

```



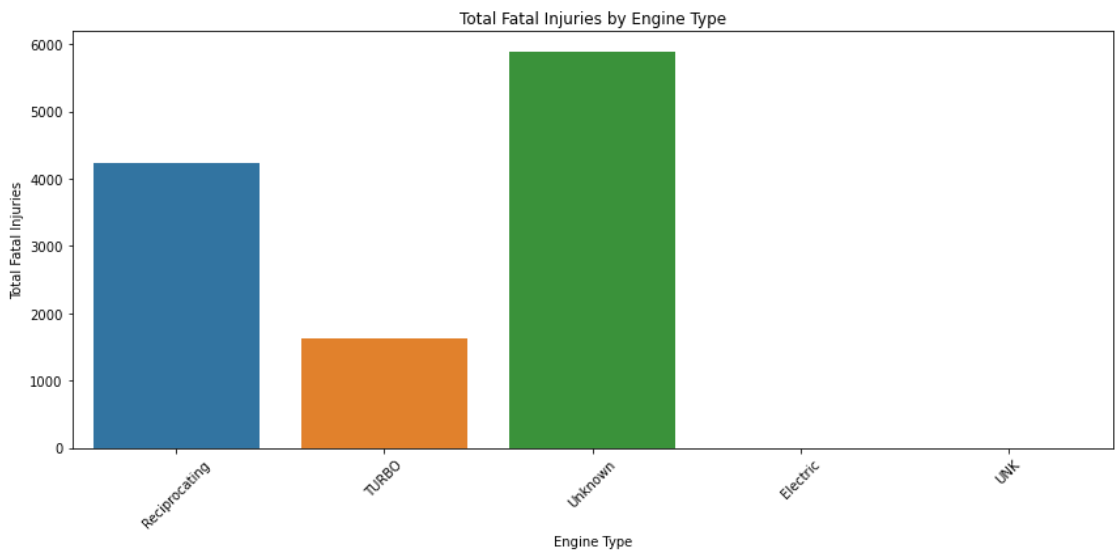
In [83]:

```
# Filter the DataFrame to include only rows with the desired 'Make'
specific_make = 'BEECH'
chart_df = df_1985[df_1985['Make'] == specific_make].copy()  # Make a

# Group the DataFrame by 'Make_Model' and calculate the average of 'T
make_model_avg_fatal_injuries = chart_df.groupby('Make_Model')['Total

# Sort the values in descending order
make_model_avg_fatal_injuries_sorted = make_model_avg_fatal_injuries.

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Create the bar plot
ax.bar(make_model_avg_fatal_injuries_sorted.index, make_model_avg_fat

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Make_Model')
ax.set_ylabel('Average Fatal Injuries per Crash')
ax.set_title(f'Average Fatal Injuries per Crash by Make_Model for {sp

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

```



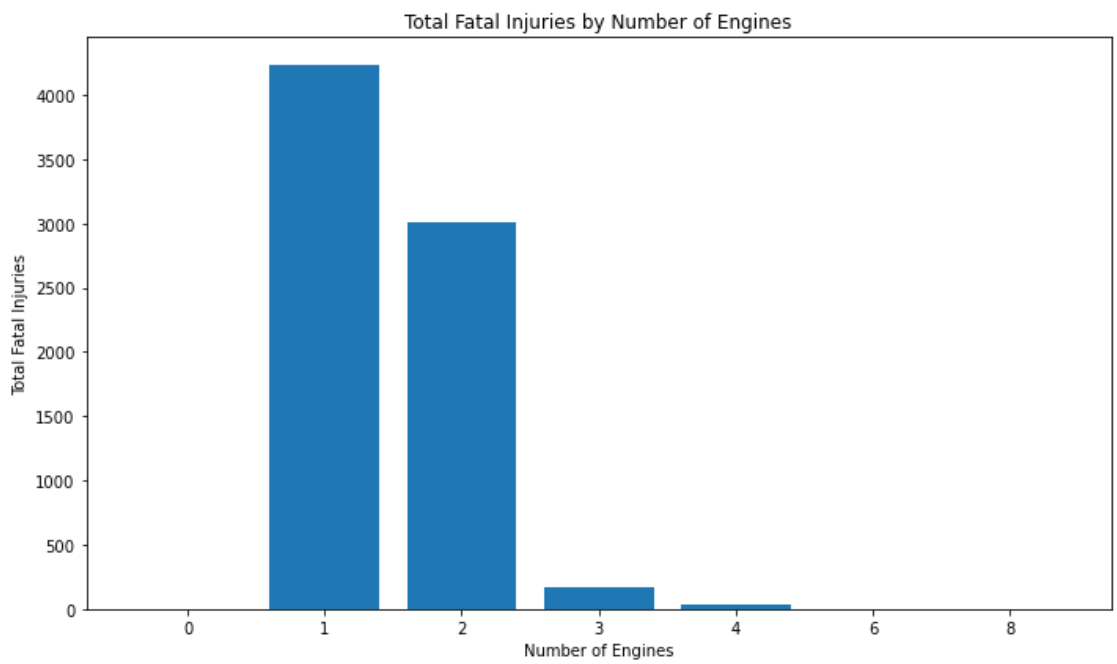
In [84]:

```
# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Use seaborn's bar plot to compare 'Engine.Type' and 'Total.Fatal.In
sns.barplot(x='Engine.Type', y='Total.Fatal.Injuries', data=df_1985,

# Set labels and title
ax.set_xlabel('Engine Type')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title('Total Fatal Injuries by Engine Type')

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=45)

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

```



In [85]:

```
# Group the DataFrame by 'Number.of.Engines' and calculate the sum of
engine_injuries_sum = df_1985.groupby('Number.of.Engines')['Total.Fat

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Create the bar plot
ax.bar(engine_injuries_sum.index, engine_injuries_sum.values)

# Set labels and title
ax.set_xlabel('Number of Engines')
ax.set_ylabel('Total Fatal Injuries')
ax.set_title('Total Fatal Injuries by Number of Engines')

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

```



In [86]:

```
# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = df_1985.groupby('Make_Model')['Total.Fata

# Get the value counts of 'Make_Model' to represent occurrences
make_model_value_counts = df_1985['Make_Model'].value_counts()

# Combine the two Series into a DataFrame
result_df = pd.DataFrame({'Occurrences': make_model_value_counts, 'To

# Calculate the average of 'Total.Fatal.Injuries' for each 'Make_Mode
result_df['Average.Fatal.Injuries'] = result_df['Total.Fatal.Injuries

# Sort the DataFrame by average of 'Total.Fatal.Injuries' in ascendin
result_df = result_df.sort_values(by='Average.Fatal.Injuries', ascend

# Display the resulting DataFrame
result_df[:50]

```

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]:

```
# Group the DataFrame by 'Make_Model' and calculate the sum of 'Total
make_model_fatal_injuries = df_1985.groupby('Make_Model')['Total.Fata

# Get the value counts of 'Make_Model' to represent occurrences
make_model_value_counts = df_1985['Make_Model'].value_counts()

# Combine the two Series into a DataFrame
result_df = pd.DataFrame({'Crashes': make_model_value_counts, 'Total.

# Calculate the average of 'Total.Fatal.Injuries' for each 'Make_Mode
result_df['Average.Fatal.Injuries'] = result_df['Total.Fatal.Injuries
result_df = result_df.sort_values(by=['Crashes', 'Average.Fatal.Injur

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]:

```
# Crash/Fatal Injury data on top
top_25_occurrences = result_df.nlargest(25, 'Crashes')
lowest_avg_fatal_injuries = top_25_occurrences.nsmallest(25, 'Average

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]:

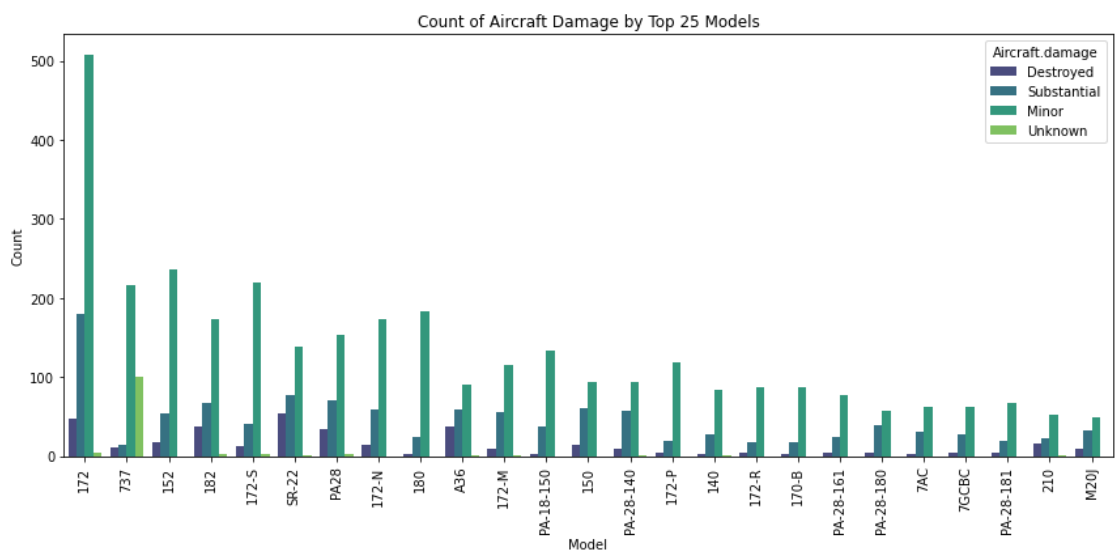
```
# Group the DataFrame by 'Model' and calculate the sum of 'Total.Unin
model_uninjured = df_1985.groupby('Make_Model')['Total.Uninjured'].su
most_uninjured_models = model_uninjured.sort_values(ascending=False)

most_uninjured_models[:25]

```

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]:

```
# Convert 'Model' column to string to avoid potential issues with plo
df_1985['Model'] = df_1985['Model'].astype(str)

# Fill missing values in 'Aircraft.damage' column with 'Unknown'
df_1985['Aircraft.damage'].fillna('Unknown', inplace=True)

# Get the top 25 unique 'Model' values based on their counts
top_25_models = df_1985['Model'].value_counts().nlargest(25).index

# Filter the DataFrame to include only the top 25 'Model' values
df_top_25_models = df_1985[df_1985['Model'].isin(top_25_models)]

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 6))

# Use seaborn's countplot to create the bar chart
sns.countplot(x='Model', hue='Aircraft.damage', data=df_top_25_models
              order=top_25_models)  # Specify the order of x-axis lab

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Set labels and title
ax.set_xlabel('Model')
ax.set_ylabel('Count')
ax.set_title('Count of Aircraft Damage by Top 25 Models')

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



Conclusions

Recommendations

- CESSNA 172 and BOEING 737 are the most viable planes for a low risk business venture.
- 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.**
- 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.
- 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 maintenence for these 2 planes

In [91]:

```
#df_1985.shape
```

In [92]:

```
df_1985.to_csv('edited_aviation_dataset.csv', index=False)
```