Importing the necessary libraries

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv('AviationData.csv',encoding='latin1',low_memory=False)
```

In [4]:
```
# Checking the Head
df.head()
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purp
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	NaN	...	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	NaN	...	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	NaN	...	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	NaN	...	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	NaN	...	

5 rows × 31 columns

In [5]:
```
# Checking the Tail
df.tail()
```

Out[5]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpos
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	NaN	...	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	NaN	...	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	PAYSON	...	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	NaN	...	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	NaN	...	

5 rows × 31 columns

Viewing the DataSet and Cleaning

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

Out[6]:
```
(88889, 31)
```

In [7]:

```
#info about the dataset
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  float64
 18  Engine.Type             81812 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82508 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [8]:

```
#Finding Missing Values
df.isnull().sum()
```

Out[8]:

```
Event.Id                      0
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 [9]:
```
#Finding Missing Values in percentage

df.isnull().sum()/df.shape[0]*100
```

Out[9]:
```
Event.Id                   0.000000
Investigation.Type         0.000000
Accident.Number            0.000000
Event.Date                 0.000000
Location                   0.058500
Country                    0.254250
Latitude                  61.320298
Longitude                 61.330423
Airport.Code              43.469946
Airport.Name              40.611324
Injury.Severity            1.124999
Aircraft.damage            3.593246
Aircraft.Category         63.677170
Registration.Number        1.481623
Make                       0.070875
Model                      0.103500
Amateur.Built              0.114750
Number.of.Engines          6.844491
Engine.Type                7.961615
FAR.Description           63.974170
Schedule                  85.845268
Purpose.of.flight          6.965991
Air.carrier               81.271023
Total.Fatal.Injuries      12.826109
Total.Serious.Injuries    14.073732
Total.Minor.Injuries      13.424608
Total.Uninjured            6.650992
Weather.Condition          5.053494
Broad.phase.of.flight     30.560587
Report.Status              7.178616
Publication.Date          15.492356
dtype: float64
```

In [10]:
```
#Dropping data before Year 1982
df= df[df['Event.Date'] >= '1982-01-01']
```

In [11]:
```
#Dropping rows with more than 10 missing values
df = df.dropna(thresh=10)
```

In [12]:
```
df.shape
```

Out[12]:
```
(88882, 31)
```

In [13]:
```
df.head()
```

Out[13]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	P
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	NaN	NaN	NaN	BLACKBURN AG STRIP	...	
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States	NaN	NaN	N58	HANOVER	...	
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	United States	NaN	NaN	JAX	JACKSONVILLE INTL	...	
10	20020909X01559	Accident	FTW82DA034	1982-01-01	HOBBS, NM	United States	NaN	NaN	NaN	NaN	...	
11	20020909X01558	Accident	ATL82DKJ10	1982-01-01	TUSKEGEE, AL	United States	NaN	NaN	NaN	TUSKEGEE	...	

5 rows × 31 columns

In [14]:
```
df.shape
```

Out[14]:
```
(88882, 31)
```

In [15]:
```
#Dropping columns with over 50% missing values
df = df.dropna(axis=1, thresh=0.5 * df.shape[0])
print(df.columns)
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Airport.Code', 'Airport.Name',
       'Injury.Severity', 'Aircraft.damage', 'Registration.Number', 'Make',
       'Model', 'Amateur.Built', 'Number.of.Engines', 'Engine.Type',
       'Purpose.of.flight', 'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight', 'Report.Status', 'Publication.Date'],
      dtype='object')
```

In [16]:
```
#checking the amount of accidents per country
country_counts = df.groupby('Country').size()
print(country_counts)
```

```
Country
ATLANTIC OCEAN    81
AY                 1
Afghanistan       14
Albania            1
Algeria            2
                  ..
West Indies       11
Wolseley           1
Yemen              1
Zambia             2
Zimbabwe           6
Length: 219, dtype: int64
```

In [17]:
```
#checking the top 10 countries with the most accidents
top_10_countries = df['Country'].value_counts().head(10)
print(top_10_countries)
```

```
United States     82241
Brazil              374
Canada              359
Mexico              358
United Kingdom      344
Australia           300
France              236
Spain               226
Bahamas             216
Germany             215
Name: Country, dtype: int64
```

In [18]:
```
#filtering my dataframe so that i can only remain with USA
df = df[(df['Investigation.Type'] == 'Accident') & (df['Country'] == 'United States')]
```

In [19]:
```
df.shape
```

Out[19]:
```
(79899, 25)
```

In [20]:
```
#checking the head of the filtered  USA data
df.head()
```

Out[20]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airport.Code	Airport.Name	Injury.Severity	Aircraft.dar
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	United States	NaN	BLACKBURN AG STRIP	Non-Fatal	Subst
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	United States	N58	HANOVER	Non-Fatal	Subst
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	United States	JAX	JACKSONVILLE INTL	Non-Fatal	Subst
10	20020909X01559	Accident	FTW82DA034	1982-01-01	HOBBS, NM	United States	NaN	NaN	Non-Fatal	Subst
11	20020909X01558	Accident	ATL82DKJ10	1982-01-01	TUSKEGEE, AL	United States	NaN	TUSKEGEE	Non-Fatal	Subst

5 rows × 25 columns

In [21]:
```
df.tail()
```

Out[21]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Airport.Code	Airport.Name	Injury.Severity	Aircraft.damage
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	Minor	NaN
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	NaN
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	PAN	PAYSON	Non-Fatal	Substantial
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	NaN
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	Minor	NaN

5 rows × 25 columns

In [22]:
```
# dropped columns which i felt they were irrelevant to my analysis
df = df.drop(['Event.Id','Investigation.Type','Accident.Number','Airport.Code','Airport.Name','Registration.Number','Public
```

In [23]:
```
print(df.columns)
```

```
Index(['Event.Date', 'Location', 'Country', 'Injury.Severity',
       'Aircraft.damage', 'Make', 'Model', 'Amateur.Built',
       'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight'],
      dtype='object')
```

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 79899 entries, 7 to 88888
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Date              79899 non-null  object 
 1   Location                79888 non-null  object 
 2   Country                 79899 non-null  object 
 3   Injury.Severity         79847 non-null  object 
 4   Aircraft.damage         78775 non-null  object 
 5   Make                    79887 non-null  object 
 6   Model                   79870 non-null  object 
 7   Amateur.Built           79884 non-null  object 
 8   Number.of.Engines       78141 non-null  float64
 9   Engine.Type             77001 non-null  object 
 10  Purpose.of.flight       78019 non-null  object 
 11  Total.Fatal.Injuries    69635 non-null  float64
 12  Total.Serious.Injuries  68916 non-null  float64
 13  Total.Minor.Injuries    69546 non-null  float64
 14  Total.Uninjured         74905 non-null  float64
 15  Weather.Condition       79338 non-null  object 
 16  Broad.phase.of.flight   59290 non-null  object 
dtypes: float64(5), object(12)
memory usage: 11.0+ MB
```

In [25]:
```
#creating new columns named type and number so that i may be able to split the Injury.Severity to numeric and text
df['Type'] = df['Injury.Severity'].str.extract(r'^([a-zA-Z-]+)')
df['Number'] = df['Injury.Severity'].str.extract(r'\((\d+)\)').fillna(0).astype(int)
```

In [26]:
```
# i dropped the original columns Total.Fatal.Injuries and  Injury.Severity then i will rename
#the new columns i created using the same names
df.drop(columns=['Injury.Severity', 'Total.Fatal.Injuries'], inplace=True, errors='ignore')
```

In [27]:
```
#renaming the 2 new columns as Total.Fatal.Injuries and  Injury.Severity
df.rename(columns={'Type': 'Injury.Severity', 'Number': 'Total.Fatal.Injuries'}, inplace=True)
```

In [28]:
```
#checking how many makes of aircrafts are there
aircraft_counts = df['Make'].value_counts()
print(aircraft_counts)
```

```
Cessna                            21339
Piper                             11521
CESSNA                             4227
Beech                              4018
PIPER                              2487
                                  ...  
Z-HI-MAX                              1
Sea & Air Sales                       1
James Browning                        1
Madera                                1
ROTORCRAFT DEVELOPEMENT CORP.         1
Name: Make, Length: 7954, dtype: int64
```

In [29]:
```
#here i made all the string to upper case so that i may get an uniform data
df['Make'] = df['Make'].str.strip().str.upper()
```

In [30]:
```
aircraft_counts = df['Make'].value_counts()
print(aircraft_counts)
```

```
CESSNA                       25566
PIPER                        14008
BEECH                         4892
BELL                          2236
MOONEY                        1272
                             ...  
SHEAHEN DANE E                   1
HENDRYX STEVE/WILEY ROSS         1
SHUEY                            1
CRESAWN PITTS                    1
LOEHLE AIRCRAFT CORP             1
Name: Make, Length: 7368, dtype: int64
```

In [31]:

```
aircraft_counts = df['Make'].value_counts().head(10)
print(aircraft_counts)
```

```
CESSNA      25566
PIPER       14008
BEECH        4892
BELL         2236
MOONEY       1272
GRUMMAN      1131
BELLANCA     1036
BOEING        931
ROBINSON      916
HUGHES        868
Name: Make, dtype: int64
```

In [32]:

```
#I decided to convert Date to a datetime.
df['Event.Date'] = pd.to_datetime(df['Event.Date'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 79899 entries, 7 to 88888
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   Event.Date              79899 non-null  datetime64[ns]
 1   Location                79888 non-null  object        
 2   Country                 79899 non-null  object        
 3   Aircraft.damage         78775 non-null  object        
 4   Make                    79887 non-null  object        
 5   Model                   79870 non-null  object        
 6   Amateur.Built           79884 non-null  object        
 7   Number.of.Engines       78141 non-null  float64       
 8   Engine.Type             77001 non-null  object        
 9   Purpose.of.flight       78019 non-null  object        
 10  Total.Serious.Injuries  68916 non-null  float64       
 11  Total.Minor.Injuries    69546 non-null  float64       
 12  Total.Uninjured         74905 non-null  float64       
 13  Weather.Condition       79338 non-null  object        
 14  Broad.phase.of.flight   59290 non-null  object        
 15  Injury.Severity         79847 non-null  object        
 16  Total.Fatal.Injuries    79899 non-null  int32         
dtypes: datetime64[ns](1), float64(4), int32(1), object(11)
memory usage: 10.7+ MB
```

In [33]:

```
# Adding a Year column,month and day
df['Year'] = df['Event.Date'].dt.year
df['Month.Abbr'] = df['Event.Date'].dt.month_name().str[:3]
df['Day.Name.Abbr'] = df['Event.Date'].dt.day_name().str[:3]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 79899 entries, 7 to 88888
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   Event.Date              79899 non-null  datetime64[ns]
 1   Location                79888 non-null  object        
 2   Country                 79899 non-null  object        
 3   Aircraft.damage         78775 non-null  object        
 4   Make                    79887 non-null  object        
 5   Model                   79870 non-null  object        
 6   Amateur.Built           79884 non-null  object        
 7   Number.of.Engines       78141 non-null  float64       
 8   Engine.Type             77001 non-null  object        
 9   Purpose.of.flight       78019 non-null  object        
 10  Total.Serious.Injuries  68916 non-null  float64       
 11  Total.Minor.Injuries    69546 non-null  float64       
 12  Total.Uninjured         74905 non-null  float64       
 13  Weather.Condition       79338 non-null  object        
 14  Broad.phase.of.flight   59290 non-null  object        
 15  Injury.Severity         79847 non-null  object        
 16  Total.Fatal.Injuries    79899 non-null  int32         
 17  Year                    79899 non-null  int64         
 18  Month.Abbr              79899 non-null  object        
 19  Day.Name.Abbr           79899 non-null  object        
dtypes: datetime64[ns](1), float64(4), int32(1), int64(1), object(13)
memory usage: 12.5+ MB
```

In [34]:

```
#Checking for missing values
df.isnull().sum()
```

Out[34]:

```
Event.Date                 0
Location                  11
Country                    0
Aircraft.damage         1124
Make                      12
Model                     29
Amateur.Built             15
Number.of.Engines       1758
Engine.Type             2898
Purpose.of.flight       1880
Total.Serious.Injuries 10983
Total.Minor.Injuries   10353
Total.Uninjured         4994
Weather.Condition        561
Broad.phase.of.flight  20609
Injury.Severity           52
Total.Fatal.Injuries       0
Year                       0
Month.Abbr                 0
Day.Name.Abbr              0
dtype: int64
```

In [35]:

```
#i want to handle missing values.i used median for this three
columns = ['Total.Minor.Injuries', 'Total.Serious.Injuries', 'Total.Uninjured']
for col in columns:
    df[col].fillna(df[col].median(), inplace=True)
```

In [36]:

```
# here i used the mode for this columns
columns = ['Location', 'Aircraft.damage', 'Make', 'Model', 'Amateur.Built',
           'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
           'Weather.Condition', 'Broad.phase.of.flight', 'Injury.Severity']
for col in columns:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

In [37]:

```
df = df[df['Purpose.of.flight'].isin(['Business', 'Personal'])]
```

In [38]:

```
# Step 1: Dynamically calculate the total number of people involved
df['Total.People'] = (
    df['Total.Fatal.Injuries'] +
    df['Total.Serious.Injuries'] +
    df['Total.Minor.Injuries'] +
    df['Total.Uninjured']
)

# Step 2: Avoid division by zero by replacing zeros with NaN
df['Total.People'] = df['Total.People'].replace(0, pd.NA)

# Step 3: Calculate the fatality rate
df['Fatality.Rate'] = (df['Total.Fatal.Injuries'] / df['Total.People']) * 100

# Step 4: Replace NaN in Fatality Rate with 0 (for cases with no people involved)
df['Fatality.Rate'] = df['Fatality.Rate'].fillna(0)

# View the updated DataFrame
print(df[['Make', 'Fatality.Rate']].head())
```

```
              Make  Fatality.Rate
7           CESSNA            0.0
8           CESSNA            0.0
9   NORTH AMERICAN            0.0
10           PIPER            0.0
11           BEECH            0.0
```

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53909 entries, 7 to 88888
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Date              53909 non-null  datetime64[ns]
 1   Location                53909 non-null  object
 2   Country                 53909 non-null  object
 3   Aircraft.damage         53909 non-null  object
 4   Make                    53909 non-null  object
 5   Model                   53909 non-null  object
 6   Amateur.Built           53909 non-null  object
 7   Number.of.Engines       53909 non-null  float64
 8   Engine.Type             53909 non-null  object
 9   Purpose.of.flight       53909 non-null  object
 10  Total.Serious.Injuries  53909 non-null  float64
 11  Total.Minor.Injuries    53909 non-null  float64
 12  Total.Uninjured         53909 non-null  float64
 13  Weather.Condition       53909 non-null  object
 14  Broad.phase.of.flight   53909 non-null  object
 15  Injury.Severity         53909 non-null  object
 16  Total.Fatal.Injuries    53909 non-null  int32
 17  Year                    53909 non-null  int64
 18  Month.Abbr              53909 non-null  object
 19  Day.Name.Abbr           53909 non-null  object
 20  Total.People            51481 non-null  float64
 21  Fatality.Rate           53909 non-null  float64
dtypes: datetime64[ns](1), float64(6), int32(1), int64(1), object(13)
memory usage: 9.3+ MB
```

In [40]:
```
#confirming whether there are still missing values
df.isnull().sum()
```

Out[40]:
```
Event.Date                 0
Location                   0
Country                    0
Aircraft.damage            0
Make                       0
Model                      0
Amateur.Built              0
Number.of.Engines          0
Engine.Type                0
Purpose.of.flight          0
Total.Serious.Injuries     0
Total.Minor.Injuries       0
Total.Uninjured            0
Weather.Condition          0
Broad.phase.of.flight      0
Injury.Severity            0
Total.Fatal.Injuries       0
Year                       0
Month.Abbr                 0
Day.Name.Abbr              0
Total.People            2428
Fatality.Rate              0
dtype: int64
```

In [ ]:
```

```

In [41]:
```
#creating the new cleaned csv file which i will use in tableau
df.to_csv('cleaned_data.csv', index=False)
```

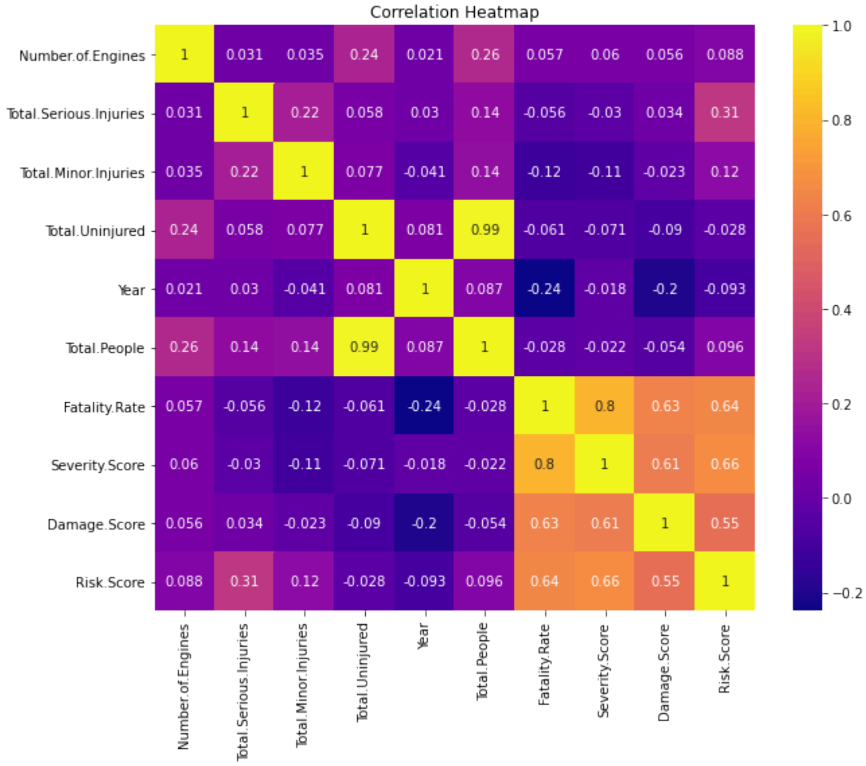
Exploratory Data Analysis(EDA)

In [42]:
```
# Calculating correlation my dataset
df.select_dtypes(include=['float64', 'int64']).corr()
```

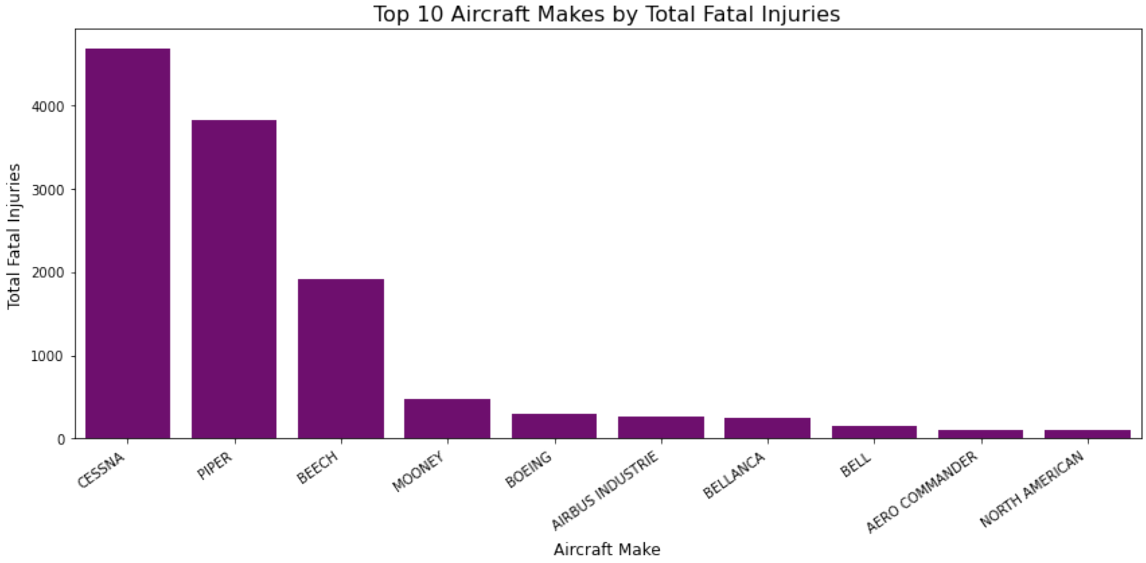
Out[42]:

	Number.of.Engines	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured	Year	Total.People	Fatality.Rate
Number.of.Engines	1.000000	0.030798	0.035286	0.244653	0.020911	0.259758	0.057403
Total.Serious.Injuries	0.030798	1.000000	0.224959	0.058397	0.030319	0.135157	-0.056305
Total.Minor.Injuries	0.035286	0.224959	1.000000	0.077004	-0.041030	0.139460	-0.123754
Total.Uninjured	0.244653	0.058397	0.077004	1.000000	0.080871	0.990131	-0.060712
Year	0.020911	0.030319	-0.041030	0.080871	1.000000	0.086721	-0.238973
Total.People	0.259758	0.135157	0.139460	0.990131	0.086721	1.000000	-0.027770
Fatality.Rate	0.057403	-0.056305	-0.123754	-0.060712	-0.238973	-0.027770	1.000000

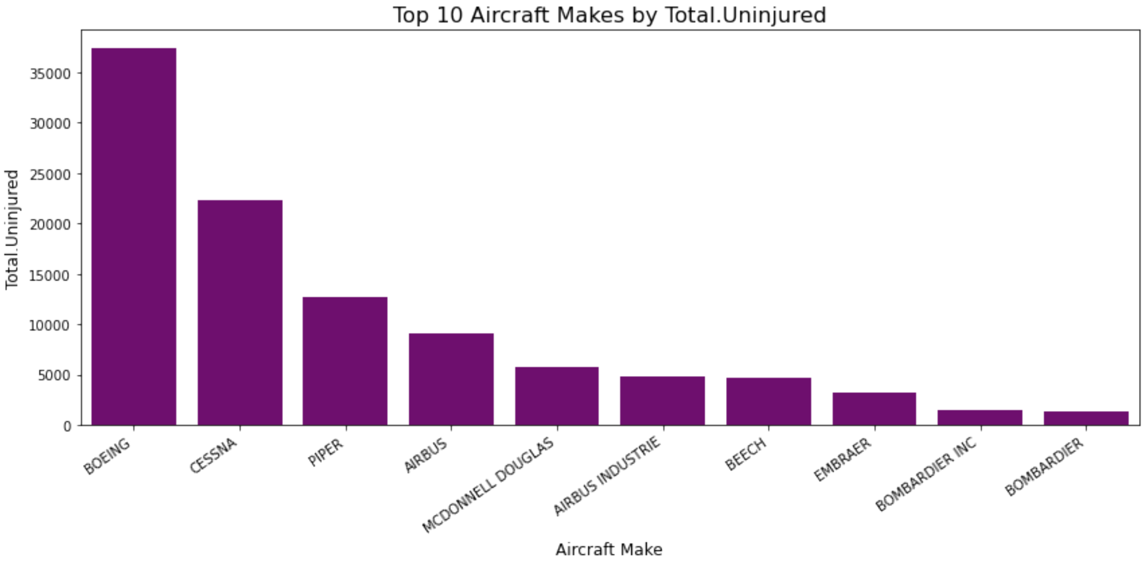
In [86]:
```
#doing virtualization of the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df.select_dtypes(include=['float64', 'int64']).corr(), annot=True, cmap='plasma')
plt.title('Correlation Heatmap')
plt.show()
```



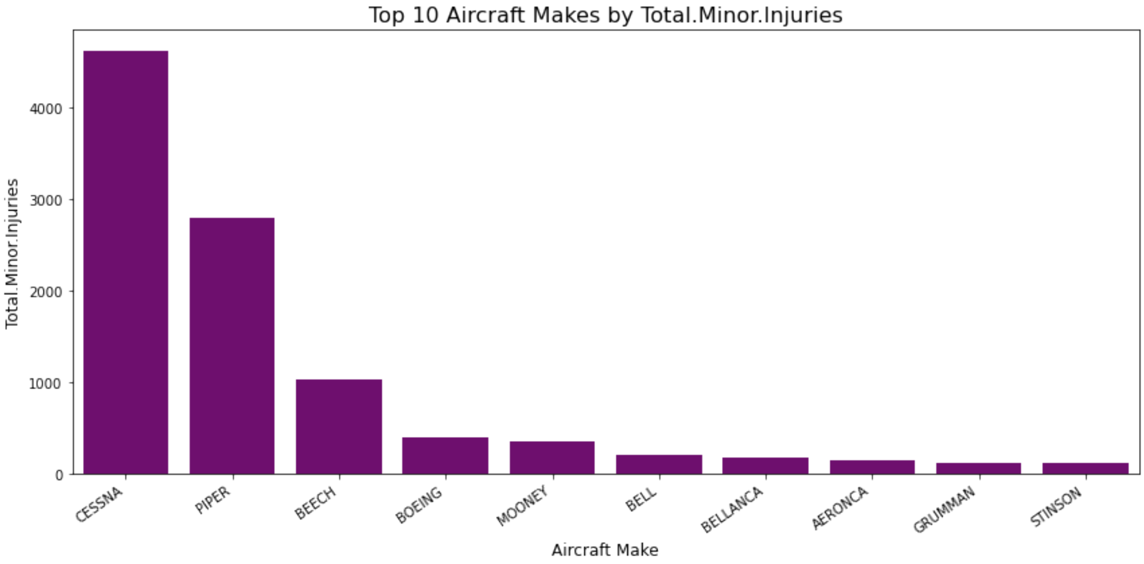
In [44]:
```
# doing a bar plot of Total Fatal injuries vs Make for top ten aircrafts
make_injuries = df.groupby('Make')['Total.Fatal.Injuries'].sum().reset_index()
# Getting  the top 10 Makes by Total.Fatal.Injuries
top_10_makes = make_injuries.nlargest(10, 'Total.Fatal.Injuries')
plt.figure(figsize=(12, 6))
sns.barplot(x='Make', y='Total.Fatal.Injuries', data=top_10_makes, color='purple')
# Customizing the plot
plt.title('Top 10 Aircraft Makes by Total Fatal Injuries', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Total Fatal Injuries', fontsize=12)
plt.xticks(rotation=35, ha='right')

plt.tight_layout()
plt.show()
```



```
In [77]: # doing a bar plot of Total.Uninjured vs Make for top ten aircrafts
make_injuries = df.groupby('Make')['Total.Uninjured'].sum().reset_index()
# Getting the top 10 Makes by Total.Fatal.Injuries
top_10_makes = make_injuries.nlargest(10, 'Total.Uninjured')
plt.figure(figsize=(12, 6))
sns.barplot(x='Make', y='Total.Uninjured', data=top_10_makes, color='purple')
# Customizing the plot
plt.title('Top 10 Aircraft Makes by Total.Uninjured', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Total.Uninjured', fontsize=12)
plt.xticks(rotation=35, ha='right')

plt.tight_layout()
plt.show()
```

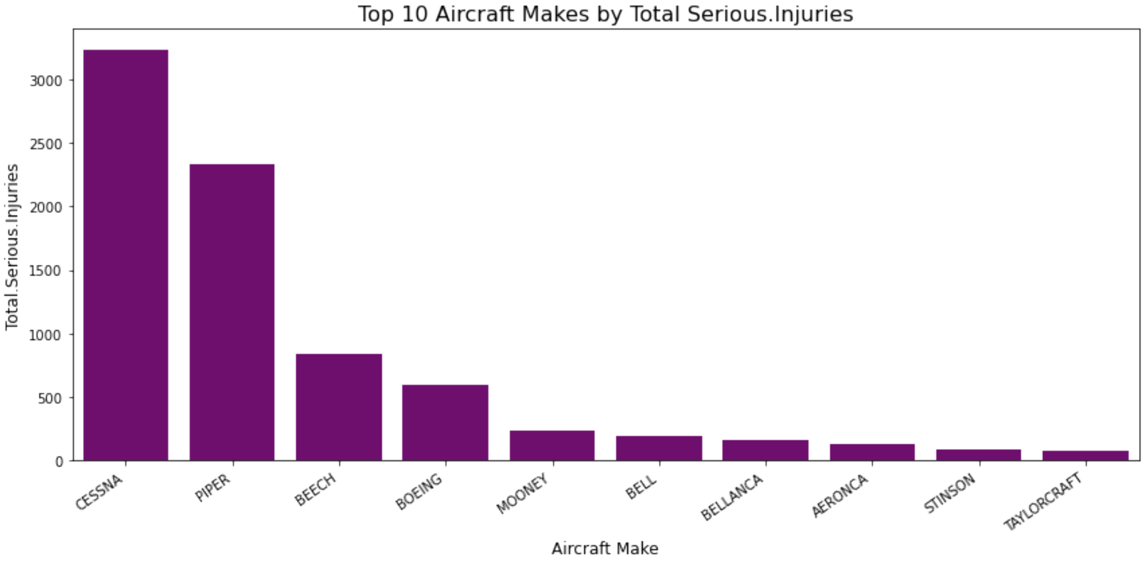


```
In [87]: #doing a bar plot of Total.Minor.Injuries vs Make for top ten aircrafts
make_injuries = df.groupby('Make')['Total.Minor.Injuries'].sum().reset_index()
# Getting the top 10 Makes by Total.Fatal.Injuries
top_10_makes = make_injuries.nlargest(10, 'Total.Minor.Injuries')
plt.figure(figsize=(12, 6))
sns.barplot(x='Make', y='Total.Minor.Injuries', data=top_10_makes, color='purple')
# Customizing the plot
plt.title('Top 10 Aircraft Makes by Total.Minor.Injuries', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Total.Minor.Injuries', fontsize=12)
plt.xticks(rotation=35, ha='right')

plt.tight_layout()
plt.show()
```



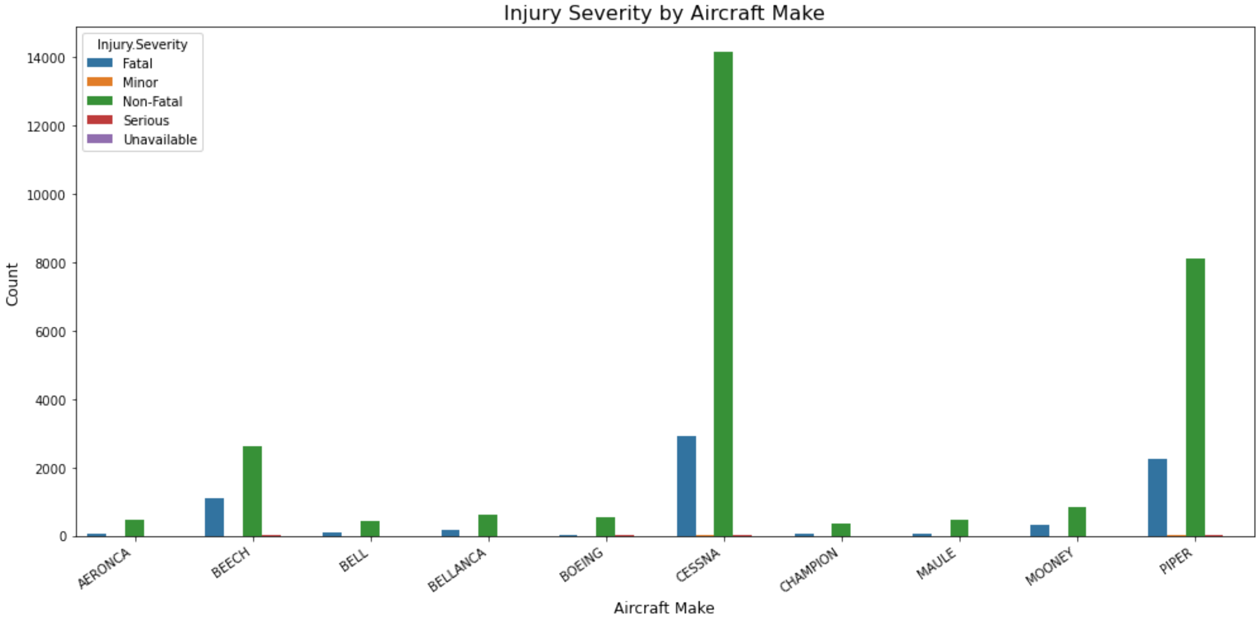
In [88]:

```
#doing a bar plot of Total.Serious.Injuries vs Make for top ten aircrafts
make_injuries = df.groupby('Make')['Total.Serious.Injuries'].sum().reset_index()
# Getting the top 10 Makes by Total.Fatal.Injuries
top_10_makes = make_injuries.nlargest(10, 'Total.Serious.Injuries')
plt.figure(figsize=(12, 6))
sns.barplot(x='Make', y='Total.Serious.Injuries', data=top_10_makes, color='purple')
# Customizing the plot
plt.title('Top 10 Aircraft Makes by Total Serious.Injuries', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Total.Serious.Injuries', fontsize=12)
plt.xticks(rotation=35, ha='right')

plt.tight_layout()
plt.show()
```



In [90]:

```
#doing a bar plot of Injury.Severity vs Make for top ten aircrafts
severity_counts = df.groupby(['Make', 'Injury.Severity']).size().reset_index(name='Count')
top_10_makes = df['Make'].value_counts().head(10).index
severity_counts_filtered = severity_counts[severity_counts['Make'].isin(top_10_makes)]
plt.figure(figsize=(14, 7))
sns.barplot(x='Make', y='Count', hue='Injury.Severity', data=severity_counts_filtered)
# Customizing the plot
plt.title('Injury Severity by Aircraft Make', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=35, ha='right')
plt.tight_layout()
plt.show()
```



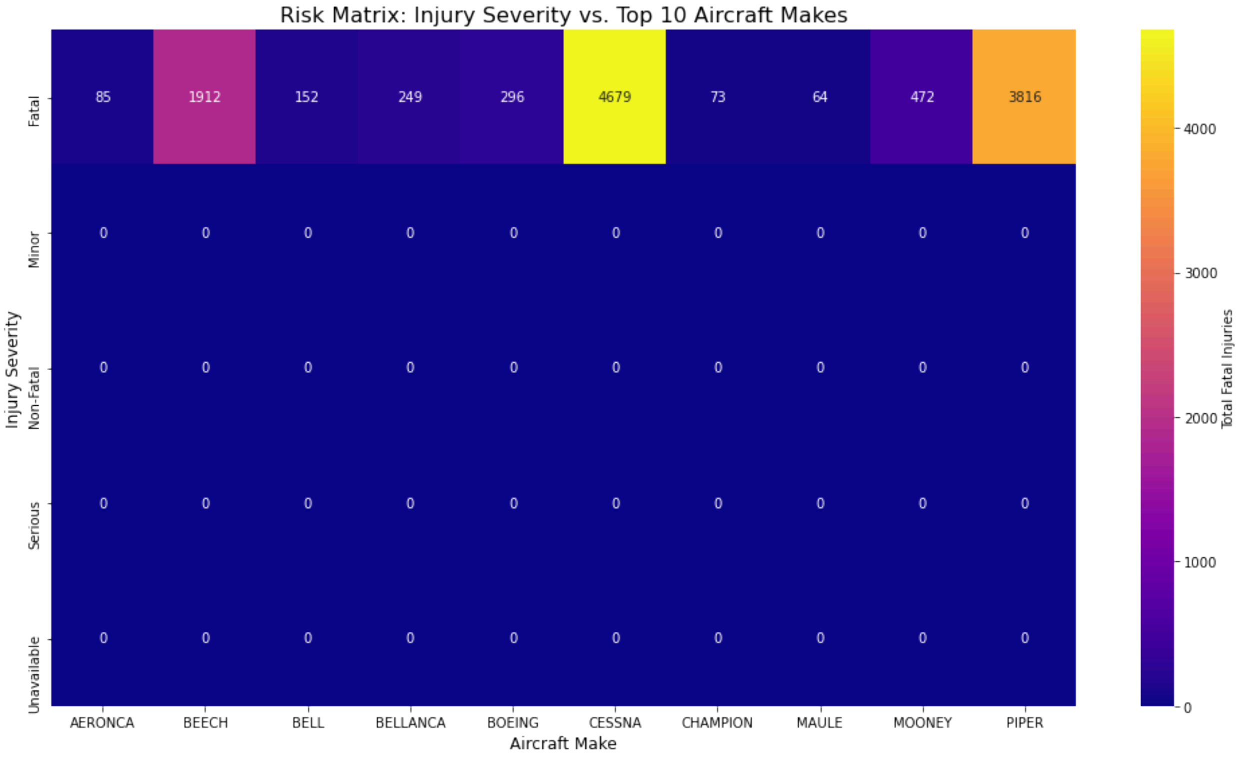
In [49]:

```
#trying to do a risk matrix by top ten aircrafts
top_10_makes = df['Make'].value_counts().head(10).index
top_10_df = df[df['Make'].isin(top_10_makes)]
# Creating a pivot table using Total.Fatal.Injuries
risk_matrix = top_10_df.pivot_table(
    index='Injury.Severity',
    columns='Make',
    values='Total.Fatal.Injuries',
    aggfunc='sum',
    fill_value=0
)
# checking  the risk matrix
print(risk_matrix)
```

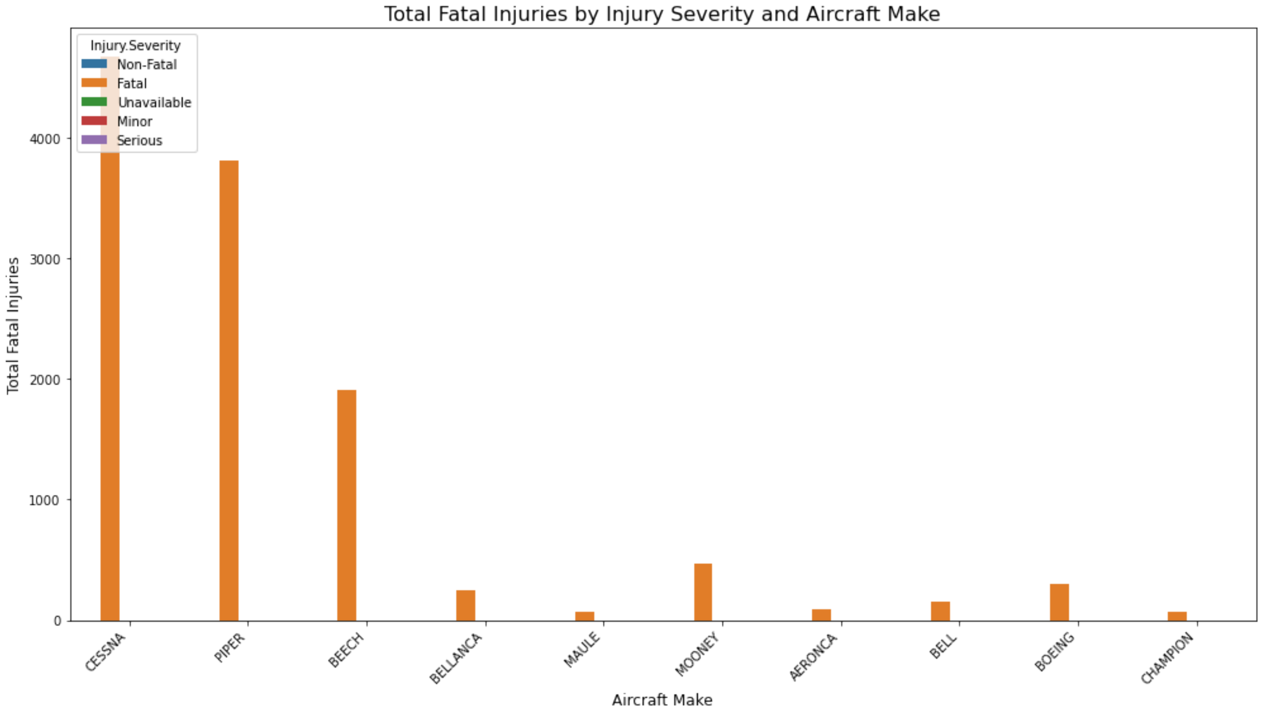
```
Make             AERONCA  BEECH  BELL  BELLANCA  BOEING  CESSNA  CHAMPION  \
Injury.Severity
Fatal                 85   1912   152       249     296    4679        73
Minor                  0      0     0         0       0       0         0
Non-Fatal              0      0     0         0       0       0         0
Serious                0      0     0         0       0       0         0
Unavailable            0      0     0         0       0       0         0

Make             MAULE  MOONEY  PIPER
Injury.Severity
Fatal               64     472   3816
Minor                0       0      0
Non-Fatal            0       0      0
Serious              0       0      0
Unavailable          0       0      0
```

In [91]:

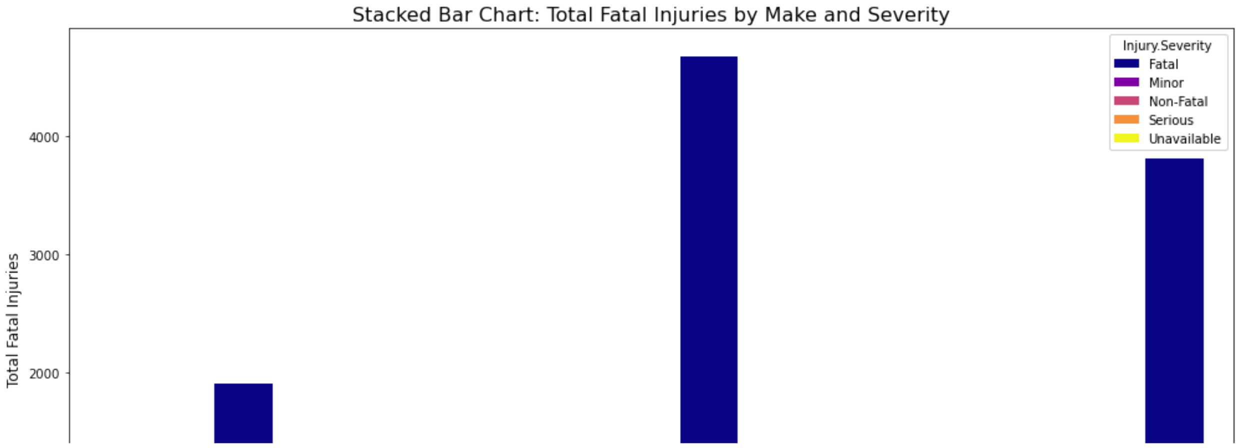
```
#plotting the risk matrix using a Heat map
plt.figure(figsize=(14, 8))
sns.heatmap(risk_matrix, annot=True, fmt="d", cmap="plasma", cbar_kws={'label': 'Total Fatal Injuries'})

# Add title and labels
plt.title('Risk Matrix: Injury Severity vs. Top 10 Aircraft Makes', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Injury Severity', fontsize=12)

# Adjust layout
plt.tight_layout()
plt.show()
```



In [51]:
```
plt.figure(figsize=(14, 8))
sns.barplot(x='Make', y='Total.Fatal.Injuries', hue='Injury.Severity', data=top_10_df, estimator=sum, ci=None)

# Customize the plot
plt.title('Total Fatal Injuries by Injury Severity and Aircraft Make', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Total Fatal Injuries', fontsize=12)
plt.xticks(rotation=45, ha='right')

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

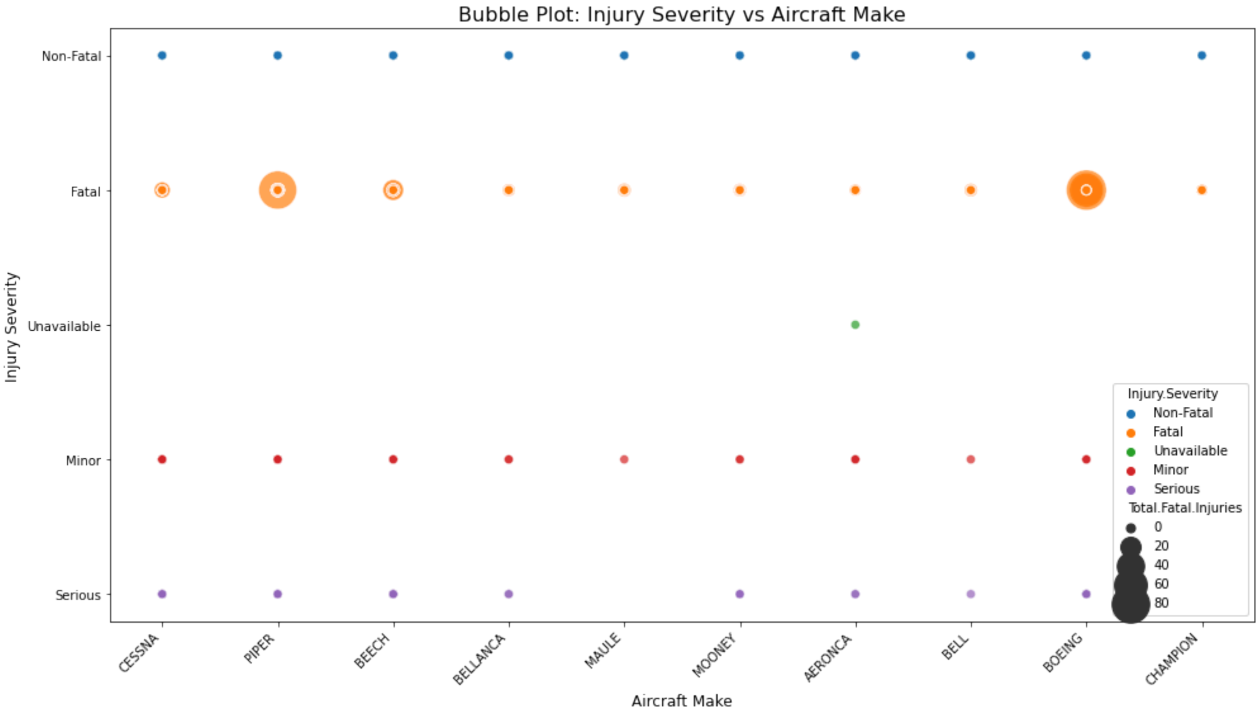


In [92]:
```
stacked_data = top_10_df.pivot_table(
    index='Make',
    columns='Injury.Severity',
    values='Total.Fatal.Injuries',
    aggfunc='sum',
    fill_value=0
)

# Ploting the stacked bar chart
stacked_data.plot(kind='bar', stacked=True, figsize=(14, 8), colormap='plasma')

# Customizing the plot
plt.title('Stacked Bar Chart: Total Fatal Injuries by Make and Severity', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Total Fatal Injuries', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

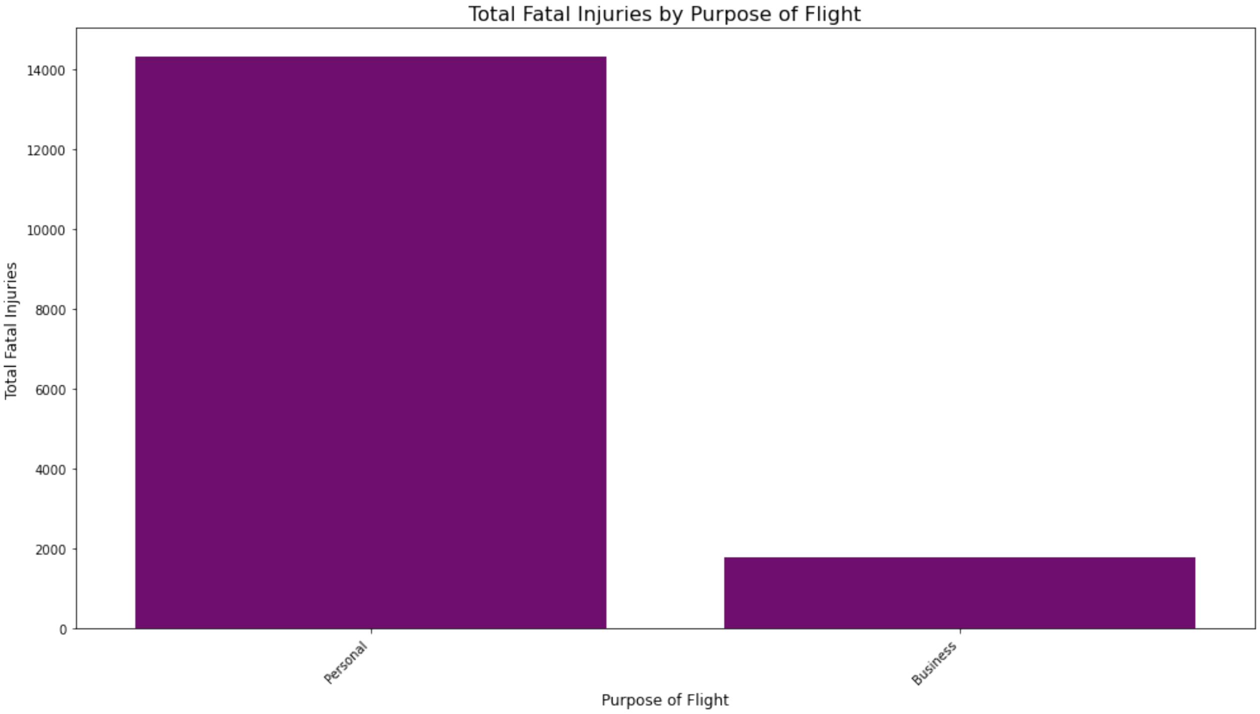


In [53]:

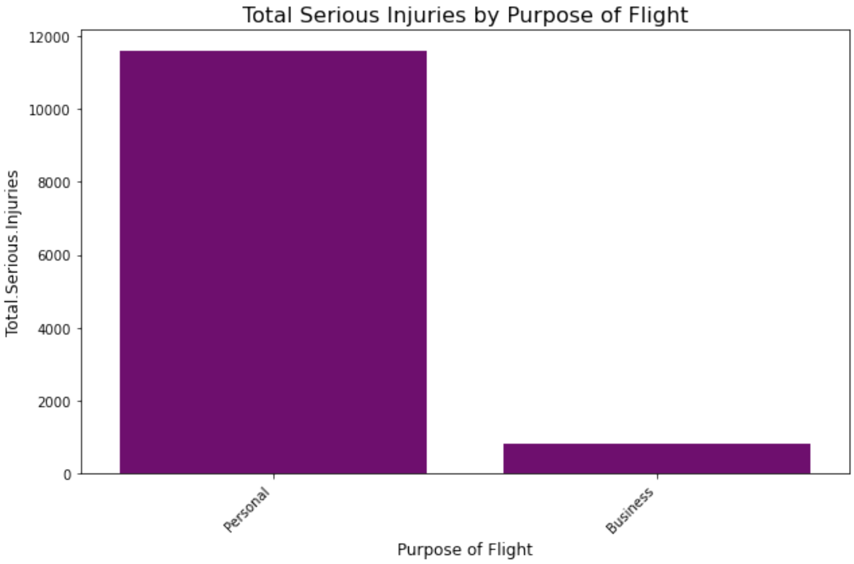
```python
# i tried plotting a Bubble Plot for top 10 aircrafts
plt.figure(figsize=(14, 8))
sns.scatterplot(
    x='Make', y='Injury.Severity', size='Total.Fatal.Injuries', hue='Injury.Severity',
    data=top_10_df, sizes=(50, 1000), alpha=0.7
)

# Customize the plot
plt.title('Bubble Plot: Injury Severity vs Aircraft Make', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Injury Severity', fontsize=12)
plt.xticks(rotation=45, ha='right')

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



In [54]:
```
# Bar plot for total fatal injuries vs purpose of Flight
plt.figure(figsize=(14, 8))
sns.barplot(x='Purpose.of.flight', y='Total.Fatal.Injuries', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title('Total Fatal Injuries by Purpose of Flight', fontsize=16)
plt.xlabel('Purpose of Flight', fontsize=12)
plt.ylabel('Total Fatal Injuries', fontsize=12)
plt.xticks(rotation=45, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```

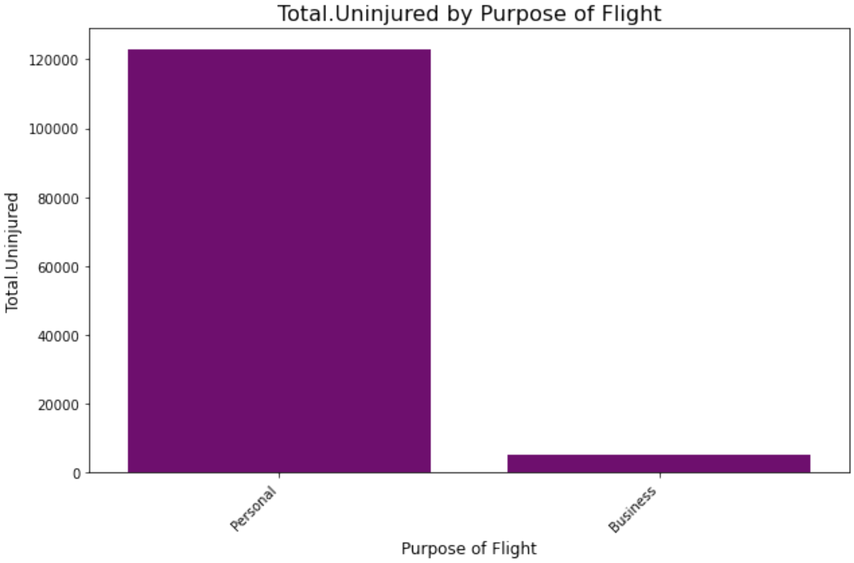


In [55]:
```
# Bar plot for Total.Serious.Injuries vs purpose of Flight
plt.figure(figsize=(9, 6))
sns.barplot(x='Purpose.of.flight', y='Total.Serious.Injuries', data=df, estimator=sum, ci=None, color='Purple')
# Customizing the plot
plt.title('Total Serious Injuries by Purpose of Flight', fontsize=16)
plt.xlabel('Purpose of Flight', fontsize=12)
plt.ylabel('Total.Serious.Injuries', fontsize=12)
plt.xticks(rotation=45, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```



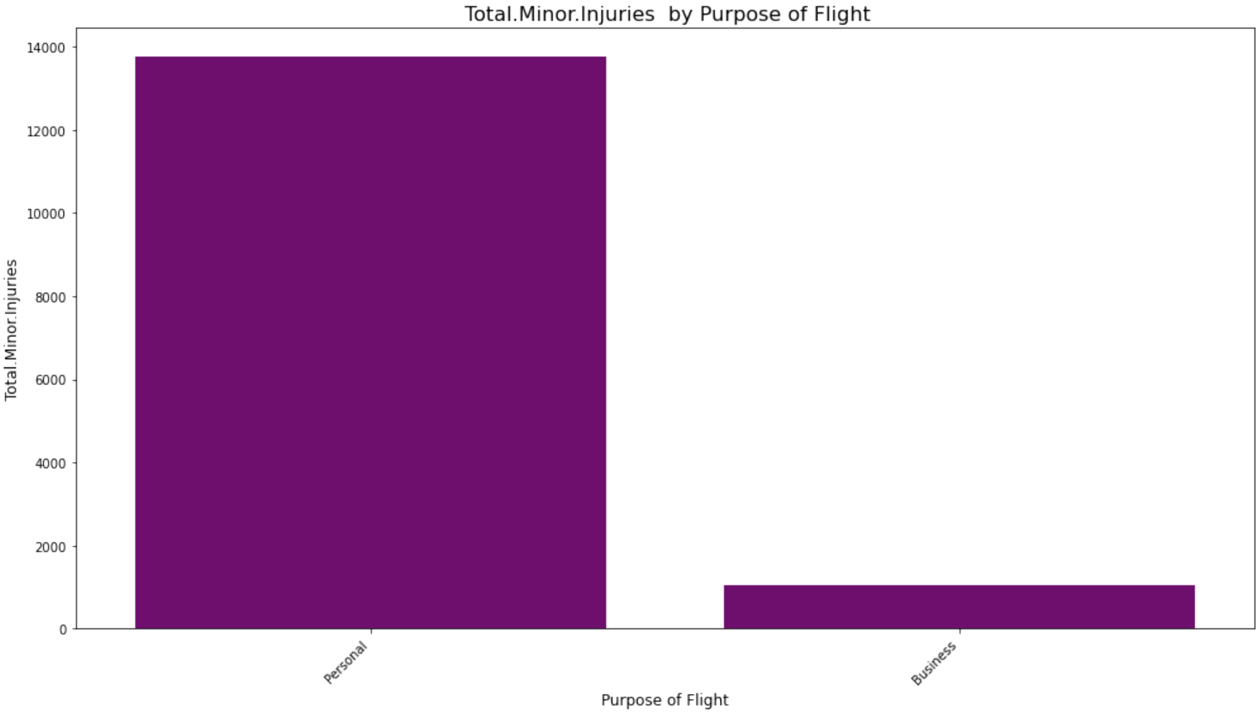
In [56]:

```
# Bar plot for Total.Uninjured vs purpose of Flight
plt.figure(figsize=(9, 6))
sns.barplot(x='Purpose.of.flight', y='Total.Uninjured', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title('Total.Uninjured by Purpose of Flight', fontsize=16)
plt.xlabel('Purpose of Flight', fontsize=12)
plt.ylabel('Total.Uninjured', fontsize=12)
plt.xticks(rotation=45, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```



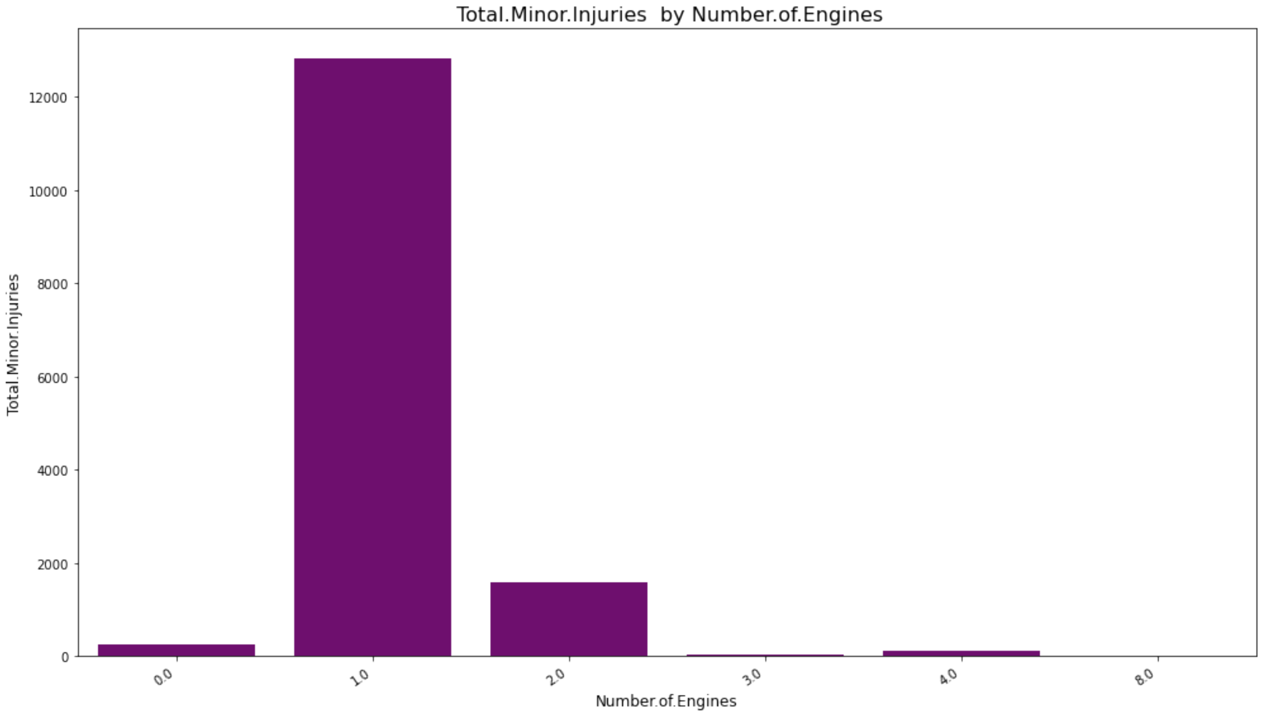
In [93]:

```
# Bar plot for  Total.Minor.Injuries  vs purpose of Flight
plt.figure(figsize=(14, 8))
sns.barplot(x='Purpose.of.flight', y='Total.Minor.Injuries', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title(' Total.Minor.Injuries  by Purpose of Flight', fontsize=16)
plt.xlabel('Purpose of Flight', fontsize=12)
plt.ylabel(' Total.Minor.Injuries ', fontsize=12)
plt.xticks(rotation=45, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```



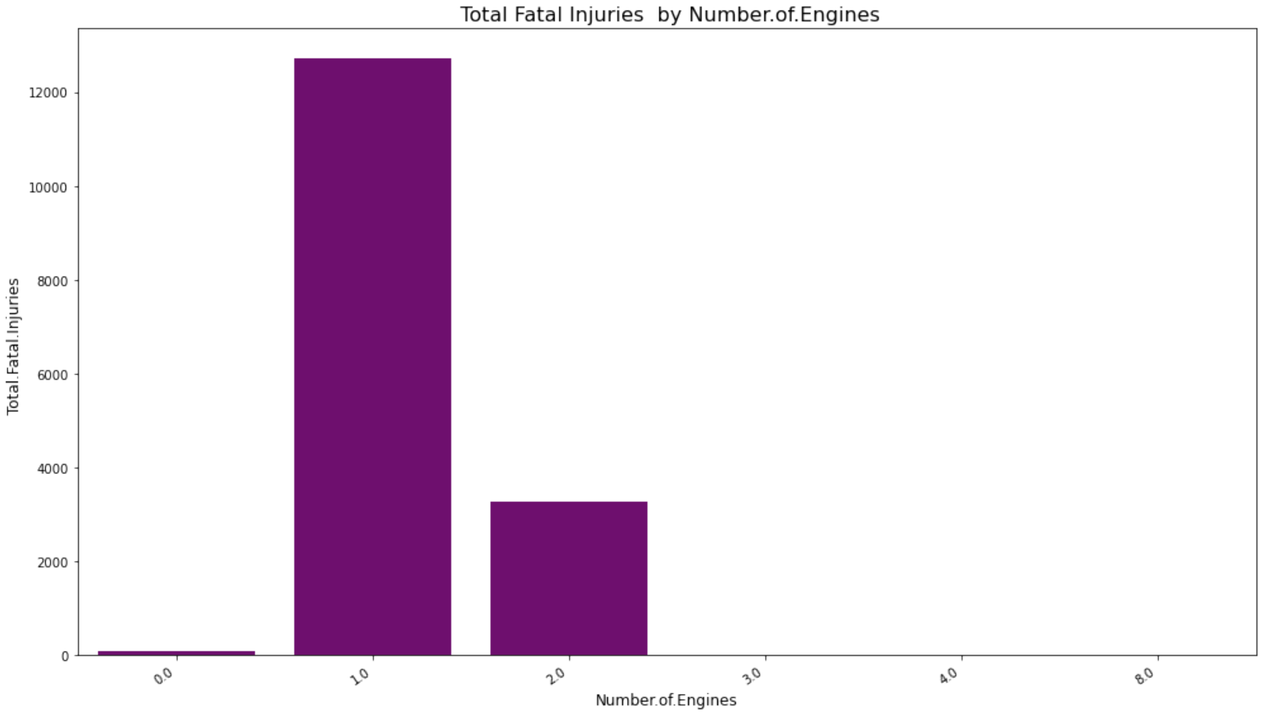
In [94]:

```
plt.figure(figsize=(14, 8))
sns.barplot(x='Number.of.Engines', y='Total.Minor.Injuries', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title(' Total.Minor.Injuries  by Number.of.Engines', fontsize=16)
plt.xlabel('Number.of.Engines', fontsize=12)
plt.ylabel(' Total.Minor.Injuries ', fontsize=12)
plt.xticks(rotation=35, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```

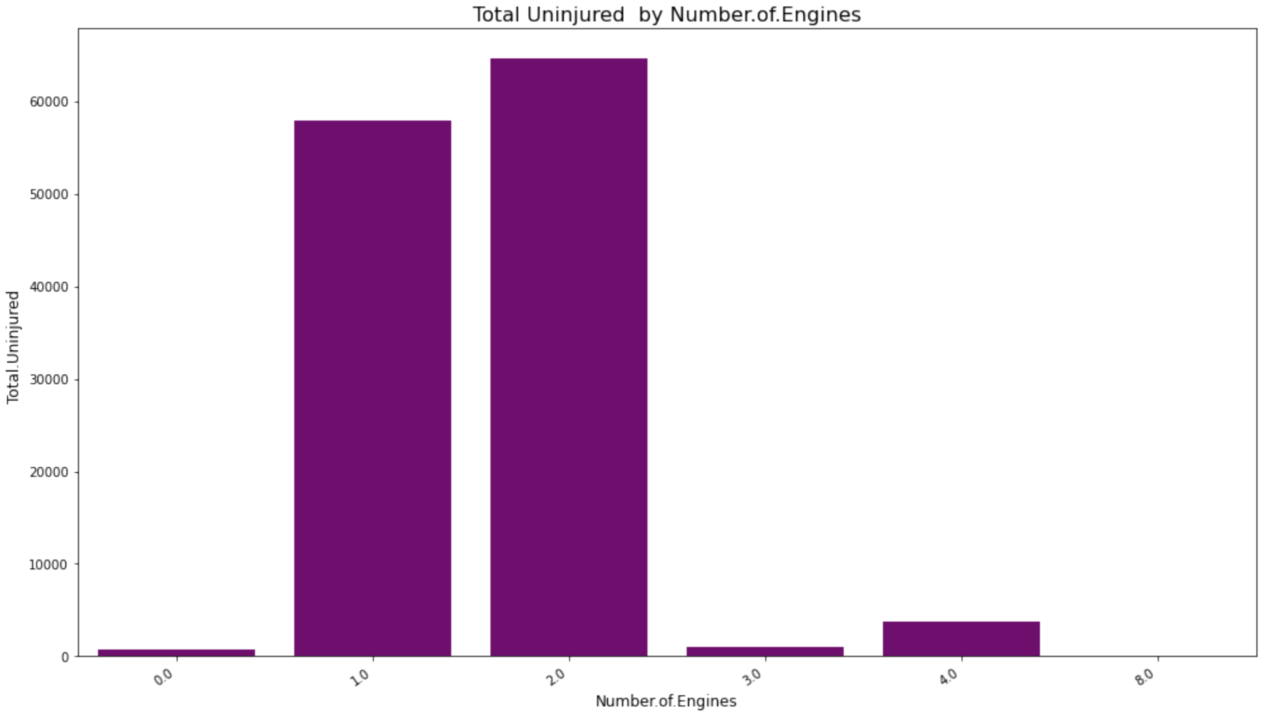


In [95]:

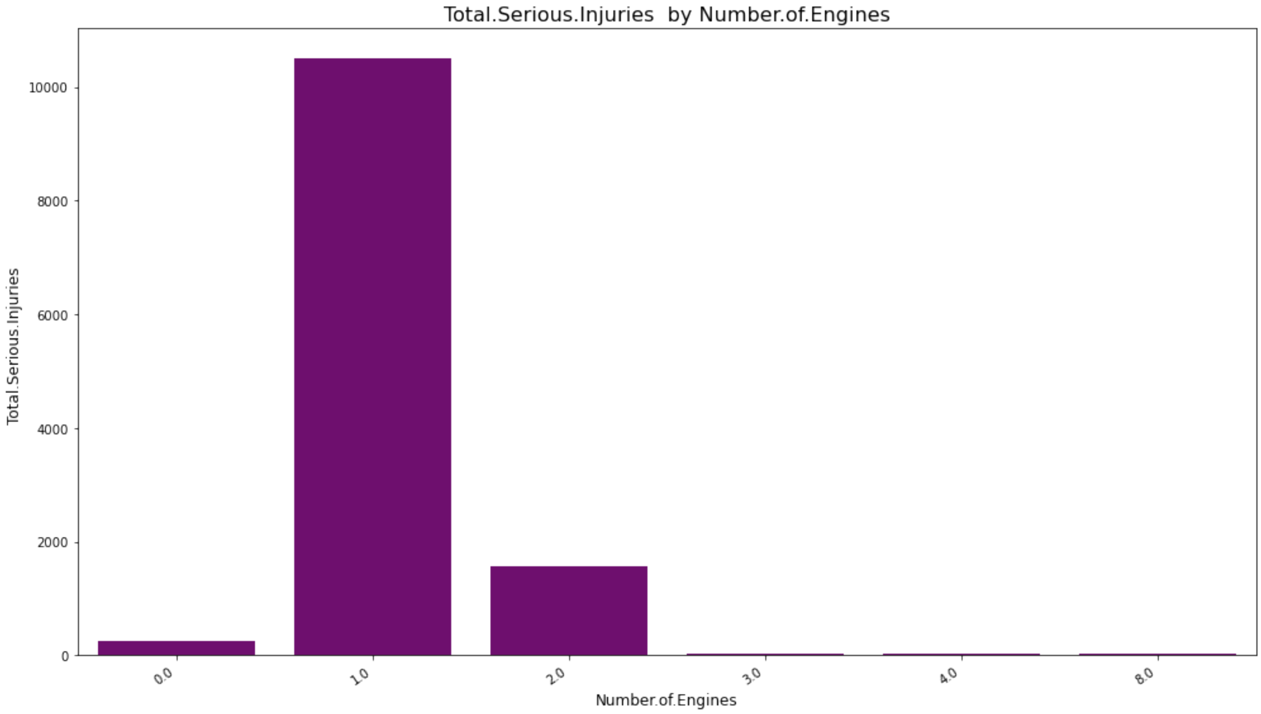
```
plt.figure(figsize=(14, 8))
sns.barplot(x='Number.of.Engines', y='Total.Fatal.Injuries', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title(' Total Fatal Injuries  by Number.of.Engines', fontsize=16)
plt.xlabel('Number.of.Engines', fontsize=12)
plt.ylabel(' Total.Fatal.Injuries  ', fontsize=12)
plt.xticks(rotation=35, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```



In [96]:
```
plt.figure(figsize=(14, 8))
sns.barplot(x='Number.of.Engines', y='Total.Uninjured', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title('Total Uninjured  by Number.of.Engines', fontsize=16)
plt.xlabel('Number.of.Engines', fontsize=12)
plt.ylabel('Total.Uninjured ', fontsize=12)
plt.xticks(rotation=35, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```



In [79]:
```
plt.figure(figsize=(14, 8))
sns.barplot(x='Number.of.Engines', y='Total.Serious.Injuries', data=df, estimator=sum, ci=None, color='purple')
# Customizing the plot
plt.title('Total.Serious.Injuries  by Number.of.Engines', fontsize=16)
plt.xlabel('Number.of.Engines', fontsize=12)
plt.ylabel('Total.Serious.Injuries ', fontsize=12)
plt.xticks(rotation=35, ha='right')
# Show the plot
plt.tight_layout()
plt.show()
```



In [62]:

```
#Lastly i wanna  Create Risk Scores
# Map numerical values to Injury.Severity
severity_mapping = {'Fatal': 3, 'Serious': 2, 'Non-Fatal': 1, 'Unavailable': 0}
df['Severity.Score'] = df['Injury.Severity'].map(severity_mapping)

# Map numerical values to Aircraft.damage
damage_mapping = {'Destroyed': 3, 'Substantial': 2, 'Minor': 1, 'Unknown' : 0}
df['Damage.Score'] = df['Aircraft.damage'].map(damage_mapping).fillna(0)
```

In [63]:

```
# Calculate the weighted risk score
df['Risk.Score'] = (
    df['Total.Fatal.Injuries'] * 3 +
    df['Total.Serious.Injuries'] * 2 +
    df['Total.Minor.Injuries'] * 1 +
    df['Severity.Score'] * 3 +
    df['Damage.Score'] * 2
)
```

In [64]:

```
# Summarizing  risk scores by Make and Model
risk_summary = df.groupby(['Make'])['Risk.Score'].mean().reset_index()

# Sort the summary by Risk.Score in ascending order (low risk first)
low_risk_aircraft = risk_summary.sort_values(by='Risk.Score',ascending =False)
```

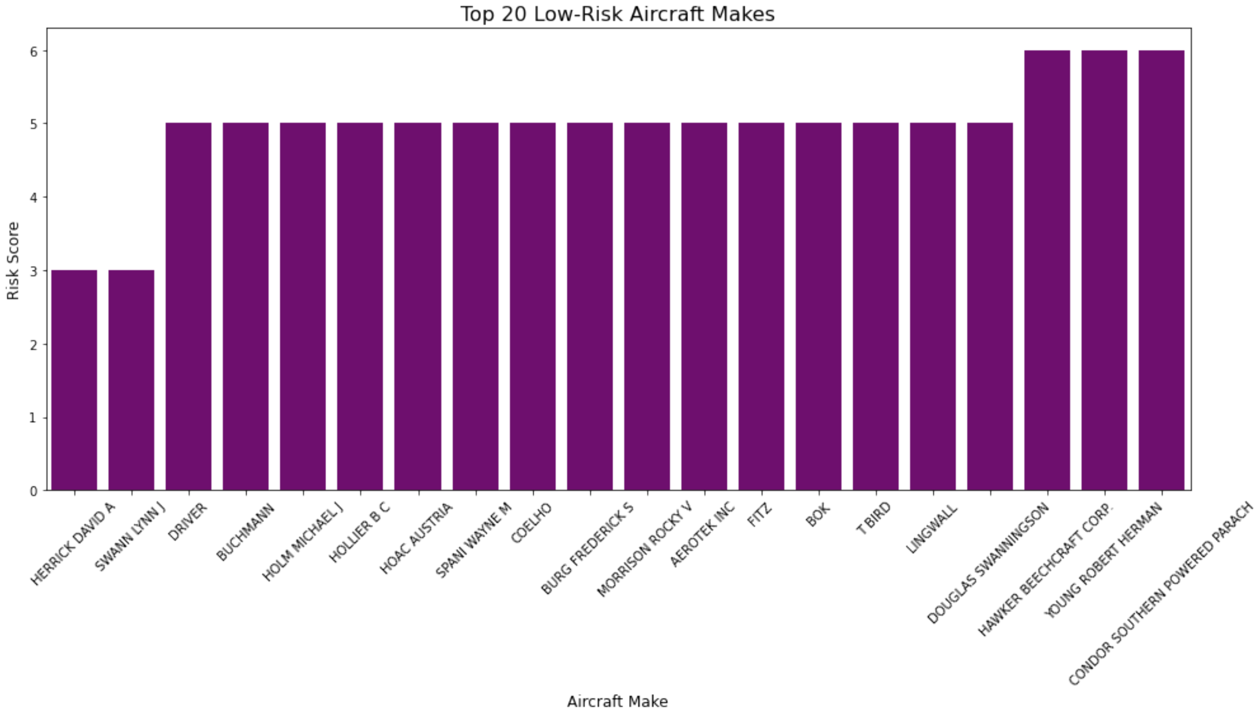
In [65]:

```
# Displaying  the top 10 low-risk aircraft
print(low_risk_aircraft.head(10))
```

```
                        Make  Risk.Score
672           BOEING COMPANY   46.000000
134         AIRBUS INDUSTRIE   33.388889
229          AMERICAN YANKEE   33.000000
5941  THUNDER BALLOONS, LTD.   32.000000
335                      ATR   30.000000
800        BRITISH AEROSPACE   27.666667
3270                  KERNER   27.000000
391                  BACHMAN   27.000000
88        AEROSPATIALE/SOCATA   27.000000
4581                 PLAVCAN   27.000000
```

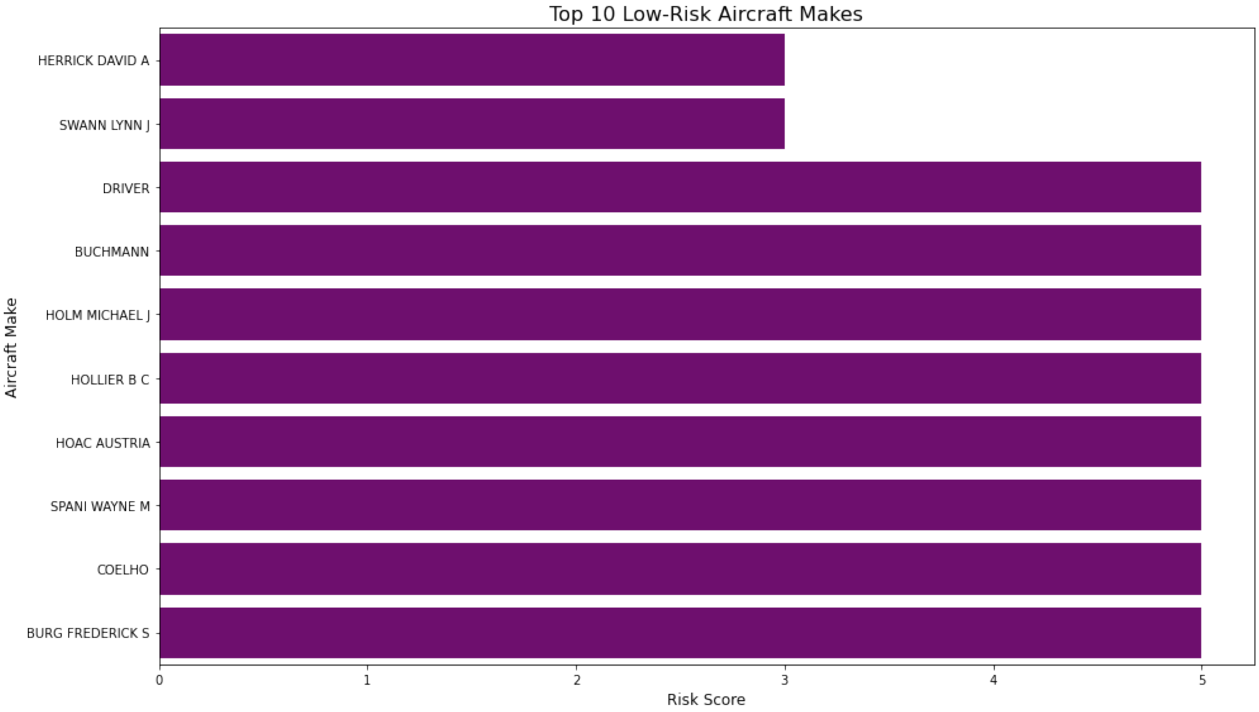
In [97]:

```
# Filtering the top 10 lowest-risk aircraft makes
top_20_low_risk = risk_summary.sort_values(by='Risk.Score', ascending=True).head(20)

plt.figure(figsize=(14, 8))
sns.barplot(y='Risk.Score', x='Make', data=top_20_low_risk, color='purple')

# Customize the plot
plt.title('Top 20 Low-Risk Aircraft Makes', fontsize=16)
plt.ylabel('Risk Score', fontsize=12)
plt.xlabel('Aircraft Make', fontsize=12)
plt.xticks(rotation=45)

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



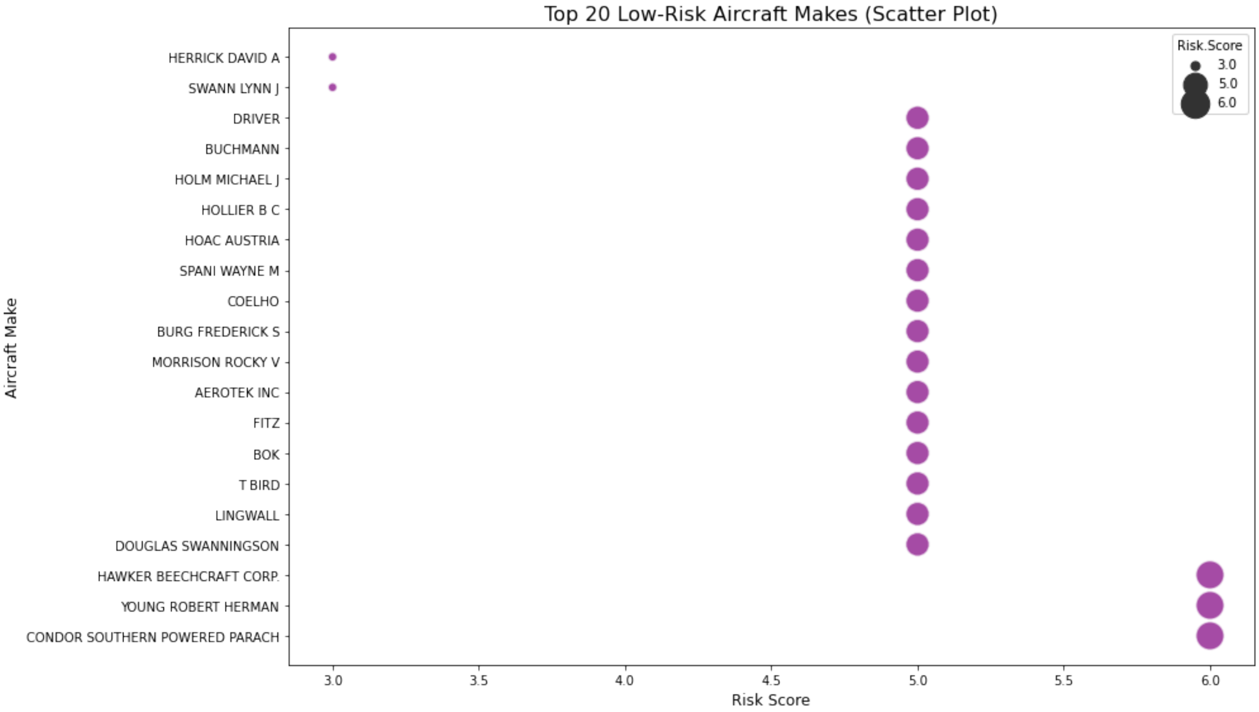
In [98]:

```
# Filter the top 10 lowest-risk aircraft makes
top_10_low_risk = risk_summary.sort_values(by='Risk.Score', ascending=True).head(10)

plt.figure(figsize=(14, 8))
sns.barplot(x='Risk.Score', y='Make', data=top_10_low_risk, color='purple')
plt.title('Top 10 Low-Risk Aircraft Makes', fontsize=16)
plt.xlabel('Risk Score', fontsize=12)
plt.ylabel('Aircraft Make', fontsize=12)
plt.tight_layout()
plt.show()
```

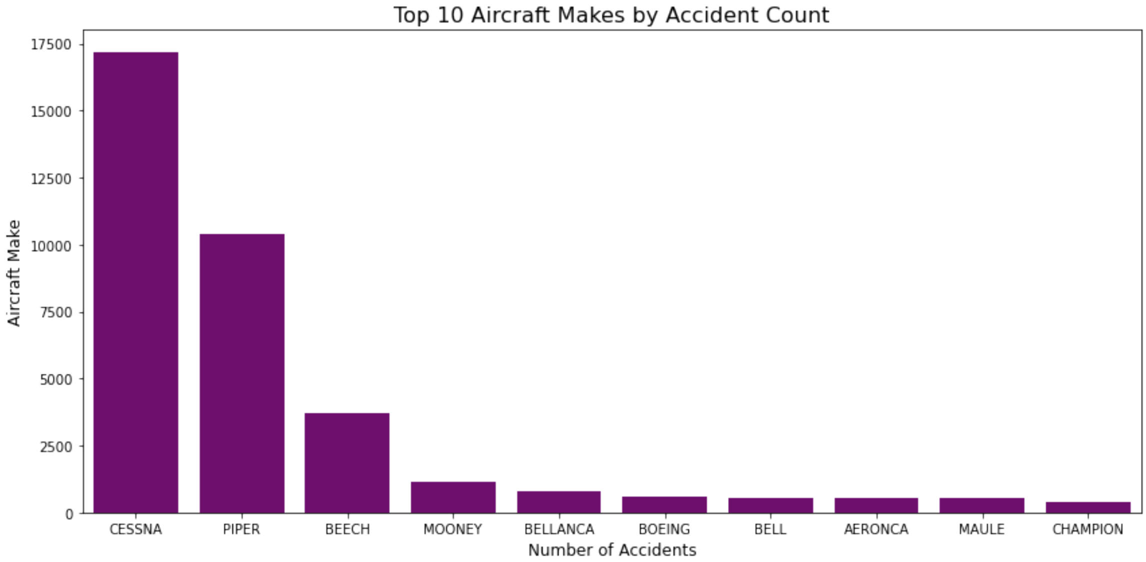


In [99]:

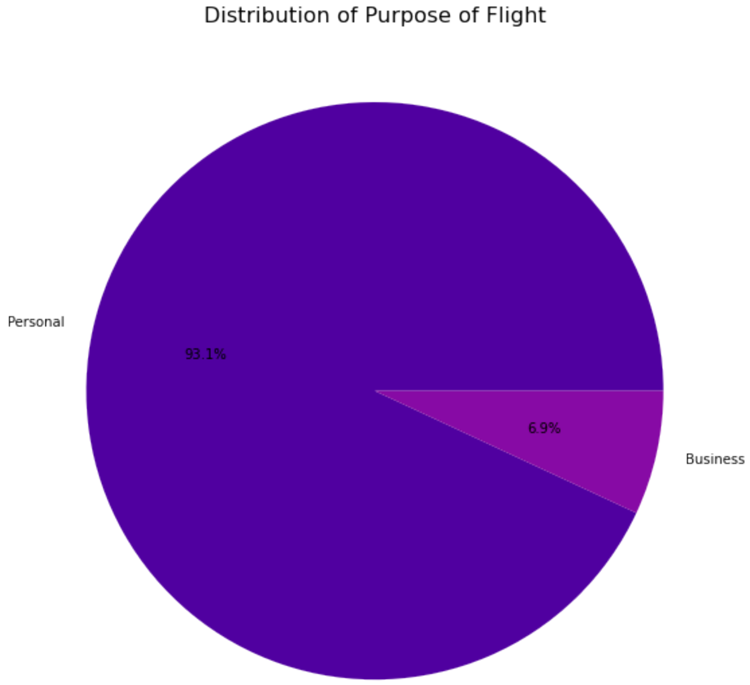
```
 #Filtering the top 10 lowest-risk aircraft makes
top_10_low_risk = risk_summary.sort_values(by='Risk.Score', ascending=True).head(50)

plt.figure(figsize=(14, 8))
sns.scatterplot(x='Risk.Score', y='Make', size='Risk.Score', data=top_20_low_risk, sizes=(50, 500), alpha=0.7, color='purpl
plt.title('Top 20 Low-Risk Aircraft Makes (Scatter Plot)', fontsize=16)
plt.xlabel('Risk Score', fontsize=12)
plt.ylabel('Aircraft Make', fontsize=12)
plt.tight_layout()
plt.show()
```



In [78]:
```
plt.figure(figsize=(12, 6))
sns.barplot(
    x=df['Make'].value_counts().index[:10],
    y=df['Make'].value_counts().values[:10],
    color='purple'
)
plt.title('Top 10 Aircraft Makes by Accident Count', fontsize=16)
plt.xlabel('Number of Accidents', fontsize=12)
plt.ylabel('Aircraft Make', fontsize=12)
plt.tight_layout()
plt.show()
```

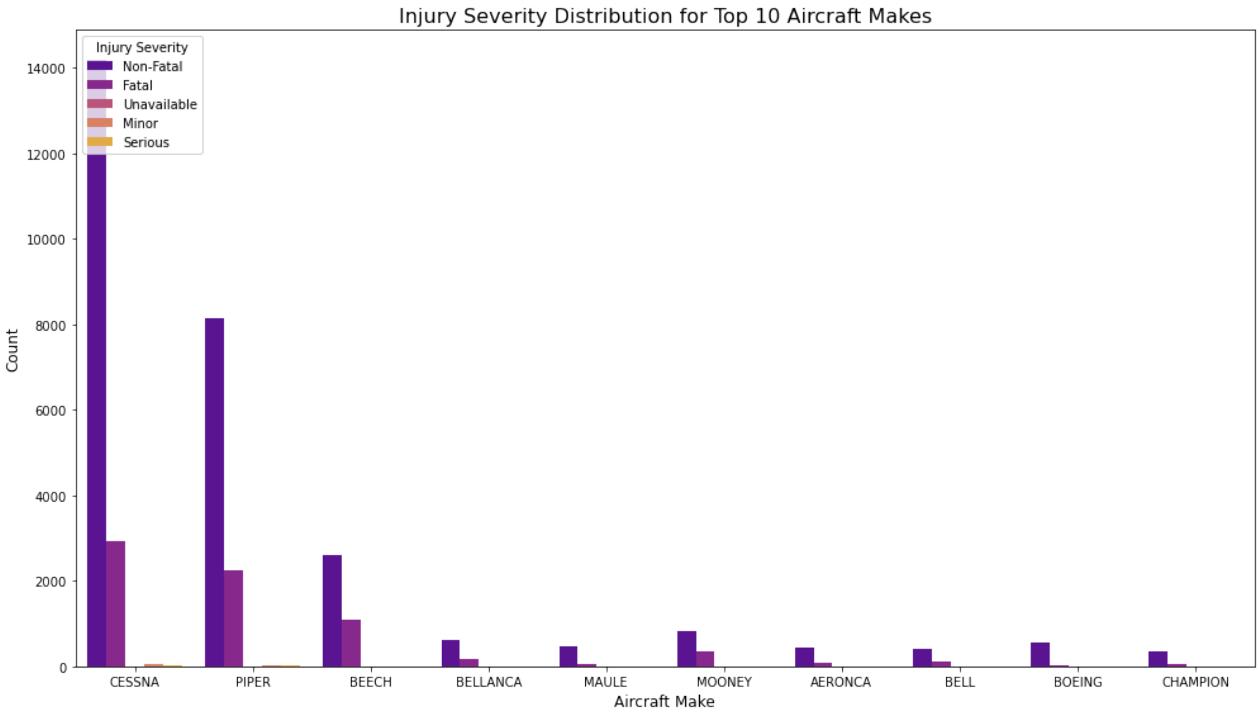


In [85]:
```
plt.figure(figsize=(8, 8))
df['Purpose.of.flight'].value_counts().plot.pie(
    autopct='%1.1f%%', colors=sns.color_palette('plasma')
)
plt.title('Distribution of Purpose of Flight', fontsize=16)
plt.ylabel('')
plt.tight_layout()
plt.show()
```



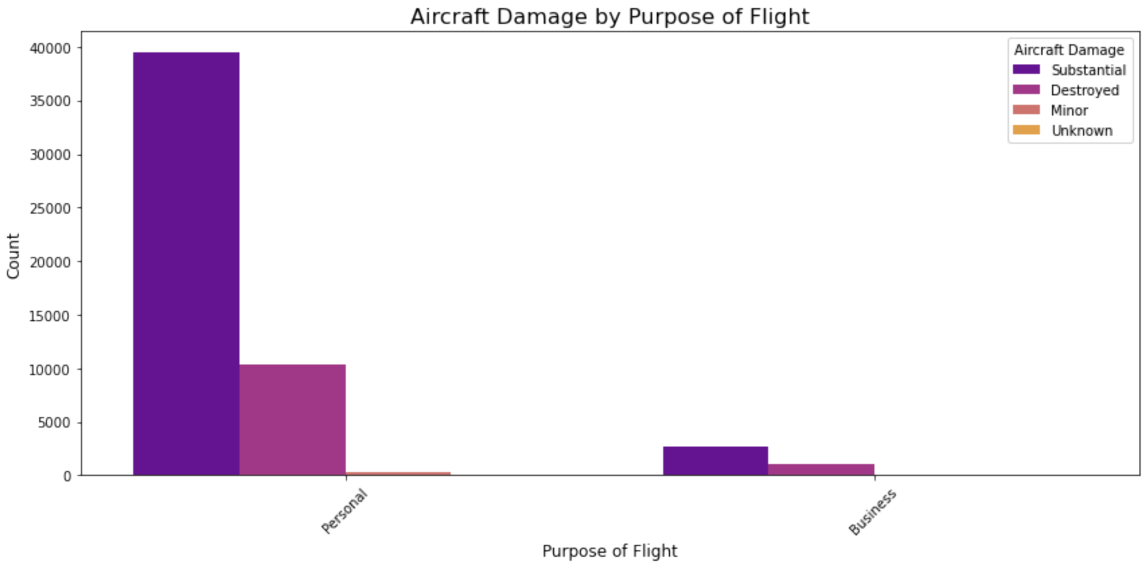
In [100]:

```
top_5_makes = df['Make'].value_counts().head(10).index
filtered_df = df[df['Make'].isin(top_5_makes)]

plt.figure(figsize=(14, 8))
sns.countplot(x='Make', hue='Injury.Severity', data=filtered_df, palette='plasma')
plt.title('Injury Severity Distribution for Top 10 Aircraft Makes', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.legend(title='Injury Severity')
plt.tight_layout()
plt.show()
```



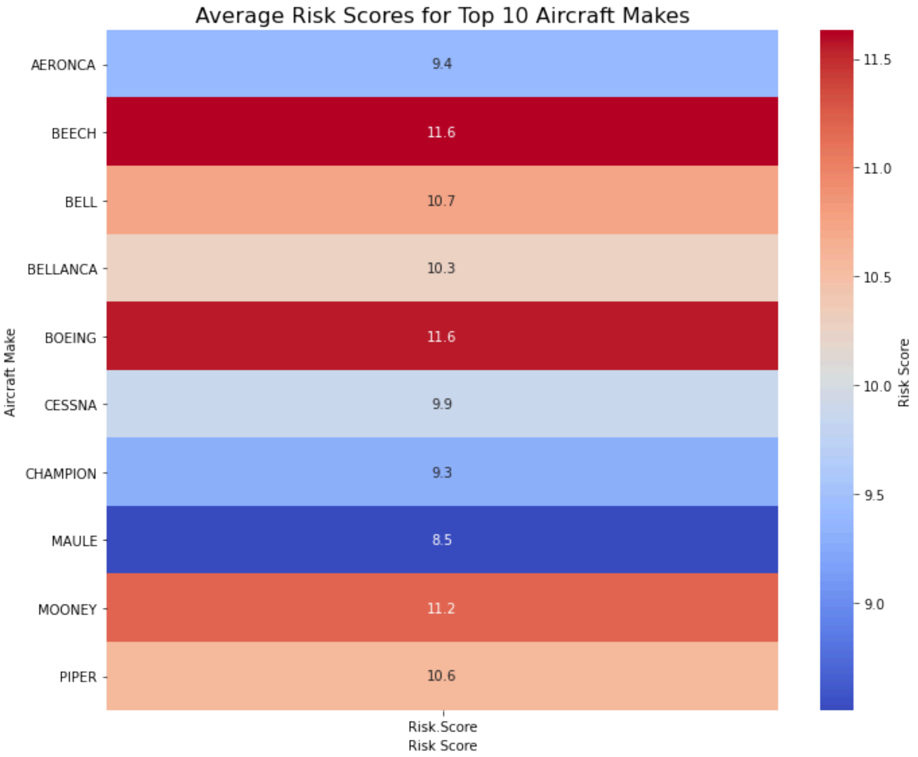
In [82]:

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Purpose.of.flight', hue='Aircraft.damage', data=df, palette='plasma')
plt.title('Aircraft Damage by Purpose of Flight', fontsize=16)
plt.xlabel('Purpose of Flight', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.legend(title='Aircraft Damage')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



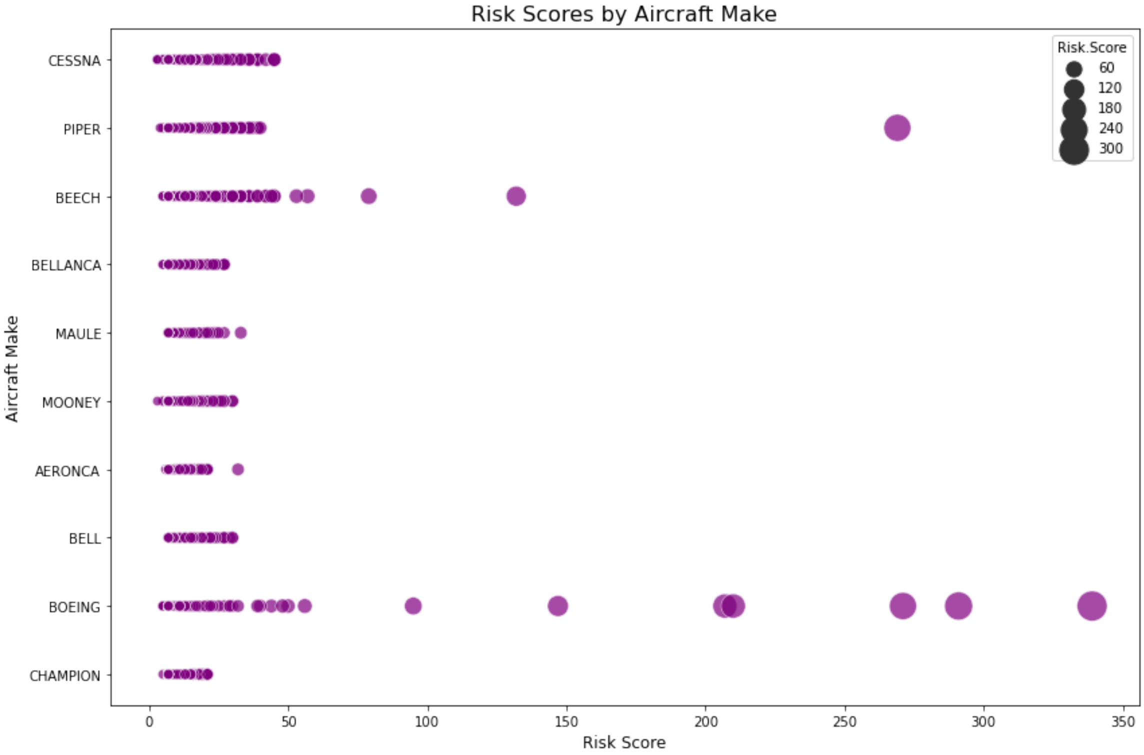
In [73]:

```
top_10_makes = df['Make'].value_counts().head(10).index
heatmap_data = df[df['Make'].isin(top_10_makes)].pivot_table(index='Make', values='Risk.Score', aggfunc='mean')

plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_data, annot=True, fmt=".1f", cmap="coolwarm", cbar_kws={'label': 'Risk Score'})
plt.title('Average Risk Scores for Top 10 Aircraft Makes', fontsize=16)
plt.xlabel('Risk Score')
plt.ylabel('Aircraft Make')
plt.tight_layout()
plt.show()
```



In [84]:

```
#doing a riskscore by aircraft make
top_10_makes = df['Make'].value_counts().head(10).index
filtered_df = df[df['Make'].isin(top_10_makes)]

plt.figure(figsize=(12, 8))
sns.scatterplot(x='Risk.Score', y='Make', size='Risk.Score', data=filtered_df, alpha=0.7, color='purple', sizes=(50, 500))
plt.title('Risk Scores by Aircraft Make', fontsize=16)
plt.xlabel('Risk Score', fontsize=12)
plt.ylabel('Aircraft Make', fontsize=12)
plt.tight_layout()
plt.show()
```



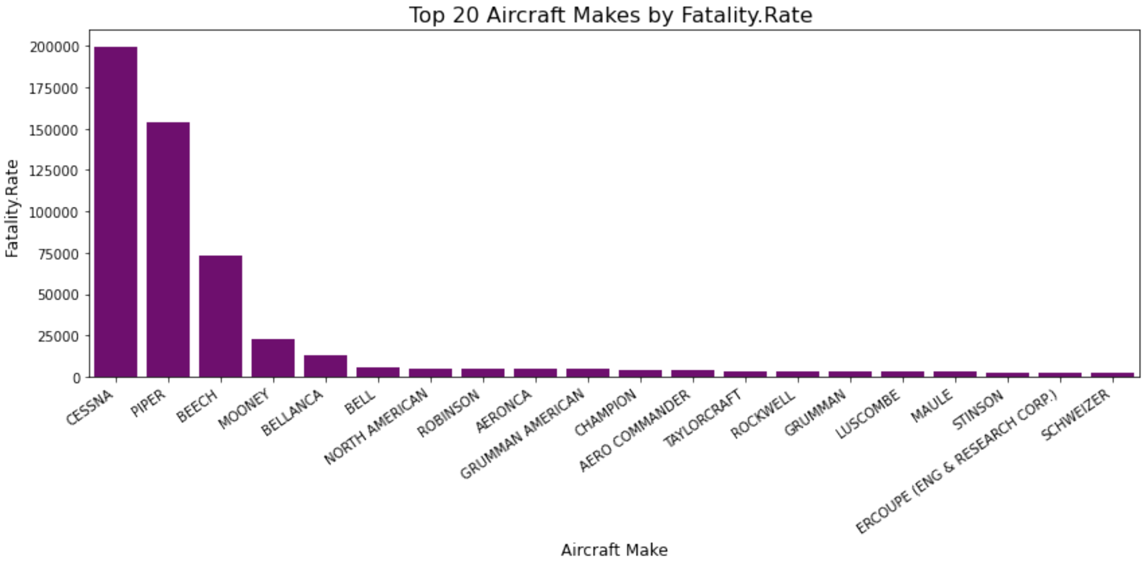
In [75]:

```
print(df.columns)
```

```
Index(['Event.Date', 'Location', 'Country', 'Aircraft.damage', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type',
       'Purpose.of.flight', 'Total.Serious.Injuries', 'Total.Minor.Injuries',
       'Total.Uninjured', 'Weather.Condition', 'Broad.phase.of.flight',
       'Injury.Severity', 'Total.Fatal.Injuries', 'Year', 'Month.Abbr',
       'Day.Name.Abbr', 'Total.People', 'Fatality.Rate', 'Severity.Score',
       'Damage.Score', 'Risk.Score'],
      dtype='object')
```

In [81]:

```
# doing a bar plot of Total Fatal injuries vs Make for top ten aircrafts
make_injuries = df.groupby('Make')['Fatality.Rate'].sum().reset_index()
# Getting  the top 10 Makes by Total.Fatal.Injuries
top_10_makes = make_injuries.nlargest(20, 'Fatality.Rate')
plt.figure(figsize=(12, 6))
sns.barplot(x='Make', y='Fatality.Rate', data=top_10_makes, color='purple')
plt.title('Top 20 Aircraft Makes by Fatality.Rate ', fontsize=16)
plt.xlabel('Aircraft Make', fontsize=12)
plt.ylabel('Fatality.Rate ', fontsize=12)
plt.xticks(rotation=35, ha='right')
plt.tight_layout()
plt.show()
```



In [ ]: