Business Objective

Our company is expanding its portfolio by entering the aviation industry, with plans to purchase and operate airplanes for both commercial and private enterprises. However, the company lacks expertise in assessing the potential risks associated with aircraft operations. This analysis aims to identify the aircraft models with the lowest risk profiles to guide the head of the new aviation division in making informed purchasing decisions.

Aviation Data Analysis Project

This notebook analyzes aviation accident and incident data from the `Aviation_Data.csv` file. The dataset contains information about aviation events. The goal is to explore the data, identify patterns, assess key risk factors, visualize key insights to recommend specific aircraft for purchase, prioritizing those with the best safety records.

 Airplane flying over Aviation Analysis dashboard

Let's do this 💪💪

Step 1: Import Libraries

We will use the following Python libraries:

- `pandas` and `numpy` for data manipulation and analysis
- `matplotlib` and `seaborn` for visualizations

In [121]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 2: Load the Dataset

Load the `Aviation_Data.csv` file into a pandas DataFrame and inspect its structure.

In [123]:
```
# Load the dataset
# As a result of this error during importing, "DtypeWarning: Columns (6,7,28)
have mixed types. Specify dtype option on import or set low_memory=False.",

df = pd.read_csv('Aviation_Data.csv', low_memory=False)

# Display the first 5 rows
df.head()
```

Out[123]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	La
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.9
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	

5 rows × 31 columns

Step 3: Data Exploration

Examining the dataset's structure and data types.

Check for missing values.

In [125]:
```
# Display basic information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      90348 non-null  object 
 2   Accident.Number         88889 non-null  object 
 3   Event.Date              88889 non-null  object 
 4   Location                88837 non-null  object 
 5   Country                 88663 non-null  object 
 6   Latitude                34382 non-null  object 
 7   Longitude               34373 non-null  object 
 8   Airport.Code            50132 non-null  object 
 9   Airport.Name            52704 non-null  object 
 10  Injury.Severity         87889 non-null  object 
 11  Aircraft.damage         85695 non-null  object 
 12  Aircraft.Category       32287 non-null  object 
 13  Registration.Number     87507 non-null  object 
 14  Make                    88826 non-null  object 
 15  Model                   88797 non-null  object 
 16  Amateur.Built           88787 non-null  object 
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object 
 19  FAR.Description         32023 non-null  object 
 20  Schedule                12582 non-null  object 
 21  Purpose.of.flight       82697 non-null  object 
 22  Air.carrier             16648 non-null  object 
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object 
 28  Broad.phase.of.flight   61724 non-null  object 
 29  Report.Status           82505 non-null  object 
 30  Publication.Date        73659 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

From the above information, we can tell the below:

1. RangeIndex: 90348 entries We have 90348 rows and any column that has less than that `Non-Null Count` as presented above, is proof of missing values.
2. Data columns (total 31 columns): We have 31 columns.
3. Data types: With exception of `Number.of.Engines` , `Total.Fatal.Injuries` , `Total.Serious.Injuries` , `Total.Minor.Injuries` , `Total.Uninjured` that have a `float64` , all the other columns have an `object` data type.

In [127]:
```
# Check for missing values
df.isnull().sum()
```

Out[127]:
```
Event.Id                 1459
Investigation.Type          0
Accident.Number          1459
Event.Date               1459
Location                 1511
Country                  1685
Latitude                55966
Longitude               55975
Airport.Code            40216
Airport.Name            37644
Injury.Severity          2459
Aircraft.damage          4653
Aircraft.Category       58061
Registration.Number      2841
Make                     1522
Model                    1551
Amateur.Built            1561
Number.of.Engines        7543
Engine.Type              8555
FAR.Description         58325
Schedule                77766
Purpose.of.flight        7651
Air.carrier             73700
Total.Fatal.Injuries    12860
Total.Serious.Injuries  13969
Total.Minor.Injuries    13392
Total.Uninjured          7371
Weather.Condition        5951
Broad.phase.of.flight   28624
Report.Status            7843
Publication.Date        16689
dtype: int64
```

In [128]:
```
# Check for missing values in %
df.isnull().mean() *100
```

Out[128]:
```
Event.Id                 1.614867
Investigation.Type       0.000000
Accident.Number          1.614867
Event.Date               1.614867
Location                 1.672422
Country                  1.865011
Latitude                61.944924
Longitude               61.954886
Airport.Code            44.512330
Airport.Name            41.665560
Injury.Severity          2.721698
Aircraft.damage          5.150086
Aircraft.Category       64.263736
Registration.Number      3.144508
Make                     1.684597
Model                    1.716695
Amateur.Built            1.727764
Number.of.Engines        8.348829
Engine.Type              9.468942
FAR.Description         64.555939
Schedule                86.073848
Purpose.of.flight        8.468367
Air.carrier             81.573471
Total.Fatal.Injuries    14.233851
Total.Serious.Injuries  15.461327
Total.Minor.Injuries    14.822686
Total.Uninjured          8.158454
Weather.Condition        6.586753
Broad.phase.of.flight   31.681941
Report.Status            8.680878
Publication.Date        18.471909
dtype: float64
```

Step 4: Data Preprocessing and Data Visualization

Step 1: Select the appropriate data to help us uncover the insights

In [131]:
```
# Filter for airplanes (exclude helicopters, gliders,...) and non-amateur-buil
t aircraft
# Amateur-built aircraft are Aircraft built by individuals, often from kits or
plans, for education or recreation.
# This is specifically to narrow down to our business need

df = df[(df['Aircraft.Category'] == 'Airplane') & (df['Amateur.Built'] == 'N
o')]
```

In [132]:
```
df.shape
```

Out[132]:
```
(24417, 31)
```

We notice that the rows have reduced from 90348 to 24417.

We will proceed to use this 27 % of the data

In [134]:
```
# Select relevant columns
columns = ['Event.Id', 'Investigation.Type', 'Event.Date', 'Location', 'Country', 'Injury.Severity',
           'Aircraft.damage', 'Aircraft.Category', 'Airport.Name', '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']

df = df[columns]
```

In [135]:
```
df['Aircraft.damage'].value_counts()
```

Out[135]:
```
Aircraft.damage
Substantial    18981
Destroyed       3144
Minor            925
Unknown           97
Name: count, dtype: int64
```

In [136]:
```
df['Weather.Condition'].value_counts()
```

Out[136]:
```
Weather.Condition
VMC    19711
IMC     1363
Unk      215
UNK      149
Name: count, dtype: int64
```

In [137]:
```
df['Broad.phase.of.flight'].value_counts()
```

Out[137]:
```
Broad.phase.of.flight
Landing        2073
Takeoff        1132
Cruise          760
Approach        556
Maneuvering     455
Taxi            232
Descent         163
Climb           143
Go-around       143
Standing         73
Unknown          55
Other            11
Name: count, dtype: int64
```

In [138]:
```
df['Purpose.of.flight'].value_counts()
```

Out[138]:
```
Purpose.of.flight
Personal                  13339
Instructional              3120
Aerial Application         1066
Unknown                     838
Business                    764
Positioning                 350
Aerial Observation          170
Skydiving                   166
Ferry                       163
Other Work Use              155
Executive/corporate         148
Flight Test                 120
Banner Tow                   89
Public Aircraft - Federal    52
Air Race show                48
Public Aircraft              42
Glider Tow                   35
Public Aircraft - State      24
Firefighting                 17
Public Aircraft - Local      12
ASHO                          5
Air Race/show                 4
Air Drop                      3
PUBS                          3
External Load                 1
Name: count, dtype: int64
```

In [139]:
```
df['Model'].value_counts()
```

Out[139]:
```
Model
172          869
152          449
737          403
182          344
172N         315
            ...
LM1            1
P-40E          1
Tierra I       1
767-432ER      1
M-8 EAGLE      1
Name: count, Length: 3779, dtype: int64
```

In [140]:
```
df['Make'].value_counts()
```

Out[140]:
```
Make
CESSNA                  4867
Cessna                  3576
PIPER                   2803
Piper                   1897
BOEING                  1037
                        ...
Air Tractor, Inc.          1
Van's Aircraft, Inc.       1
M7Aero                     1
Piper/Cub Crafters         1
ORLICAN S R O              1
Name: count, Length: 1382, dtype: int64
```

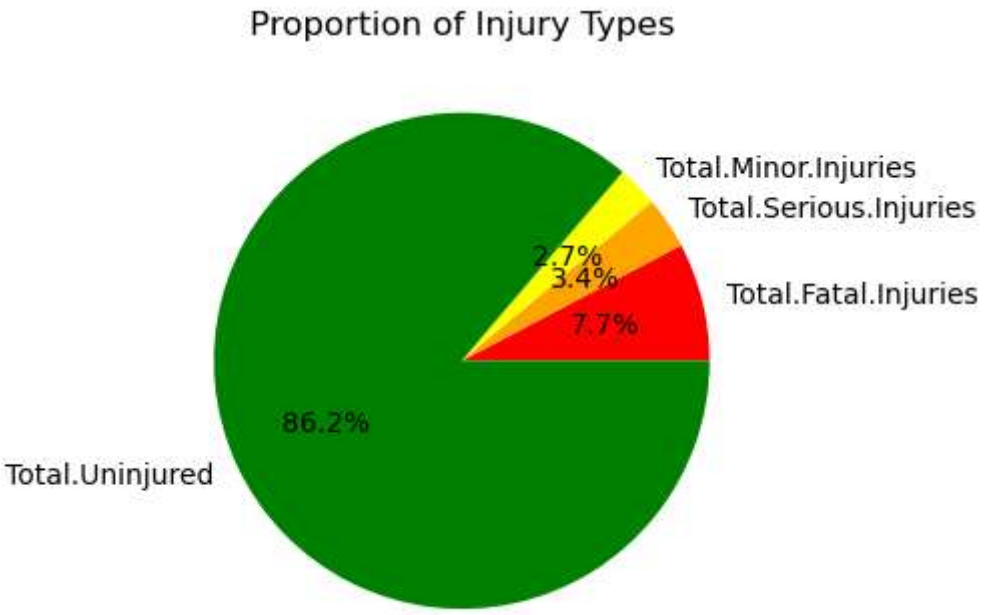
Confirmation that we are dealing with data just relevant to `Airplanes`

In [142]:
```
df['Aircraft.Category'].value_counts()
```

Out[142]:
```
Aircraft.Category
Airplane    24417
Name: count, dtype: int64
```

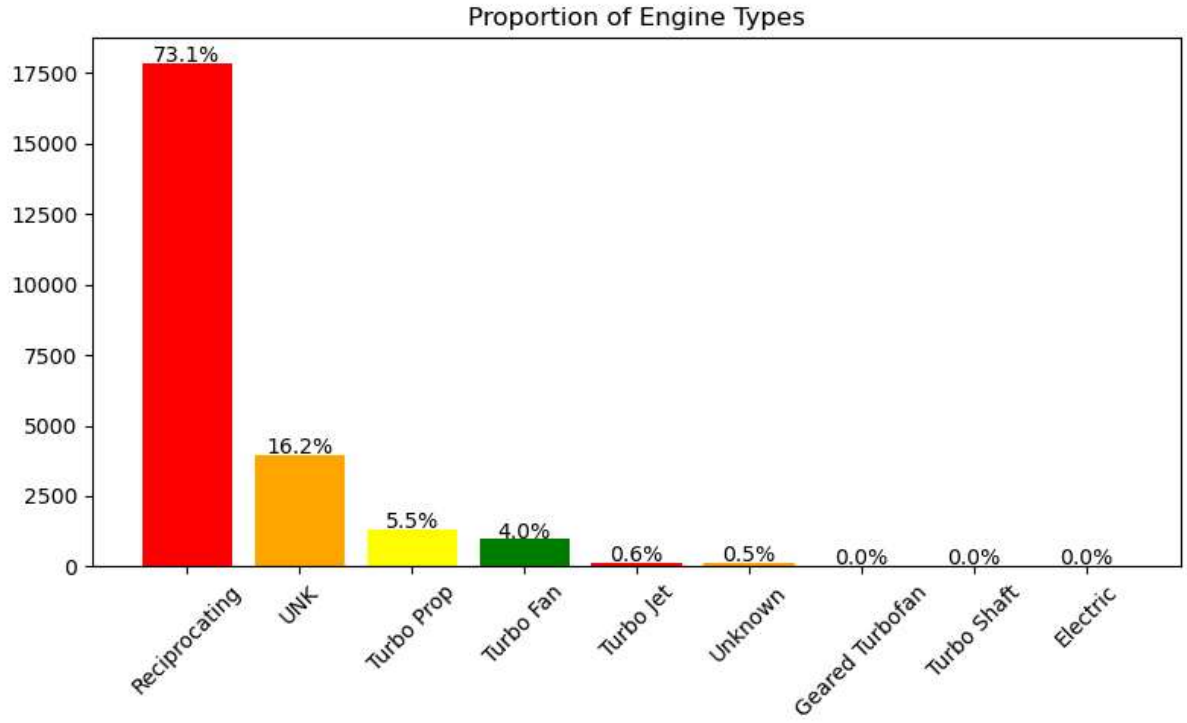
In [143]:
```
# Handle missing values for Quantitative data
# Handle missing values for injuries and damage
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].fillna(0)
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].fillna(0)
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].fillna(0)
df['Total.Uninjured'] = df['Total.Uninjured'].fillna(0)
```

In [144]:
```
# Let's understand the distribution of those injured
injury_totals = df[['Total.Fatal.Injuries','Total.Serious.Injuries','Total.Minor.Injuries','Total.Uninjured']].sum()
injury_totals.plot(kind='pie',autopct='%1.1f%%',colors=['red', 'orange', 'yellow', 'green'],figsize=(5, 5))

plt.title('Proportion of Injury Types')
plt.ylabel('')  # Remove y-axis label
plt.tight_layout()
plt.show()
```

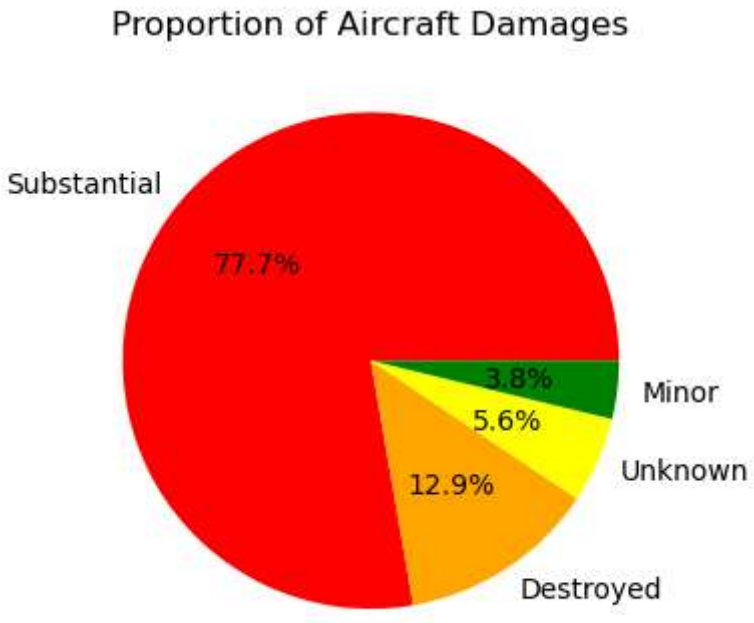


In [145]:
```
# Handle missing values for Qualitative data
df['Aircraft.damage'] = df['Aircraft.damage'].fillna('Unknown')
df['Weather.Condition'] = df['Weather.Condition'].fillna('UNK')
df['Broad.phase.of.flight'] = df['Broad.phase.of.flight'].fillna('Unknown')
df['Purpose.of.flight'] = df['Purpose.of.flight'].fillna('Unknown')
df['Engine.Type'] = df['Engine.Type'].fillna('UNK')
```

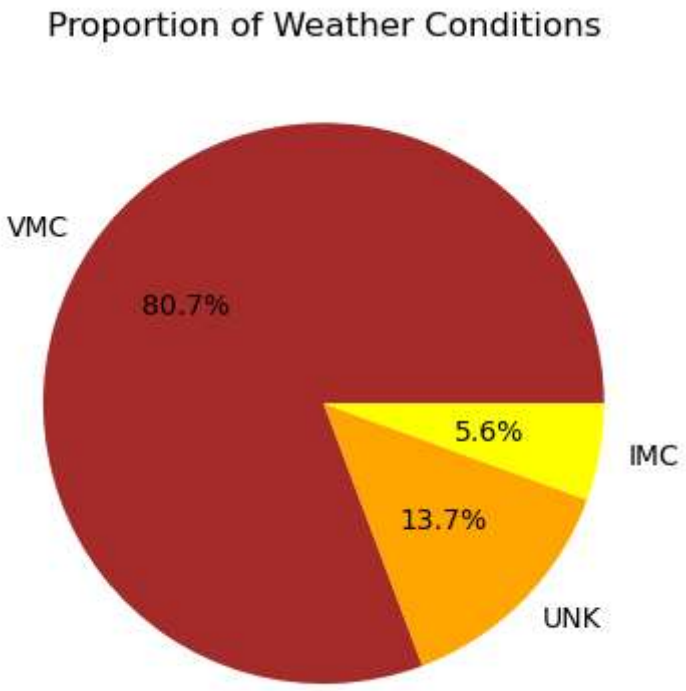
In [146]:
```
# Count and compute percentage
engine_type_counts = df['Engine.Type'].value_counts()
engine_type_percent = (engine_type_counts / engine_type_counts.sum()) * 100

# Plot
plt.figure(figsize=(8, 5))
bars = plt.bar(
    engine_type_counts.index,
    engine_type_counts.values,
    color=['red', 'orange', 'yellow', 'green']
)

# Annotate bars with percentage labels
for bar, pct in zip(bars, engine_type_percent):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 50,
             f'{pct:.1f}%', ha='center')

plt.title('Proportion of Engine Types')
plt.ylabel('')  # Optional: hide y-axis label
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



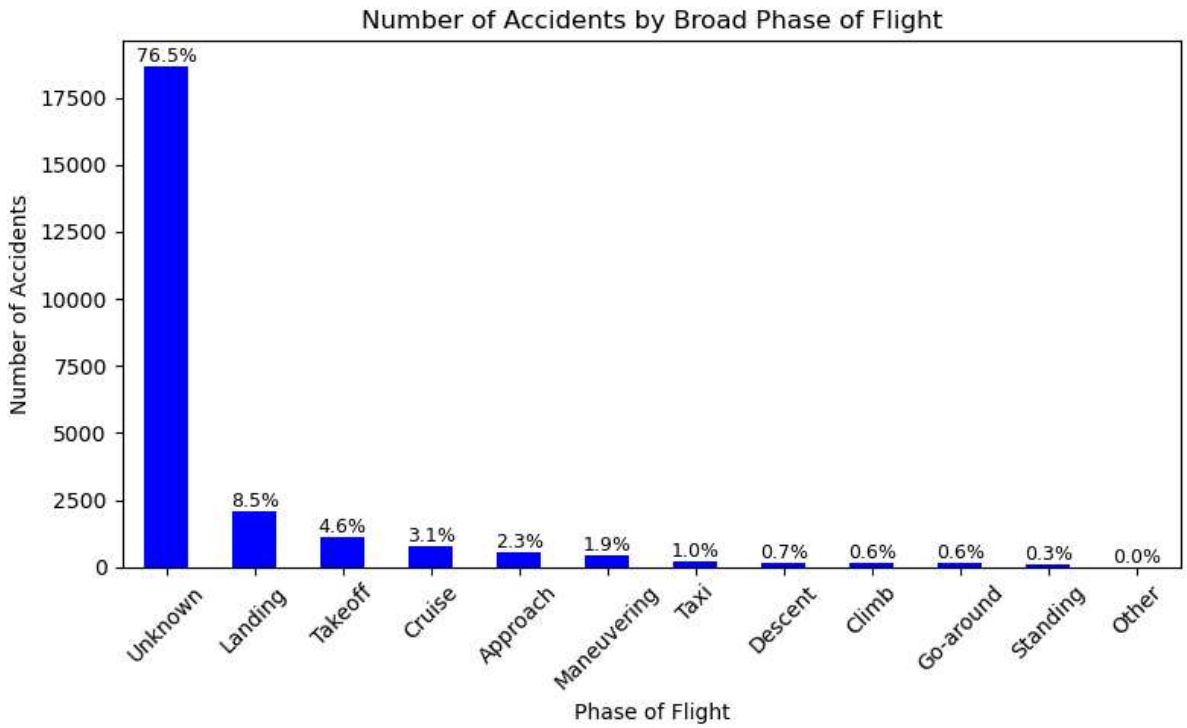
In [147]:
```
# Proportion of aircraft damages on number of accidents
aircraft_damage = df['Aircraft.damage'].value_counts()

aircraft_damage.plot(kind='pie',autopct='%1.1f%%',colors=['red', 'orange', 'ye
llow', 'green'],figsize=(4, 4))

plt.title('Proportion of Aircraft Damages')
plt.ylabel('')  # Remove y-axis label
plt.tight_layout()
plt.show()
```



In [148]:
```
# The below code is added after realizing that the pie sub-divides because of
different cases.
df['Weather.Condition'] = df['Weather.Condition'].str.upper()

# Proportion of Weather Conditions on number of accidents
weather_conditions = df['Weather.Condition'].value_counts()

weather_conditions.plot(kind='pie',autopct='%1.1f%%',colors=['brown', 'orang
e', 'yellow', 'green'],figsize=(4, 4))

plt.title('Proportion of Weather Conditions')
plt.ylabel('')  # Remove y-axis label
plt.tight_layout()
plt.show()
```



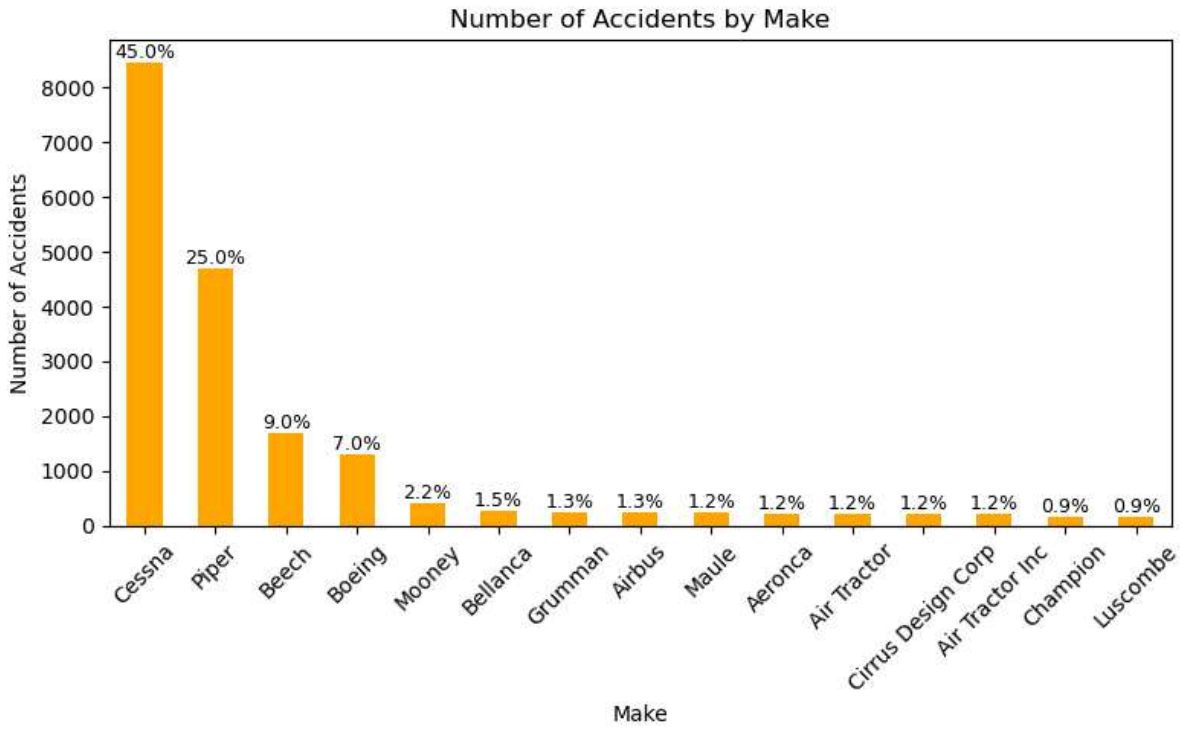
In [149]:
```
# Count each phase
flight_phases = df['Broad.phase.of.flight'].value_counts()

# Total number of accidents
total = flight_phases.sum()

# Plot bar chart
ax = flight_phases.plot(
    kind='bar',
    color='blue',
    figsize=(8, 5)
)

# Add percentage labels on top of each bar
for i, value in enumerate(flight_phases):
    percentage = (value / total) * 100
    ax.text(i, value + 1, f'{percentage:.1f}%', ha='center', va='bottom', font
size=9)

# Labels and title
plt.title('Number of Accidents by Broad Phase of Flight')
plt.xlabel('Phase of Flight')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



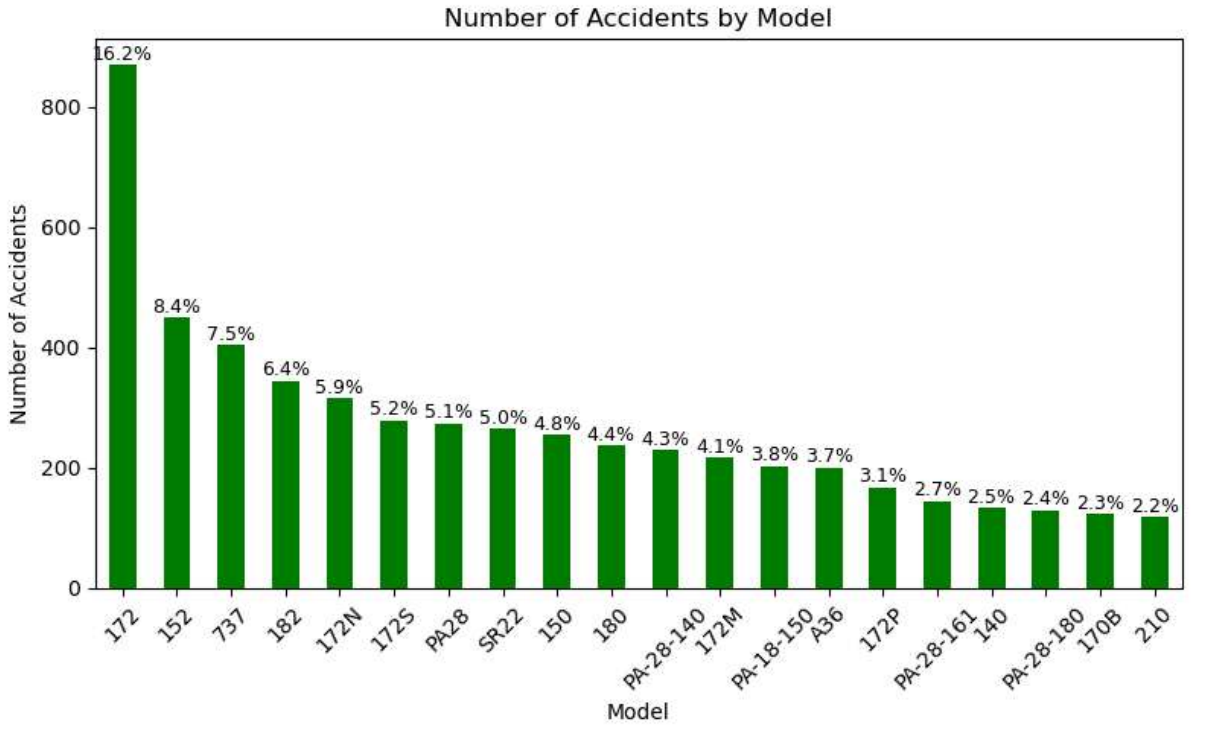
This is a good column, however about 77% of it's data is unknown, so it won't be used in decision-making.

In [151]:
```
# Standardize text fields
df['Make'] = df['Make'].str.title().str.strip()
df['Model'] = df['Model'].str.upper().str.strip()
df['Aircraft.Category'] = df['Aircraft.Category'].str.title().str.strip()
```

In [152]:
```
# Proportion of top 15 makes
# Count each phase
makes = df['Make'].value_counts().head(15)

# Total number of accidents
total = makes.sum()

# Plot bar chart
ax = makes.plot(
    kind='bar',
    color='orange',
    figsize=(8, 5)
)

# Add percentage labels on top of each bar
for i, value in enumerate(makes):
    percentage = (value / total) * 100
    ax.text(i, value + 1, f'{percentage:.1f}%', ha='center', va='bottom', font
size=9)

# Labels and title
plt.title('Number of Accidents by Make')
plt.xlabel('Make')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [153]:
```
# Proportion of top 20 models
# Count each phase
makes = df['Model'].value_counts().head(20)

# Total number of accidents
total = makes.sum()

# Plot bar chart
ax = makes.plot(
    kind='bar',
    color='green',
    figsize=(8, 5)
)

# Add percentage labels on top of each bar
for i, value in enumerate(makes):
    percentage = (value / total) * 100
    ax.text(i, value + 1, f'{percentage:.1f}%', ha='center', va='bottom', font
size=9)

# Labels and title
plt.title('Number of Accidents by Model')
plt.xlabel('Model')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Confirmation we are working with aeroplanes only 😅

In [155]:
```
df['Aircraft.Category'].value_counts()
```

Out[155]:
```
Aircraft.Category
Airplane    24417
Name: count, dtype: int64
```

Good 😊

Step 2: Define Risk Metrics

In [158]:
```
# Calculate total injuries
df['Total.Injuries'] = df['Total.Fatal.Injuries'] + df['Total.Serious.Injuries'] + df['Total.Minor.Injuries']
```

In [159]:
```
# Assign numerical scores to damage severity
damage_scores = {'Destroyed': 3, 'Substantial': 2, 'Minor': 1, 'Unknown': 0, 'None': 0}
df['Damage.Score'] = df['Aircraft.damage'].map(damage_scores)
```

In [160]:
```
df['Damage.Score']
```

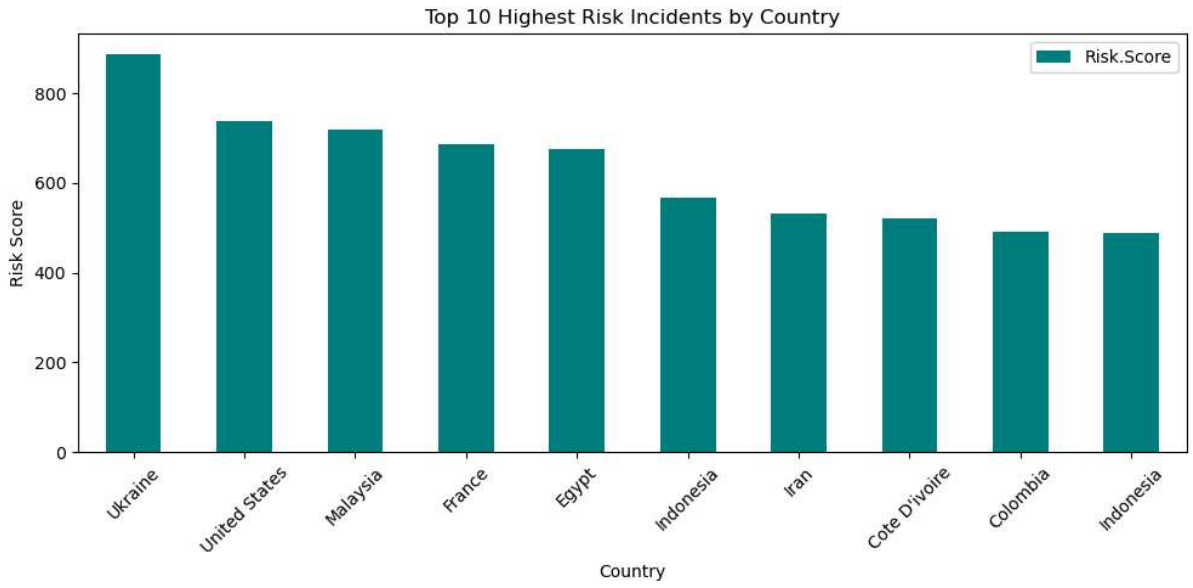
Out[160]:
```
5        2
7        2
8        2
12       3
13       3
        ..
90328    2
90332    2
90335    2
90336    2
90345    2
Name: Damage.Score, Length: 24417, dtype: int64
```

Creating a risk score, which is sum of quantified damage and the injuries

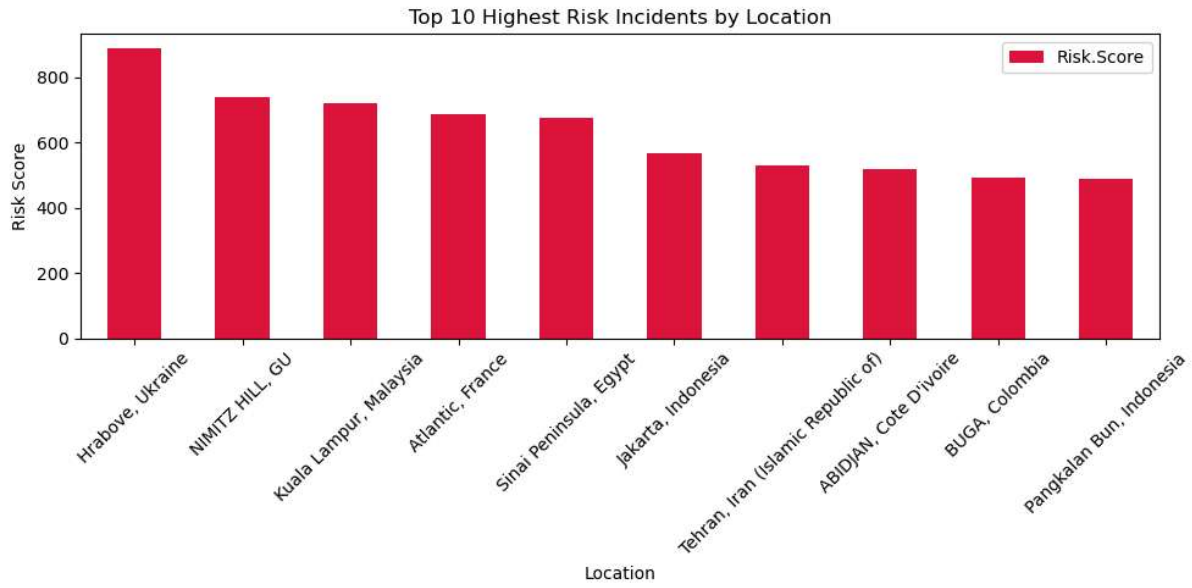
In [162]:
```
# Create a risk score: weighted sum of fatal injuries (x3), serious injuries (x2), minor injuries (x1), and damage score
df['Risk.Score'] = (3 * df['Total.Fatal.Injuries'] + 2 * df['Total.Serious.Injuries'] +
                    df['Total.Minor.Injuries'] + df['Damage.Score'])
```

Step 3: Risk Analysis

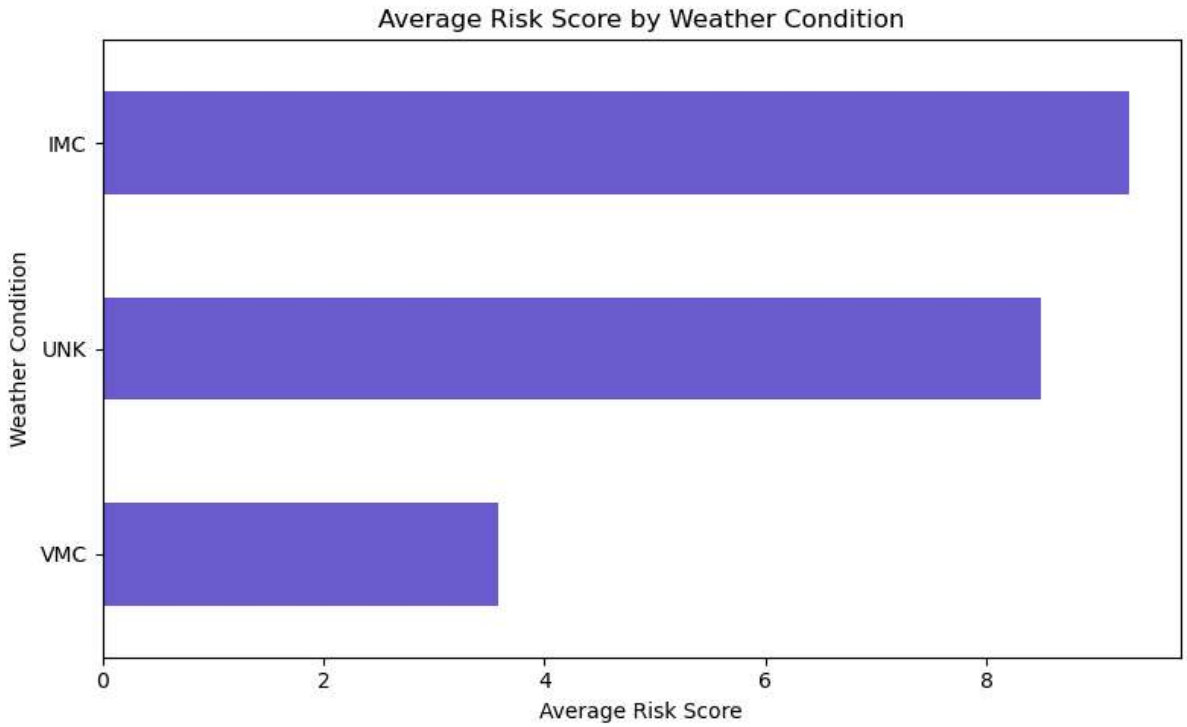
In [164]:
```
top_risks = df.sort_values('Risk.Score', ascending=False).head(10)

top_risks.plot(
    x='Country',
    y='Risk.Score',
    kind='bar',
    color='teal',
    figsize=(10, 5)
)

plt.title('Top 10 Highest Risk Incidents by Country')
plt.xlabel('Country')
plt.ylabel('Risk Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



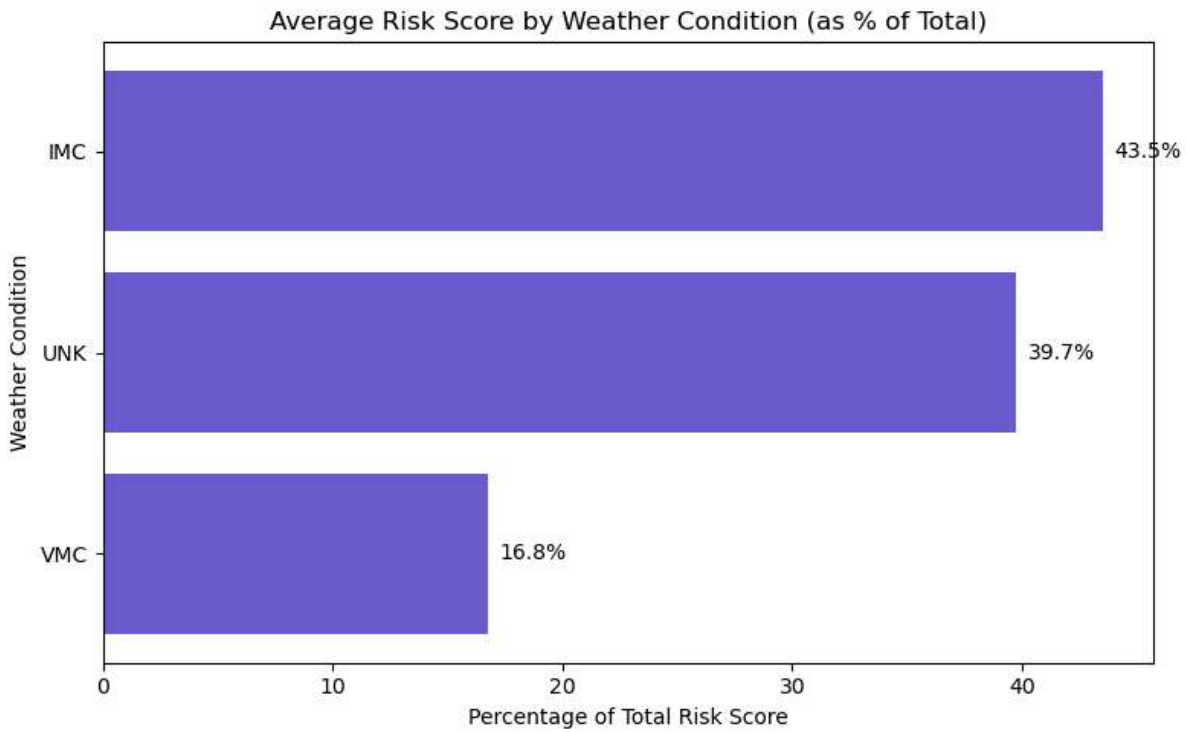
In [165]:
```
top_risks = df.sort_values('Risk.Score', ascending=False).head(10)

top_risks.plot(
    x='Location',
    y='Risk.Score',
    kind='bar',
    color='crimson',
    figsize=(10, 5)
)

plt.title('Top 10 Highest Risk Incidents by Location')
plt.xlabel('Location')
plt.ylabel('Risk Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



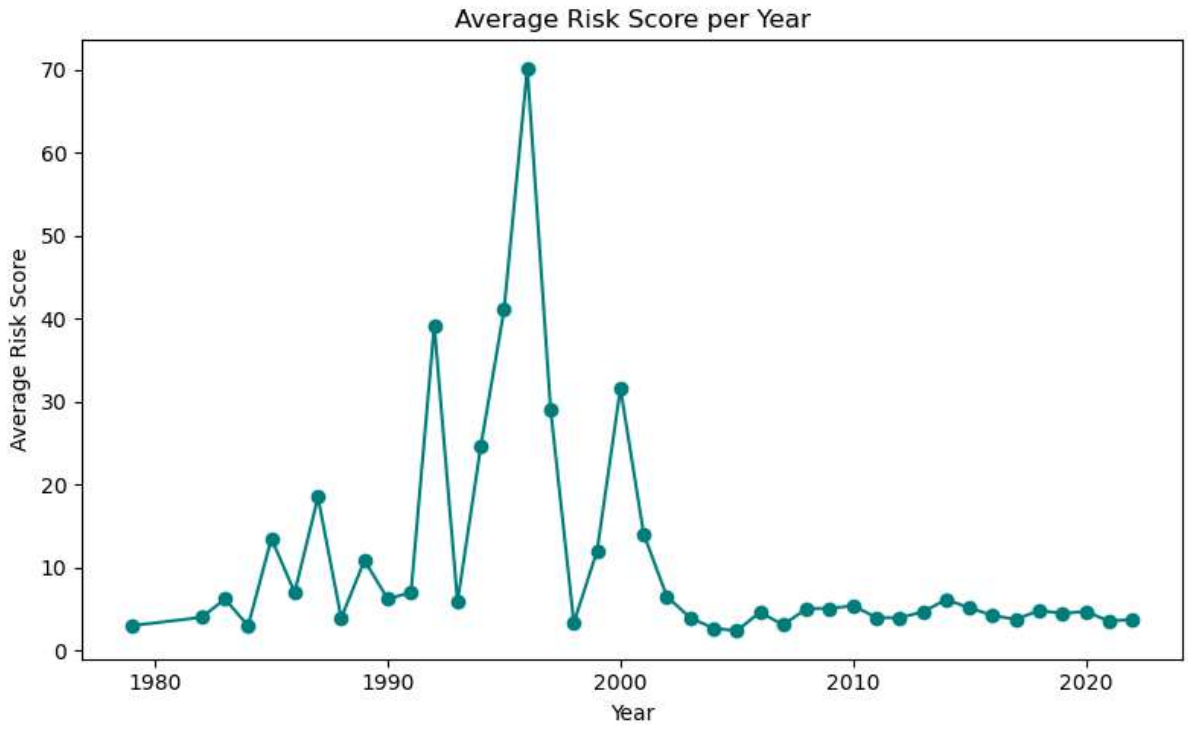
In [166]:
```
# Risk Score by Weather Conditions
avg_risk_weather = df.groupby('Weather.Condition')['Risk.Score'].mean().sort_v
alues()

avg_risk_weather.plot(kind='barh', figsize=(8, 5), color='slateblue')

plt.title('Average Risk Score by Weather Condition')
plt.xlabel('Average Risk Score')
plt.ylabel('Weather Condition')
plt.tight_layout()
plt.show()
```



In [167]:
```
# Group by and calculate average risk score
avg_risk_weather = df.groupby('Weather.Condition')['Risk.Score'].mean().sort_v
alues()

# Normalize to percentages (optional, based on total)
percentage = (avg_risk_weather / avg_risk_weather.sum()) * 100

# Plot
plt.figure(figsize=(8, 5))
bars = plt.barh(percentage.index, percentage.values, color='slateblue')

# Title and labels
plt.title('Average Risk Score by Weather Condition (as % of Total)')
plt.xlabel('Percentage of Total Risk Score')
plt.ylabel('Weather Condition')

# Annotate each bar with percentage value
for bar in bars:
    width = bar.get_width()
    plt.text(width + 0.5, bar.get_y() + bar.get_height()/2,
             f'{width:.1f}%', va='center')

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

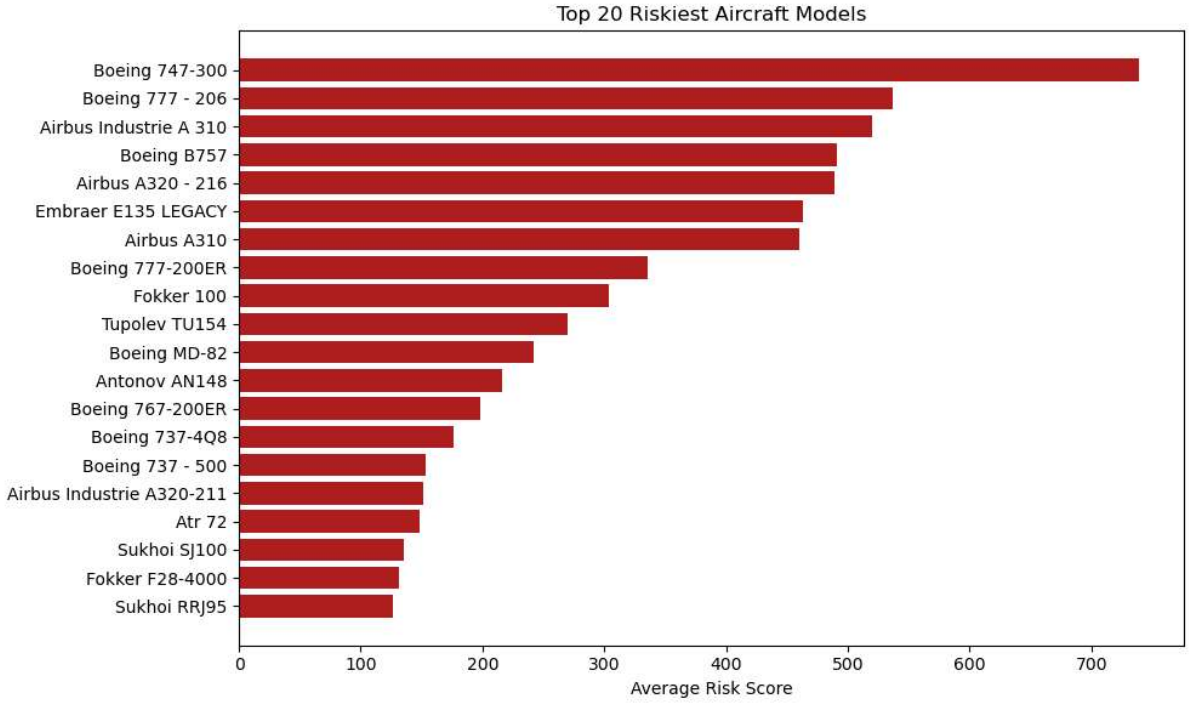


In [168]:
```
# Risk trend over time
df['Event.Year'] = pd.to_datetime(df['Event.Date']).dt.year

risk_trend = df.groupby('Event.Year')['Risk.Score'].mean()

risk_trend.plot(kind='line', marker='o', color='teal', figsize=(8, 5))

plt.title('Average Risk Score per Year')
plt.xlabel('Year')
plt.ylabel('Average Risk Score')
plt.tight_layout()
plt.show()
```

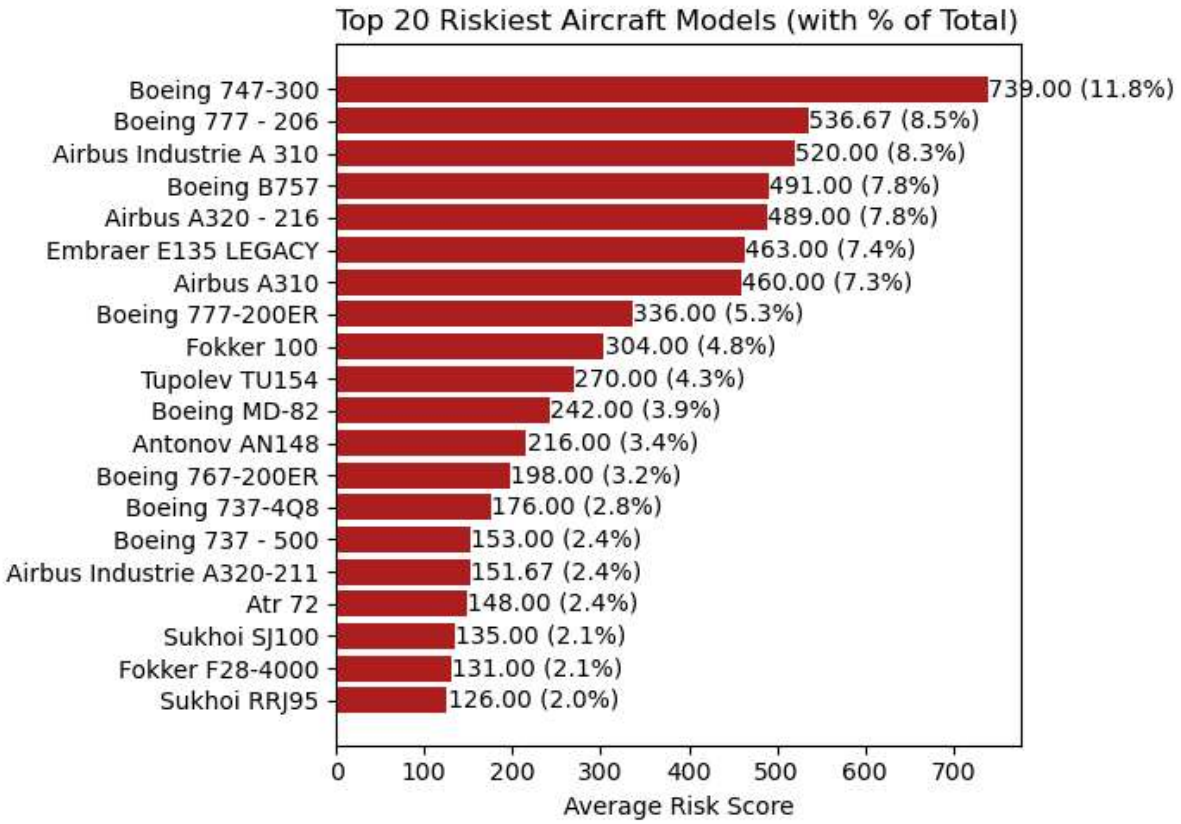


In [169]:
```
# Group by Make and Model
risk_by_model = df.groupby(['Make', 'Model']).agg({
    'Event.Id': 'count',
    'Total.Fatal.Injuries': 'sum',
    'Total.Serious.Injuries': 'sum',
    'Total.Minor.Injuries': 'sum',
    'Total.Uninjured': 'sum',
    'Risk.Score': 'mean'
}).rename(columns={'Event.Id': 'Accident.Count'}).reset_index()
```

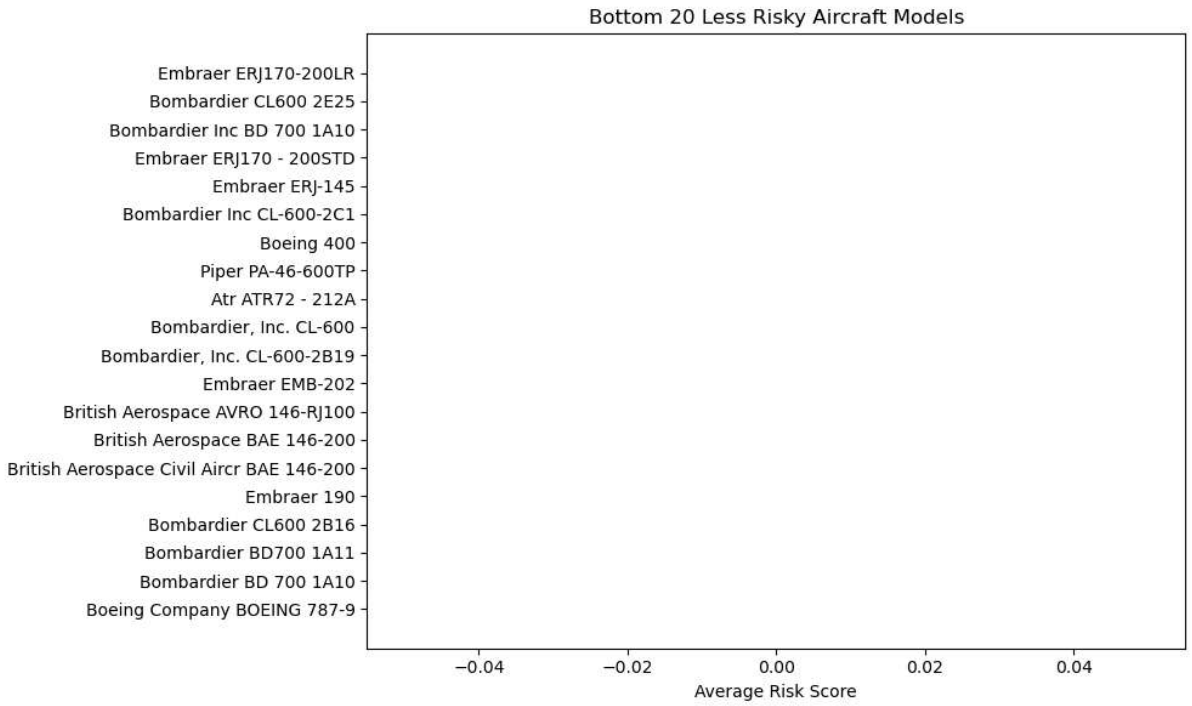
In [170]:
```
# Top 20 Riskiest Aircraft Make and Models (by Average Risk Score)
top_risk_models = risk_by_model.sort_values('Risk.Score', ascending=False).hea
d(20)

plt.figure(figsize=(10, 6))
plt.barh(
    top_risk_models['Make'] + ' ' + top_risk_models['Model'],
    top_risk_models['Risk.Score'],
    color='firebrick'
)
plt.xlabel('Average Risk Score')
plt.title('Top 20 Riskiest Aircraft Models')
plt.gca().invert_yaxis()  # highest at top
plt.tight_layout()
plt.show()
```



In [171]:
```
# Select top 20 risky models
top_risk_models = risk_by_model.sort_values('Risk.Score', ascending=False).hea
d(20)

# Calculate total for percentage
total_risk = top_risk_models['Risk.Score'].sum()

# Create the plot
plt.figure(figsize=(7, 5))
bars = plt.barh(
    top_risk_models['Make'] + ' ' + top_risk_models['Model'],
    top_risk_models['Risk.Score'],
    color='firebrick'
)

# Add annotations (percentage)
for bar, value in zip(bars, top_risk_models['Risk.Score']):
    percent = (value / total_risk) * 100
    plt.text(
        value + 0.1,
        bar.get_y() + bar.get_height() / 2,
        f'{value:.2f} ({percent:.1f}%)',
        va='center'
    )

# Add titles and labels
plt.xlabel('Average Risk Score')
plt.title('Top 20 Riskiest Aircraft Models (with % of Total)')
plt.gca().invert_yaxis()  # highest at top
plt.tight_layout()
plt.show()
```


Top 20 Riskiest Aircraft Models (with % of Total)
Boeing 747-300
Boeing 777 - 206
Airbus Industrie A 310
Boeing B757
Airbus A320 - 216
Embraer E135 LEGACY
Airbus A310
Boeing 777-200ER
Fokker 100
Tupolev TU154
Boeing MD-82
Antonov AN148
Boeing 767-200ER
Boeing 737-4Q8
Boeing 737 - 500
Airbus Industrie A320-211
Atr 72
Sukhoi SJ100
Fokker F28-4000
Sukhoi RRJ95
739.00 (11.8%)
536.67 (8.5%)
520.00 (8.3%)
491.00 (7.8%)
489.00 (7.8%)
463.00 (7.4%)
460.00 (7.3%)
336.00 (5.3%)
304.00 (4.8%)
270.00 (4.3%)
242.00 (3.9%)
216.00 (3.4%)
198.00 (3.2%)
176.00 (2.8%)
153.00 (2.4%)
151.67 (2.4%)
148.00 (2.4%)
135.00 (2.1%)
131.00 (2.1%)
126.00 (2.0%)
0
100
200
300
400
500
600
700
Average Risk Score

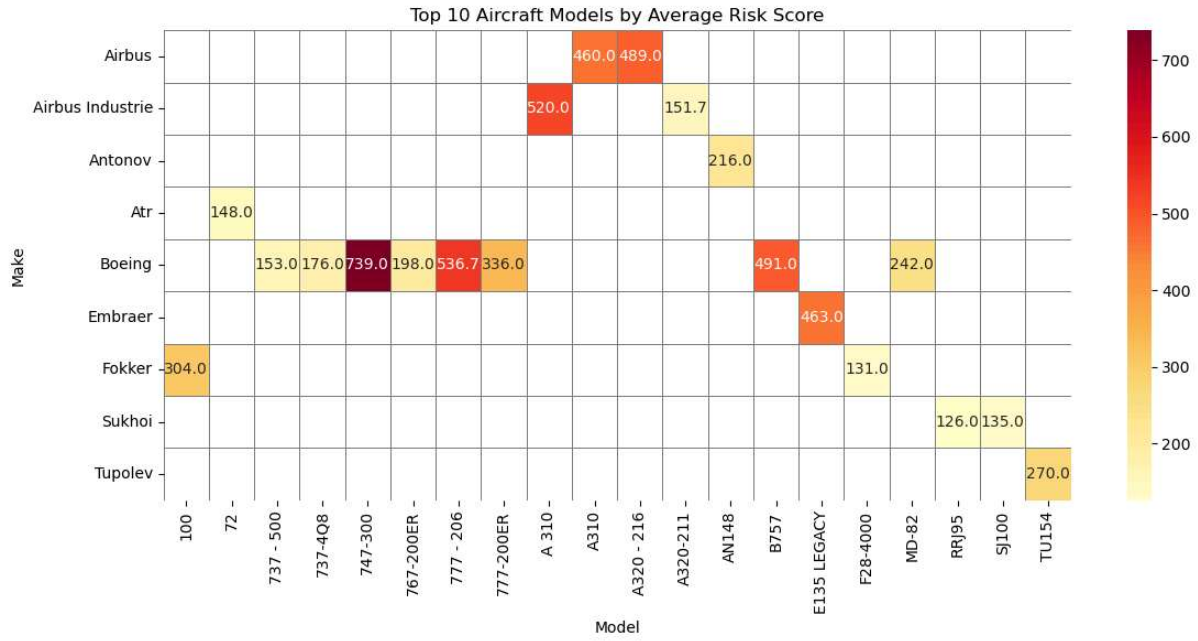
In [172]:
```
# Top 20 Less Riskiest Aircraft Models and Make (by Average Risk Score)
bottom_risk_models = risk_by_model.sort_values('Risk.Score', ascending=True).h
ead(20)

plt.figure(figsize=(10, 6))
plt.barh(
    bottom_risk_models['Make'] + ' ' + bottom_risk_models['Model'],
    bottom_risk_models['Risk.Score'],
    color='firebrick'
)
plt.xlabel('Average Risk Score')
plt.title('Bottom 20 Less Risky Aircraft Models')
plt.gca().invert_yaxis()  # highest at top
plt.tight_layout()
plt.show()
```



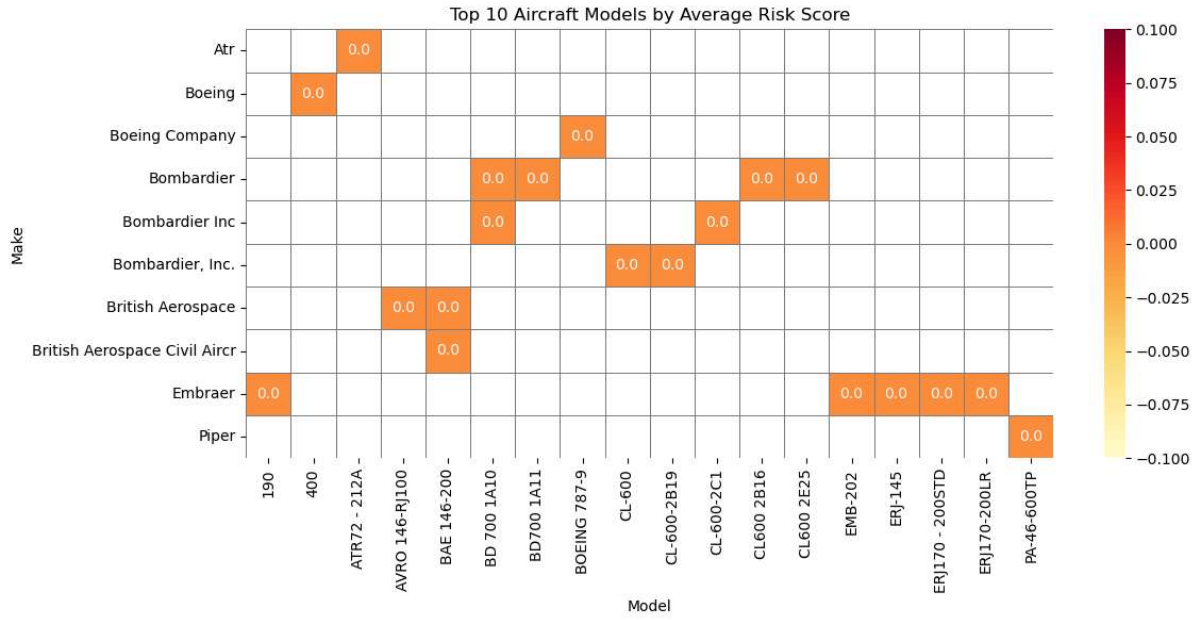
In [173]:
```
df.columns
```

Out[173]:
```
Index(['Event.Id', 'Investigation.Type', 'Event.Date', 'Location', 'Country',
       'Injury.Severity', 'Aircraft.damage', 'Aircraft.Category',
       'Airport.Name', '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', 'Total.Injuries',
       'Damage.Score', 'Risk.Score', 'Event.Year'],
      dtype='object')
```

In [174]:
```
#  top_20_risk by make and model
top_20_risk = risk_by_model.sort_values('Risk.Score', ascending=False).head(20)

# Pivot only the top 20 rows
pivot = top_20_risk.pivot_table(index='Make', columns='Model', values='Risk.Score')

plt.figure(figsize=(12, 6))
sns.heatmap(pivot, annot=True, cmap='YlOrRd', fmt='.1f', linewidths=0.5, linecolor='gray')
plt.title('Top 10 Aircraft Models by Average Risk Score')
plt.xlabel('Model')
plt.ylabel('Make')
plt.tight_layout()
plt.show()
```

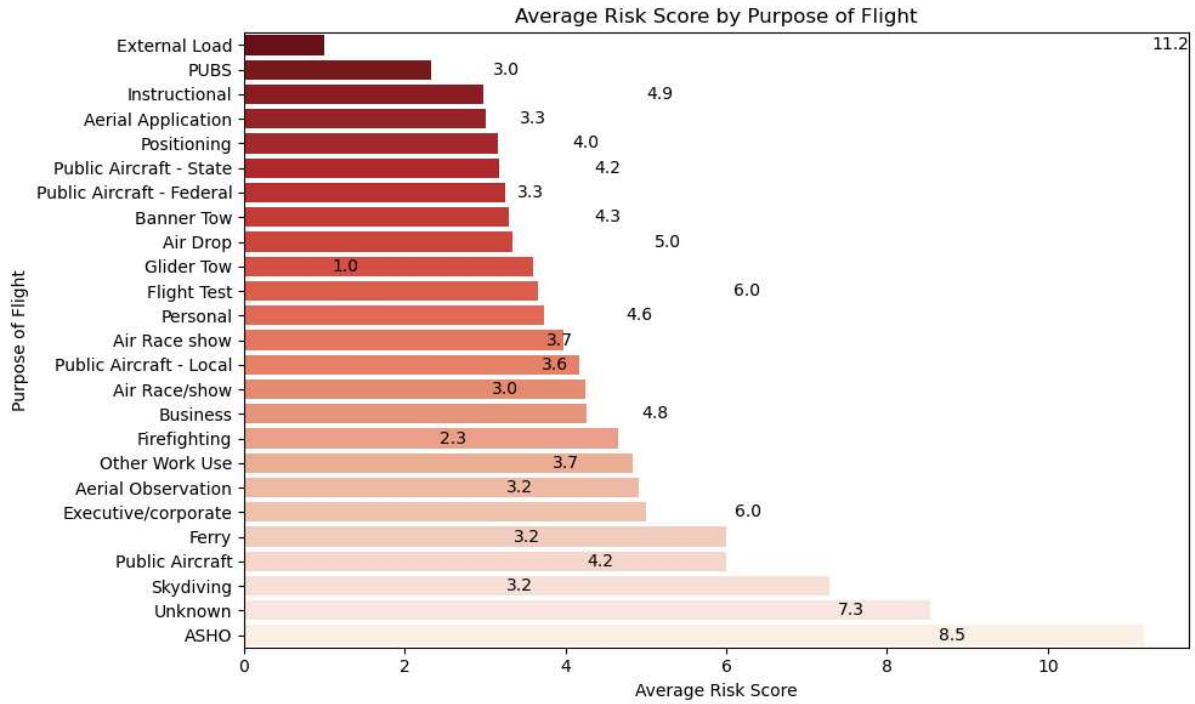


In [175]:
```
#  bottom_20_risk by make and model
bottom_20_risk = risk_by_model.sort_values('Risk.Score', ascending=True).head
(20)

# Pivot only the bottom 20 rows
pivot = bottom_20_risk.pivot_table(index='Make', columns='Model', values='Ris
k.Score')

plt.figure(figsize=(12, 6))
sns.heatmap(pivot, annot=True, cmap='YlOrRd', fmt='.1f', linewidths=0.5, linec
olor='gray')
plt.title('Top 10 Aircraft Models by Average Risk Score')
plt.xlabel('Model')
plt.ylabel('Make')
plt.tight_layout()
plt.show()
```



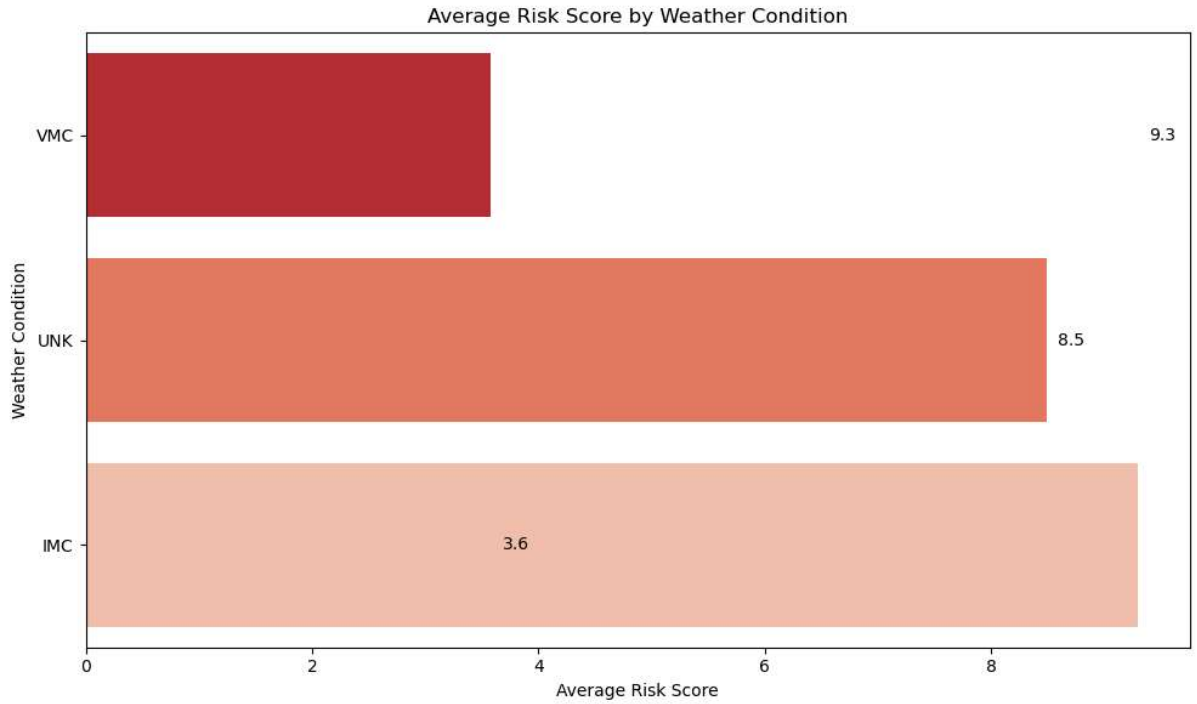
In [176]:
```
# Risk by purpose of flight
risk_by_purpose = df.groupby('Purpose.of.flight').agg({
    'Event.Id': 'count',
    'Total.Fatal.Injuries': 'sum',
    'Risk.Score': 'mean'
}).rename(columns={'Event.Id': 'Accident.Count'}).reset_index()
```

In [177]:
```
# Risk by Purpose of Flight Visual
risk_by_purpose = risk_by_purpose.sort_values('Risk.Score', ascending=True)

plt.figure(figsize=(10, 6))
sns.barplot(
    data=risk_by_purpose,
    x='Risk.Score',
    y='Purpose.of.flight',
    hue='Purpose.of.flight',
    dodge=False,
    palette='Reds_r',
    legend=False
)

# Add values on bars
for index, row in risk_by_purpose.iterrows():
    plt.text(row['Risk.Score'] + 0.1, index, f"{row['Risk.Score']:.1f}", va='c
enter')

plt.title('Average Risk Score by Purpose of Flight')
plt.xlabel('Average Risk Score')
plt.ylabel('Purpose of Flight')
plt.tight_layout()
plt.show()
```



In [178]:
```
# Risk by weather condition
risk_by_weather = df.groupby('Weather.Condition').agg({
    'Event.Id': 'count',
    'Total.Fatal.Injuries': 'sum',
    'Risk.Score': 'mean'
}).rename(columns={'Event.Id': 'Accident.Count'}).reset_index()
```

In [179]:
```
# Risk by Weather Condition Visual
risk_by_weather = risk_by_weather.sort_values('Risk.Score', ascending=True)

plt.figure(figsize=(10, 6))
sns.barplot(
    data=risk_by_weather,
    x='Risk.Score',
    y='Weather.Condition',
    hue='Weather.Condition',
    dodge=False,                    # Keep bars single (no separation)
    palette='Reds_r',
    legend=False                    # Avoid showing unnecessary legend
)

# Add values on bars
for index, row in risk_by_weather.iterrows():
    plt.text(row['Risk.Score'] + 0.1, index, f"{row['Risk.Score']:.1f}", va='c
enter')

plt.title('Average Risk Score by Weather Condition')
plt.xlabel('Average Risk Score')
plt.ylabel('Weather Condition')
plt.tight_layout()
plt.show()
```



Step 4: Conclusion

In [181]:
```
print("✅ Top 5 Lowest-Risk Aircraft Models:")
print(risk_by_model.sort_values('Risk.Score', ascending=True)[['Make', 'Mode
l', 'Total.Fatal.Injuries','Accident.Count','Risk.Score']].head())

print("\n✅ Risk by Purpose of Flight:")
print(risk_by_purpose.sort_values('Risk.Score', ascending=True)[['Purpose.of.f
light', 'Accident.Count', 'Total.Fatal.Injuries', 'Risk.Score']].head())

print("\n✅ Risk by Weather Condition:")
print(risk_by_weather.sort_values('Risk.Score', ascending=True)[['Weather.Cond
ition', 'Accident.Count', 'Total.Fatal.Injuries', 'Risk.Score']].head())
```

```
✅ Top 5 Lowest-Risk Aircraft Models:
                Make            Model  Total.Fatal.Injuries  Accident.Count
\
2325         Embraer     ERJ170-200LR                   0.0               1
1218      Bombardier       CL600 2E25                   0.0               1
1227  Bombardier Inc      BD 700 1A10                   0.0               1
2323         Embraer  ERJ170 - 200STD                   0.0               1
2321         Embraer          ERJ-145                   0.0               1

      Risk.Score
2325         0.0
1218         0.0
1227         0.0
2323         0.0
2321         0.0

✅ Risk by Purpose of Flight:
     Purpose.of.flight  Accident.Count  Total.Fatal.Injuries  Risk.Score
9        External Load               1                   0.0    1.000000
16                PUBS               3                   0.0    2.333333
14       Instructional            3120                 509.0    2.975641
1   Aerial Application            1066                 153.0    3.003752
18         Positioning             350                  99.0    3.160000

✅ Risk by Weather Condition:
  Weather.Condition  Accident.Count  Total.Fatal.Injuries  Risk.Score
2               VMC           19711                5515.0    3.579829
1               UNK            3343                6693.0    8.487885
0               IMC            1363                2781.0    9.292737
```

The data to use in Visualization in Tableau

In [183]:
```
df.to_csv('cleaned_aviation_data.csv', index=False)
```

Link to tableau visuals: https://public.tableau.com/app/profile/ann.wahu/viz/AviationAccidentsNTSB-Phase1/AviationRiskDashboard?publish=yes (https://public.tableau.com/app/profile/ann.wahu/viz/AviationAccidentsNTSB-Phase1/AviationRiskDashboard?publish=yes)