Phase 1 Project: Aviation Safety Analysis

Business Understanding

Objective

The purpose of this project is to identify low-risk aircraft models and operational strategies to support the company's entry into the aviation sector.

Key Deliverables

1. Recommendations for low-risk aircraft.
2. Insights into operational risks (e.g., flight phases, weather conditions).
3. Strategic guidance for improving safety in operations.

Data Understanding

Dataset Overview

- **Source**: National Transportation Safety Board (1962–2023)
- **Scope**: Aviation accidents and incidents across various models, flight phases, and weather conditions.

Initial Observations

- The dataset contained missing values, mixed data types, and columns with varying relevance to the analysis.
- Preprocessing was needed to clean and prepare the data for meaningful analysis.

In [2]:

```
# Import necessary libraries
import pandas as pd

# Load the dataset with the correct encoding
file_path = 'C:\\Users\\jakeg\\Phase_1_Project\\AviationData.csv'  # Upda
aviation_data = pd.read_csv(file_path, encoding='latin1')

# Display the first few rows
aviation_data.head()
```

```
C:\Users\jakeg\anaconda3\envs\learn-env\lib\site-packages\IPython\core\i
nteractiveshell.py:3145: DtypeWarning: Columns (6,7,28) have mixed type
s.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

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 [4]:

```
# Import necessary library
import pandas as pd

# Load the dataset with low_memory=False to handle mixed data types
file_path = "C:\\Users\\jakeg\\Phase_1_Project\\AviationData.csv"  # Repla
aviation_data = pd.read_csv(file_path, encoding="latin1", low_memory=False

# Display basic information about the dataset
aviation_data.info()

# Preview the first few rows
aviation_data.head()
```

```
<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
```

Out[4]:

	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 [5]:

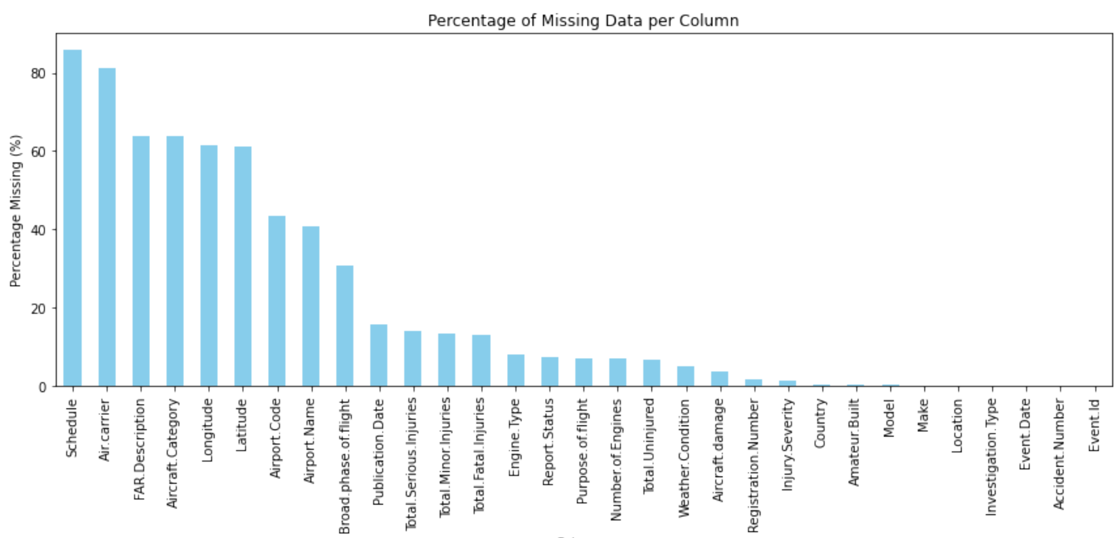
```
# Check for missing values in each column
missing_data = aviation_data.isnull().sum()

# Display missing values for each column
print("Missing values per column:\n", missing_data)

# Visualize missing data using a bar plot
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
missing_data.plot(kind='bar', color='skyblue')
plt.title("Missing Values per Column")
plt.ylabel("Number of Missing Values")
plt.xlabel("Columns")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

```
Missing values per column:
 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
```

In [6]:

```
# Calculate missing values per column
missing_data = aviation_data.isnull().sum()

# Calculate the percentage of missing data for each column
missing_percentage = (missing_data / len(aviation_data)) * 100

# Combine into a single DataFrame for better readability
missing_summary = pd.DataFrame({
    "Missing Values": missing_data,
    "Percentage Missing (%)": missing_percentage
}).sort_values(by="Percentage Missing (%)", ascending=False)

# Display the summary
print("Summary of Missing Data:")
print(missing_summary)

# Optional: Visualize missing data
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
missing_summary["Percentage Missing (%)"].plot(kind='bar', color='skyblue
plt.title("Percentage of Missing Data per Column")
plt.ylabel("Percentage Missing (%)")
plt.xlabel("Columns")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

```
Summary of Missing Data:
                        Missing Values  Percentage Missing (%)
Schedule                         76307               85.845268
Air.carrier                      72241               81.271023
FAR.Description                  56866               63.974170
Aircraft.Category                56602               63.677170
Longitude                        54516               61.330423
Latitude                         54507               61.320298
Airport.Code                     38640               43.469946
Airport.Name                     36099               40.611324
Broad.phase.of.flight            27165               30.560587
Publication.Date                 13771               15.492356
Total.Serious.Injuries           12510               14.073732
Total.Minor.Injuries             11933               13.424608
Total.Fatal.Injuries             11401               12.826109
Engine.Type                       7077                7.961615
Report.Status                     6381                7.178616
Purpose.of.flight                 6192                6.965991
Number.of.Engines                 6084                6.844491
Total.Uninjured                   5912                6.650992
Weather.Condition                 4492                5.053494
Aircraft.damage                   3194                3.593246
Registration.Number               1317                1.481623
Injury.Severity                   1000                1.124999
Country                            226                0.254250
Amateur.Built                      102                0.114750
Model                               92                0.103500
Make                                63                0.070875
Location                            52                0.058500
Investigation.Type                   0                0.000000
Event.Date                           0                0.000000
Accident.Number                      0                0.000000
Event.Id                             0                0.000000
```



In [7]:

```
# Function to summarize key columns
columns_to_check = ['Schedule', 'Air.carrier', 'FAR.Description', 'Aircra

for column in columns_to_check:
    print(f"\nColumn: {column}")
    print(aviation_data[column].value_counts(dropna=False).head(10))  # Sh
    print(f"Unique values: {aviation_data[column].nunique()}")
    print(f"Missing values: {aviation_data[column].isnull().sum()}")
```

```
Column: Schedule
NaN     76307
NSCH     4474
UNK      4099
SCHD     4009
Name: Schedule, dtype: int64
Unique values: 3
Missing values: 76307

Column: Air.carrier
NaN                      72241
Pilot                      258
American Airlines           90
United Airlines             89
Delta Air Lines             53
SOUTHWEST AIRLINES CO       42
DELTA AIR LINES INC         37
AMERICAN AIRLINES INC       29
ON FILE                     27
Continental Airlines        27
Name: Air.carrier, dtype: int64
Unique values: 13590
Missing values: 72241

Column: FAR.Description
NaN                          56866
091                          18221
Part 91: General Aviation     6486
NUSN                          1584
NUSC                          1013
137                           1010
135                            746
121                            679
Part 137: Agricultural         437
UNK                            371
Name: FAR.Description, dtype: int64
Unique values: 31
Missing values: 56866

Column: Aircraft.Category
NaN                  56602
Airplane             27617
Helicopter            3440
Glider                 508
Balloon                231
Gyrocraft              173
Weight-Shift           161
Powered Parachute       91
Ultralight              30
Unknown                 14
Name: Aircraft.Category, dtype: int64
Unique values: 15
Missing values: 56602

Column: Latitude
NaN          54507
```

```
332739N         19
335219N         18
32.815556       17
334118N         17
324934N         16
039405N         16
34.654444       15
393412N         14
391132N         14
Name: Latitude, dtype: int64
Unique values: 25589
Missing values: 54507

Column: Longitude
NaN            54516
0112457W          24
1114342W          18
1151140W          17
-104.673056       17
-112.0825         16
1114840W          16
-111.728334       15
-117.139444       15
0010572W          15
Name: Longitude, dtype: int64
Unique values: 27154
Missing values: 54516
```

In [3]:

```
# Reload the dataset
import pandas as pd

# Specify the file path to your dataset
file_path = "C:\\Users\\jakeg\\Phase_1_Project\\AviationData.csv"  # Repl

# Load the dataset with proper encoding and low_memory=False to handle mi
aviation_data = pd.read_csv(file_path, encoding="latin1", low_memory=False

# Verify the dataset is loaded
aviation_data.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 [4]:

```
# Drop less relevant columns with high missingness
columns_to_drop = ['Schedule', 'Air.carrier', 'Latitude', 'Longitude']
aviation_data_cleaned = aviation_data.drop(columns=columns_to_drop)

# Fill missing values for retained columns
aviation_data_cleaned['FAR.Description'].fillna('Unknown', inplace=True)
aviation_data_cleaned['Aircraft.Category'].fillna('Unknown', inplace=True)

# Fill injury-related columns with 0
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total
aviation_data_cleaned[injury_columns] = aviation_data_cleaned[injury_colur

# Impute categorical columns with mode or 'Unknown'
aviation_data_cleaned['Weather.Condition'].fillna(aviation_data_cleaned['W
aviation_data_cleaned['Broad.phase.of.flight'].fillna('Unknown', inplace=T

# Verify cleaning
aviation_data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 27 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   Airport.Code            50249 non-null  object 
 7   Airport.Name            52790 non-null  object 
 8   Injury.Severity         87889 non-null  object 
 9   Aircraft.damage         85695 non-null  object 
 10  Aircraft.Category       88889 non-null  object 
 11  Registration.Number     87572 non-null  object 
 12  Make                    88826 non-null  object 
 13  Model                   88797 non-null  object 
 14  Amateur.Built           88787 non-null  object 
 15  Number.of.Engines       82805 non-null  float64
 16  Engine.Type             81812 non-null  object 
 17  FAR.Description         88889 non-null  object 
 18  Purpose.of.flight       82697 non-null  object 
 19  Total.Fatal.Injuries    88889 non-null  float64
 20  Total.Serious.Injuries  88889 non-null  float64
 21  Total.Minor.Injuries    88889 non-null  float64
 22  Total.Uninjured         88889 non-null  float64
 23  Weather.Condition       88889 non-null  object 
 24  Broad.phase.of.flight   88889 non-null  object 
 25  Report.Status           82508 non-null  object 
 26  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(22)
memory usage: 18.3+ MB
```

Data Preparation

Data Cleaning Steps

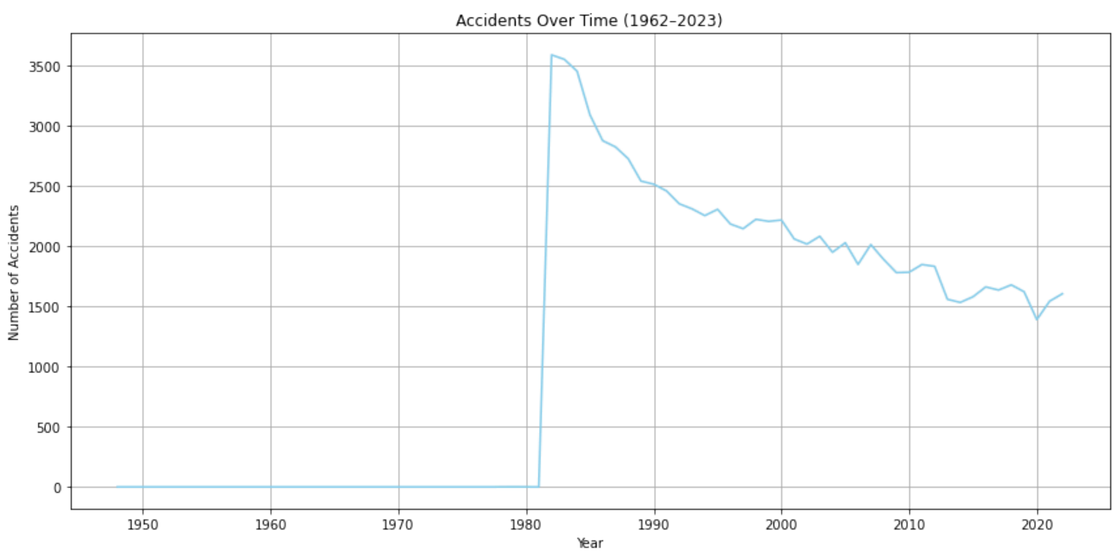
- **Handling Missing Data**:
  - Categorical fields (e.g., `FAR.Description` , `Aircraft.Category`) were imputed with "Unknown" to preserve rows for analysis.
  - Numeric fields related to injuries (e.g., `Total.Fatal.Injuries`) were imputed with 0.
- **Column Removal**:
  - Columns like `Latitude` , `Longitude` , and `Schedule` were removed due to high percentages of missing values and limited relevance.

Outcome

A clean and focused dataset enabling meaningful analysis of aircraft models, operational phases, and conditions.

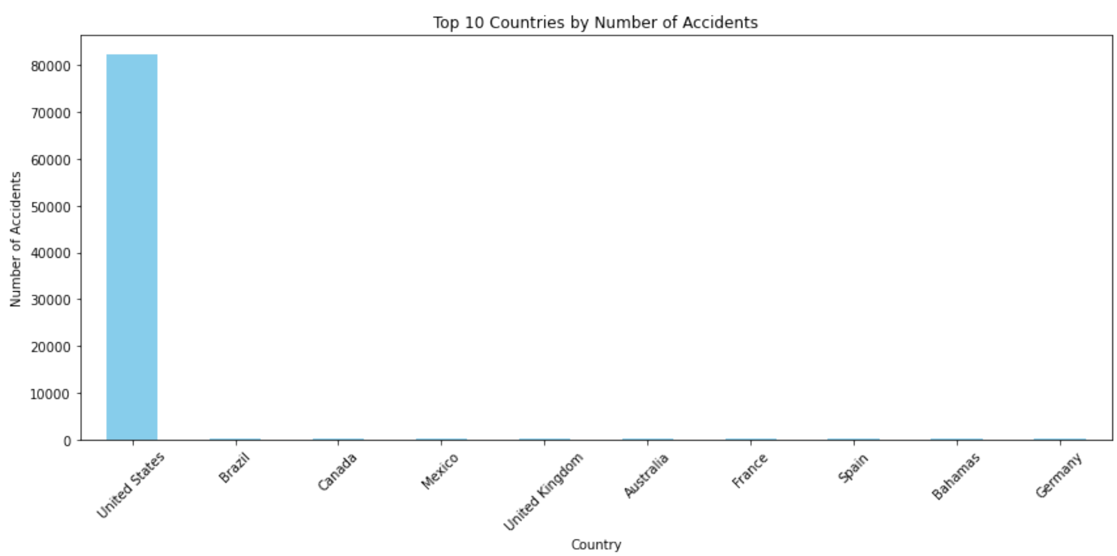
In [5]:

```
import matplotlib.pyplot as plt

# Convert Event.Date to datetime
aviation_data_cleaned['Event.Date'] = pd.to_datetime(aviation_data_cleaned

# Extract the year from Event.Date
aviation_data_cleaned['Year'] = aviation_data_cleaned['Event.Date'].dt.yea

# Group by year and count accidents
accidents_per_year = aviation_data_cleaned.groupby('Year').size()

# Plot accidents over time
plt.figure(figsize=(12, 6))
accidents_per_year.plot(kind='line', color='skyblue')
plt.title("Accidents Over Time (1962-2023)")
plt.xlabel("Year")
plt.ylabel("Number of Accidents")
plt.grid(True)
plt.tight_layout()
plt.show()
```



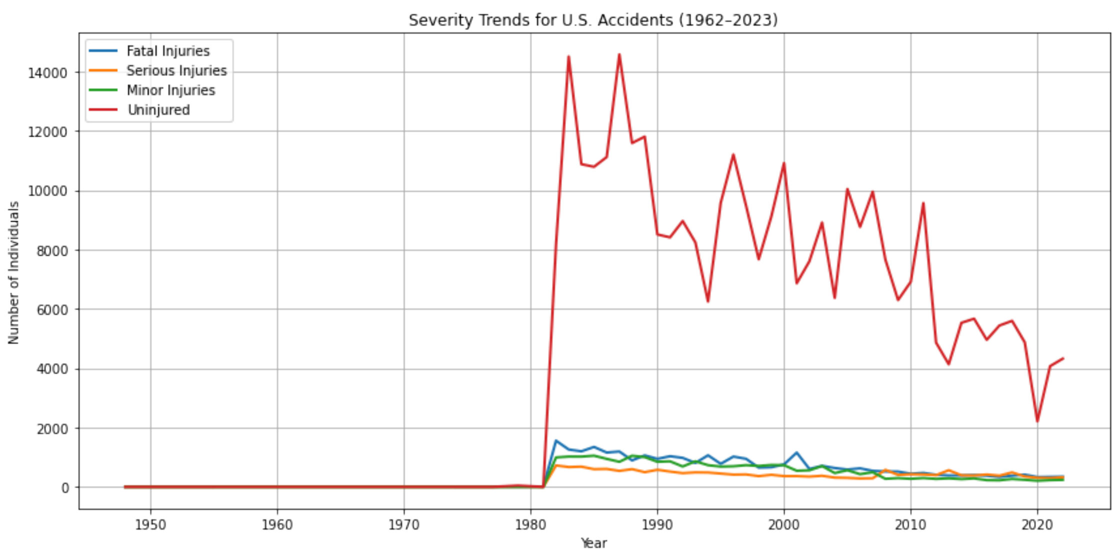
In [6]:
```
# Count accidents by country
accidents_by_country = aviation_data_cleaned['Country'].value_counts()

# Display the top 10 countries with the most accidents
top_countries = accidents_by_country.head(10)
print("Top 10 Countries by Number of Accidents:")
print(top_countries)

# Plot the top 10 countries
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
top_countries.plot(kind='bar', color='skyblue')
plt.title("Top 10 Countries by Number of Accidents")
plt.xlabel("Country")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top 10 Countries by Number of Accidents:
United States     82248
Brazil              374
Canada              359
Mexico              358
United Kingdom      344
Australia           300
France              236
Spain               226
Bahamas             216
Germany             215
Name: Country, dtype: int64
```



In [7]:

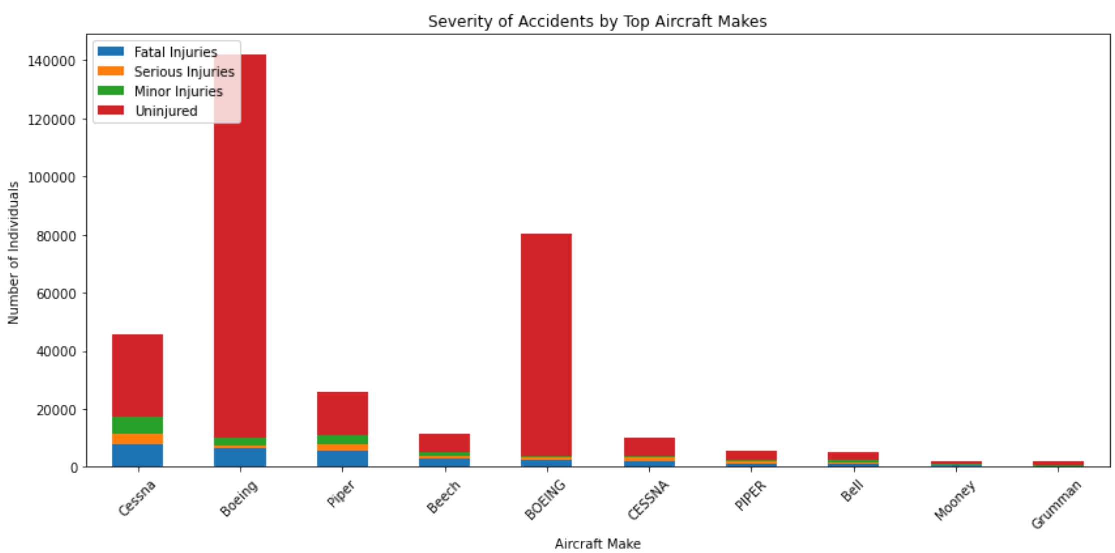
```
# Filter dataset for U.S. accidents
us_accidents = aviation_data_cleaned[aviation_data_cleaned['Country'] ==

# Group by year and sum injury counts
severity_trends = us_accidents.groupby('Year')[['Total.Fatal.Injuries', 'T

# Plot severity trends
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
severity_trends.plot(kind='line', ax=plt.gca(), linewidth=2)
plt.title("Severity Trends for U.S. Accidents (1962-2023)")
plt.xlabel("Year")
plt.ylabel("Number of Individuals")
plt.legend(["Fatal Injuries", "Serious Injuries", "Minor Injuries", "Uninj
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [8]:

```python
# Count accidents by aircraft make
make_counts = aviation_data_cleaned['Make'].value_counts().head(10)
print("Top 10 Aircraft Makes by Accident Count:")
print(make_counts)

# Group by Make and calculate severity statistics
make_severity = aviation_data_cleaned.groupby('Make')[['Total.Fatal.Injuri

# Filter for the top 10 makes and sort by fatal injuries
top_make_severity = make_severity.loc[make_counts.index].sort_values(by='

# Visualize severity for top aircraft makes
top_make_severity.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Severity of Accidents by Top Aircraft Makes")
plt.xlabel("Aircraft Make")
plt.ylabel("Number of Individuals")
plt.legend(["Fatal Injuries", "Serious Injuries", "Minor Injuries", "Uninj
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top 10 Aircraft Makes by Accident Count:
Cessna     22227
Piper      12029
CESSNA      4922
Beech       4330
PIPER       2841
Bell        2134
Boeing      1594
BOEING      1151
Grumman     1094
Mooney      1092
Name: Make, dtype: int64
```



Data Analysis

Aircraft Model Safety Analysis

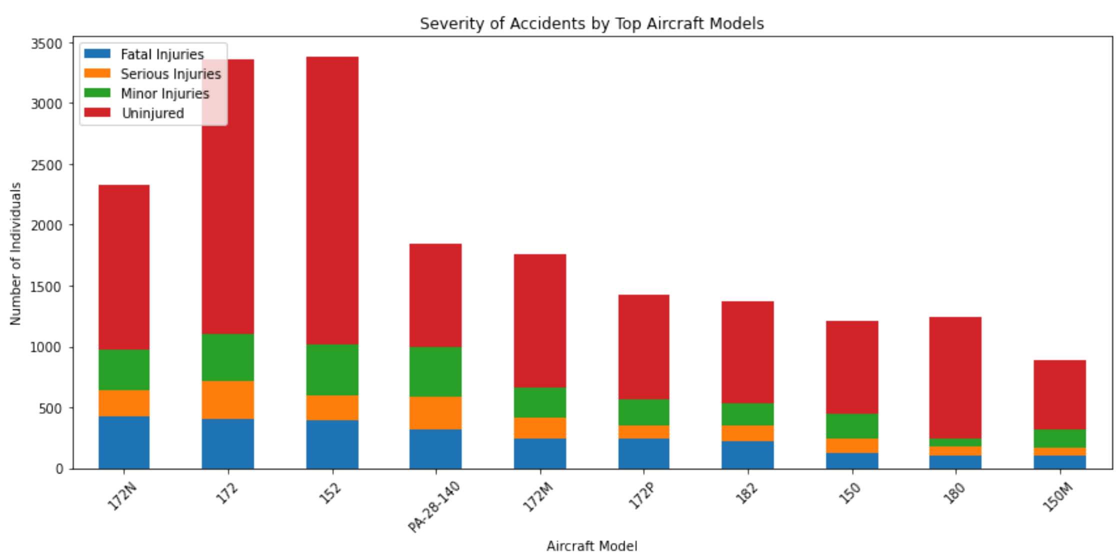
The following steps were conducted to evaluate the safety performance of aircraft models:

- Counted accidents for each model to identify the most frequently involved.
- Analyzed the severity of accidents (fatalities, serious injuries, etc.) for the top models.

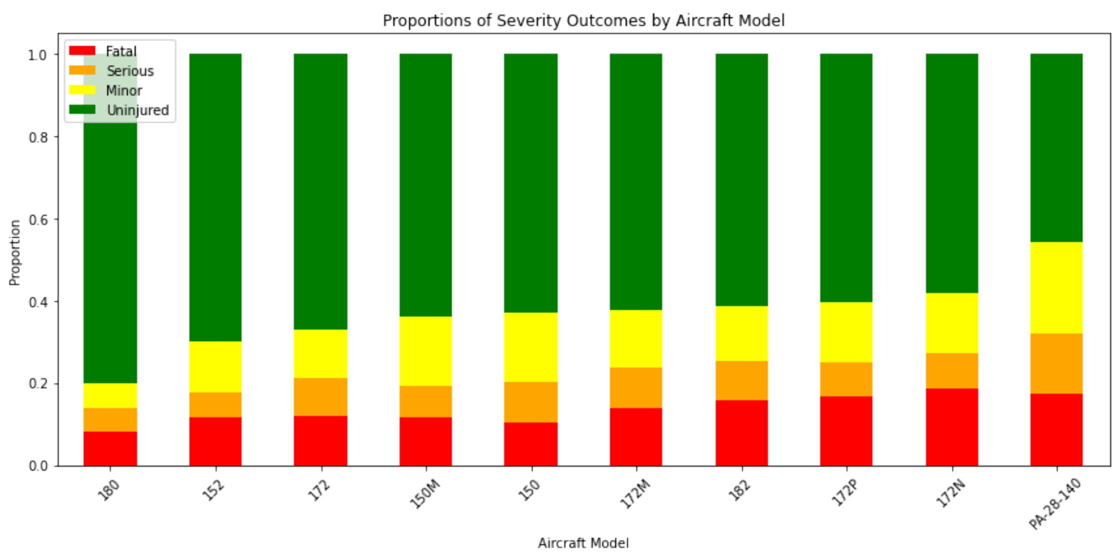
In [9]:

```
# Count accidents by aircraft model
model_counts = aviation_data_cleaned['Model'].value_counts().head(10)
print("Top 10 Aircraft Models by Accident Count:")
print(model_counts)

# Group by Model and calculate severity statistics
model_severity = aviation_data_cleaned.groupby('Model')[['Total.Fatal.Inju

# Filter for the top 10 models and sort by fatal injuries
top_model_severity = model_severity.loc[model_counts.index].sort_values(by

# Visualize severity for top aircraft models
top_model_severity.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Severity of Accidents by Top Aircraft Models")
plt.xlabel("Aircraft Model")
plt.ylabel("Number of Individuals")
plt.legend(["Fatal Injuries", "Serious Injuries", "Minor Injuries", "Uninj
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top 10 Aircraft Models by Accident Count:
152          2367
172          1756
172N         1164
PA-28-140     932
150           829
172M          798
172P          689
182           659
180           622
150M          585
Name: Model, dtype: int64
```



In [10]:
```
# Calculate total individuals involved in accidents for each model
model_severity['Total.Individuals'] = (
    model_severity['Total.Fatal.Injuries'] +
    model_severity['Total.Serious.Injuries'] +
    model_severity['Total.Minor.Injuries'] +
    model_severity['Total.Uninjured']
)

# Calculate proportions for each severity category
model_severity['Fatal.Proportion'] = model_severity['Total.Fatal.Injuries'
model_severity['Serious.Proportion'] = model_severity['Total.Serious.Injur
model_severity['Minor.Proportion'] = model_severity['Total.Minor.Injuries'
model_severity['Uninjured.Proportion'] = model_severity['Total.Uninjured']

# Filter for the top 10 models
top_model_proportions = model_severity.loc[model_counts.index]

# Sort by Uninjured.Proportion
top_model_proportions = top_model_proportions.sort_values(by='Uninjured.Pr

# Visualize proportions for the top models
top_model_proportions[['Fatal.Proportion', 'Serious.Proportion', 'Minor.Pr
    kind='bar', stacked=True, figsize=(12, 6), color=['red', 'orange', 'ye
)
plt.title("Proportions of Severity Outcomes by Aircraft Model")
plt.xlabel("Aircraft Model")
plt.ylabel("Proportion")
plt.xticks(rotation=45)
plt.legend(["Fatal", "Serious", "Minor", "Uninjured"], loc="upper left")
plt.tight_layout()
plt.show()
```



Flight Phase Risks

Objective

To identify the most high-risk phases of flight, we analyzed the frequency and severity of accidents across different flight phases.

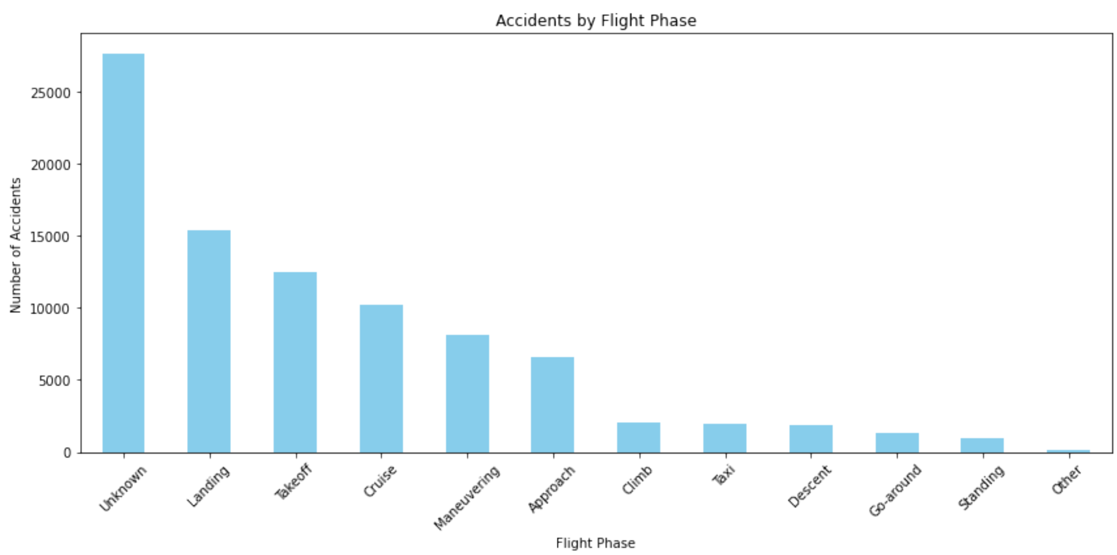
Analysis Steps

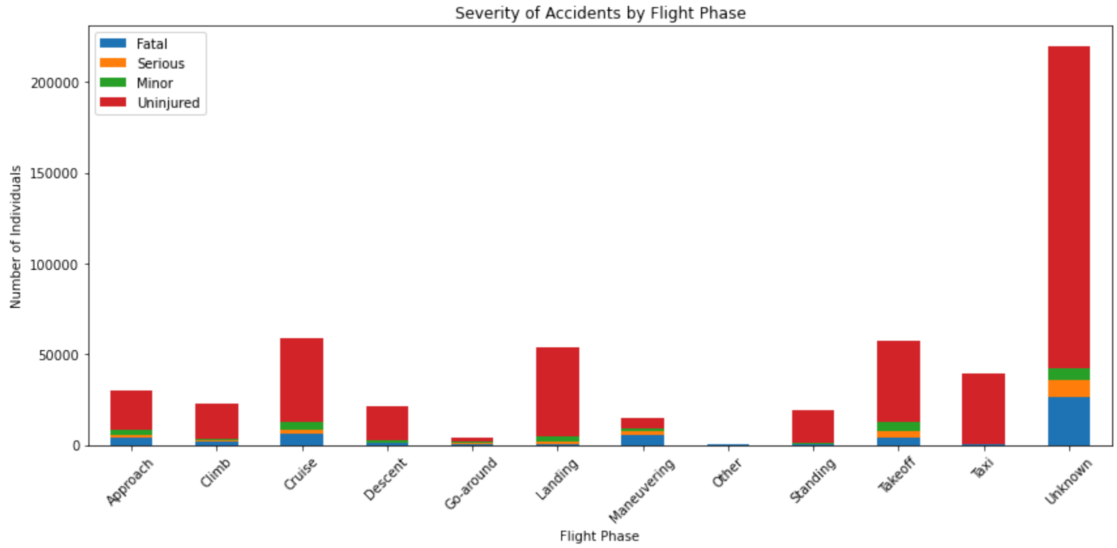
1. Grouped accidents by the `Broad.phase.of.flight` column to calculate the total accidents for each phase.
2. Analyzed the proportions of fatal, serious, minor injuries, and uninjured outcomes in each flight phase.
3. Visualized the results to highlight the most critical phases (e.g., landing, takeoff).

Key Questions

- Which phases of flight (e.g., landing, takeoff, cruise) have the highest frequency of accidents?
- How severe are the accidents in each phase, and where should safety measures be

In [11]:
```
# Count accidents by flight phase
flight_phase_counts = aviation_data_cleaned['Broad.phase.of.flight'].value

# Plot accidents by flight phase
flight_phase_counts.plot(kind='bar', figsize=(12, 6), color='skyblue')
plt.title("Accidents by Flight Phase")
plt.xlabel("Flight Phase")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Analyze severity by flight phase
flight_phase_severity = aviation_data_cleaned.groupby('Broad.phase.of.flig
    ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuri
].sum()

# Plot severity distribution by flight phase
flight_phase_severity.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Severity of Accidents by Flight Phase")
plt.xlabel("Flight Phase")
plt.ylabel("Number of Individuals")
plt.legend(["Fatal", "Serious", "Minor", "Uninjured"], loc="upper left")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```





Weather Impact on Safety

Objective

To understand how weather conditions (VMC: Visual Meteorological Conditions vs. IMC: Instrument Meteorological Conditions) impact accident severity and safety outcomes.

Analysis Steps

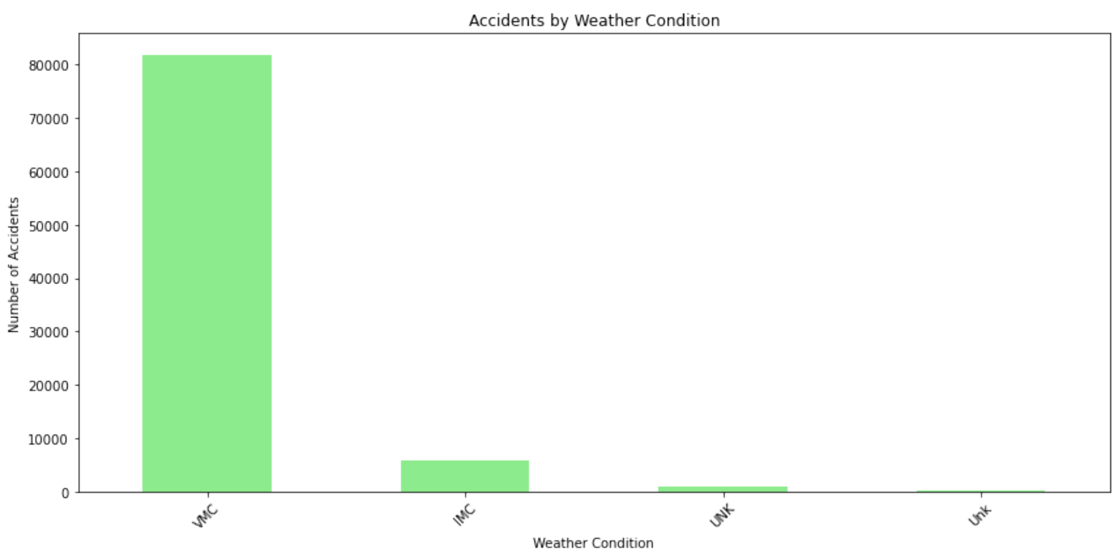
1. Categorized accidents by the `Weather.Condition` column into VMC, IMC, and unknown conditions.
2. Calculated the proportions of uninjured individuals and injury severities under each weather condition.
3. Compared safety outcomes for the top aircraft models in both VMC and IMC.

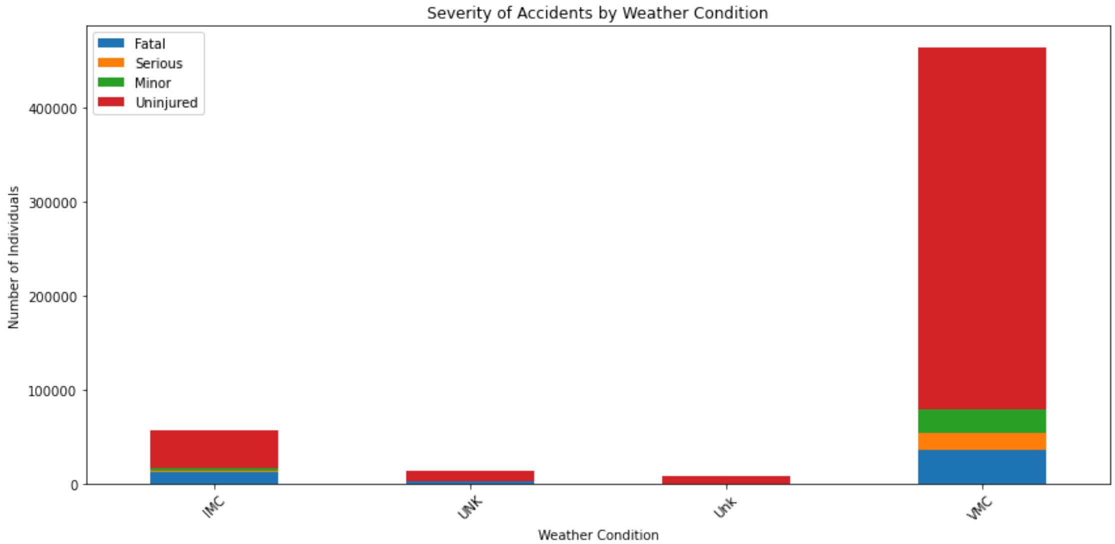
Key Questions

- How do safety outcomes differ under VMC and IMC conditions?
- Which aircraft models perform better under adverse (IMC) conditions, and how can this guide aircraft selection?

In [12]:

```
# Count accidents by weather condition
weather_counts = aviation_data_cleaned['Weather.Condition'].value_counts()

# Plot accidents by weather condition
weather_counts.plot(kind='bar', figsize=(12, 6), color='lightgreen')
plt.title("Accidents by Weather Condition")
plt.xlabel("Weather Condition")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Analyze severity by weather condition
weather_severity = aviation_data_cleaned.groupby('Weather.Condition')[
    ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuri
].sum()

# Plot severity distribution by weather condition
weather_severity.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Severity of Accidents by Weather Condition")
plt.xlabel("Weather Condition")
plt.ylabel("Number of Individuals")
plt.legend(["Fatal", "Serious", "Minor", "Uninjured"], loc="upper left")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```





Flight Phase Analysis for Top Models

Objective

To evaluate how the top-performing aircraft models perform during the most dangerous flight phases: landing and takeoff.

Analysis Steps

1. Filtered the dataset for the top models (e.g., Cessna 172M, 172P, etc.).
2. Segmented the data by `Broad.phase.of.flight` to focus on landing and takeoff phases.
3. Calculated and visualized injury proportions (fatal, serious, minor, and uninjured) for each top model during these critical phases.

Key Questions

- Which top models exhibit the best safety performance during landing and takeoff?
- Are there specific models that perform significantly worse during these phases?

In [13]:
```
# Filter dataset for critical phases (Landing, Takeoff)
critical_phases = ['Landing', 'Takeoff']
phase_filtered = aviation_data_cleaned[aviation_data_cleaned['Broad.phase.

# Count accidents by model and flight phase
phase_model_counts = phase_filtered.groupby(['Model', 'Broad.phase.of.flig

# Select top models for analysis
top_models = model_counts.head(10).index
phase_model_counts = phase_model_counts.loc[top_models]

# Plot accidents by model and phase
phase_model_counts.plot(kind='bar', stacked=True, figsize=(12, 6), color=[
plt.title("Accidents by Aircraft Model and Flight Phase (Landing vs. Takeo
plt.xlabel("Aircraft Model")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.legend(["Landing", "Takeoff"], loc="upper left")
plt.tight_layout()
plt.show()
```

Flight Phase Analysis Results

Key Findings

1. **Landing**:

   - The majority of accidents occur during the landing phase.
   - **Cessna 172M** consistently exhibits higher uninjured proportions compared to other top models during landing.

2. **Takeoff**:

   - Takeoff is also a high-risk phase, but accidents here tend to be less frequent compared to landing.
   - **Cessna 172P** shows strong safety performance in takeoff-related accidents.

Implications

- Training programs and safety protocols should prioritize landing and takeoff operations.
- Models like **Cessna 172M** and **Cessna 172P** can be recommended for their reliability during critical phases.

Weather Condition Analysis for Top Models

Objective

To analyze how top-performing aircraft models perform under different weather conditions: VMC (good visibility) and IMC (poor visibility).

Analysis Steps

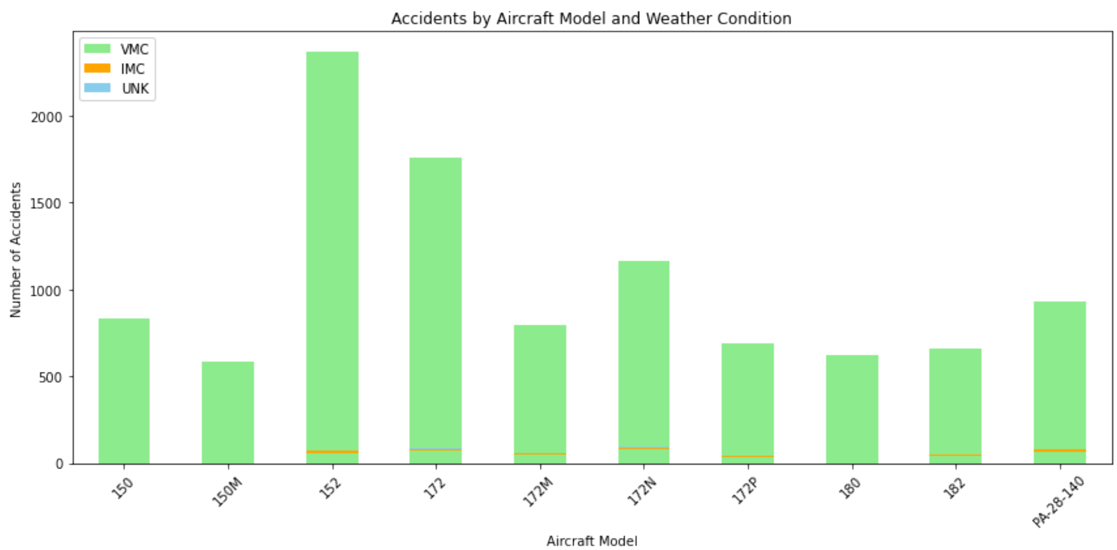
1. Filtered the dataset for the top models.
2. Segmented the data by `Weather.Condition` to categorize accidents into VMC and IMC.
3. Calculated and visualized injury proportions (fatal, serious, minor, and uninjured) for each top model under these conditions.

Key Questions

- Which models maintain good safety performance under IMC conditions (poor weather)?
- Are there any models that show significant safety risks in IMC compared to VMC?

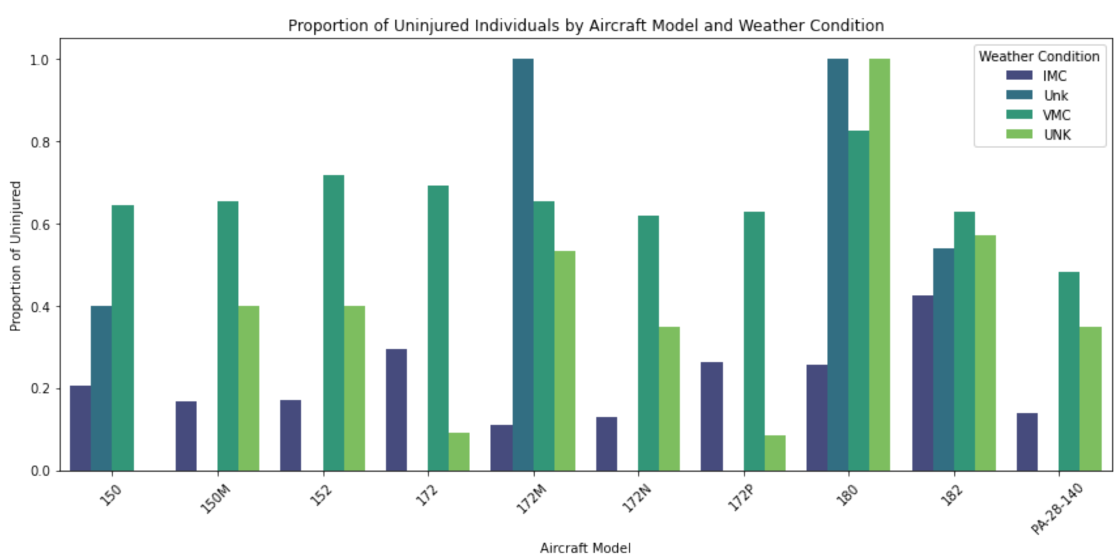
In [14]:

```
# Filter for top models
top_models_weather = aviation_data_cleaned[aviation_data_cleaned['Model'].

# Group by Model and Weather Condition
weather_model_counts = top_models_weather.groupby(['Model', 'Weather.Condi

# Plot accidents by model and weather condition
weather_model_counts.plot(kind='bar', stacked=True, figsize=(12, 6), color
plt.title("Accidents by Aircraft Model and Weather Condition")
plt.xlabel("Aircraft Model")
plt.ylabel("Number of Accidents")
plt.xticks(rotation=45)
plt.legend(["VMC", "IMC", "UNK"], loc="upper left")
plt.tight_layout()
plt.show()

# Analyze severity by model and weather condition
weather_severity = top_models_weather.groupby(['Model', 'Weather.Conditior
    ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuri
].sum()

# Optional: Add proportions for deeper insights
```



In [15]:

```
# Filter for top models
top_models_weather = aviation_data_cleaned[aviation_data_cleaned['Model'].

# Group by Model and Weather Condition to calculate severity totals
weather_severity = top_models_weather.groupby(['Model', 'Weather.Conditio
    ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuri
].sum()

# Calculate total individuals involved in accidents
weather_severity['Total.Individuals'] = (
    weather_severity['Total.Fatal.Injuries'] +
    weather_severity['Total.Serious.Injuries'] +
    weather_severity['Total.Minor.Injuries'] +
    weather_severity['Total.Uninjured']
)

# Calculate proportions for each severity category
weather_severity['Fatal.Proportion'] = weather_severity['Total.Fatal.Injur
weather_severity['Serious.Proportion'] = weather_severity['Total.Serious.I
weather_severity['Minor.Proportion'] = weather_severity['Total.Minor.Injur
weather_severity['Uninjured.Proportion'] = weather_severity['Total.Uninjur

# Reset index for easier plotting
weather_severity = weather_severity.reset_index()

# Plot proportions by weather condition and model
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.barplot(
    x='Model', y='Uninjured.Proportion', hue='Weather.Condition', data=wea
)
plt.title("Proportion of Uninjured Individuals by Aircraft Model and Weath
plt.xlabel("Aircraft Model")
plt.ylabel("Proportion of Uninjured")
plt.xticks(rotation=45)
plt.legend(title="Weather Condition")
plt.tight_layout()
plt.show()
```



Weather Condition Analysis Results

Key Findings

1. **Visual Meteorological Conditions (VMC)**:
    - Most accidents occur under VMC (clear weather), which suggests that pilot error or other non-weather-related factors are major contributors.
2. **Instrument Meteorological Conditions (IMC)**:
    - Accidents under IMC (poor visibility) are less frequent but more severe, with higher proportions of fatalities and serious injuries.
    - **Cessna 172M** demonstrates strong performance under IMC, making it a suitable choice for operations in adverse weather.

Implications

- Aircraft equipped with advanced avionics and training for IMC conditions are critical for mitigating risks.
- Models like **Cessna 172M** and **Cessna 172P** are preferred for their resilience in poor weather conditions.

In [16]:

```
# Select relevant columns for the dashboard
dashboard_data = aviation_data_cleaned[
    ['Model', 'Broad.phase.of.flight', 'Weather.Condition', 'Total.Fatal.I
     'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
]

# Save the cleaned dataset to a CSV file
file_path = 'aviation_dashboard_data.csv'
dashboard_data.to_csv(file_path, index=False)

print(f"Dashboard data has been saved to {file_path}")
```

```
Dashboard data has been saved to aviation_dashboard_data.csv
```

In [18]:
```
import pandas as pd

# Paths to your datasets
original_data_path = "C:\\Users\\jakeg\\Phase_1_Project\\AviationData.csv'
cleaned_data_path = "C:\\Users\\jakeg\\aviation_dashboard_data.csv"  # Upd

# Load the original and cleaned datasets
original_data = pd.read_csv(original_data_path, encoding='latin1')
aviation_data_cleaned = pd.read_csv(cleaned_data_path)

print("Datasets successfully loaded.")
```

```
C:\Users\jakeg\anaconda3\envs\learn-env\lib\site-packages\IPython\core\i
nteractiveshell.py:3145: DtypeWarning: Columns (6,7,28) have mixed type
s.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,

Datasets successfully loaded.
```

In [19]:
```
# Replace the injury columns in the exported dataset with original values
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Tota

# Restore original injury totals
aviation_data_cleaned[injury_columns] = original_data[injury_columns]

# Fill any missing values with 0
aviation_data_cleaned[injury_columns] = aviation_data_cleaned[injury_colu

# Save the corrected dataset
aviation_data_cleaned.to_csv('aviation_dashboard_data_corrected.csv', inde
print("Corrected dataset saved as 'aviation_dashboard_data_corrected.csv''
```

```
Corrected dataset saved as 'aviation_dashboard_data_corrected.csv'
```

In [2]:
```
import pandas as pd

# Load the cleaned dataset
aviation_data_cleaned = pd.read_csv('aviation_dashboard_data_corrected.csv
print("Cleaned dataset loaded successfully.")
```

```
Cleaned dataset loaded successfully.
```

In [4]:
```
# Display all column names in the dataset
print(aviation_data_cleaned.columns)
```

```
Index(['Model', 'Broad.phase.of.flight', 'Weather.Condition',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Year'],
      dtype='object')
```

In [8]:

```
# Load the original dataset if not already loaded
import pandas as pd

original_data_path = "C:\\Users\\jakeg\\Phase_1_Project\\AviationData.csv'
original_data = pd.read_csv(original_data_path, encoding='latin1')

# Check if the 'Make' column exists
print(original_data.columns)
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Dat
e',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descrip
tion',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Inju
ries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')

C:\Users\jakeg\anaconda3\envs\learn-env\lib\site-packages\IPython\core\i
nteractiveshell.py:3145: DtypeWarning: Columns (6,7,28) have mixed type
s.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [9]:

```
# Add the 'Make' column back to the cleaned dataset
aviation_data_cleaned['Make'] = original_data['Make']

# Standardize the 'Make' column to title case for consistency
aviation_data_cleaned['Make'] = aviation_data_cleaned['Make'].str.title()

# Verify the changes
print(aviation_data_cleaned['Make'].value_counts())
```

```
Cessna                  27149
Piper                   14870
Beech                    5372
Boeing                   2745
Bell                     2722
                        ...
Hanek                       1
Turcotte Jr Robert L        1
Bell-Cont 42G               1
Gilchrist                   1
Noakes B J                  1
Name: Make, Length: 7587, dtype: int64
```

In [10]:

```
# Save the updated dataset with the 'Make' column
aviation_data_cleaned.to_csv('aviation_dashboard_data_updated.csv', index=
print("Updated dataset saved as 'aviation_dashboard_data_updated.csv'")
```

```
Updated dataset saved as 'aviation_dashboard_data_updated.csv'
```

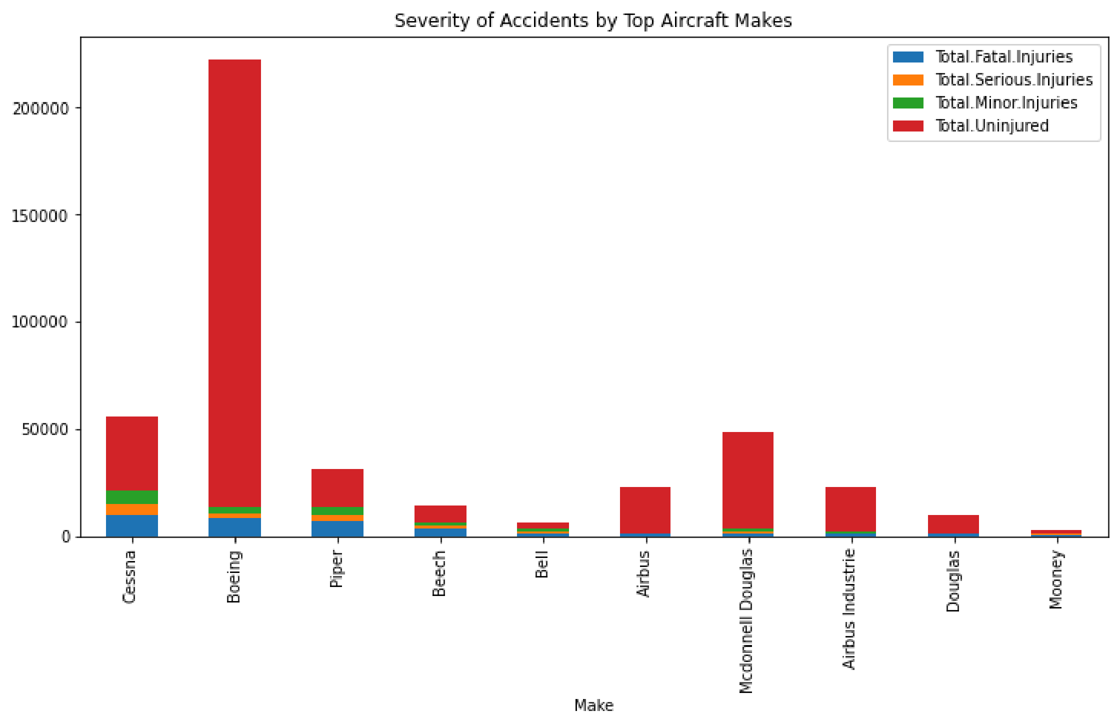
In [11]:

```
# Group by 'Make' and calculate total severity counts
severity_by_make = aviation_data_cleaned.groupby('Make')[[
    'Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injurie
]].sum().sort_values(by='Total.Fatal.Injuries', ascending=False)

# Display the top 10 aircraft makes by fatalities
print(severity_by_make.head(10))
```

```
                   Total.Fatal.Injuries  Total.Serious.Injuries  \
Make
Cessna                           9641.0                  4894.0
Boeing                           8748.0                  2157.0
Piper                            6689.0                  3059.0
Beech                            3784.0                  1095.0
Bell                             1332.0                   878.0
Airbus                           1325.0                   192.0
Mcdonnell Douglas                1286.0                   556.0
Airbus Industrie                 1174.0                   138.0
Douglas                           984.0                   105.0
Mooney                            685.0                   248.0

                   Total.Minor.Injuries  Total.Uninjured
Make
Cessna                           6876.0          34423.0
Boeing                           2761.0         208375.0
Piper                            3757.0          17832.0
Beech                            1341.0           7891.0
Bell                             1122.0           3072.0
Airbus                            106.0          21261.0
Mcdonnell Douglas                1505.0          45102.0
Airbus Industrie                  399.0          21261.0
Douglas                           247.0           8805.0
Mooney                            391.0           1303.0
```

In [12]:
```
# Visualize the severity of accidents for the top 10 makes
severity_by_make.head(10).plot(
    kind='bar', stacked=True, figsize=(12, 6),
    title='Severity of Accidents by Top Aircraft Makes'
)
```

Out[12]:
```
<AxesSubplot:title={'center':'Severity of Accidents by Top Aircraft Make
s'}, xlabel='Make'>
```



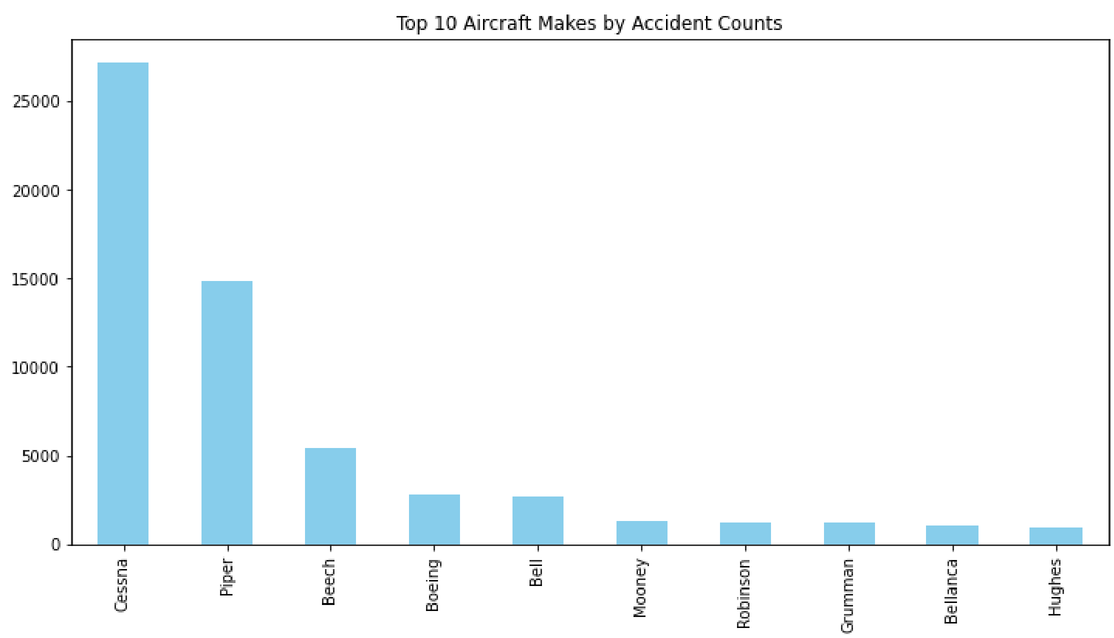
In [13]:

```
# Calculate total accidents for each make
accident_counts_by_make = aviation_data_cleaned['Make'].value_counts()

# Display the top 10 makes by accident counts
print(accident_counts_by_make.head(10))

# Visualize the top 10 makes by accident counts
accident_counts_by_make.head(10).plot(
    kind='bar', figsize=(12, 6), color='skyblue',
    title='Top 10 Aircraft Makes by Accident Counts'
)
```

```
Cessna      27149
Piper       14870
Beech        5372
Boeing       2745
Bell         2722
Mooney       1334
Robinson     1230
Grumman      1172
Bellanca     1045
Hughes        932
Name: Make, dtype: int64
```

Out[13]:

```
<AxesSubplot:title={'center':'Top 10 Aircraft Makes by Accident Counts'}
>
```



In [15]:

```
# Check the columns in the cleaned dataset
print(aviation_data_cleaned.columns)
```

```
Index(['Model', 'Broad.phase.of.flight', 'Weather.Condition',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Year', 'Make'],
      dtype='object')
```

In [16]:
```
# Add 'Aircraft.Category' back to the cleaned dataset
aviation_data_cleaned['Aircraft.Category'] = original_data['Aircraft.Categ

# Verify the reintroduced column
print(aviation_data_cleaned['Aircraft.Category'].unique())
```

```
[nan 'Airplane' 'Helicopter' 'Glider' 'Balloon' 'Gyrocraft' 'Ultralight'
 'Unknown' 'Blimp' 'Powered-Lift' 'Weight-Shift' 'Powered Parachute'
 'Rocket' 'WSFT' 'UNK' 'ULTR']
```

In [17]:
```
# Check the relationship between 'Model' and 'Aircraft.Category'
model_category_check = original_data[['Model', 'Aircraft.Category']].drop_
print(model_category_check[model_category_check['Model'].isin(['172M', '17
```

```
     Model Aircraft.Category
2     172M               NaN
54    172M          Airplane
204   172P          Airplane
3786  172P               NaN
```

In [18]:
```
# Verify the 'Aircraft.Category' for recommended and cautioned models
models_to_check = ['172M', '172P', '152']
model_category_check = original_data[['Model', 'Aircraft.Category']].drop_
print(model_category_check[model_category_check['Model'].isin(models_to_ch
```

```
     Model Aircraft.Category
2     172M               NaN
20     152          Airplane
54    172M          Airplane
204   172P          Airplane
3592   152               NaN
3786  172P               NaN
```

US vs. International Flights Analysis

Objective

To compare safety performance and accident trends for US flights and international flights, identifying models and operational conditions best suited for each setting.

In [21]:
```
# Try reading the file with a different encoding
original_data = pd.read_csv("C:\\Users\\jakeg\\Phase_1_Project\\AviationDa

# If 'latin1' doesn't work, try 'iso-8859-1' or 'cp1252'
# original_data = pd.read_csv("C:\\Users\\jakeg\\Phase_1_Project\\Aviation
# original_data = pd.read_csv("C:\\Users\\jakeg\\Phase_1_Project\\Aviation
```

```
C:\Users\jakeg\anaconda3\envs\learn-env\lib\site-packages\IPython\core\i
nteractiveshell.py:3145: DtypeWarning: Columns (6,7,28) have mixed type
s.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [26]:
```
# Check if the 'Country' column exists in the original dataset
print("Columns in original_data:")
print(original_data.columns)
```

```
Columns in original_data:
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Dat
e',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descrip
tion',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Inju
ries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [27]:
```
# Add 'Country' column from original_data to aviation_data_cleaned based c
aviation_data_cleaned['Country'] = original_data['Country']
```

In [28]:
```
print("Columns in aviation_data_cleaned after adding 'Country':")
print(aviation_data_cleaned.columns)
```

```
Columns in aviation_data_cleaned after adding 'Country':
Index(['Model', 'Broad.phase.of.flight', 'Weather.Condition',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Year', 'Make',
       'Aircraft.Category', 'Country'],
      dtype='object')
```

In [29]:
```
# Segment the data into US and international flights
us_flights = aviation_data_cleaned[aviation_data_cleaned['Country'] == "Ur
international_flights = aviation_data_cleaned[aviation_data_cleaned['Count

# Verify the segmentation
print(f"Number of US flights: {us_flights.shape[0]}")
print(f"Number of international flights: {international_flights.shape[0]}'
```

```
Number of US flights: 82248
Number of international flights: 6641
```

In [30]:

```
# Check the distribution of accidents by country in each subset
print("Unique countries in US flights subset:")
print(us_flights['Country'].unique())

print("\nUnique countries in international flights subset:")
print(international_flights['Country'].unique())
```

```
Unique countries in US flights subset:
['United States']

Unique countries in international flights subset:
[nan 'GULF OF MEXICO' 'Puerto Rico' 'ATLANTIC OCEAN' 'HIGH ISLAND'
 'Bahamas' 'MISSING' 'Pakistan' 'Angola' 'Germany' 'Korea, Republic Of'
 'Martinique' 'American Samoa' 'PACIFIC OCEAN' 'Canada' 'Bolivia' 'Mexic
o'
 'Dominica' 'Netherlands Antilles' 'Iceland' 'Greece' 'Guam' 'Australia'
 'CARIBBEAN SEA' 'West Indies' 'Japan' 'Philippines' 'Venezuela' 'Bermud
a'
 'San Juan Islands' 'Colombia' 'El Salvador' 'United Kingdom'
 'British Virgin Islands' 'Netherlands' 'Costa Rica' 'Mozambique'
 'Jamaica' 'Panama' 'Guyana' 'Norway' 'Hong Kong' 'Portugal' 'Malaysia'
 'Turks And Caicos Islands' 'Northern Mariana Islands'
 'Dominican Republic' 'Suriname' 'Honduras' 'Congo' 'Belize' 'Guatemala'
 'Anguilla' 'France' 'St Vincent And The Grenadines' 'Haiti' 'Montserra
t'
 'Papua New Guinea' 'Cayman Islands' 'Sweden' 'Taiwan' 'Senegal'
 'Barbados' 'BLOCK 651A' 'Brazil' 'Mauritius' 'Argentina' 'Kenya'
 'Ecuador' 'Aruba' 'Saudi Arabia' 'Cuba' 'Italy' 'French Guiana' 'Denmar
k'
 'Sudan' 'Spain' 'Federated States Of Micronesia' 'St Lucia' 'Switzerlan
d'
 'Central African Republic' 'Algeria' 'Turkey' 'Nicaragua'
 'Marshall Islands' 'Trinidad And Tobago' 'Poland' 'Belarus' 'Austria'
 'Malta' 'Cameroon' 'Solomon Islands' 'Zambia' 'Peru' 'Croatia' 'Fiji'
 'South Africa' 'India' 'Ethiopia' 'Ireland' 'Chile' 'Antigua And Barbud
a'
 'Uganda' 'China' 'Cambodia' 'Paraguay' 'Thailand' 'Belgium' 'Gambia'
 'Uruguay' 'Tanzania' 'Mali' 'Indonesia' 'Bahrain' 'Kazakhstan' 'Egypt'
 'Russia' 'Cyprus' "Cote D'ivoire" 'Nigeria' 'Greenland' 'Vietnam'
 'New Zealand' 'Singapore' 'Ghana' 'Gabon' 'Nepal' 'Slovakia' 'Finland'
 'Liberia' 'Romania' 'Maldives' 'Antarctica' 'Zimbabwe' 'Botswana'
 'Isle of Man' 'Latvia' 'Niger' 'French Polynesia' 'Guadeloupe'
 'Ivory Coast' 'Tunisia' 'Eritrea' 'Gibraltar' 'Namibia' 'Czech Republi
c'
 'Benin' 'Bosnia And Herzegovina' 'Israel' 'Estonia' 'St Kitts And Nevi
s'
 'Sierra Leone' 'Corsica' 'Scotland' 'Reunion' 'United Arab Emirates'
 'Afghanistan' 'Ukraine' 'Hungary' 'Bangladesh' 'Morocco' 'Iraq' 'Jorda
n'
 'Qatar' 'Madagascar' 'Malawi' 'Unknown' 'Central Africa' 'South Sudan'
 'Saint Barthelemy' 'Micronesia' 'South Korea' 'Kyrgyzstan'
 'Turks And Caicos' 'Eswatini' 'Tokelau' 'Sint Maarten' 'Macao'
 'Seychelles' 'Rwanda' 'Palau' 'Luxembourg' 'Lebanon'
 'Bosnia and Herzegovina' 'Libya' 'Guinea'
 'Saint Vincent and the Grenadines' 'UN' 'Iran' 'Lithuania' 'Malampa'
 'Antigua and Barbuda' 'AY' 'Chad' 'Cayenne' 'New Caledonia' 'Yemen'
 'Slovenia' 'Nauru' 'Niue' 'Bulgaria' 'Republic of North Macedonia'
 'Virgin Islands' 'Somalia' 'Pacific Ocean' 'Obyan' 'Mauritania' 'Albani
a'
 'Wolseley' 'Wallis and Futuna' 'Saint Pierre and Miquelon' 'Georgia'
 "Côte d'Ivoire" 'South Korean' 'Serbia' 'MU' 'Guernsey' 'Great Britain'
 'Turks and Caicos Islands']
```

In [31]:
```
# Inspect a few rows from each subset
print("\nSample of US flights:")
print(us_flights.head())

print("\nSample of international flights:")
print(international_flights.head())
```

```
Sample of US flights:
      Model Broad.phase.of.flight Weather.Condition  Total.Fatal.Injurie
s  \
0     108-3                Cruise               UNK                    2.
0
1  PA24-180               Unknown               UNK                    4.
0
2      172M                Cruise               IMC                    3.
0
3       112                Cruise               IMC                    2.
0
4       501              Approach               VMC                    1.
0

   Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  Year
\
0                     0.0                   0.0              0.0  1948
1                     0.0                   0.0              0.0  1962
2                     0.0                   0.0              0.0  1974
3                     0.0                   0.0              0.0  1977
4                     2.0                   0.0              0.0  1979

       Make Aircraft.Category        Country
0   Stinson               NaN  United States
1     Piper               NaN  United States
2    Cessna               NaN  United States
3  Rockwell               NaN  United States
4    Cessna               NaN  United States

Sample of international flights:
      Model Broad.phase.of.flight Weather.Condition  Total.Fatal.Injurie
s  \
36      206                  Taxi               VMC                    1.
0
237  206L-1               Takeoff               VMC                    0.
0
333     172              Approach               VMC                    0.
0
402     210                Cruise               VMC                    0.
0
463    206B              Approach               VMC                    2.
0

     Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  Year
\
36                      0.0                   1.0              0.0  1982
237                     0.0                   0.0              1.0  1982
333                     0.0                   0.0              1.0  1982
402                     2.0                   0.0              0.0  1982
463                     0.0                   0.0              0.0  1982

       Make Aircraft.Category         Country
36   Cessna          Airplane             NaN
237    Bell        Helicopter  GULF OF MEXICO
333  Cessna          Airplane     Puerto Rico
```

```
402  Cessna         Airplane  ATLANTIC OCEAN
463    Bell       Helicopter     HIGH ISLAND
```

In [32]:

```
# Count the number of NaN values in the 'Country' column
nan_countries_count = aviation_data_cleaned['Country'].isna().sum()
print(f"Number of rows with NaN in 'Country': {nan_countries_count}")
```

```
Number of rows with NaN in 'Country': 226
```

In [33]:

```
# Drop rows where 'Country' is NaN
aviation_data_cleaned = aviation_data_cleaned.dropna(subset=['Country'])

# Re-segment the data
us_flights = aviation_data_cleaned[aviation_data_cleaned['Country'] == "Ur
international_flights = aviation_data_cleaned[aviation_data_cleaned['Count

# Verify the counts
print(f"Number of US flights: {us_flights.shape[0]}")
print(f"Number of international flights: {international_flights.shape[0]}'
```

```
Number of US flights: 82248
Number of international flights: 6415
```

In [34]:

```
# Verify the updated counts
print(f"Number of US flights: {us_flights.shape[0]}")
print(f"Number of international flights: {international_flights.shape[0]}'

# Double-check the unique values in the 'Country' column
print("\nUnique values in the 'Country' column for US flights:")
print(us_flights['Country'].unique())

print("\nUnique values in the 'Country' column for international flights:'
print(international_flights['Country'].unique())
```

```
Number of US flights: 82248
Number of international flights: 6415

Unique values in the 'Country' column for US flights:
['United States']

Unique values in the 'Country' column for international flights:
['GULF OF MEXICO' 'Puerto Rico' 'ATLANTIC OCEAN' 'HIGH ISLAND' 'Bahamas'
 'MISSING' 'Pakistan' 'Angola' 'Germany' 'Korea, Republic Of' 'Martiniqu
e'
 'American Samoa' 'PACIFIC OCEAN' 'Canada' 'Bolivia' 'Mexico' 'Dominica'
 'Netherlands Antilles' 'Iceland' 'Greece' 'Guam' 'Australia'
 'CARIBBEAN SEA' 'West Indies' 'Japan' 'Philippines' 'Venezuela' 'Bermud
a'
 'San Juan Islands' 'Colombia' 'El Salvador' 'United Kingdom'
 'British Virgin Islands' 'Netherlands' 'Costa Rica' 'Mozambique'
 'Jamaica' 'Panama' 'Guyana' 'Norway' 'Hong Kong' 'Portugal' 'Malaysia'
 'Turks And Caicos Islands' 'Northern Mariana Islands'
 'Dominican Republic' 'Suriname' 'Honduras' 'Congo' 'Belize' 'Guatemala'
 'Anguilla' 'France' 'St Vincent And The Grenadines' 'Haiti' 'Montserra
t'
 'Papua New Guinea' 'Cayman Islands' 'Sweden' 'Taiwan' 'Senegal'
 'Barbados' 'BLOCK 651A' 'Brazil' 'Mauritius' 'Argentina' 'Kenya'
 'Ecuador' 'Aruba' 'Saudi Arabia' 'Cuba' 'Italy' 'French Guiana' 'Denmar
k'
 'Sudan' 'Spain' 'Federated States Of Micronesia' 'St Lucia' 'Switzerlan
d'
 'Central African Republic' 'Algeria' 'Turkey' 'Nicaragua'
 'Marshall Islands' 'Trinidad And Tobago' 'Poland' 'Belarus' 'Austria'
 'Malta' 'Cameroon' 'Solomon Islands' 'Zambia' 'Peru' 'Croatia' 'Fiji'
 'South Africa' 'India' 'Ethiopia' 'Ireland' 'Chile' 'Antigua And Barbud
a'
 'Uganda' 'China' 'Cambodia' 'Paraguay' 'Thailand' 'Belgium' 'Gambia'
 'Uruguay' 'Tanzania' 'Mali' 'Indonesia' 'Bahrain' 'Kazakhstan' 'Egypt'
 'Russia' 'Cyprus' "Cote D'ivoire" 'Nigeria' 'Greenland' 'Vietnam'
 'New Zealand' 'Singapore' 'Ghana' 'Gabon' 'Nepal' 'Slovakia' 'Finland'
 'Liberia' 'Romania' 'Maldives' 'Antarctica' 'Zimbabwe' 'Botswana'
 'Isle of Man' 'Latvia' 'Niger' 'French Polynesia' 'Guadeloupe'
 'Ivory Coast' 'Tunisia' 'Eritrea' 'Gibraltar' 'Namibia' 'Czech Republi
c'
 'Benin' 'Bosnia And Herzegovina' 'Israel' 'Estonia' 'St Kitts And Nevi
s'
 'Sierra Leone' 'Corsica' 'Scotland' 'Reunion' 'United Arab Emirates'
 'Afghanistan' 'Ukraine' 'Hungary' 'Bangladesh' 'Morocco' 'Iraq' 'Jorda
n'
 'Qatar' 'Madagascar' 'Malawi' 'Unknown' 'Central Africa' 'South Sudan'
 'Saint Barthelemy' 'Micronesia' 'South Korea' 'Kyrgyzstan'
 'Turks And Caicos' 'Eswatini' 'Tokelau' 'Sint Maarten' 'Macao'
 'Seychelles' 'Rwanda' 'Palau' 'Luxembourg' 'Lebanon'
 'Bosnia and Herzegovina' 'Libya' 'Guinea'
 'Saint Vincent and the Grenadines' 'UN' 'Iran' 'Lithuania' 'Malampa'
 'Antigua and Barbuda' 'AY' 'Chad' 'Cayenne' 'New Caledonia' 'Yemen'
 'Slovenia' 'Nauru' 'Niue' 'Bulgaria' 'Republic of North Macedonia'
 'Virgin Islands' 'Somalia' 'Pacific Ocean' 'Obyan' 'Mauritania' 'Albani
a'
 'Wolseley' 'Wallis and Futuna' 'Saint Pierre and Miquelon' 'Georgia'
```

```
"Côte d'Ivoire" 'South Korean' 'Serbia' 'MU' 'Guernsey' 'Great Britain'
'Turks and Caicos Islands']
```

Handling Missing `Country` Values

- **Decision**: Rows with missing `Country` values were excluded from the analysis.
- **Reason**: These rows represented only 0.25% of the data and could not be confidently classified as US or international flights.
- **Impact**: This ensures the segmentation into US and international subsets is clean and precise.

In [35]:

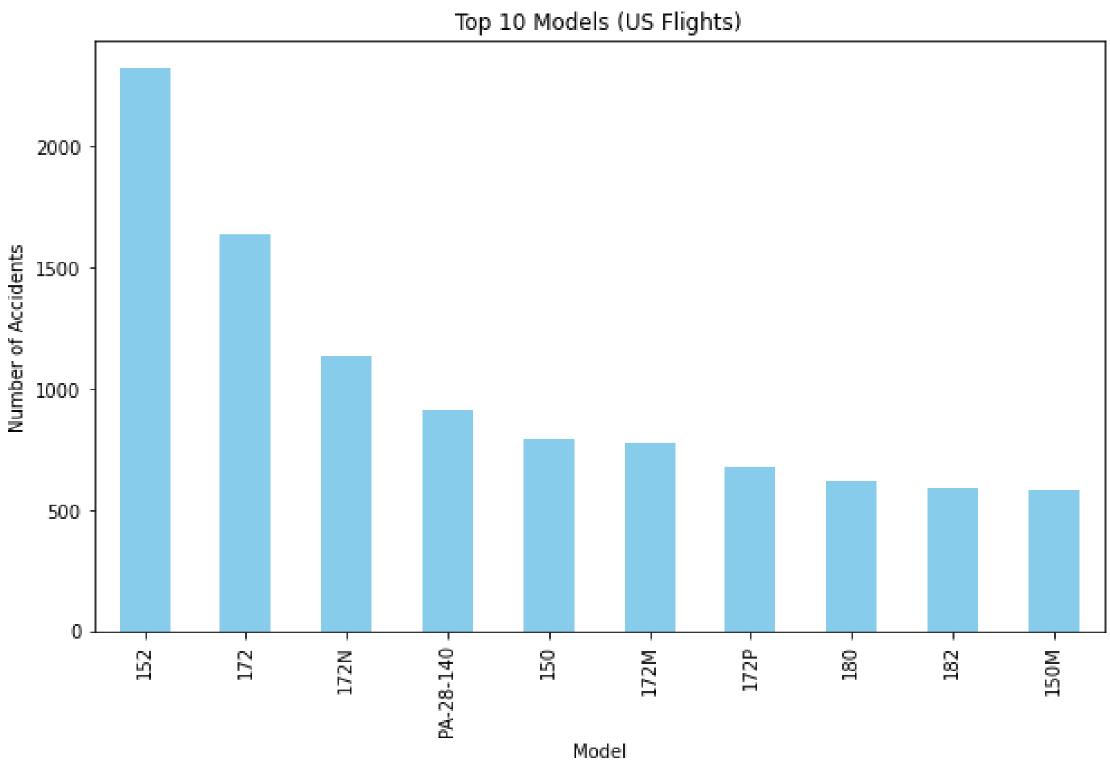
```
# Top 10 models for US flights
top_models_us = us_flights['Model'].value_counts().head(10)
print("Top 10 Models for US Flights:")
print(top_models_us)

# Top 10 models for international flights
top_models_international = international_flights['Model'].value_counts().h
print("\nTop 10 Models for International Flights:")
print(top_models_international)

# Analyze safety performance for US flights (proportions of uninjured pass
safety_us = us_flights.groupby('Model')[['Total.Fatal.Injuries', 'Total.Se
                                         'Total.Minor.Injuries', 'Total.Un
safety_us['Uninjured_Proportion'] = safety_us['Total.Uninjured'] / (
    safety_us['Total.Fatal.Injuries'] + safety_us['Total.Serious.Injuries'
    safety_us['Total.Minor.Injuries'] + safety_us['Total.Uninjured']
)
print("\nSafety Performance for Top US Models:")
print(safety_us.loc[top_models_us.index])

# Analyze safety performance for international flights (proportions of uni
safety_international = international_flights.groupby('Model')[['Total.Fata
                                                               'Total.Seri
                                                               'Total.Mino
                                                               'Total.Unin
safety_international['Uninjured_Proportion'] = safety_international['Total
    safety_international['Total.Fatal.Injuries'] + safety_international['T
    safety_international['Total.Minor.Injuries'] + safety_international['T
)
print("\nSafety Performance for Top International Models:")
print(safety_international.loc[top_models_international.index])
```

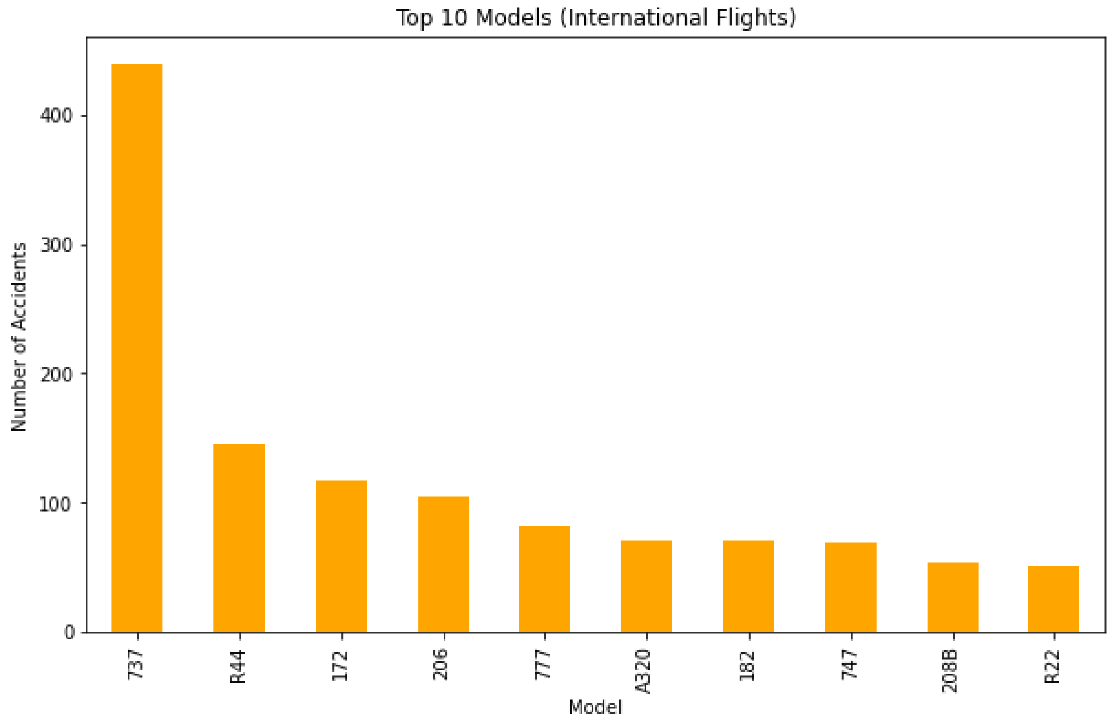
```
Top 10 Models for US Flights:
152          2323
172          1637
172N         1136
PA-28-140     910
150           790
172M          773
172P          680
180           617
182           589
150M          578
Name: Model, dtype: int64

Top 10 Models for International Flights:
737     439
R44     145
172     117
206     105
777      81
A320     71
182      70
747      69
208B     54
R22      50
Name: Model, dtype: int64

Safety Performance for Top US Models:
           Total.Fatal.Injuries  Total.Serious.Injuries  Total.Minor.Inj
uries  \
152                       364.0                   195.0
417.0
172                       254.0                   290.0
363.0
172N                      390.0                   199.0
334.0
PA-28-140                 301.0                   258.0
401.0
150                        93.0                   115.0
200.0
172M                      221.0                   168.0
247.0
172P                      218.0                   113.0
208.0
180                       101.0                    70.0
73.0
182                       126.0                   119.0
158.0
150M                       99.0                    64.0
149.0

           Total.Uninjured  Uninjured_Proportion
152                 2340.0              0.705669
172                 2205.0              0.708548
172N                1336.0              0.591412
PA-28-140            832.0              0.464286
150                  746.0              0.646447
172M                1068.0              0.626761
```

```
172P                856.0              0.613620
180                 990.0              0.802269
182                 813.0              0.668586
150M                562.0              0.643021

Safety Performance for Top International Models:
      Total.Fatal.Injuries  Total.Serious.Injuries  Total.Minor.Injuries
\
737                 1348.0                   368.0                  74.0
R44                  209.0                    35.0                  42.0
172                  146.0                    20.0                  28.0
206                  162.0                    18.0                  49.0
777                    0.0                    24.0                  19.0
A320                 513.0                    11.0                   7.0
182                   92.0                    12.0                  23.0
747                    6.0                    50.0                   6.0
208B                 131.0                    60.0                  94.0
R22                   48.0                     5.0                   3.0

      Total.Uninjured  Uninjured_Proportion
737           20126.0              0.918325
R44              52.0              0.153846
172              45.0              0.188285
206             147.0              0.390957
777            7559.0              0.994344
A320           1706.0              0.762629
182              31.0              0.196203
747            2349.0              0.974285
208B            104.0              0.267352
R22              16.0              0.222222
```

In [37]:

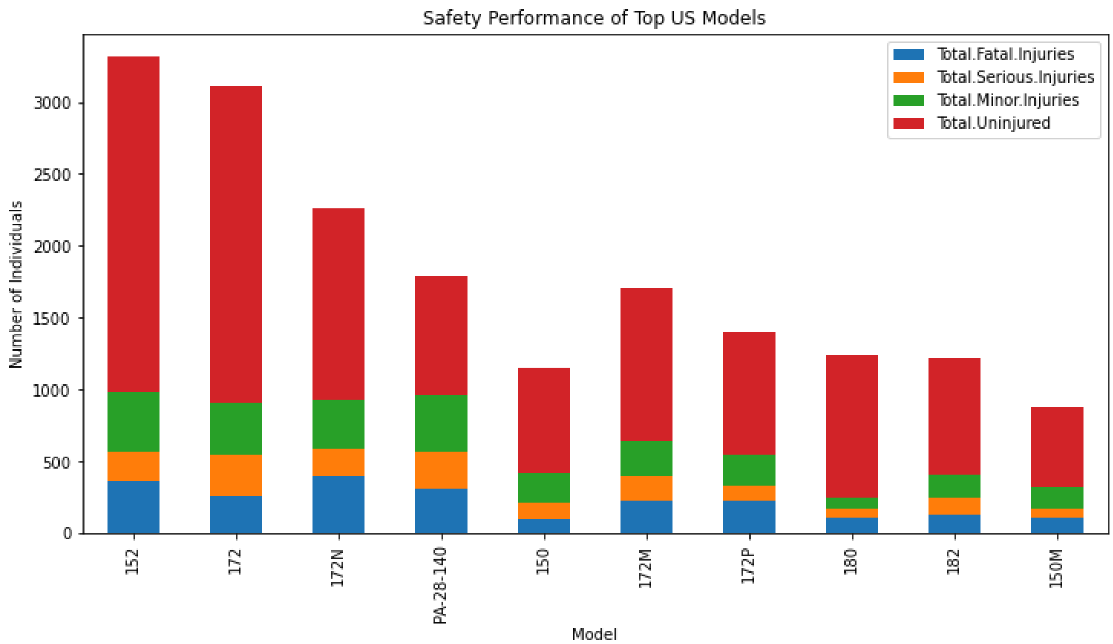
```
# Import matplotlib if not already imported
import matplotlib.pyplot as plt

# Bar chart for top US models
top_models_us.plot(kind='bar', figsize=(10, 6), color='skyblue', title='To
plt.xlabel("Model")
plt.ylabel("Number of Accidents")
plt.show()

# Bar chart for top international models
top_models_international.plot(kind='bar', figsize=(10, 6), color='orange',
plt.xlabel("Model")
plt.ylabel("Number of Accidents")
plt.show()
```

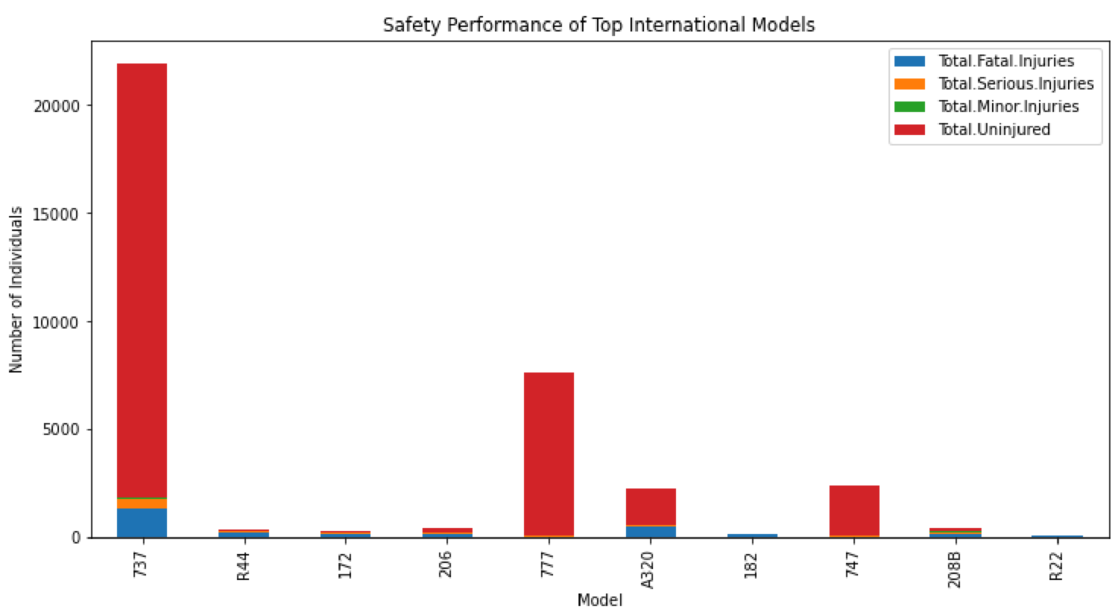



Top 10 Models (International Flights)
Number of Accidents
0
100
200
300
400
737
R44
172
206
777
A320
182
747
208B
R22
Model

In [38]:

```
# Safety performance for top US models
safety_us.loc[top_models_us.index][['Total.Fatal.Injuries', 'Total.Serious
                                    'Total.Minor.Injuries', 'Total.Uninjur
    kind='bar', stacked=True, figsize=(12, 6), title='Safety Performance o
)
plt.xlabel("Model")
plt.ylabel("Number of Individuals")
plt.show()

# Safety performance for top international models
safety_international.loc[top_models_international.index][['Total.Fatal.Inj
                                                          'Total.Minor.Inj
    kind='bar', stacked=True, figsize=(12, 6), title='Safety Performance o
)
plt.xlabel("Model")
plt.ylabel("Number of Individuals")
plt.show()
```





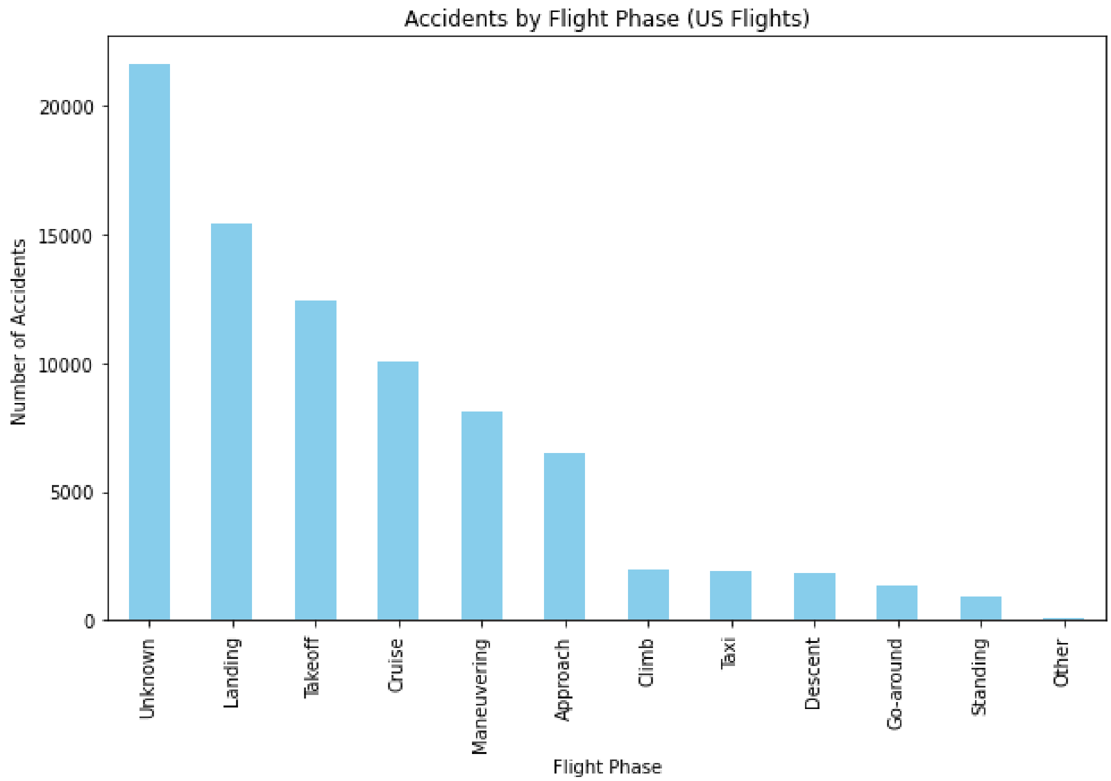
In [39]:

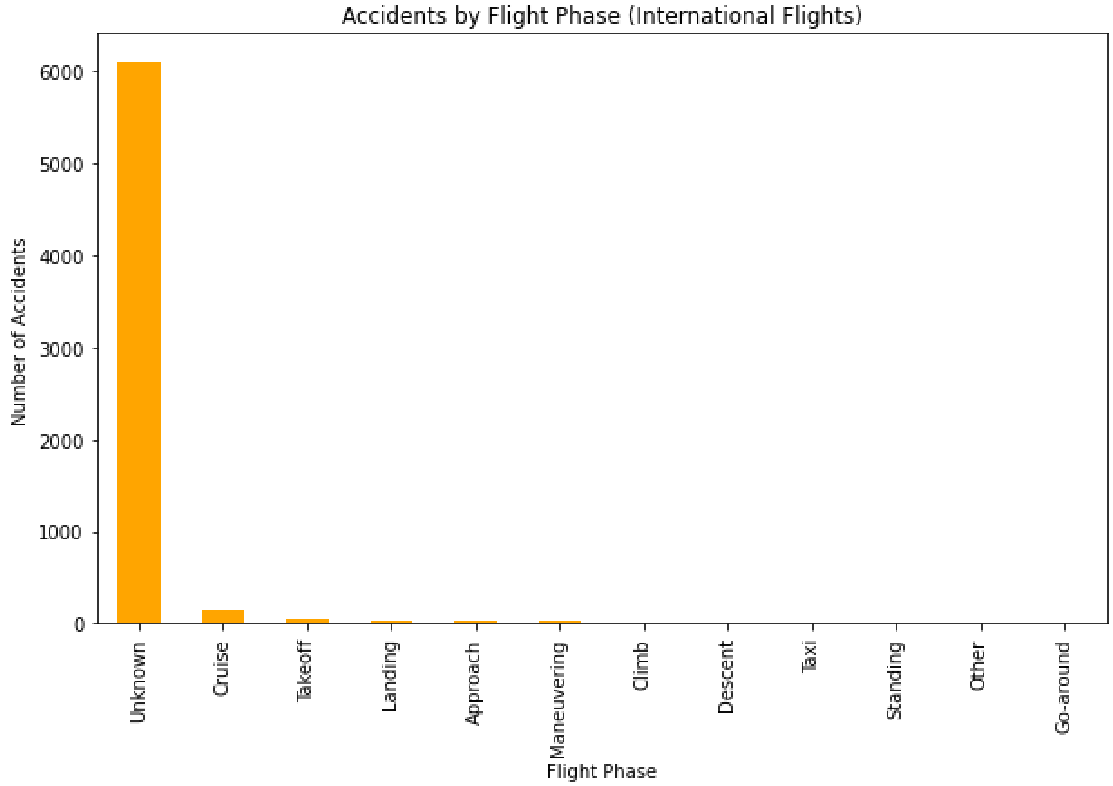
```
# Accident distribution by flight phase for US flights
phase_distribution_us = us_flights['Broad.phase.of.flight'].value_counts()
print("US Flight Phase Distribution:")
print(phase_distribution_us)

# Accident distribution by flight phase for international flights
phase_distribution_international = international_flights['Broad.phase.of.f
print("\nInternational Flight Phase Distribution:")
print(phase_distribution_international)
```

```
US Flight Phase Distribution:
Unknown        21595
Landing        15365
Takeoff        12412
Cruise         10073
Maneuvering     8100
Approach        6502
Climb           2006
Taxi            1941
Descent         1862
Go-around       1350
Standing         926
Other            116
Name: Broad.phase.of.flight, dtype: int64

International Flight Phase Distribution:
Unknown        6101
Cruise          147
Takeoff          40
Landing          28
Approach         24
Maneuvering      23
Climb            18
Descent          13
Taxi              9
Standing          8
Other             3
Go-around         1
Name: Broad.phase.of.flight, dtype: int64
```

In [40]:

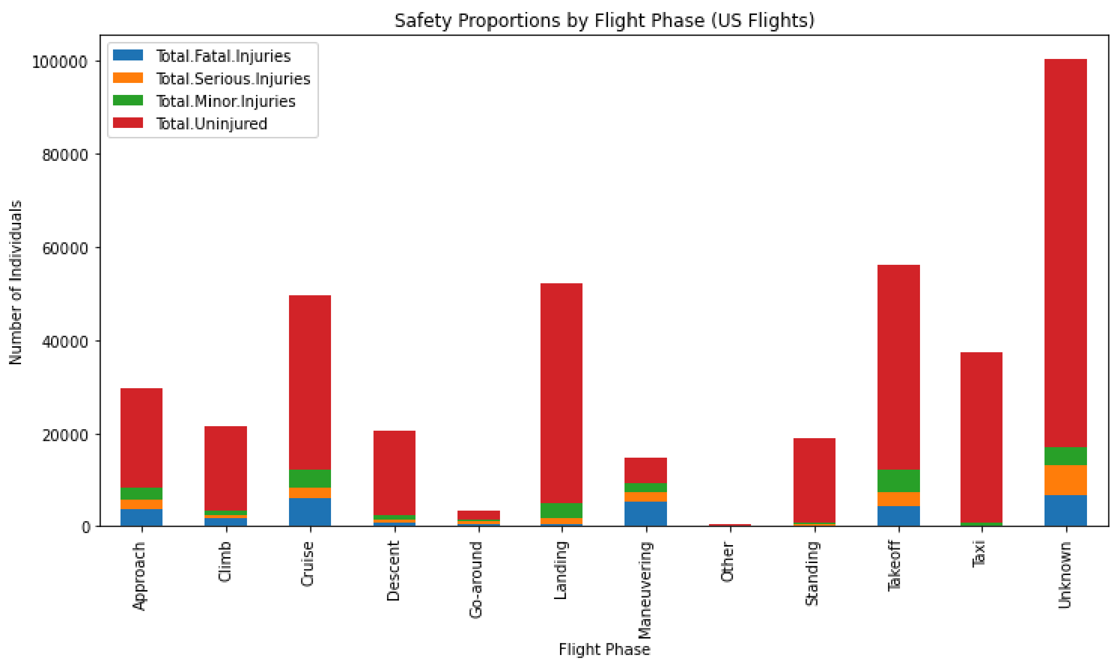
```
# Bar chart for US flight phases
phase_distribution_us.plot(kind='bar', figsize=(10, 6), color='skyblue', t
plt.xlabel("Flight Phase")
plt.ylabel("Number of Accidents")
plt.show()

# Bar chart for international flight phases
phase_distribution_international.plot(kind='bar', figsize=(10, 6), color=
plt.xlabel("Flight Phase")
plt.ylabel("Number of Accidents")
plt.show()
```

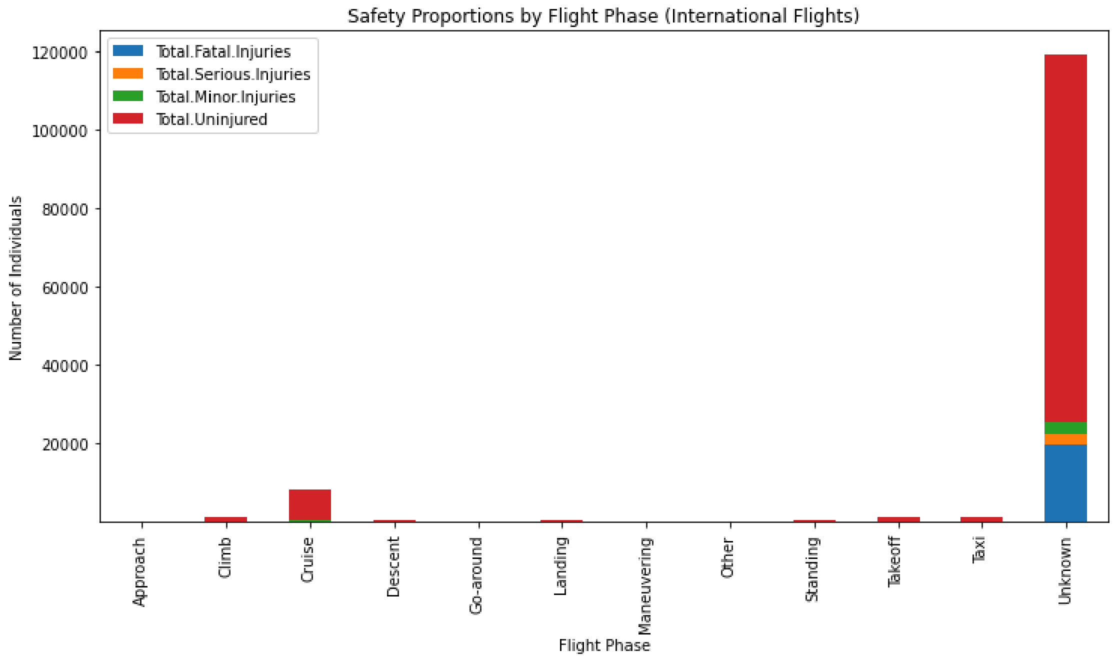



Accidents by Flight Phase (International Flights)
Number of Accidents
0
1000
2000
3000
4000
5000
6000
Unknown
Cruise
Takeoff
Landing
Approach
Maneuvering
Climb
Descent
Taxi
Standing
Other
Go-around
Flight Phase

In [41]:

```
# Safety proportions for US flights by flight phase
safety_by_phase_us = us_flights.groupby('Broad.phase.of.flight')[['Total.F
                                                                   'Total.
safety_by_phase_us.plot(kind='bar', stacked=True, figsize=(12, 6), title=
plt.xlabel("Flight Phase")
plt.ylabel("Number of Individuals")
plt.show()

# Safety proportions for international flights by flight phase
safety_by_phase_international = international_flights.groupby('Broad.phase



safety_by_phase_international.plot(kind='bar', stacked=True, figsize=(12,
plt.xlabel("Flight Phase")
plt.ylabel("Number of Individuals")
plt.show()
```



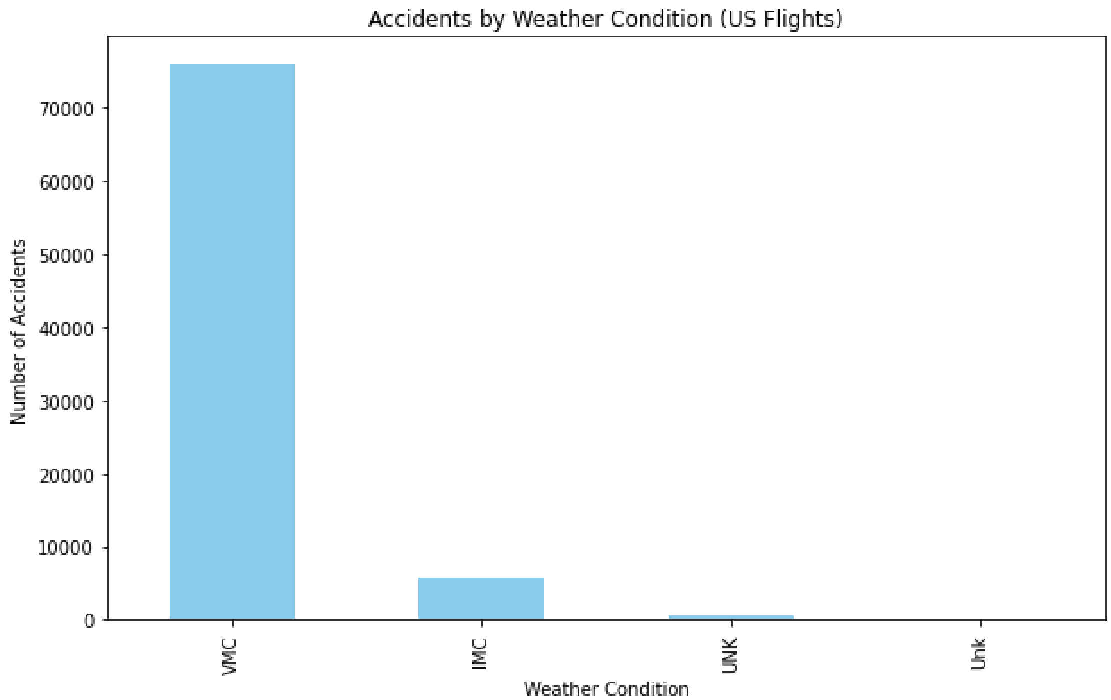


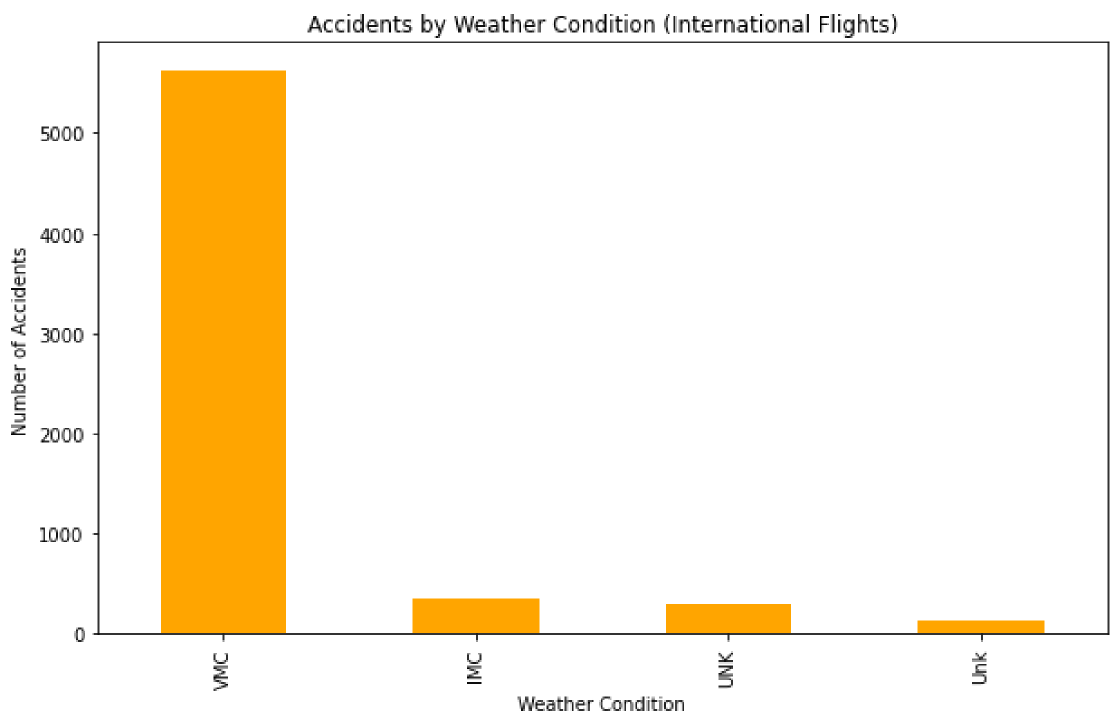
In [42]:

```
# Accident distribution by weather condition for US flights
weather_distribution_us = us_flights['Weather.Condition'].value_counts()
print("US Weather Condition Distribution:")
print(weather_distribution_us)

# Accident distribution by weather condition for international flights
weather_distribution_international = international_flights['Weather.Condit
print("\nInternational Weather Condition Distribution:")
print(weather_distribution_international)
```

```
US Weather Condition Distribution:
VMC    75962
IMC     5618
UNK      547
Unk      121
Name: Weather.Condition, dtype: int64

International Weather Condition Distribution:
VMC    5627
IMC     345
UNK     302
Unk     141
Name: Weather.Condition, dtype: int64
```

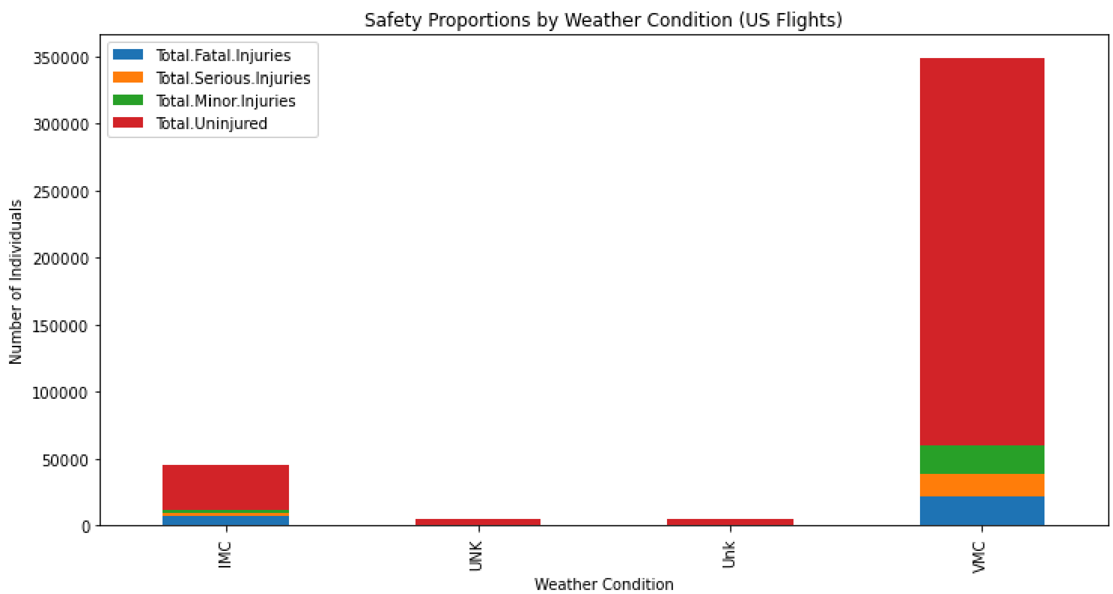
In [43]:
```
# Bar chart for US weather conditions
weather_distribution_us.plot(kind='bar', figsize=(10, 6), color='skyblue',
plt.xlabel("Weather Condition")
plt.ylabel("Number of Accidents")
plt.show()

# Bar chart for international weather conditions
weather_distribution_international.plot(kind='bar', figsize=(10, 6), color
plt.xlabel("Weather Condition")
plt.ylabel("Number of Accidents")
plt.show()
```

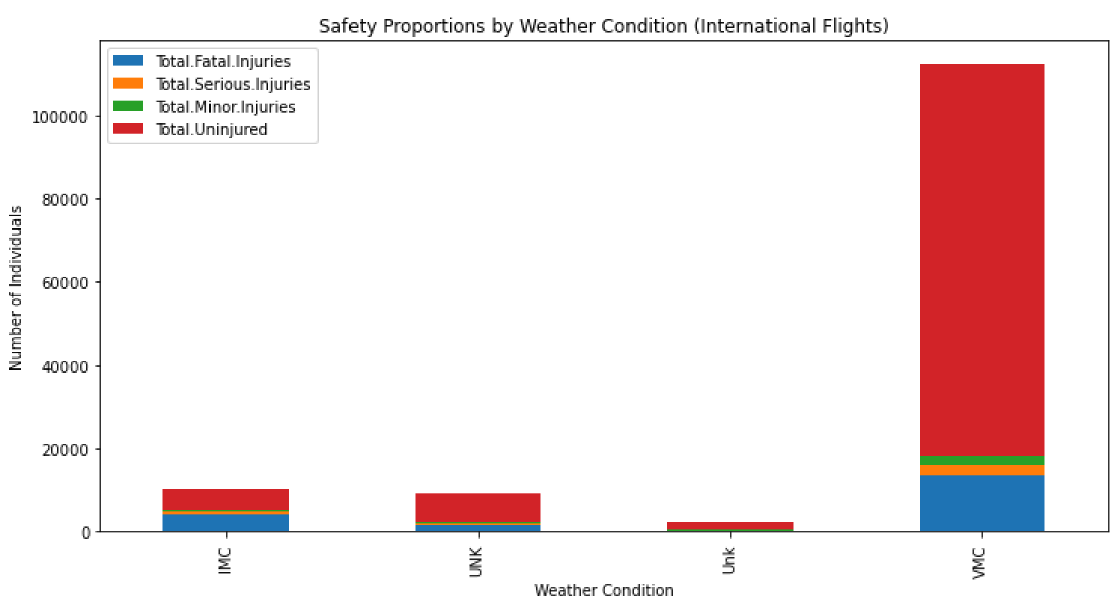




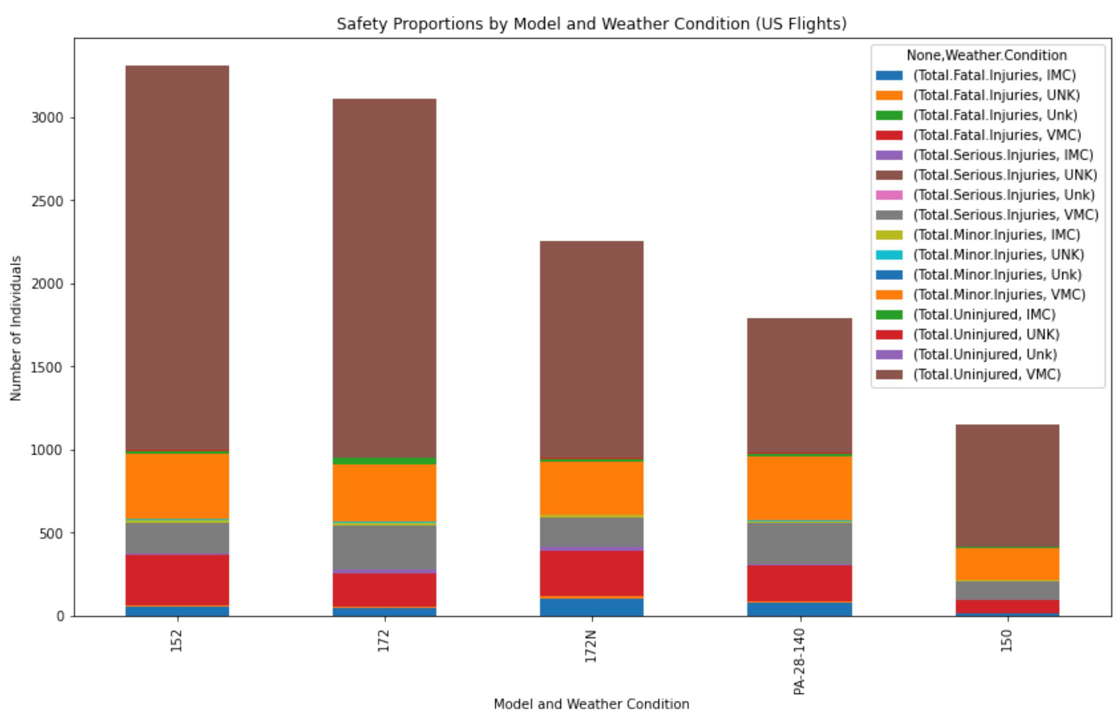
In [44]:

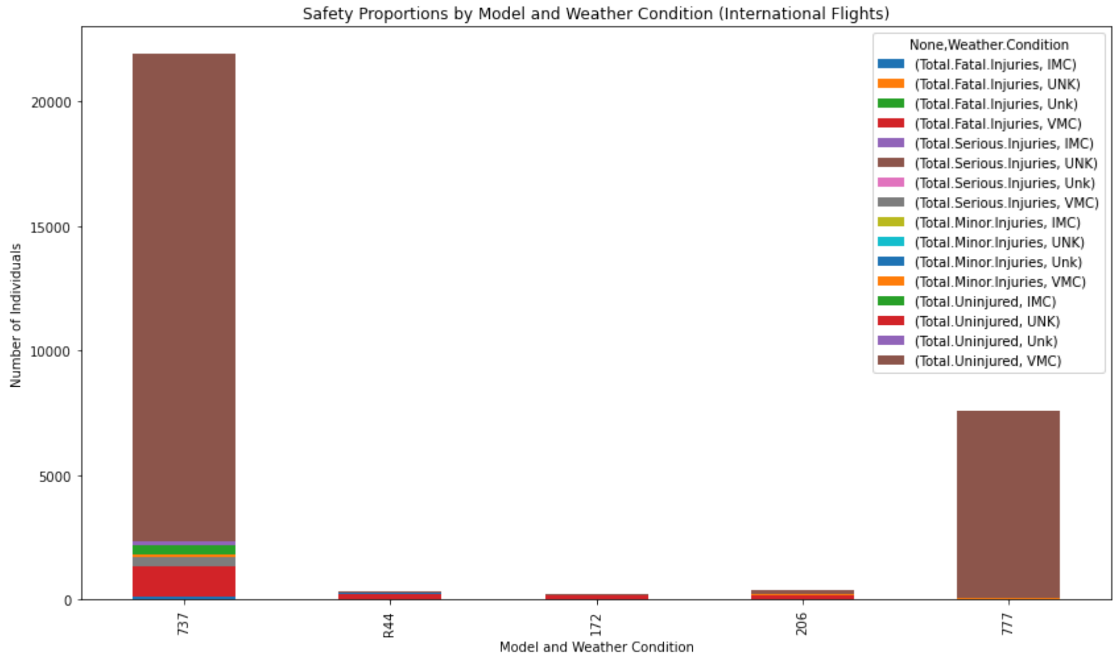
```
# Safety proportions for US flights by weather condition
safety_by_weather_us = us_flights.groupby('Weather.Condition')[['Total.Fat
                                                                 'Total.Mi
safety_by_weather_us.plot(kind='bar', stacked=True, figsize=(12, 6), title
plt.xlabel("Weather Condition")
plt.ylabel("Number of Individuals")
plt.show()

# Safety proportions for international flights by weather condition
safety_by_weather_international = international_flights.groupby('Weather.C



safety_by_weather_international.plot(kind='bar', stacked=True, figsize=(12
plt.xlabel("Weather Condition")
plt.ylabel("Number of Individuals")
plt.show()
```





In [45]:
```
# Group by 'Model' and 'Weather.Condition' for US flights
safety_by_model_weather_us = us_flights.groupby(['Model', 'Weather.Conditi


# Select top 5 models by accident frequency for visualization
top_models_us_weather = us_flights['Model'].value_counts().head(5).index
safety_by_model_weather_us_top = safety_by_model_weather_us.loc[top_models

# Visualize safety proportions for top US models under VMC and IMC
safety_by_model_weather_us_top.unstack().plot(kind='bar', stacked=True, fi
plt.xlabel("Model and Weather Condition")
plt.ylabel("Number of Individuals")
plt.show()

# Repeat for international flights
safety_by_model_weather_international = international_flights.groupby(['Mo




# Select top 5 models by accident frequency for visualization
top_models_international_weather = international_flights['Model'].value_co
safety_by_model_weather_international_top = safety_by_model_weather_intern

# Visualize safety proportions for top international models under VMC and
safety_by_model_weather_international_top.unstack().plot(kind='bar', stack
plt.xlabel("Model and Weather Condition")
plt.ylabel("Number of Individuals")
plt.show()
```





In [47]:

```
# Safely update the 'Year' column using .loc
top_models_data = top_models_data.copy()  # Create an explicit copy to avo
top_models_data['Year'] = pd.to_numeric(top_models_data['Year'], errors='c
```

In [50]:

```
# Group data by Model and Year, summing up injury counts
model_yearly_stats = top_models_data.groupby(['Model', 'Year']).sum().rese

# Display the first few rows to verify
print(model_yearly_stats.head())
```

```
  Model  Year  Total.Fatal.Injuries  Total.Serious.Injuries  \
0   150  1982                   7.0                    11.0
1   150  1983                   5.0                    12.0
2   150  1984                   7.0                     4.0
3   150  1985                   2.0                     7.0
4   150  1986                   3.0                     3.0

   Total.Minor.Injuries  Total.Uninjured
0                  25.0             68.0
1                  14.0             54.0
2                  11.0             39.0
3                  12.0             37.0
4                   9.0             33.0
```

In [51]:

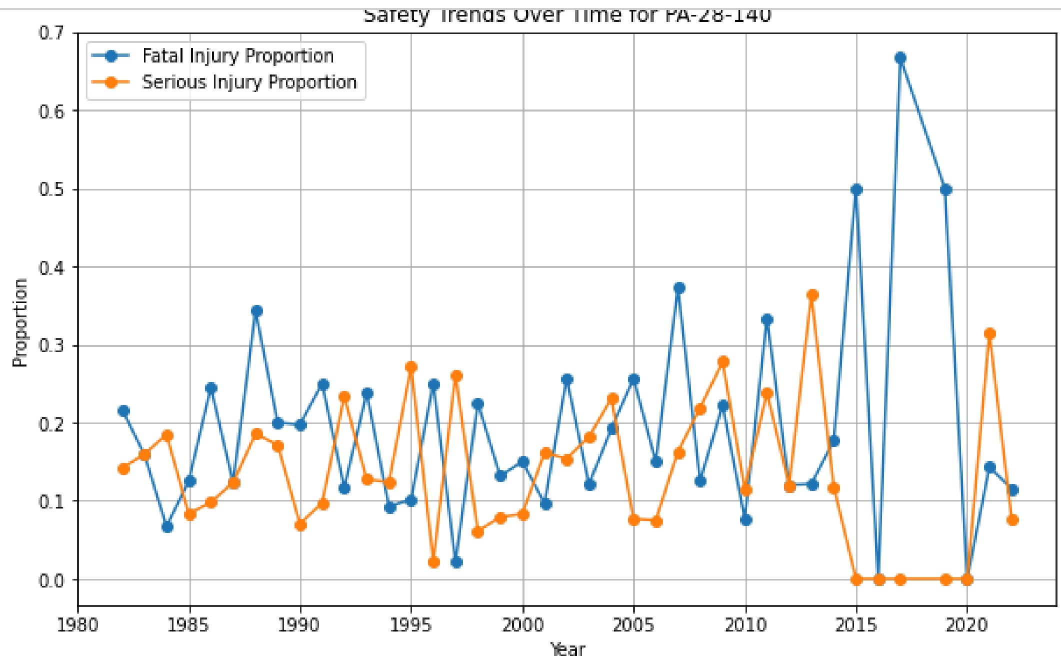
```
# Calculate proportions for each severity type
model_yearly_stats['Fatal.Injury.Proportion'] = model_yearly_stats['Total.
    model_yearly_stats['Total.Fatal.Injuries'] +
    model_yearly_stats['Total.Serious.Injuries'] +
    model_yearly_stats['Total.Minor.Injuries'] +
    model_yearly_stats['Total.Uninjured']
)

model_yearly_stats['Serious.Injury.Proportion'] = model_yearly_stats['Tota
    model_yearly_stats['Total.Fatal.Injuries'] +
    model_yearly_stats['Total.Serious.Injuries'] +
    model_yearly_stats['Total.Minor.Injuries'] +
    model_yearly_stats['Total.Uninjured']
)
```

In [52]:

```
# Display the first few rows to check the calculated proportions
print(model_yearly_stats[['Model', 'Year', 'Fatal.Injury.Proportion', 'Ser
```

```
  Model  Year  Fatal.Injury.Proportion  Serious.Injury.Proportion
0   150  1982                 0.063063                   0.099099
1   150  1983                 0.058824                   0.141176
2   150  1984                 0.114754                   0.065574
3   150  1985                 0.034483                   0.120690
4   150  1986                 0.062500                   0.062500
```

In [53]:
```
import matplotlib.pyplot as plt

# Plot trends for each model
for model in model_yearly_stats['Model'].unique():
    model_data = model_yearly_stats[model_yearly_stats['Model'] == model]

    plt.figure(figsize=(10, 6))
    plt.plot(model_data['Year'], model_data['Fatal.Injury.Proportion'], la
    plt.plot(model_data['Year'], model_data['Serious.Injury.Proportion'],

    plt.title(f"Safety Trends Over Time for {model}")
    plt.xlabel("Year")
    plt.ylabel("Proportion")
    plt.legend()
    plt.grid(True)
    plt.show()
```



In [ ]: