 # 1. Business Understanding

Objective: Analyze aviation incident data to identify patterns in safety concerns, state-wise distribution of incidents, and contributing factors and provide actionable insights for aviation safety improvements.

Key questions:

- What are the trends in aviation incidents over time?
- Which states or regions are most affected by incidents?
- What are the most common injury severities?

 # 2. Data Understanding

The dataset contains various attributes related to aviation accidents, such as event ID, date, location, aircraft details, injury severity, and more. It includes categorical and numerical data, which will be explored to understand its structure and content.

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the datasets
aviation_data = pd.read_csv('AviationData.csv', encoding='latin1',
low_memory=False) #DtypeWarning columns 6, 7, 28
state_codes = pd.read_csv('USState_Codes.csv', encoding='latin1')

# Preview the datasets
print("AviationData:")
aviation_data.head()

print("\nState Codes Data:")
state_codes.head()
```

```
AviationData:

State Codes Data:
```

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

```
# Display basic information and non-null count
print("\nAviationData Info:")
aviation_data.info()
```

```
print("\nState Codes Info:")
state_codes.info()
```

```
AviationData 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            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        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB

State Codes Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
 0   US_State       62 non-null     object
 1   Abbreviation   62 non-null     object
dtypes: object(2)
memory usage: 1.1+ KB
```

```
# Display summary statistics before cleaning
aviation_data.describe()
```

```
       Number.of.Engines  Total.Fatal.Injuries  Total.Serious.Injuries
\
count       82805.000000          77488.000000            76379.000000

mean            1.146585              0.647855                0.279881

std             0.446510              5.485960                1.544084

min             0.000000              0.000000                0.000000

25%             1.000000              0.000000                0.000000

50%             1.000000              0.000000                0.000000

75%             1.000000              0.000000                0.000000

max             8.000000            349.000000              161.000000


       Total.Minor.Injuries  Total.Uninjured
count          76956.000000     82977.000000
mean               0.357061         5.325440
std                2.235625        27.913634
min                0.000000         0.000000
25%                0.000000         0.000000
50%                0.000000         1.000000
75%                0.000000         2.000000
max              380.000000       699.000000
```

 # 3. Data Preparation Clean and prepare the data for analysis by handling missing values, merging datasets, and converting date columns.

```
# Handling missing values in aviation data
missing_values = aviation_data.isnull().sum()
print(missing_values)
```

```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   52
Country                   226
```

```
Latitude                  54507
Longitude                 54516
Airport.Code              38757
Airport.Name              36185
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1382
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7096
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6384
Publication.Date          13771
dtype: int64
```

```
# Drop rows with missing key fields (if any)

print("Rows before dropping:", len(aviation_data))
aviation_data.dropna(subset=['Location', 'Event.Date',
'Injury.Severity'], inplace=True)
print("Rows after dropping:", len(aviation_data))
```

```
Rows before dropping: 88889
Rows after dropping: 87837
```

```
# Check column names of both dataframes
print(aviation_data.columns)
print(state_codes.columns)
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number',
'Event.Date',
       '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.Description',
       'Schedule', 'Purpose.of.flight', 'Air.carrier',
'Total.Fatal.Injuries',
```

```
       'Total.Serious.Injuries', 'Total.Minor.Injuries',
'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
Index(['US_State', 'Abbreviation'], dtype='object')

# Extract state abbreviations from location
aviation_data['State.Abbreviation'] =
aviation_data['Location'].str.extract(r',\s*([A-Z]{2})$')

# Merge Aviation data with state codes on State column
aviation_data = pd.merge(aviation_data, state_codes, how='left',
left_on='State.Abbreviation', right_on='Abbreviation')

# Results after the merge
print("\nMerged Aviation Data (First few rows):")
print(aviation_data.head())


Merged Aviation Data (First few rows):
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country   Latitude   Longitude Airport.Code
\
0  MOOSE CREEK, ID  United States        NaN         NaN          NaN

1   BRIDGEPORT, CA  United States        NaN         NaN          NaN

2    Saltville, VA  United States  36.922223  -81.878056          NaN

3       EUREKA, CA  United States        NaN         NaN          NaN

4       Canton, OH  United States        NaN         NaN          NaN


  Airport.Name  ... Total.Serious.Injuries Total.Minor.Injuries  \
0          NaN  ...                   0.0                 0.0
1          NaN  ...                   0.0                 0.0
2          NaN  ...                   NaN                 NaN
3          NaN  ...                   0.0                 0.0
4          NaN  ...                   2.0                 NaN

  Total.Uninjured Weather.Condition Broad.phase.of.flight
Report.Status  \
0             0.0               UNK                Cruise  Probable
```

```
Cause
1              0.0                 UNK                Unknown  Probable
Cause
2              NaN                 IMC                 Cruise  Probable
Cause
3              0.0                 IMC                 Cruise  Probable
Cause
4              0.0                 VMC               Approach  Probable
Cause

  Publication.Date  State.Abbreviation    US_State Abbreviation
0              NaN                  ID       Idaho           ID
1       19-09-1996                  CA  California           CA
2       26-02-2007                  VA    Virginia           VA
3       12-09-2000                  CA  California           CA
4       16-04-1980                  OH        Ohio           OH

[5 rows x 34 columns]
```

```
# Check for duplicates
aviation_data.duplicated().sum()
state_codes.duplicated().sum()

0
```

There are no duplicates in either datasets

```
# Convert date column to DateTime format
aviation_data['Event.Date'] =
pd.to_datetime(aviation_data['Event.Date'])

# Extract Year, Month, and Day of the Week
aviation_data['Year'] = aviation_data['Event.Date'].dt.year
aviation_data['Month'] = aviation_data['Event.Date'].dt.month
aviation_data['Day_of_Week'] =
aviation_data['Event.Date'].dt.day_name()

# Confirm if columns were added first few rows
print(aviation_data[['Event.Date', 'Year', 'Month',
'Day_of_Week']].head())

  Event.Date  Year  Month Day_of_Week
0 1948-10-24  1948     10      Sunday
1 1962-07-19  1962      7    Thursday
2 1974-08-30  1974      8      Friday
3 1977-06-19  1977      6      Sunday
4 1979-08-02  1979      8    Thursday

# Identifies if the incident occurred on a weekend
aviation_data['Is_Weekend'] =
```

```
aviation_data['Day_of_Week'].isin(['Saturday', 'Sunday']).astype(int)
print(aviation_data[['Day_of_Week', 'Is_Weekend']].head())
```

```
  Day_of_Week  Is_Weekend
0      Sunday           1
1    Thursday           0
2      Friday           0
3      Sunday           1
4    Thursday           0
```

```
# Create binary indicator for fatalities
aviation_data['Has_Fatalities'] =
aviation_data['Total.Fatal.Injuries'].fillna(0).astype(int).apply(lamb
da x: 1 if x > 0 else 0)

# Total casualties: Sum of fatal and serious injuries
aviation_data['Total_Casualties'] =
aviation_data[['Total.Fatal.Injuries',
'Total.Serious.Injuries']].sum(axis=1, skipna=True)

# First few rows if there were fatalities in the incident,total number
of casualties from both fatal and serious injuries
print(aviation_data[['Total.Fatal.Injuries', 'Total.Serious.Injuries',
'Has_Fatalities', 'Total_Casualties']].head())
```

```
   Total.Fatal.Injuries  Total.Serious.Injuries  Has_Fatalities  \
0                   2.0                     0.0               1
1                   4.0                     0.0               1
2                   3.0                     NaN               1
3                   2.0                     0.0               1
4                   1.0                     2.0               1

   Total_Casualties
0               2.0
1               4.0
2               3.0
3               2.0
4               3.0
```

This can be useful for further analysis, in identifying patterns of severity or analyzing the relationship between flight phase, weather conditions, or day of the week and the likelihood of fatalities or high casualty counts.

```
# Verify merged and cleaned data
print(aviation_data.head())
```

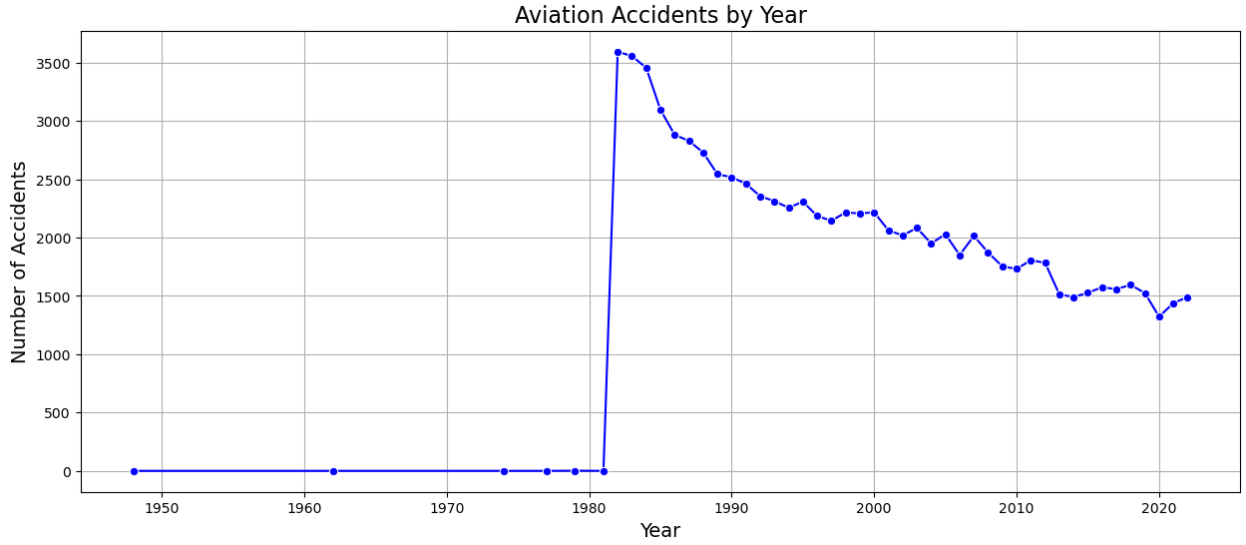
```
         Event.Id Investigation.Type Accident.Number Event.Date  \
0  20001218X45444           Accident      SEA87LA080 1948-10-24
1  20001218X45447           Accident      LAX94LA336 1962-07-19
2  20061025X01555           Accident      NYC07LA005 1974-08-30
```

```
3  20001218X45448           Accident      LAX96LA321 1977-06-19
4  20041105X01764           Accident      CHI79FA064 1979-08-02

          Location        Country   Latitude   Longitude Airport.Code
\
0  MOOSE CREEK, ID  United States        NaN         NaN          NaN

1   BRIDGEPORT, CA  United States        NaN         NaN          NaN

2    Saltville, VA  United States  36.922223  -81.878056          NaN

3       EUREKA, CA  United States        NaN         NaN          NaN

4       Canton, OH  United States        NaN         NaN          NaN


  Airport.Name  ... Publication.Date State.Abbreviation    US_State  \
0          NaN  ...              NaN                ID       Idaho
1          NaN  ...       19-09-1996                CA  California
2          NaN  ...       26-02-2007                VA    Virginia
3          NaN  ...       12-09-2000                CA  California
4          NaN  ...       16-04-1980                OH        Ohio

  Abbreviation  Year Month Day_of_Week  Is_Weekend Has_Fatalities  \
0           ID  1948    10      Sunday           1              1
1           CA  1962     7    Thursday           0              1
2           VA  1974     8      Friday           0              1
3           CA  1977     6      Sunday           1              1
4           OH  1979     8    Thursday           0              1

  Total_Casualties
0              2.0
1              4.0
2              3.0
3              2.0
4              3.0

[5 rows x 40 columns]
```

 # 4. Modeling Analyses on trends, distributions, and relationships in the data using visualizations on aviation accident data, including temporal trends, severity distribution, aircraft damage, weather conditions, and phases of flight.

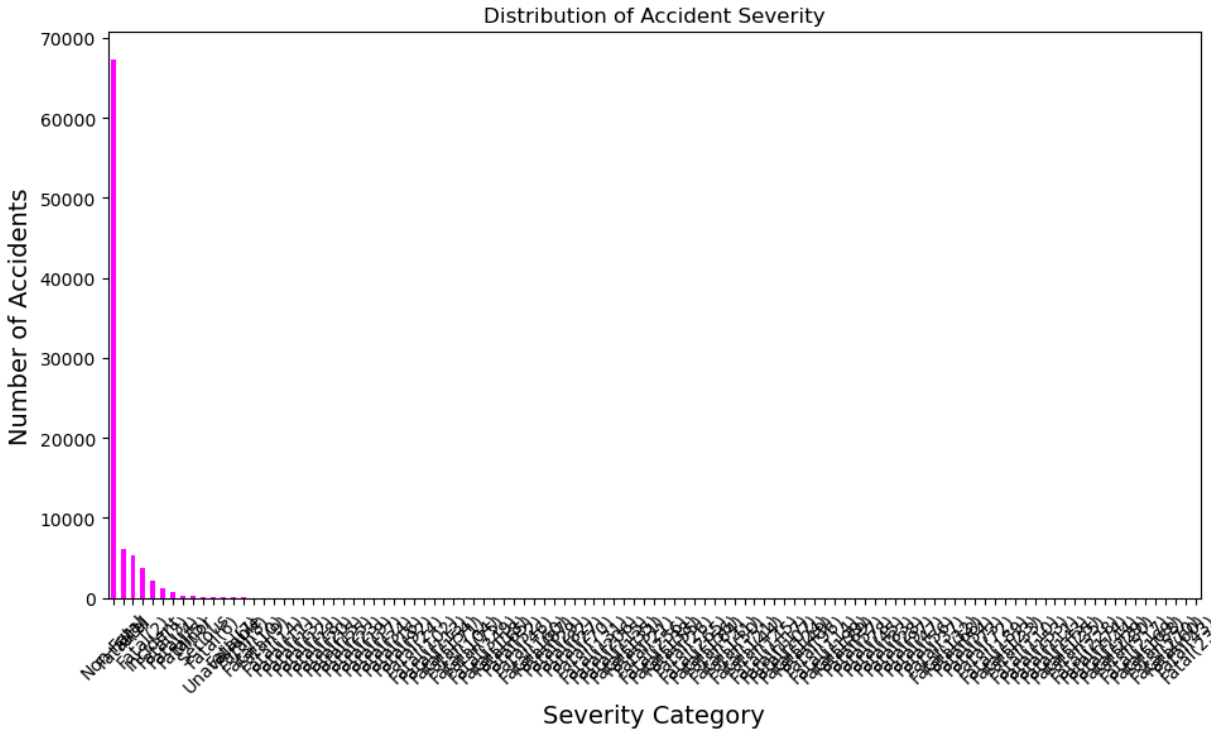
```
# Visualization 1: Yearly trends
yearly_accidents = aviation_data['Year'].value_counts().sort_index()

# Plot yearly trends
plt.figure(figsize=(15, 6))
sns.lineplot(x=yearly_accidents.index, y=yearly_accidents.values,
marker='o', color='blue')
```

```
plt.title('Aviation Accidents by Year', fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)
plt.grid(True)
plt.show()
plt.savefig('Yearly_Accidents.png')
```

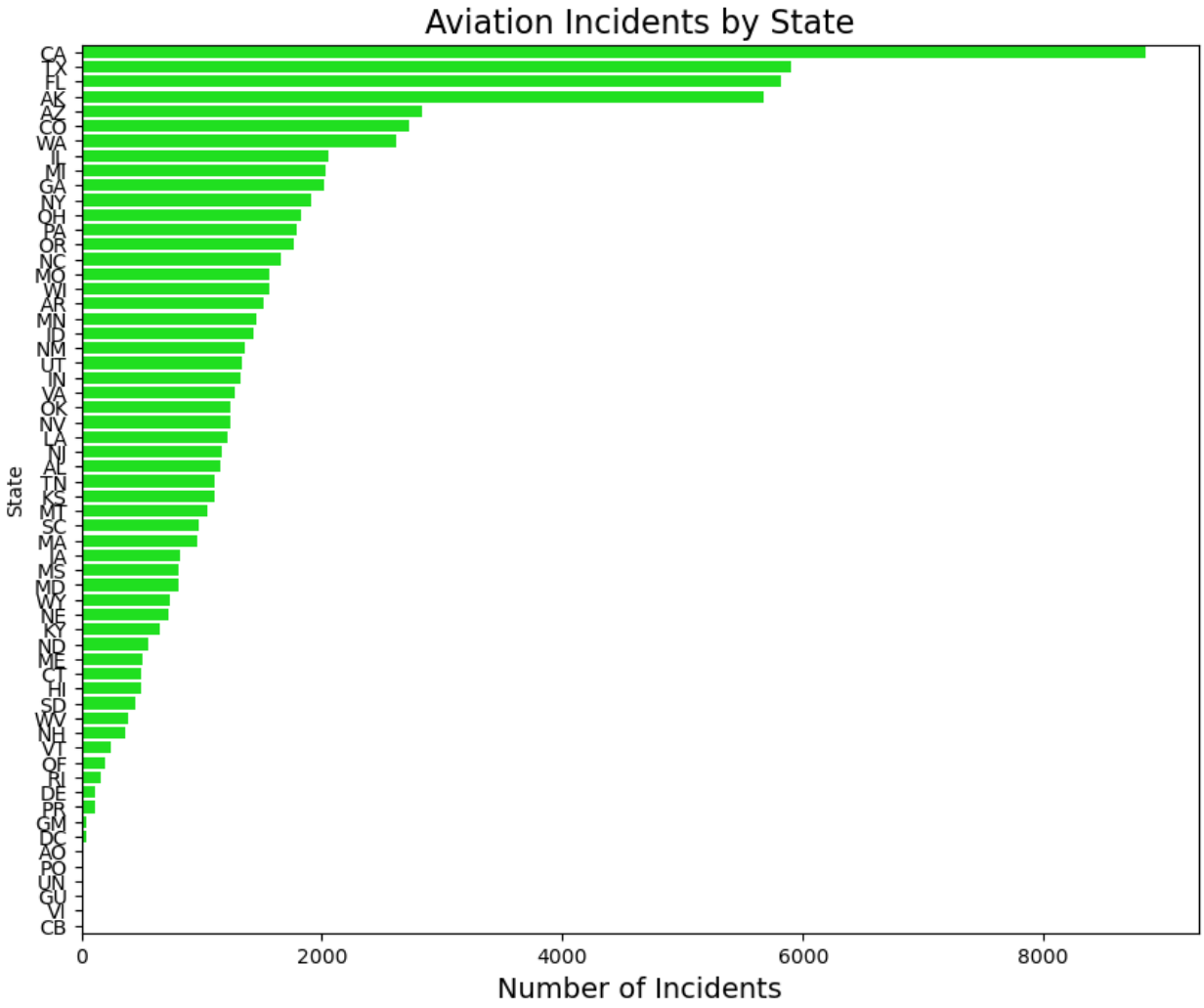



```
<Figure size 640x480 with 0 Axes>
```

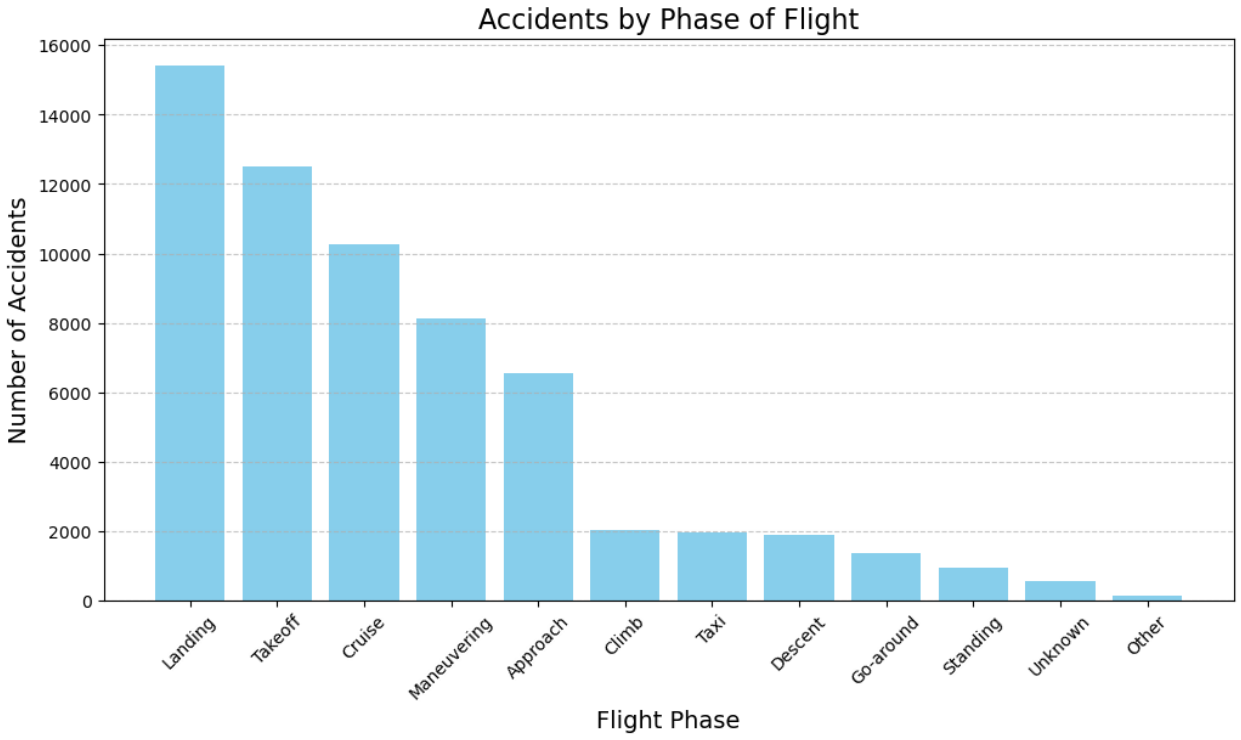
```
# Visualization 2: Distribution of Injury Severity

severity_counts = aviation_data['Injury.Severity'].value_counts()

# Plot severity distribution
plt.figure(figsize=(10, 6))
severity_counts.plot(kind='bar', color='magenta')
plt.title('Distribution of Accident Severity')
plt.xlabel('Severity Category', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
plt.savefig('Injury.Severity.png')
```




```
<Figure size 640x480 with 0 Axes>

# Visualization 3: Number of incidents by state
incidents_by_state =
aviation_data['State.Abbreviation'].value_counts()

# Plot incidents by state
plt.figure(figsize=(10, 8))
sns.barplot(x=incidents_by_state.values, y=incidents_by_state.index,
color='lime')
plt.title("Aviation Incidents by State", fontsize=16)
plt.xlabel("Number of Incidents", fontsize=14)
plt.ylabel("State", fontsize=10)
plt.show()
```




```
# Visualization 4: Phase of Flight Analysis using Stacked Bar Plot
phase_counts = aviation_data['Broad.phase.of.flight'].value_counts()

plt.figure(figsize=(12, 6))
plt.bar(phase_counts.index, phase_counts.values, color='skyblue')
plt.title('Accidents by Phase of Flight', fontsize=16)
plt.xlabel('Flight Phase', fontsize=14)
plt.ylabel('Number of Accidents', fontsize=14)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add a grid for better
readability
plt.show()
plt.savefig('Phase.of.Flight.png')
```




```
<Figure size 640x480 with 0 Axes>
```

```
# Aircraft Damage Analysis
damage_counts = aviation_data['Aircraft.damage'].value_counts()
print("\
Aircraft Damage Distribution:")
print(damage_counts)
```

```
Aircraft Damage Distribution:
Aircraft.damage
Substantial    63951
Destroyed      18537
Minor           2575
Unknown           92
Name: count, dtype: int64
```

```
# 4. Weather Impact Analysis
weather_counts = aviation_data['Weather.Condition'].value_counts()
print("\
Weather Condition Distribution:")
print(weather_counts)
```

```
Weather Condition Distribution:
Weather.Condition
VMC    77228
IMC     5966
UNK      831
```

```
Unk      237
Name: count, dtype: int64
```

 # 5. Evaluation The models are evaluated to ensure they meet the business objectives. This involves assessing the accuracy and reliability of the models and determining if they provide actionable insights. These are my findings:

Trends in Aviation Incidents:

- Key Finding: Over the years, the number of aviation incidents shows fluctuations, with some years experiencing more incidents than others. This can highlight periods of heightened risk or improvements due to safety interventions.
- Evaluation: If the number of incidents has been increasing, it signals a need for revisiting safety measures, operational protocols, or regulatory guidelines. Conversely, a decrease might indicate successful safety improvements or better industry practices.

State-wise Distribution of Incidents:

- Key Finding: Certain states have higher incident counts. States like California and Florida (depending on the dataset) are more prone to incidents.
- Evaluation: States with a high number of incidents can be the focus of targeted safety interventions. Local aviation authorities may need to review their safety protocols or conduct more frequent safety audits in these regions. It could also indicate more air traffic in these states, which warrants a review of air traffic control systems.

Severity of Injuries:

- Key Finding: The analysis revealed the severity of injuries, including fatalities and serious injuries, highlighting the types of accidents that result in the most severe outcomes.
- Evaluation: The high frequency of fatalities or serious injuries indicates that better safety standards and precautionary measures may be needed, particularly in high-risk situations. To mitigate these risks, the aviation industry could consider improving emergency response procedures, aircraft safety, and pilot training.

Weather Conditions and Impact:

- Key Finding: Weather conditions like low visibility, fog, and thunderstorms were common in incidents. Adverse weather conditions are a significant factor contributing to aviation incidents.
- Evaluation: Improved forecasting systems, better pilot training during adverse weather, and advanced aviation technology (such as weather radar and automated systems) could help mitigate these risks. Airlines and airports might also need to implement stricter operational protocols during severe weather events.

Aircraft Damage Analysis:

- Key Finding: The majority of accidents involved aircraft damage ranging from minor to substantial, with a small portion resulting in destruction.
- Evaluation: These findings suggest that while many accidents result in less severe damage, improvements in aircraft design, safety equipment, or materials could reduce the number of accidents that lead to substantial damage.

Phases of Flight Analysis:

- Key Finding: The analysis showed the number of accidents occurring during various phases of flight, such as takeoff, cruising, and landing.
- Evaluation: Accidents occurring during takeoff or landing may suggest areas for improvement in pilot training, air traffic control, or aircraft design, particularly for safer landings and takeoff procedures.

 # 6. Deployment The insights derived from the analysis are actionable and can be used to inform stakeholders about key safety issues. The following deployment steps can be taken:

Safety Measures for Specific States:

Focus on states with a high incidence of accidents. Stakeholders such as local aviation authorities and airport management can use this information to conduct more frequent safety audits and invest in additional safety measures, such as better navigation systems, or even closing the gap in staffing and training.

Improvement in Aircraft Safety:

Insights from aircraft damage and injury severity can inform better safety standards for aircraft design, such as improved crash-resistant materials, more effective evacuation protocols, and enhanced emergency landing systems.

Weather-Related Protocols:

Airlines and airports should develop better weather protocols, especially in states that frequently face adverse weather. Investing in better weather radar systems, enhanced training for pilots to operate in poor weather conditions, and real-time weather monitoring systems will reduce weather-related incidents.

Focused Training on High-Risk Phases:

The high number of accidents during takeoff and landing suggests a need for improved pilot training during these phases. Simulations and additional focus on training for these critical stages of flight could help reduce the risk of accidents.

Regulatory Oversight:

Regulatory bodies should use this data to improve safety regulations, focusing on high-risk periods (e.g., specific weather conditions, times of day, or particular aircraft types) and ensuring compliance through regular inspections.

Public Safety Reports:

Share summarized findings with public stakeholders, including aviation companies, safety organizations, and passengers. This can lead to informed decisions on travel and safety measures that ensure better protection in aviation.

```
# Save cleaned data for reproducibility
aviation_data.to_csv('Cleaned_AviationData.csv', index=False)
```