```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

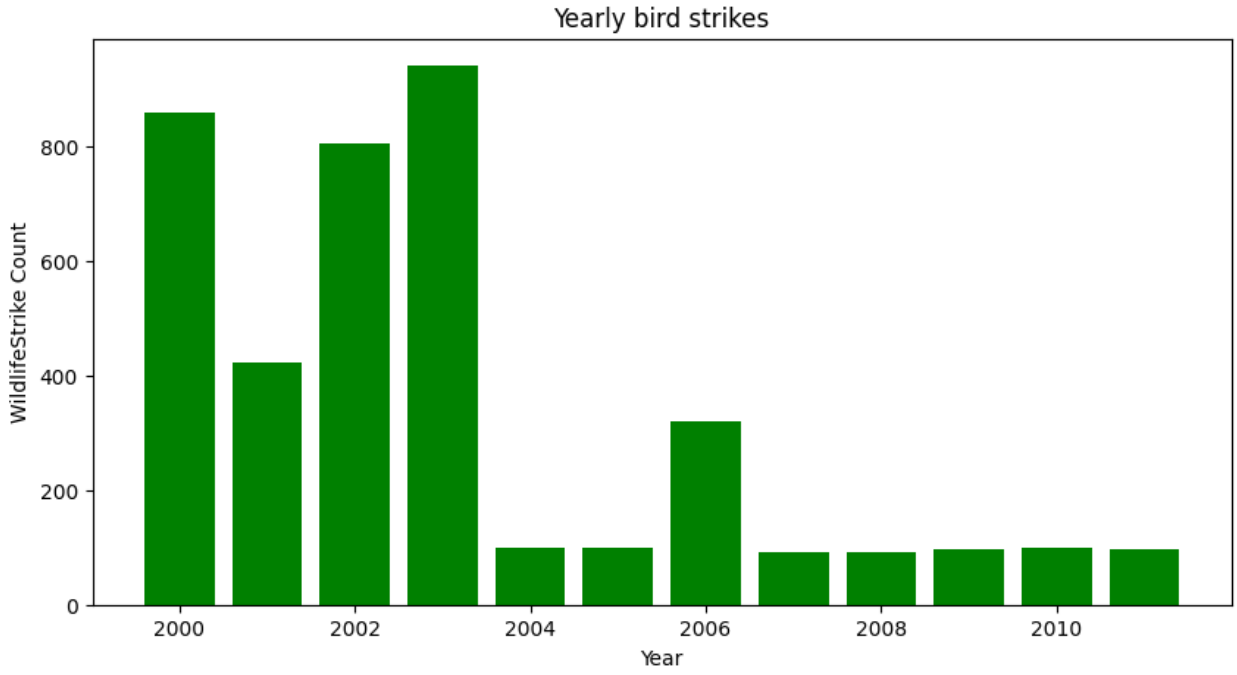
Importing the Necessary Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def Data_Reading(path):
    df = pd.read_csv(path)
    return df

def plot_bird_strikes_over_years(df):
    df['FlightDate'] = pd.to_datetime(df['FlightDate'])
    df['Year'] = df['FlightDate'].dt.year
    plt.figure(figsize=(10, 5))
    plt.bar(df['Year'], df['Wildlife: Number Struck Actual'],
color='green')
    plt.xlabel('Year')
    plt.ylabel('WildlifeStrike Count')
    plt.title('Yearly bird strikes')
    plt.show()

def us_airlines_with_highest_strikes(df):
    airlines_data = df[df['Aircraft: Airline/Operator'].notna() &
df['Aircraft: Airline/Operator'].str.contains('AIRLINES')]
    plt.figure(figsize=(10, 5))
    top_airlines = airlines_data['Aircraft:
Airline/Operator'].value_counts().nlargest(10)
    sns.barplot(x=top_airlines.values, y=top_airlines.index,
palette='viridis')
    plt.title('Top ten US airlines which encountered bird strikes')
    plt.xlabel('Number of Bird Strikes')
    plt.ylabel('Airline and Operator')
    plt.show()

def most_incident_airports(df):
    plt.figure(figsize=(15, 5))
    top_airports = df['Airport: Name'].value_counts().nlargest(50)
    sns.barplot(x=top_airports.values, y=top_airports.index,
palette='viridis')
    plt.title('Top 50 Airports with the most incidents of bird
strikes')
    plt.xlabel('Number of Bird Strikes')
```

```
    plt.ylabel('Name of Airport')
    plt.show()

def annual_expense_caused_by_incidents(df):
    plt.figure(figsize=(10, 5))
    sns.lineplot(x='Year', y='Cost: Total $', data=df, estimator=sum,
ci=None, marker='o', color='blue')
    plt.yscale('log')
    plt.title('Annual Expense Caused by Bird Incidentss')
    plt.xlabel('Year')
    plt.ylabel('Total Expense in dollars')
    plt.show()

def plot_bird_strike_distribution(df):
    plt.figure(figsize=(12, 6))
    sns.countplot(x='When: Phase of flight', data=df, order=df['When:
Phase of flight'].value_counts().index, palette='viridis')
    plt.title('Distribution of Bird Strikes Across Flight Phases')
    plt.xlabel('Flight Phase')
    plt.ylabel('Bird Strike Count')
    plt.xticks(rotation=45, ha='right')
    plt.show()

def altitude_of_airplanes_at_time_of_strike(df):
    df['Feet above ground'] = pd.to_numeric(df['Feet above ground'],
errors='coerce')
    plt.figure(figsize=(10, 5))
    plt.bar(df['Feet above ground'], df['Wildlife: Number Struck
Actual'])
    plt.title('Airplane Altitude During Bird Strikes')
    plt.xlabel('Altitude in feet')
    plt.ylabel('Count of Bird Strikes')
    plt.show()

def plot_flight_phase_at_strike(df):
    plt.figure(figsize=(10, 5))
    sns.countplot(x='When: Phase of flight', data=df, order=df['When:
Phase of flight'].value_counts().index, palette='viridis')
    plt.title('Flight Phase at Time of Bird Strike')
    plt.xlabel('Flight Phase')
    plt.ylabel('Count of Bird Strikes')
    plt.xticks(rotation=45, ha='right')
    plt.show()

def plot_airplane_altitudes(df):
    plt.figure(figsize=(10, 5))
    sns.boxplot(x='When: Phase of flight', y='Feet above ground',
hue='When: Phase of flight', data=df,
                order=df['When: Phase of
flight'].value_counts().index, palette='viridis')
```

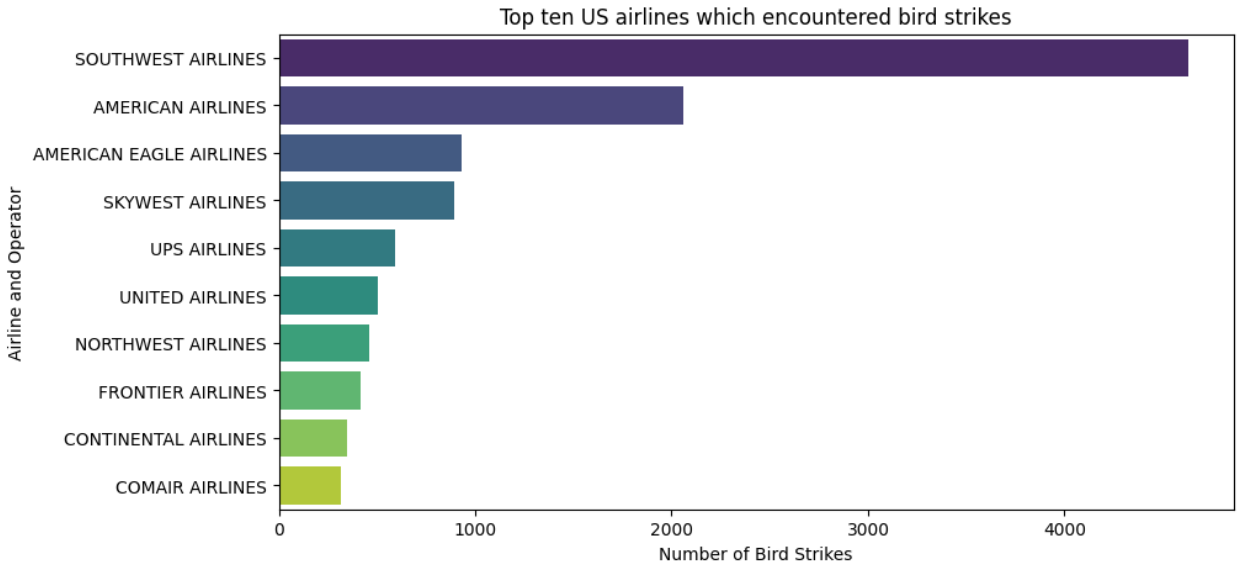
```
    plt.title('Average Altitude of Airplanes in Various Flight Phases
During Bird Strike')
    plt.xlabel('Flight Phase')
    plt.ylabel('Altitude in feet')
    plt.legend(title='Flight Phase', loc='upper right',
bbox_to_anchor=(1.2, 1))
    plt.show()

def visualize_bird_strike_impact(df):
    plt.figure(figsize=(10, 5))
    sns.countplot(x='Effect: Impact to flight', data=df,
order=df['Effect: Impact to flight'].value_counts().index,
palette='Set2')
    plt.title('Bird Strikes and Their Impact on Flight')
    plt.xlabel('Flight Impact')
    plt.ylabel('Count of Bird Strikes')
    plt.xticks(rotation=45, ha='right')
    plt.show()

def plot_bird_strike_impact_by_altitude(df):
    plt.figure(figsize=(10, 5))
    sns.countplot(x='Effect: Impact to flight', data=df,
palette='viridis')
    plt.title('Bird Strike Impact at Various Altitudes')
    plt.xlabel('Flight Impact')
    plt.ylabel('Count of Bird Strikes')
    plt.xticks(rotation=45, ha='right')
    plt.show()

def visualize_bird_strike_impact(df):
    plt.figure(figsize=(10, 5))
    sns.countplot(x='Effect: Impact to flight', hue='Pilot warned of
birds or wildlife?', data=df, palette='pastel')
    plt.title('Impact of Bird Strikes and Pilot Alerts')
    plt.xlabel('Flight Impact')
    plt.ylabel('Count of Bird Strikes')
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Pilot Warned', loc='upper right',
bbox_to_anchor=(1.25, 1))
    plt.show()

path = "/content/drive/MyDrive/Unified Mentor/Bird Strikes data.csv"
df = Data_Reading(path)
df.head()

{"type":"dataframe","variable_name":"df"}

plot_bird_strikes_over_years(df)

<ipython-input-4-88bb02e3ae61>:2: UserWarning: Could not infer format,
so each element will be parsed individually, falling back to
```

```
`dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df['FlightDate'] = pd.to_datetime(df['FlightDate'])
```



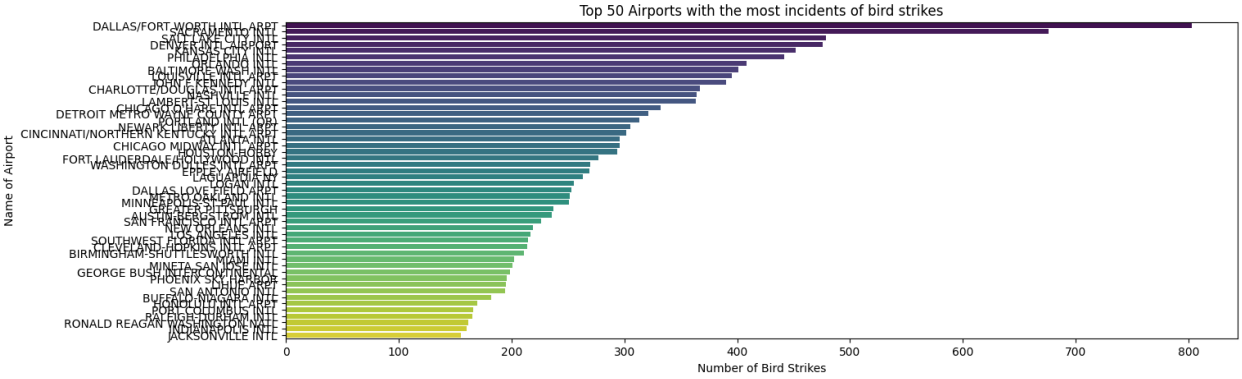

```
us_airlines_with_highest_strikes(df)
```

```
<ipython-input-5-3369de733b6e>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_airlines.values, y=top_airlines.index, palette='viridis')
```



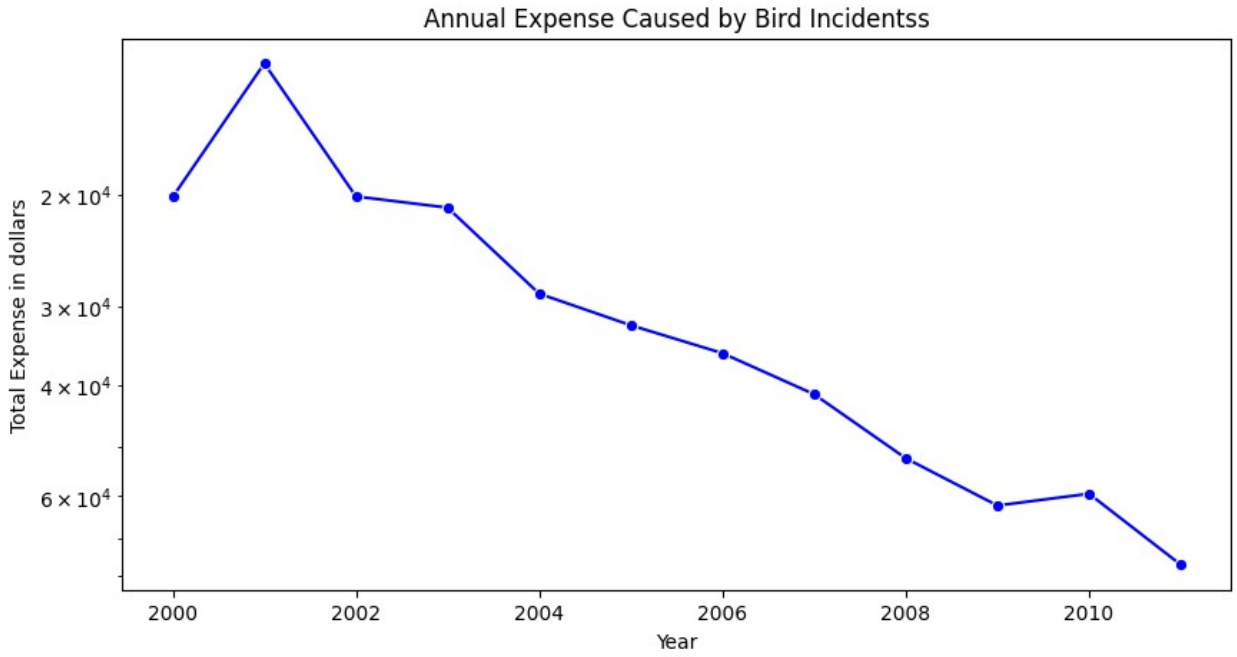

```
most_incident_airports(df)
```

```
<ipython-input-27-f628da4b3423>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_airports.values, y=top_airports.index, palette='viridis')
```



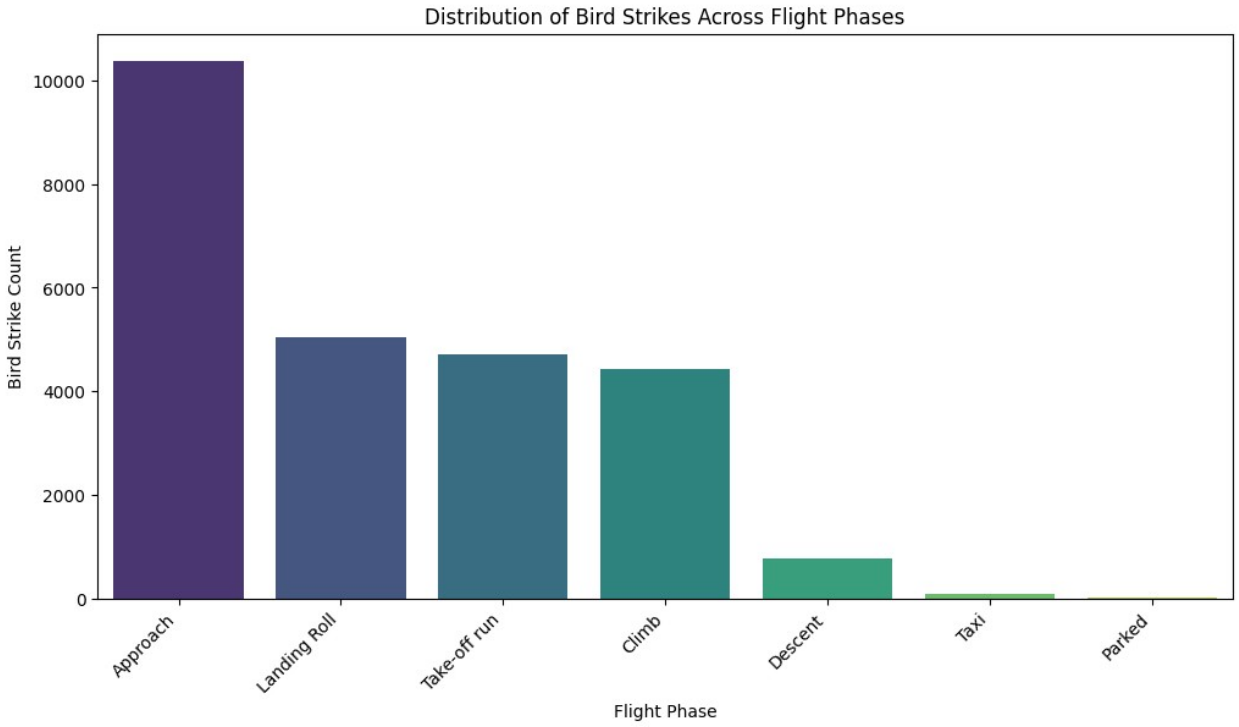

```
annual_expense_caused_by_incidents(df)
```

```
<ipython-input-7-06a3652a9ff0>:3: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(x='Year', y='Cost: Total $', data=df, estimator=sum, ci=None, marker='o', color='blue')
```

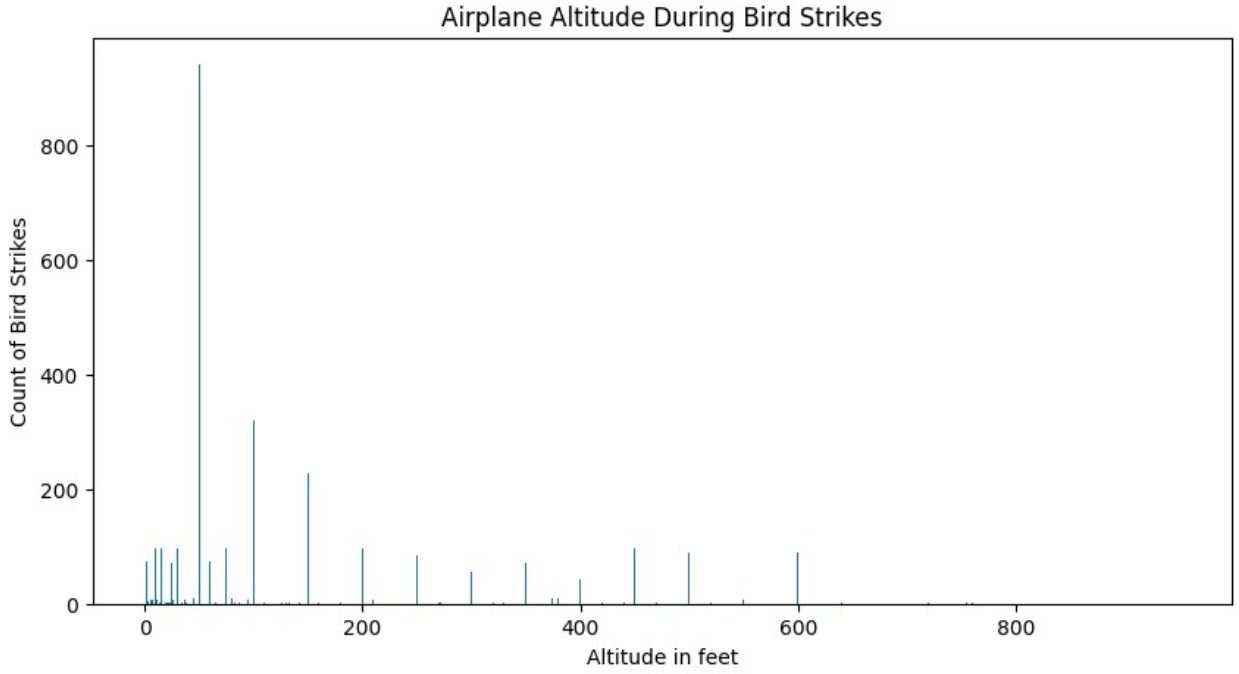



```
plot_bird_strike_distribution(df)
```

```
<ipython-input-8-d365c3dc5f3b>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='When: Phase of flight', data=df, order=df['When: Phase of flight'].value_counts().index, palette='viridis')
```



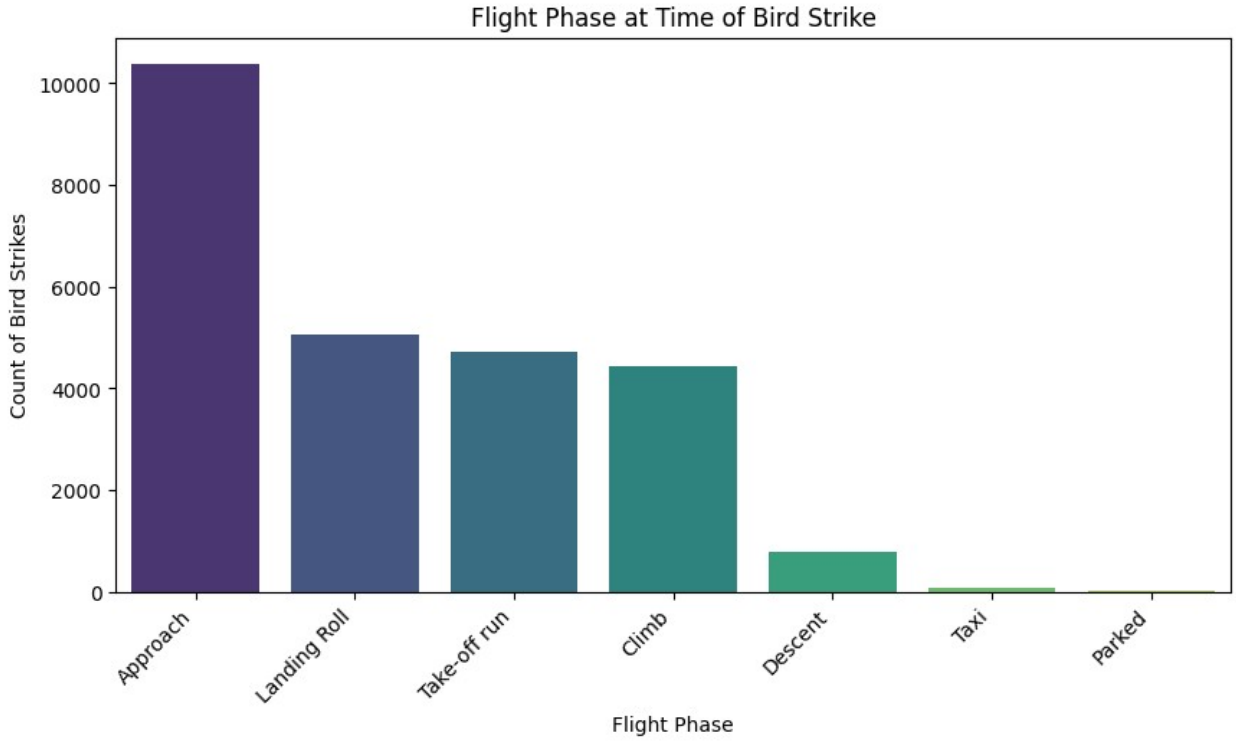

```
altitude_of_airplanes_at_time_of_strike(df)
```



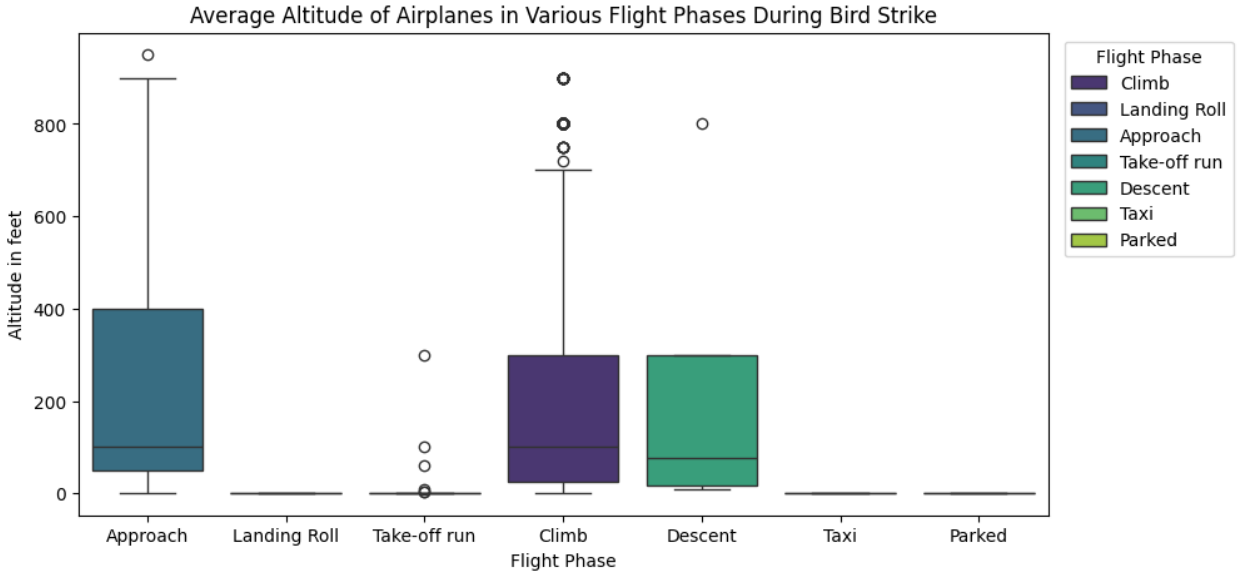

```
plot_flight_phase_at_strike(df)
```

```
<ipython-input-10-a66839e3d25b>:3: FutureWarning:
```

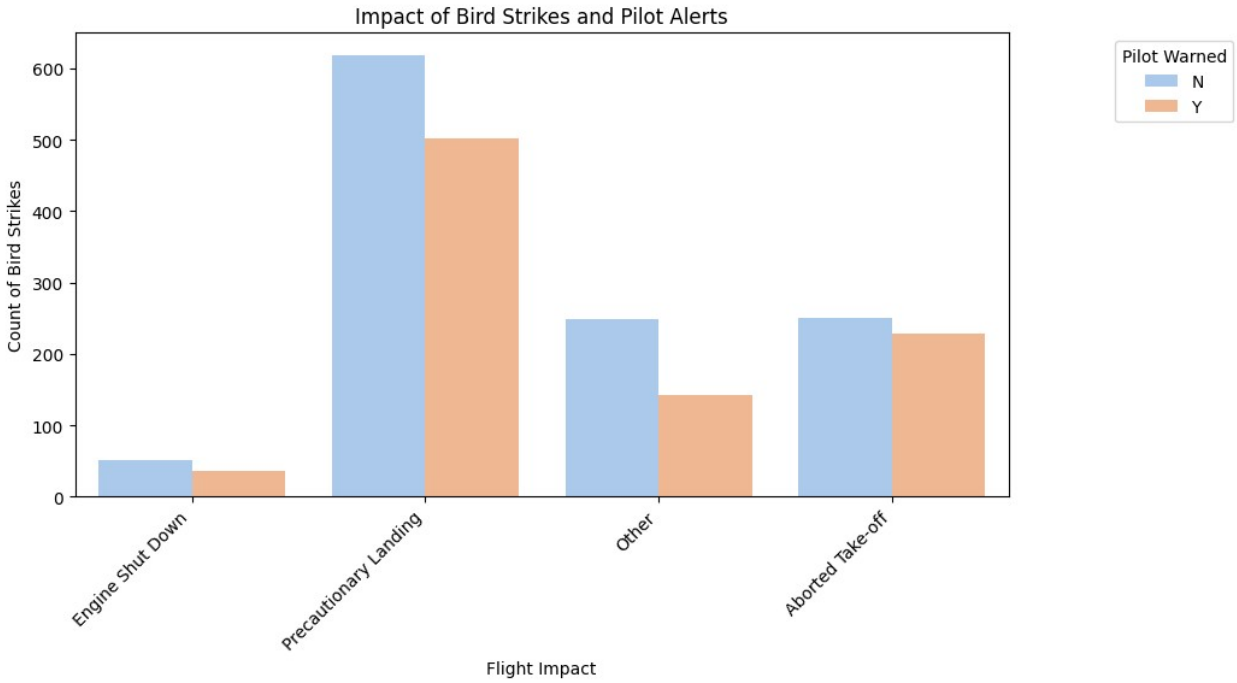
```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='When: Phase of flight', data=df, order=df['When: Phase of flight'].value_counts().index, palette='viridis')
```



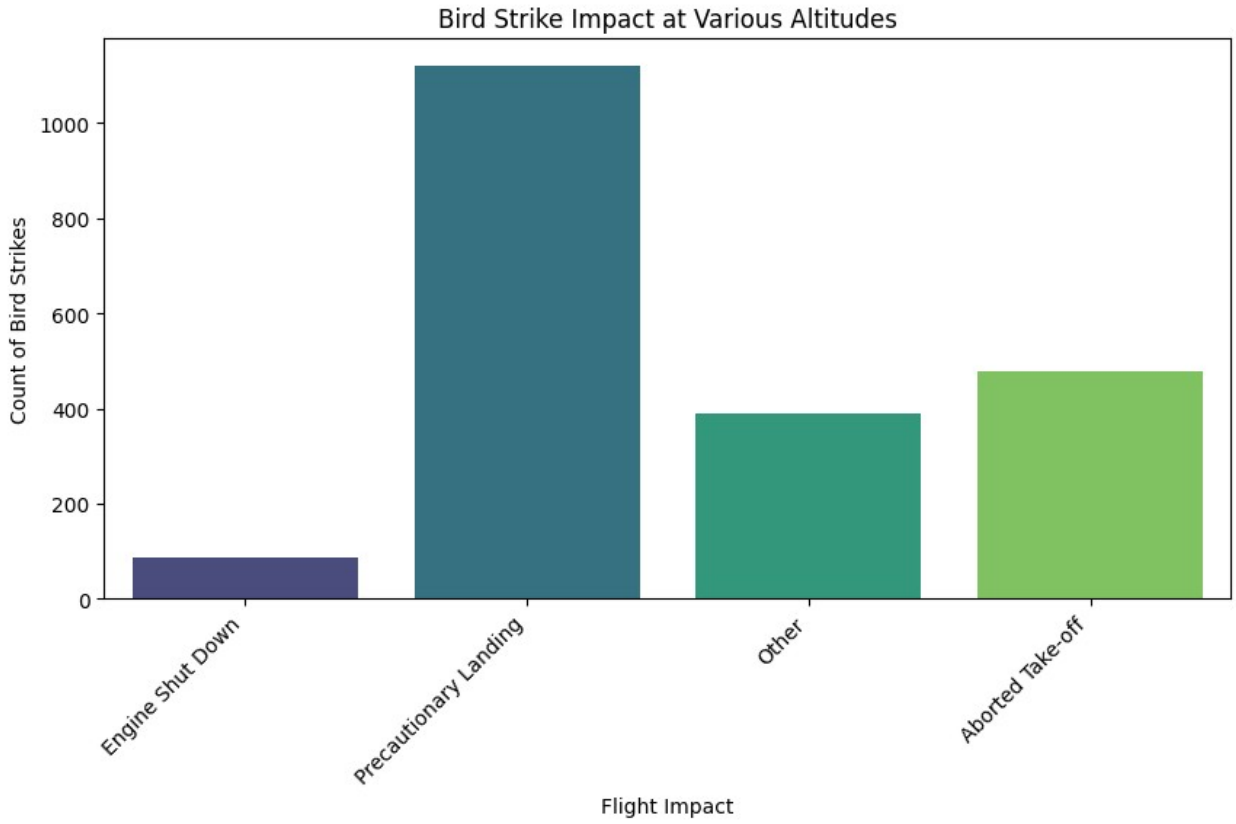

```
plot_airplane_altitudes(df)
```



```
visualize_bird_strike_impact(df)
```




```
plot_bird_strike_impact_by_altitude(df)

<ipython-input-13-da10a5538e32>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Effect: Impact to flight', data=df, palette='viridis')
```




```
visualize_bird_strike_impact(df)
```