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

In [3]:
```
df = pd.read_csv(r'C:\Users\kriti\OneDrive\Desktop\UM\Bird Strikes Data
```

In [4]:
```
df.head()
```

Out[4]:

	Record ID	Aircraft: Type	Airport: Name	Altitude bin	Aircraft: Make/Model	Wildlife: Number struck	Wildlife: Number Struck Actual	Effe Impact flig
0	202152	Airplane	LAGUARDIA NY	> 1000 ft	B-737-400	Over 100	859	Engine Sh Dov
1	208159	Airplane	DALLAS/FORT WORTH INTL ARPT	< 1000 ft	MD-80	Over 100	424	No
2	207601	Airplane	LAKEFRONT AIRPORT	< 1000 ft	C-500	Over 100	261	No
3	215953	Airplane	SEATTLE-TACOMA INTL	< 1000 ft	B-737-400	Over 100	806	Precautiona Landi
4	219878	Airplane	NORFOLK INTL	< 1000 ft	CL-RJ100/200	Over 100	942	No

5 rows × 26 columns

In [5]:
```
df.shape
```

Out[5]:
```
(25558, 26)
```

In [6]:

```
# check the columns
df.columns
```

Out[6]:

```
Index(['Record ID', 'Aircraft: Type', 'Airport: Name', 'Altitude bin',
       'Aircraft: Make/Model', 'Wildlife: Number struck',
       'Wildlife: Number Struck Actual', 'Effect: Impact to flight',
       'FlightDate', 'Effect: Indicated Damage',
       'Aircraft: Number of engines?', 'Aircraft: Airline/Operator',
       'Origin State', 'When: Phase of flight', 'Conditions: Precipita
tion',
       'Remains of wildlife collected?',
       'Remains of wildlife sent to Smithsonian', 'Remarks', 'Wildlif
e: Size',
       'Conditions: Sky', 'Wildlife: Species',
       'Pilot warned of birds or wildlife?', 'Cost: Total $',
       'Feet above ground', 'Number of people injured', 'Is Aircraft L
arge?'],
      dtype='object')
```

In [7]:

```
# information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25558 entries, 0 to 25557
Data columns (total 26 columns):
 #   Column                                   Non-Null Count  Dtype 
---  ------                                   --------------  ----- 
 0   Record ID                                25558 non-null  int64 
 1   Aircraft: Type                           25429 non-null  object
 2   Airport: Name                            25429 non-null  object
 3   Altitude bin                             25429 non-null  object
 4   Aircraft: Make/Model                     25558 non-null  object
 5   Wildlife: Number struck                  25429 non-null  object
 6   Wildlife: Number Struck Actual           25558 non-null  int64 
 7   Effect: Impact to flight                 25429 non-null  object
 8   FlightDate                               25429 non-null  object
 9   Effect: Indicated Damage                 25558 non-null  object
 10  Aircraft: Number of engines?             25291 non-null  object
 11  Aircraft: Airline/Operator               25429 non-null  object
 12  Origin State                             25109 non-null  object
 13  When: Phase of flight                    25429 non-null  object
 14  Conditions: Precipitation                25558 non-null  object
 15  Remains of wildlife collected?           25558 non-null  bool  
 16  Remains of wildlife sent to Smithsonian  25558 non-null  bool  
 17  Remarks                                  20787 non-null  object
 18  Wildlife: Size                           25429 non-null  object
 19  Conditions: Sky                          25558 non-null  object
 20  Wildlife: Species                        25558 non-null  object
 21  Pilot warned of birds or wildlife?       25429 non-null  object
 22  Cost: Total $                            25558 non-null  object
 23  Feet above ground                        25429 non-null  object
 24  Number of people injured                 25558 non-null  int64 
 25  Is Aircraft Large?                       25429 non-null  object
dtypes: bool(2), int64(3), object(21)
memory usage: 4.7+ MB
```

In [8]:

```
# columns with categorical values
df.select_dtypes(include=['object']).columns
```

Out[8]:

```
Index(['Aircraft: Type', 'Airport: Name', 'Altitude bin',
       'Aircraft: Make/Model', 'Wildlife: Number struck',
       'Effect: Impact to flight', 'FlightDate', 'Effect: Indicated Da
mage',
       'Aircraft: Number of engines?', 'Aircraft: Airline/Operator',
       'Origin State', 'When: Phase of flight', 'Conditions: Precipita
tion',
       'Remarks', 'Wildlife: Size', 'Conditions: Sky', 'Wildlife: Spec
ies',
       'Pilot warned of birds or wildlife?', 'Cost: Total $',
       'Feet above ground', 'Is Aircraft Large?'],
      dtype='object')
```

In [9]:

```
# columns with numerical values
df.select_dtypes(include=['int64', 'float64']).columns
```

Out[9]:

```
Index(['Record ID', 'Wildlife: Number Struck Actual',
       'Number of people injured'],
      dtype='object')
```

In [10]:

```
# check if there are any null values
df.isnull().values.any() # this function returns true and false
```

Out[10]:

```
True
```

In [11]:

```
# check how many null values
df.isnull().values.sum()
```

Out[11]:

```
7035
```

In [12]:

```
# Drop rows with null values
df_cleaned = df.dropna()

# Fill null values with a specific value
df_filled = df.fillna(0)  # Replace null values with 0
```

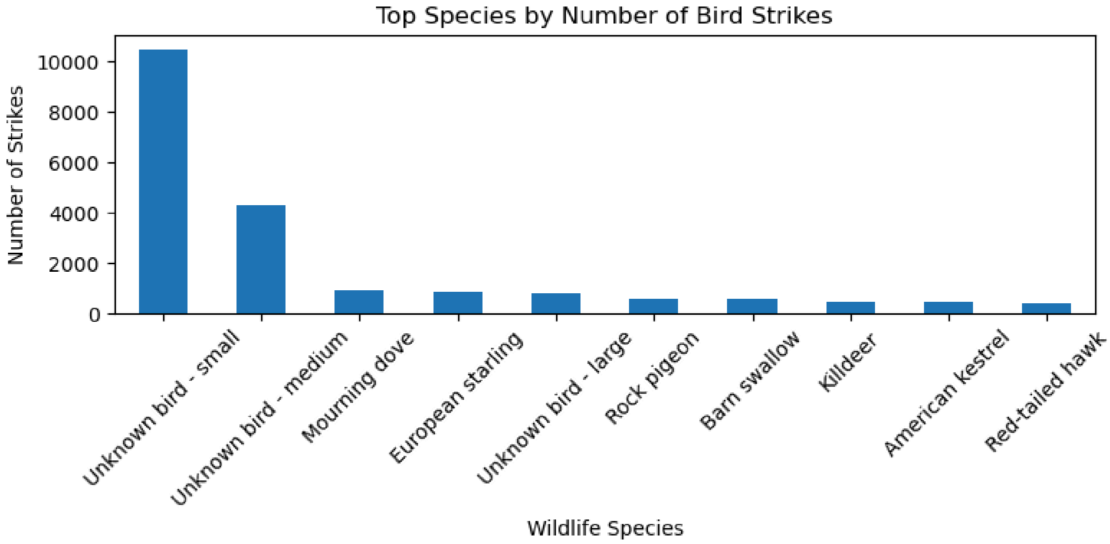
In [13]:

```
# Count the number of bird strikes by Wildlife Species
bird_strikes_count = df['Wildlife: Species'].value_counts()
```

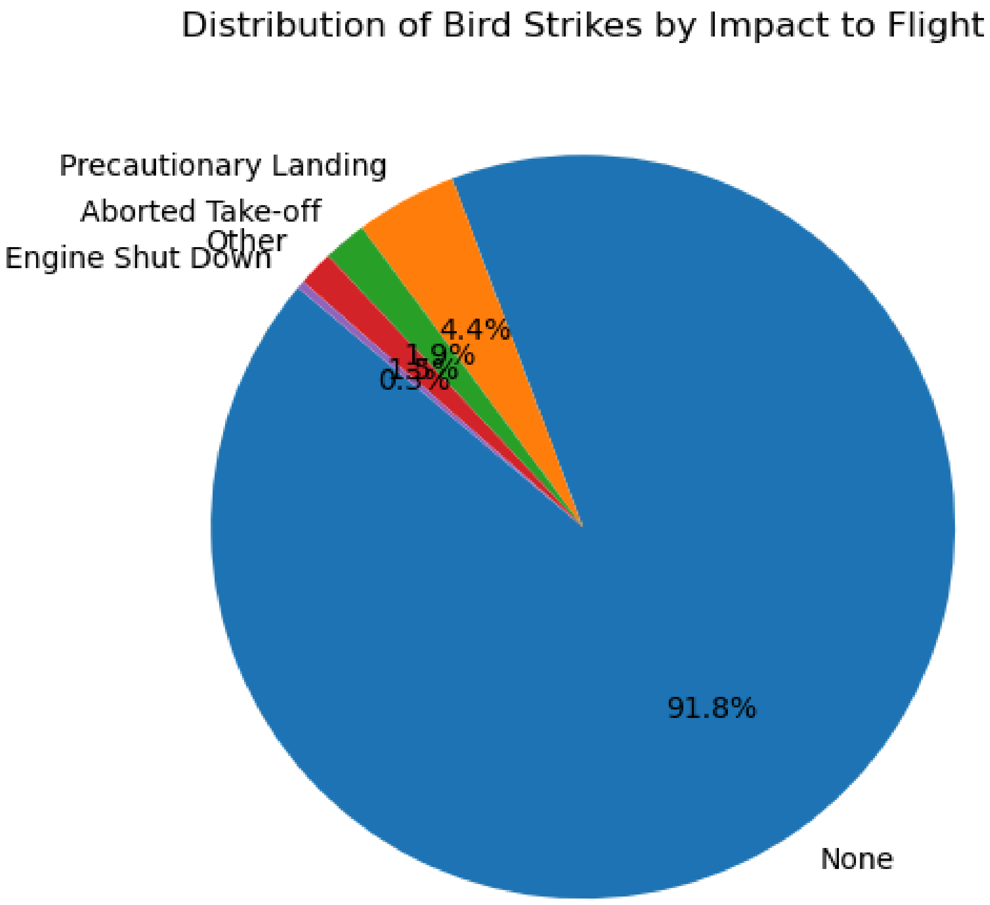
In [14]:

```
# Choose the top N species with the highest number of strikes
top_species_count = bird_strikes_count.head(10)  # Change 10 to the des
```

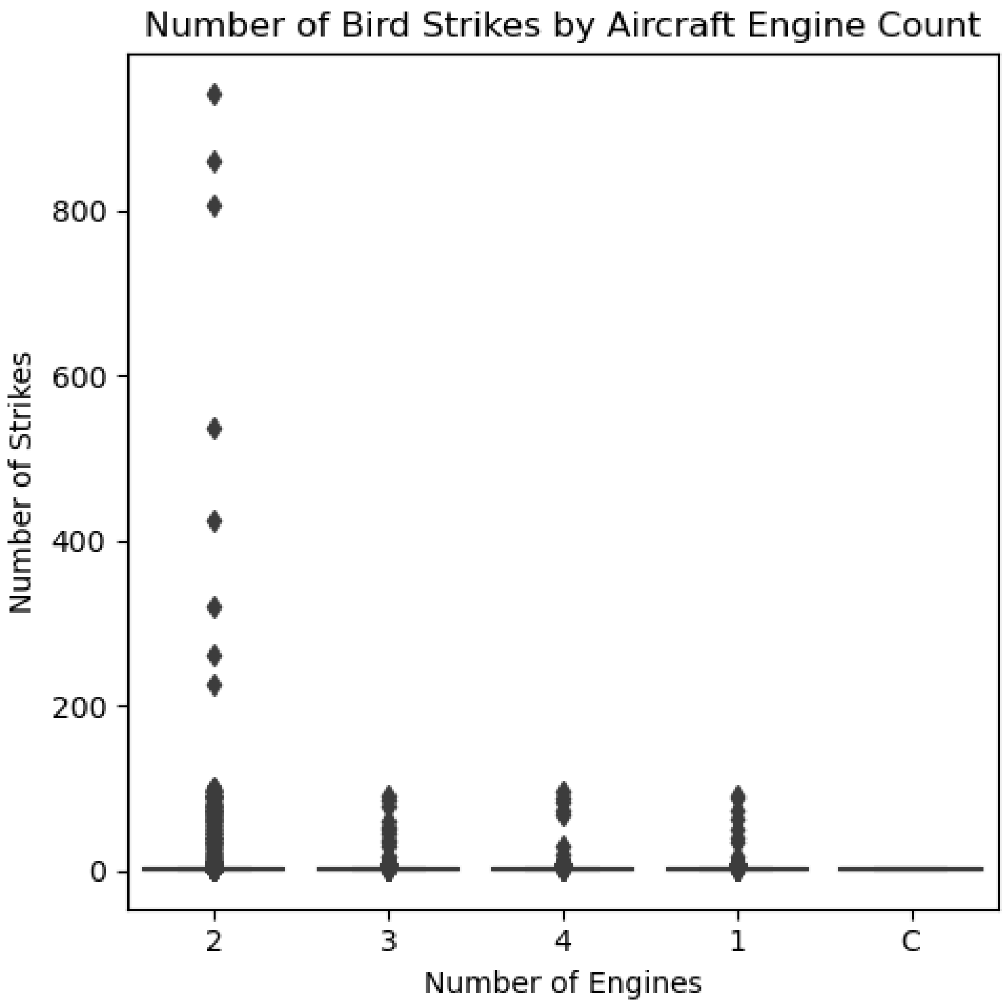
```
In [18]: # Plot a bar chart for the top N species
         plt.figure(figsize=(8, 4))
         top_species_count.plot(kind='bar')
         plt.title('Top Species by Number of Bird Strikes')
         plt.xlabel('Wildlife Species')
         plt.ylabel('Number of Strikes')
         plt.xticks(rotation=45)
         plt.tight_layout()
         plt.show()
```



In [19]:
```python
# Plot a pie chart for the distribution of bird strikes by Effect: Impa
impact_counts = df['Effect: Impact to flight'].value_counts()
plt.figure(figsize=(5, 5))
plt.pie(impact_counts, labels=impact_counts.index, autopct='%1.1f%%', s
plt.title('Distribution of Bird Strikes by Impact to Flight')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

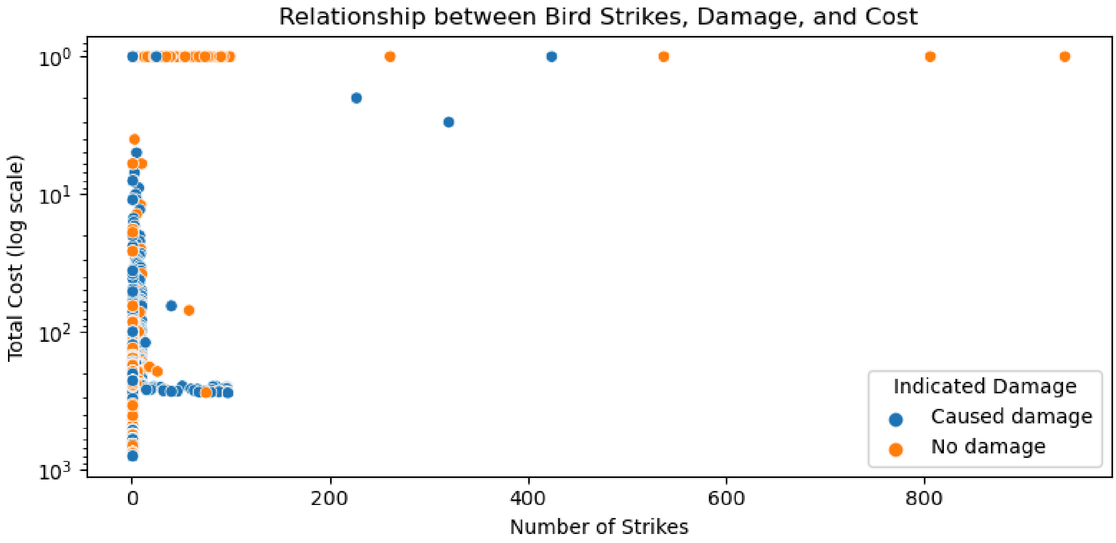


In [20]:
```
# Plot a box plot for the number of bird strikes by Aircraft: Number of
plt.figure(figsize=(5, 5))
sns.boxplot(data=df, x='Aircraft: Number of engines?', y='Wildlife: Num
plt.title('Number of Bird Strikes by Aircraft Engine Count')
plt.xlabel('Number of Engines')
plt.ylabel('Number of Strikes')
plt.tight_layout()
plt.show()
```



In [21]:

```python
# Plot a scatter plot with log scale for the y-axis
plt.figure(figsize=(8, 4))
sns.scatterplot(data=df, x='Wildlife: Number Struck Actual', y='Cost: T
plt.title('Relationship between Bird Strikes, Damage, and Cost')
plt.xlabel('Number of Strikes')
plt.ylabel('Total Cost (log scale)')
plt.yscale('log')  # Set y-axis to log scale
plt.legend(title='Indicated Damage')
plt.tight_layout()
plt.show()
```



When using a log scale, the values on the axis are not evenly spaced like in a linear scale. Instead, each increment on the axis represents a multiple of the base of the logarithm. For example, in a logarithmic base 10 scale, each increment on the y-axis represents a multiple of 10.

The main benefit of using a log scale in a scatter plot is that it can help in visualizing data that has a wide range of values or data points that are clustered at low values with a few extreme outliers. It compresses the higher values, making the plot more interpretable and highlighting patterns that may not be apparent on a linear scale.

In the context of the scatter plot code provided earlier, setting the y-axis to a log scale (plt.yscale('log')) adjusts the scale of the y-axis to better accommodate the wide range of values in the 'Cost: Total $' column, making it easier to read and interpret the data points.

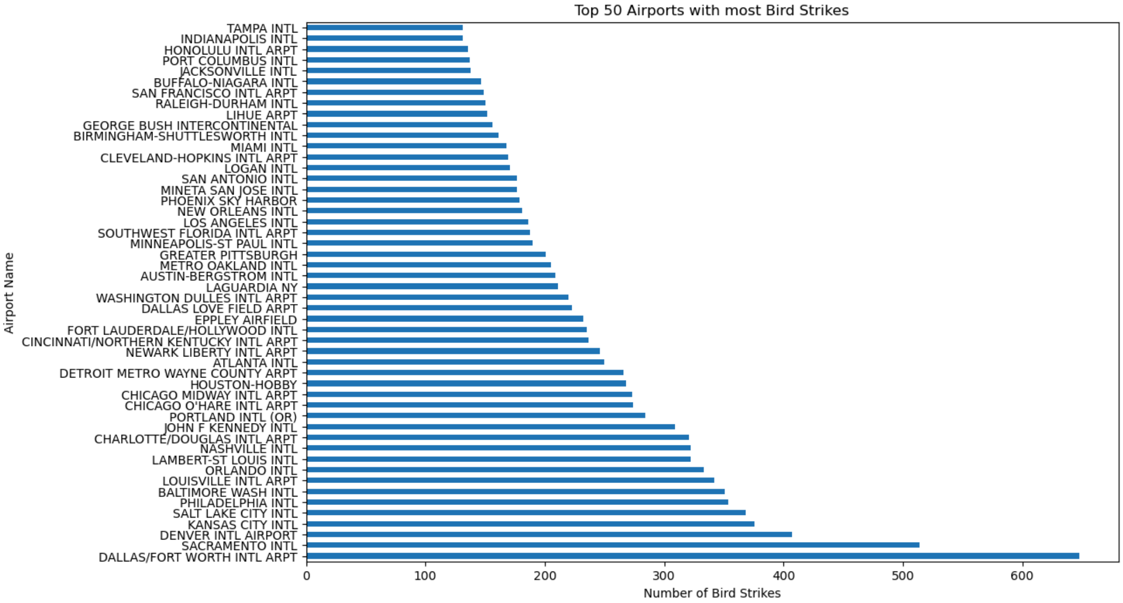
In [18]:

```python
# Filter data for US airlines
us_airlines = df[df['Origin State'] == 'US']
```

In [35]:

```python
# Top 10 US Airlines with most bird strikes
top_10_airlines = us_airlines.groupby('Aircraft: Airline/Operator')['Wi
```

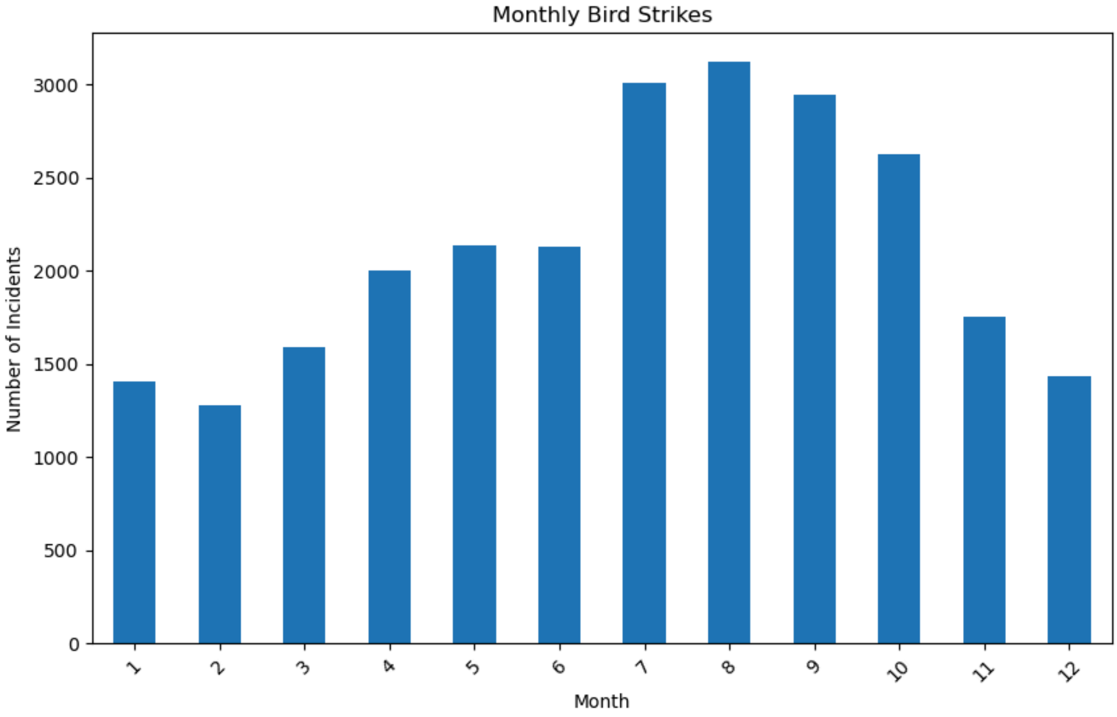
```
In [36]: # Convert 'Wildlife: Number struck' column to numeric
         df['Wildlife: Number struck'] = pd.to_numeric(df['Wildlife: Number stru

         # Filter data for incidents with non-null 'Wildlife: Number struck'
         df_filtered = df.dropna(subset=['Wildlife: Number struck'])

         # Group by 'Airport: Name' and sum the 'Wildlife: Number struck' for ea
         top_airports = df_filtered.groupby('Airport: Name')['Wildlife: Number s

         # Plotting Top 50 Airports with most bird strikes
         plt.figure(figsize=(12, 8))
         top_airports.plot(kind='barh')
         plt.title('Top 50 Airports with most Bird Strikes')
         plt.xlabel('Number of Bird Strikes')
         plt.ylabel('Airport Name')
         plt.show()
```



In [38]:

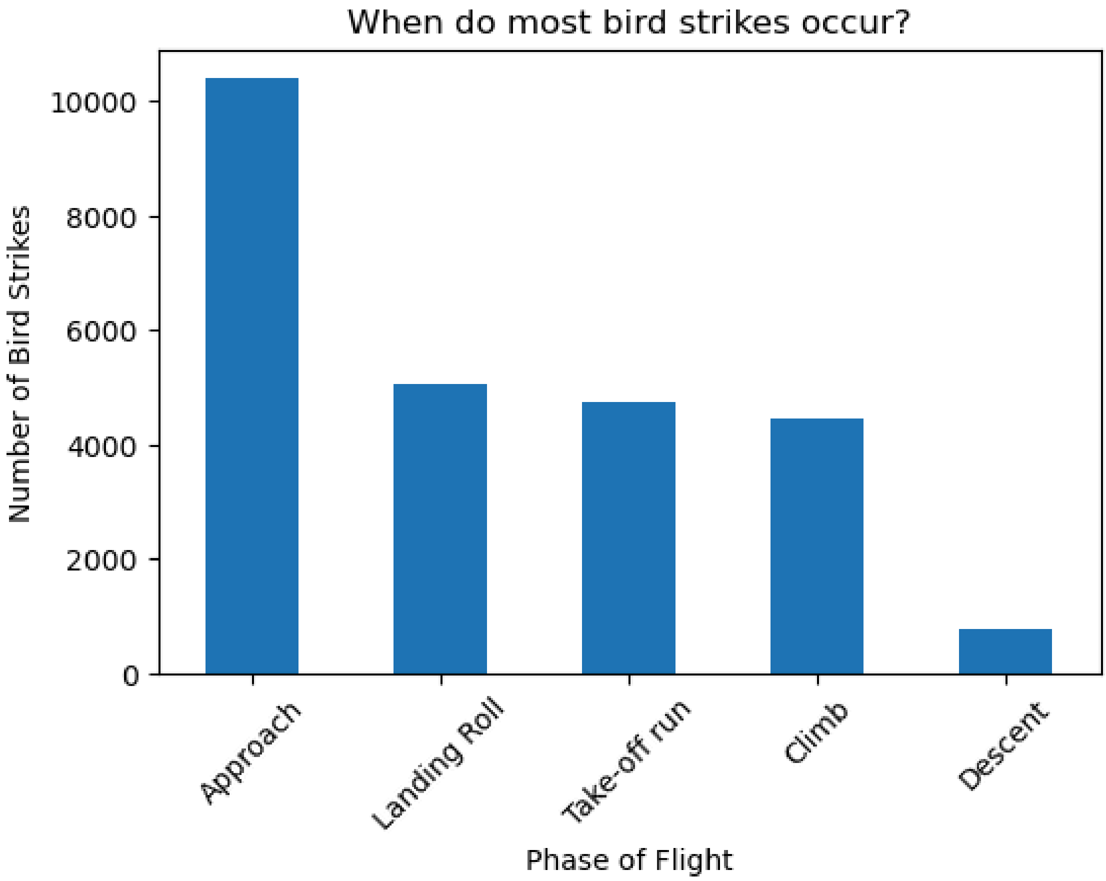
```python
# Convert 'FlightDate' column to datetime
df['FlightDate'] = pd.to_datetime(df['FlightDate'], errors='coerce')

# Filter data for incidents with non-null 'FlightDate'
df_filtered = df.dropna(subset=['FlightDate'])

# Extract month from 'FlightDate' and create a new 'Month' column
df_filtered['Month'] = df_filtered['FlightDate'].dt.month

# Group by month and count the number of incidents
monthly_incidents = df_filtered.groupby('Month').size()

# Plotting Monthly Bird Strikes
plt.figure(figsize=(10, 6))
monthly_incidents.plot(kind='bar')
plt.title('Monthly Bird Strikes')
plt.xlabel('Month')
plt.ylabel('Number of Incidents')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\kriti\AppData\Local\Temp\ipykernel_18176\1838022889.py:8: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  df_filtered['Month'] = df_filtered['FlightDate'].dt.month
```



In [23]:

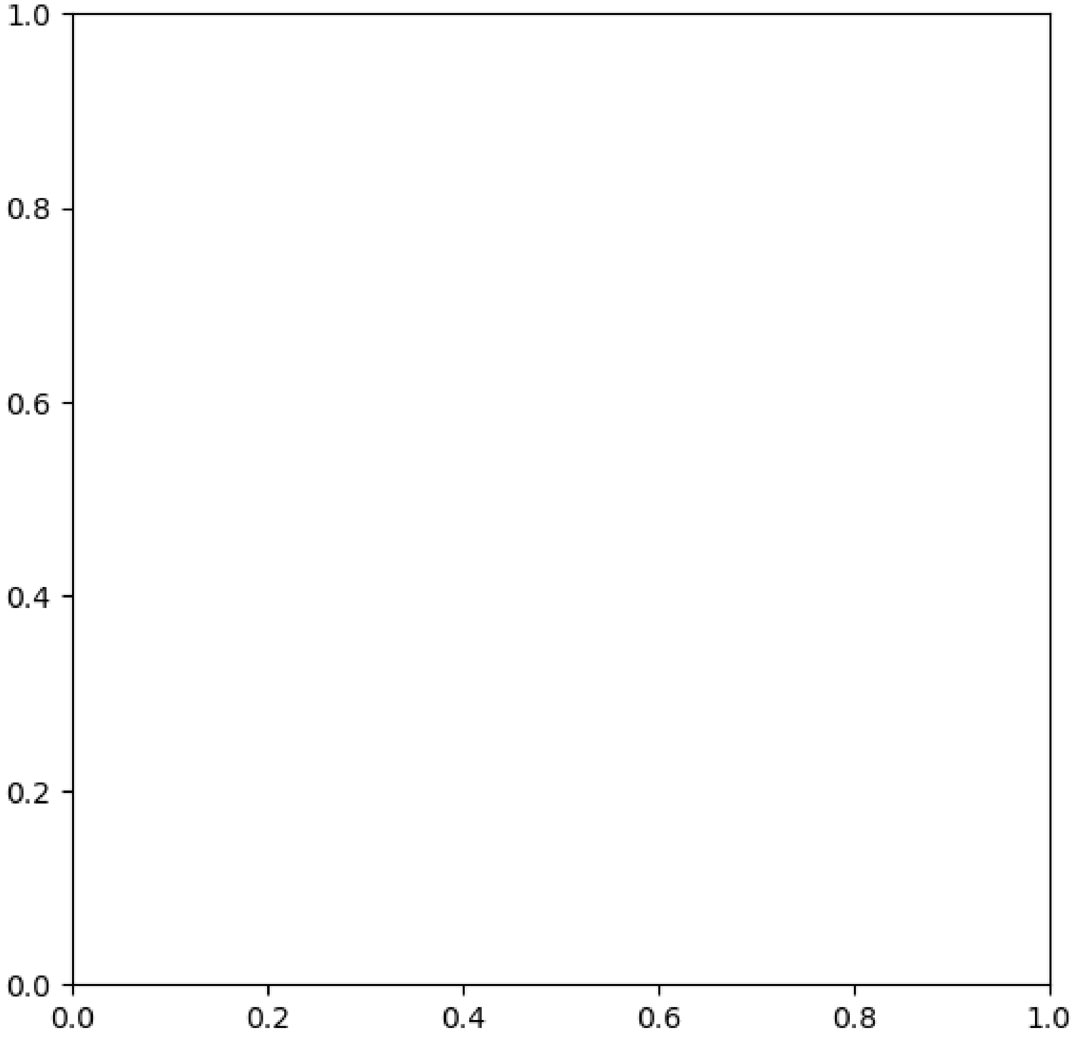
```
# When do most bird strikes occur?
most_strikes_phase = df['When: Phase of flight'].value_counts().nlarges

# Plotting When do most bird strikes occur
plt.figure(figsize=(6, ))
most_strikes_phase.plot(kind='bar')
plt.title('When do most bird strikes occur?')
plt.xlabel('Phase of Flight')
plt.ylabel('Number of Bird Strikes')
plt.xticks(rotation=45)
plt.show()
```



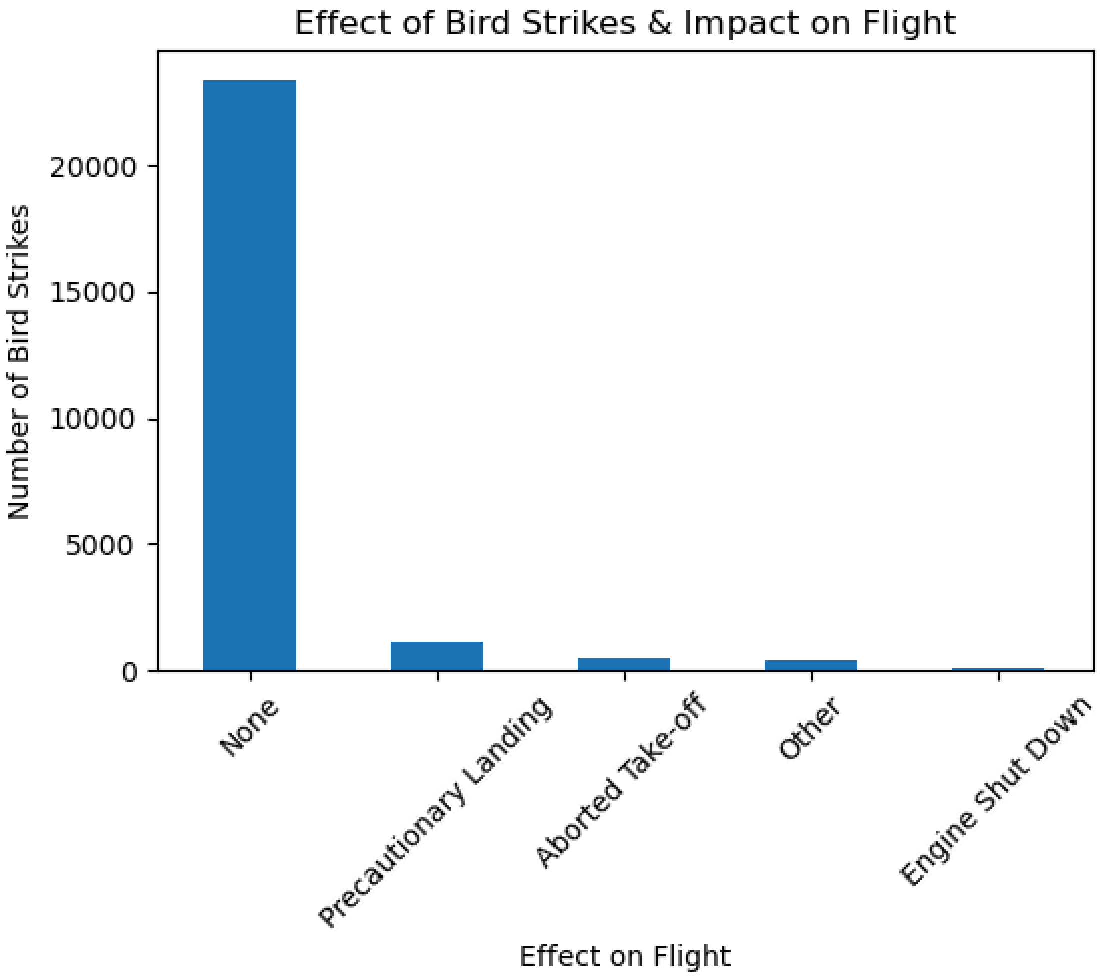
In [25]:

```python
# Altitude of airplanes at the time of strike
altitude_strike = df.groupby('Altitude bin')['Wildlife: Number struck']

# Plotting Altitude of airplanes at the time of strike
plt.figure(figsize=(6, 6))
altitude_strike.plot(kind='bar')
plt.title('Altitude of airplanes at the time of Bird Strikes')
plt.xlabel('Altitude Bin')
plt.ylabel('Number of Bird Strikes')
plt.xticks(rotation=45)
plt.show()
```

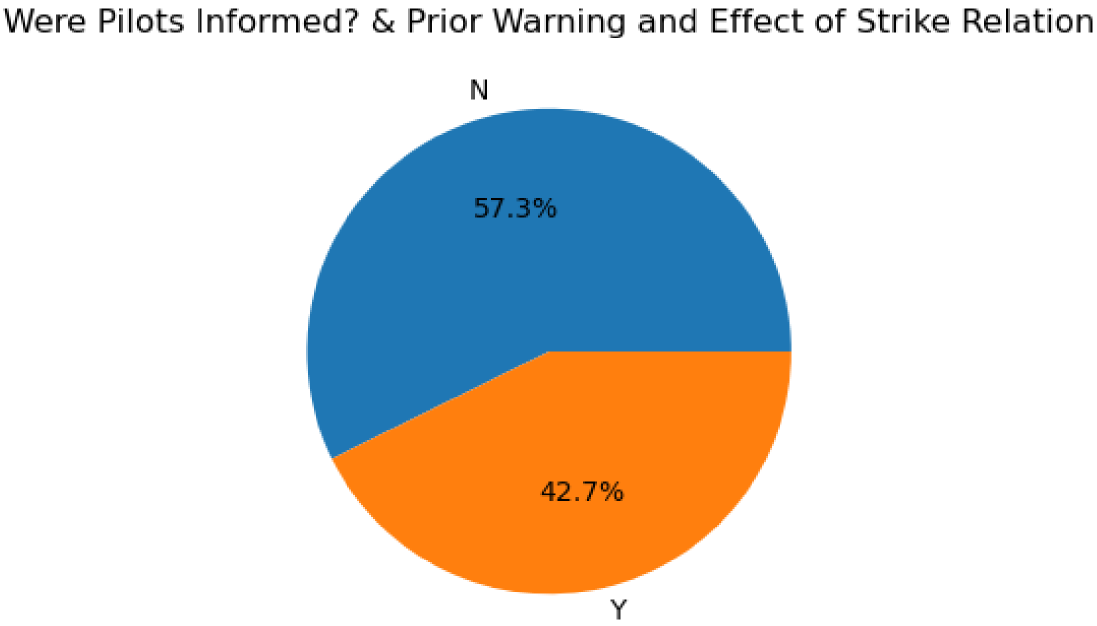
```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[25], line 6
      4 # Plotting Altitude of airplanes at the time of strike
      5 plt.figure(figsize=(6, 6))
----> 6 altitude_strike.plot(kind='bar')
      7 plt.title('Altitude of airplanes at the time of Bird Strikes')
      8 plt.xlabel('Altitude Bin')

File E:\Anaconda\Lib\site-packages\pandas\plotting\_core.py:1000, in P
lotAccessor.__call__(self, *args, **kwargs)
    997             label_name = label_kw or data.columns
    998             data.columns = label_name
-> 1000 return plot_backend.plot(data, kind=kind, **kwargs)

File E:\Anaconda\Lib\site-packages\pandas\plotting\_matplotlib\__init_
_.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File E:\Anaconda\Lib\site-packages\pandas\plotting\_matplotlib\core.p
y:450, in MPLPlot.generate(self)
    448 def generate(self) -> None:
    449     self._args_adjust()
--> 450     self._compute_plot_data()
    451     self._setup_subplots()
    452     self._make_plot()

File E:\Anaconda\Lib\site-packages\pandas\plotting\_matplotlib\core.p
y:635, in MPLPlot._compute_plot_data(self)
    633 # no non-numeric frames or series allowed
    634 if is_empty:
--> 635     raise TypeError("no numeric data to plot")
    637 self.data = numeric_data.apply(self._convert_to_ndarray)

TypeError: no numeric data to plot
```


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

In [26]:
```
# Effect of Bird Strikes & Impact on Flight
effect_impact = df['Effect: Impact to flight'].value_counts()

# Plotting Effect of Bird Strikes & Impact on Flight
plt.figure(figsize=(6, 4))
effect_impact.plot(kind='bar')
plt.title('Effect of Bird Strikes & Impact on Flight')
plt.xlabel('Effect on Flight')
plt.ylabel('Number of Bird Strikes')
plt.xticks(rotation=45)
plt.show()
```



```
In [42]: # Were Pilots Informed? & Prior Warning and Effect of Strike Relation
         pilot_warning = df['Pilot warned of birds or wildlife?'].value_counts()

         # Plotting Were Pilots Informed? & Prior Warning and Effect of Strike R
         plt.figure(figsize=(6, 4))
         pilot_warning.plot(kind='pie', autopct='%1.1f%%')
         plt.title('Were Pilots Informed? & Prior Warning and Effect of Strike R
         plt.ylabel('')
         plt.show()
```



Were Pilots Informed? & Prior Warning and Effect of Strike Relation

In [27]:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report



# Data preprocessing
# Encode categorical variables
le = LabelEncoder()
df['Effect: Impact to flight'] = le.fit_transform(df['Effect: Impact to
df['When: Phase of flight'] = le.fit_transform(df['When: Phase of fligh

# Select features and target variable
X = df[['Effect: Impact to flight', 'When: Phase of flight']]
y = df['Wildlife: Number Struck Actual']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier()

# Train the classifier
rf_classifier.fit(X_train, y_train)

# Make predictions
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8151408450704225
              precision    recall  f1-score   support

           1       0.82      1.00      0.90      4167
           2       0.00      0.00      0.00       112
           3       0.00      0.00      0.00        99
           4       0.00      0.00      0.00       114
           5       0.00      0.00      0.00        90
           6       0.00      0.00      0.00       104
           7       0.00      0.00      0.00        92
           8       0.00      0.00      0.00        90
           9       0.00      0.00      0.00        96
          10       0.00      0.00      0.00        87
          22       0.00      0.00      0.00         1
          28       0.00      0.00      0.00         1
          32       0.00      0.00      0.00         1
          33       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         4
          36       0.00      0.00      0.00         1
          39       0.00      0.00      0.00         4
          44       0.00      0.00      0.00         2
          46       0.00      0.00      0.00         1
          51       0.00      0.00      0.00         1
          52       0.00      0.00      0.00         2
          56       0.00      0.00      0.00         2
          58       0.00      0.00      0.00         1
          59       0.00      0.00      0.00         1
          60       0.00      0.00      0.00         3
          61       0.00      0.00      0.00         1
          62       0.00      0.00      0.00         3
          63       0.00      0.00      0.00         1
          64       0.00      0.00      0.00         1
          66       0.00      0.00      0.00         1
          68       0.00      0.00      0.00         1
          69       0.00      0.00      0.00         1
          72       0.00      0.00      0.00         1
          73       0.00      0.00      0.00         5
          74       0.00      0.00      0.00         2
          75       0.00      0.00      0.00         1
          76       0.00      0.00      0.00         1
          77       0.00      0.00      0.00         2
          78       0.00      0.00      0.00         1
          79       0.00      0.00      0.00         1
          80       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1
          82       0.00      0.00      0.00         1
          84       0.00      0.00      0.00         1
          85       0.00      0.00      0.00         1
          88       0.00      0.00      0.00         1
          89       0.00      0.00      0.00         1
          92       0.00      0.00      0.00         1
          95       0.00      0.00      0.00         1
          96       0.00      0.00      0.00         1
          98       0.00      0.00      0.00         1
          99       0.00      0.00      0.00         1

    accuracy                           0.82      5112
   macro avg       0.02      0.02      0.02      5112
```

```
weighted avg       0.66      0.82      0.73      5112

E:\Anaconda\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
E:\Anaconda\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision and F-score are ill-defined and bein
g set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

In [ ]:

In [ ]: