Bird Strikes Data Analysis

Analysis By: **Dipean Dasgupta**

UNID: **UMIP4841**

Importing Libraries

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

Connecting Drive

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [3]:
```
DS=pd.read_csv('/content/drive/MyDrive/UM_Projects_data/Bird Strikes data.csv')
```

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

Out[4]:

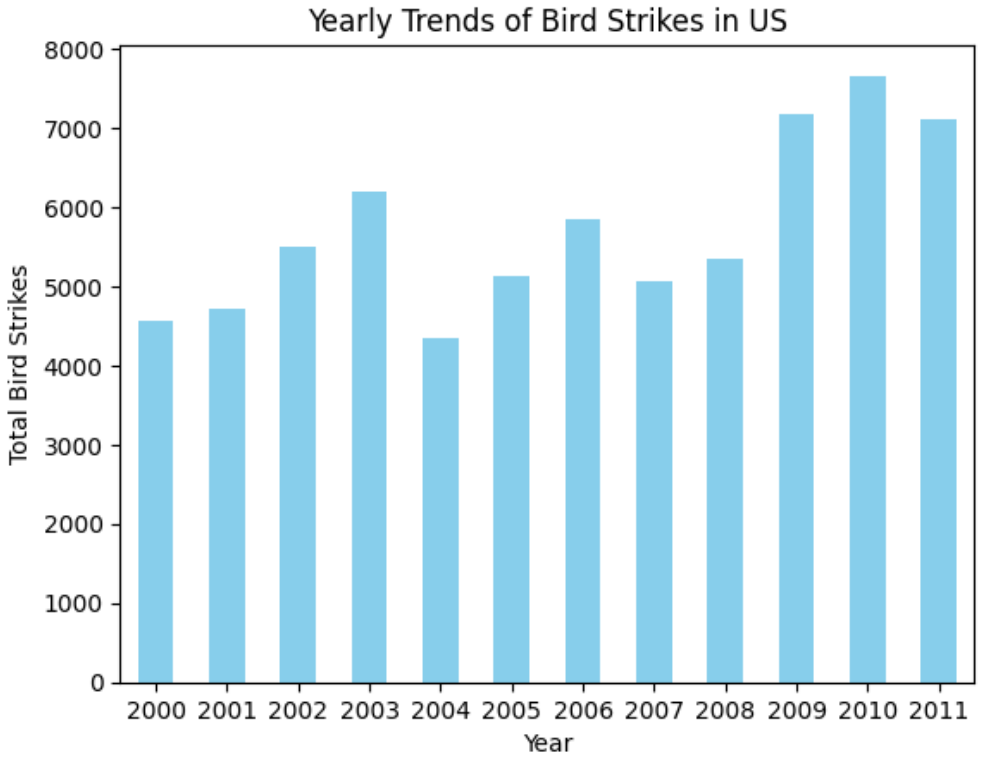
	Record ID	Aircraft: Type	Airport: Name	Altitude bin	Aircraft: Make/ Model	Wildlife: Number struck	Wildlife: Number Struck Actual	Effect: Impact to flight	FlightDate	Effect: Indicated Damage	...
0	202152	Airplane	LAGUARDIA NY	> 1000 ft	B-737-400	Over 100	859	Engine Shut Down	11/23/00 0:00	Caused damage	...
1	208159	Airplane	DALLAS/ FORT WORTH INTL ARPT	< 1000 ft	MD-80	Over 100	424	NaN	7/25/01 0:00	Caused damage	...
2	207601	Airplane	LAKEFRONT AIRPORT	< 1000 ft	C-500	Over 100	261	NaN	9/14/01 0:00	No damage	...
3	215953	Airplane	SEATTLE-TACOMA INTL	< 1000 ft	B-737-400	Over 100	806	Precautionary Landing	9/5/02 0:00	No damage	...
4	219878	Airplane	NORFOLK INTL	< 1000 ft	CL-RJ100/200	Over 100	942	NaN	6/23/03 0:00	No damage	...

5 rows × 26 columns

In [5]:
```
DF1=pd.DataFrame(DS)
```

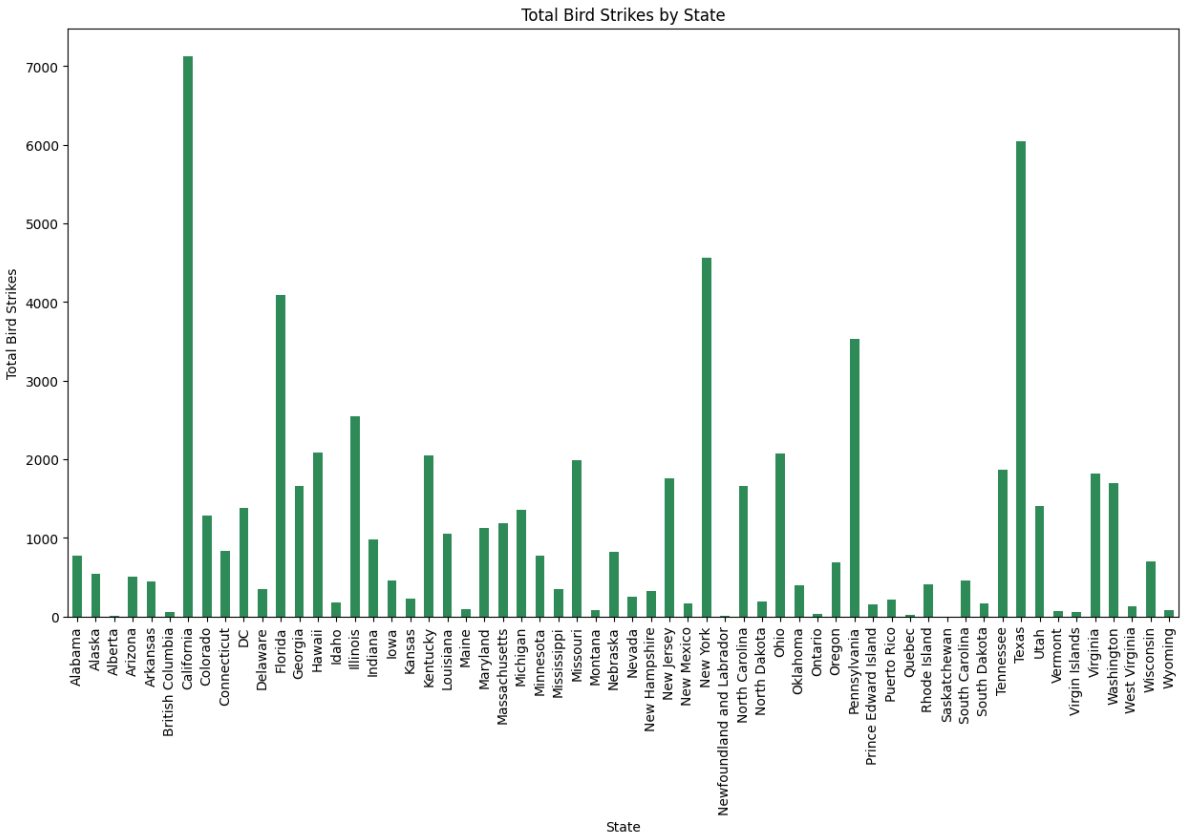
Yearly Analysis & Bird Strikes in the US

In [ ]:
```
DS['FlightDate'] = pd.to_datetime(DS['FlightDate'])
DS['Year'] = DS['FlightDate'].dt.year

# Grouping by year and sum the number of bird strikes
yearly_bird_strikes = DS.groupby('Year')['Wildlife: Number Struck Actual'].sum()

# Plotting the bar chart
ax = yearly_bird_strikes.plot(kind='bar',
                              title='Yearly Trends of Bird Strikes in US',
                              xlabel='Year',
                              ylabel='Total Bird Strikes',
                              color='skyblue')


ax.set_xticks(range(len(yearly_bird_strikes.index)))
ax.set_xticklabels(yearly_bird_strikes.index.astype(int), rotation=0)

plt.show()
```

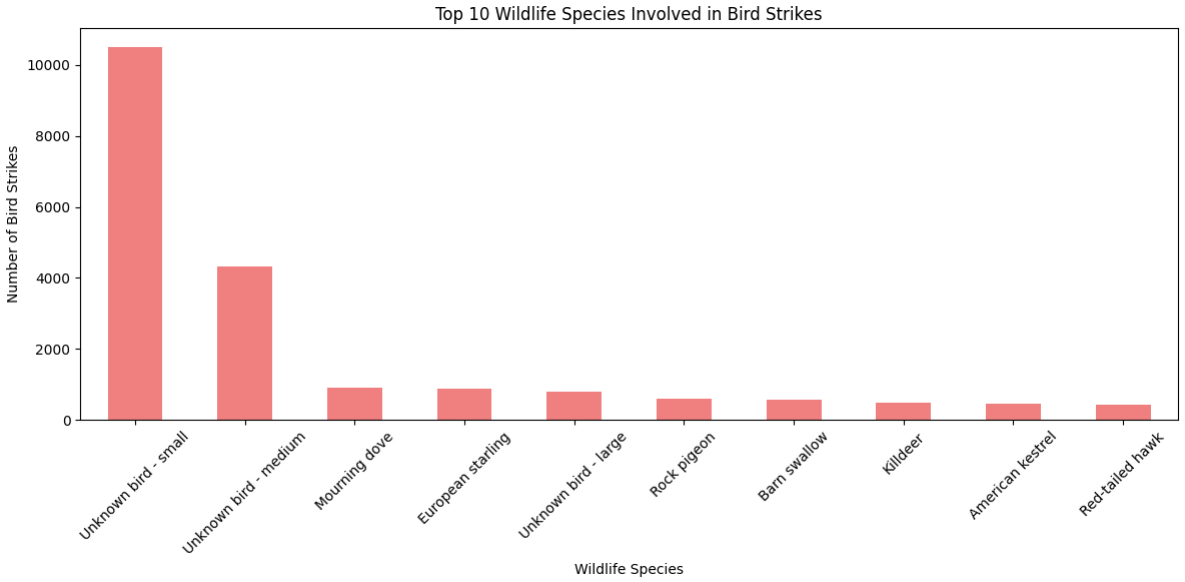


Visuals Depicting the Number of Bird Strikes

In [ ]:
```
# Grouping by 'Origin State' and sum the number of bird strikes
state_bird_strikes = DS.groupby('Origin State')['Wildlife: Number Struck Actual'].sum()

# Plotting the bar chart
fig, ax = plt.subplots(figsize=(15, 8))
state_bird_strikes.plot(kind='bar',
                        title='Total Bird Strikes by State',
                        xlabel='State',
                        ylabel='Total Bird Strikes',
                        color='seagreen',
                        ax=ax)

plt.show()
```

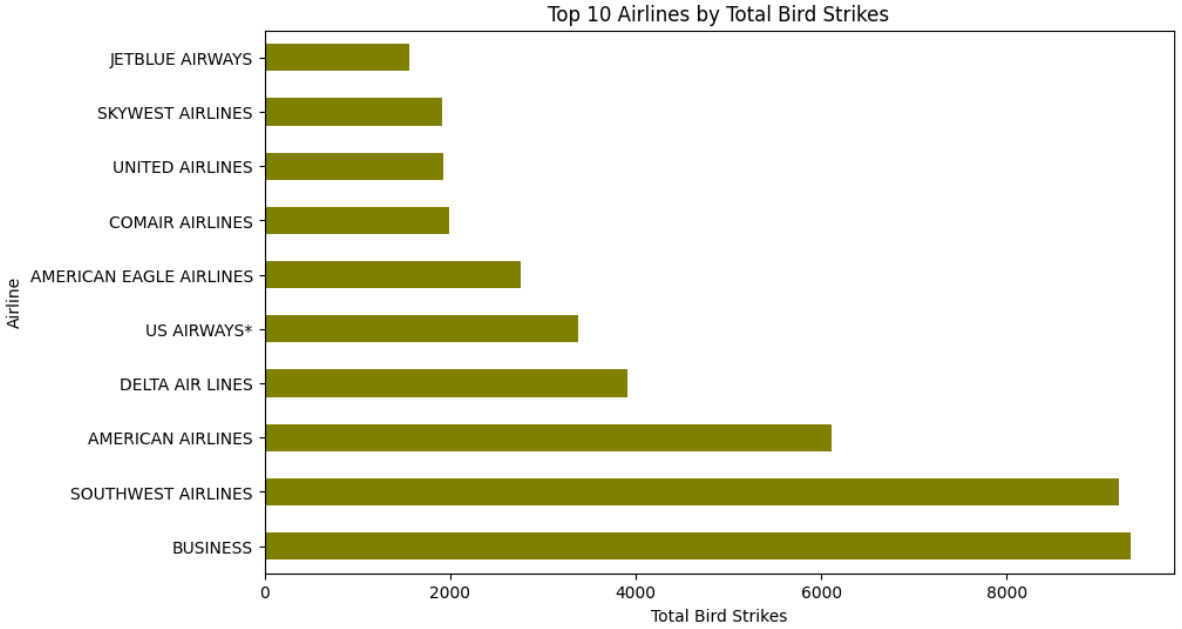


In [ ]:
```
plt.figure(figsize=(12, 6))
DS['Wildlife: Species'].value_counts().head(10).plot(kind='bar', color='lightcoral')
plt.title('Top 10 Wildlife Species Involved in Bird Strikes')
plt.xlabel('Wildlife Species')
plt.ylabel('Number of Bird Strikes')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



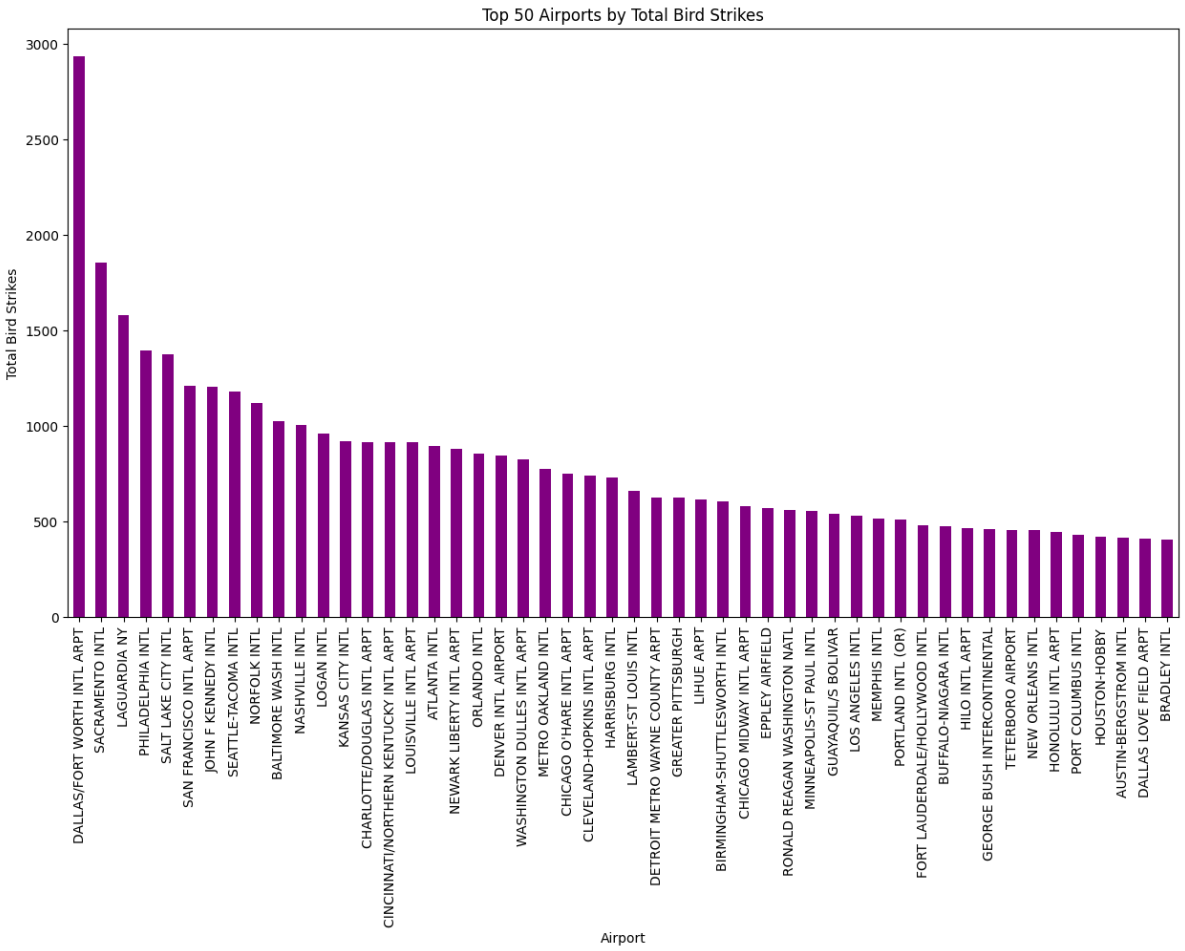
Top 10 US Airlines in terms of having encountered bird strikes

In [ ]:
```
# Grouping by 'Aircraft: Airline/Operator' and sum the number of bird strikes
top_airlines = DS.groupby('Aircraft: Airline/Operator')['Wildlife: Number Struck Actual'].sum().

# Plotting the horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 6))
top_airlines.plot(kind='barh',
                  title='Top 10 Airlines by Total Bird Strikes',
                  xlabel='Total Bird Strikes',
                  ylabel='Airline',
                  color='olive',
                  ax=ax)

plt.show()
```



Airports with most incidents of bird strikes – Top 50

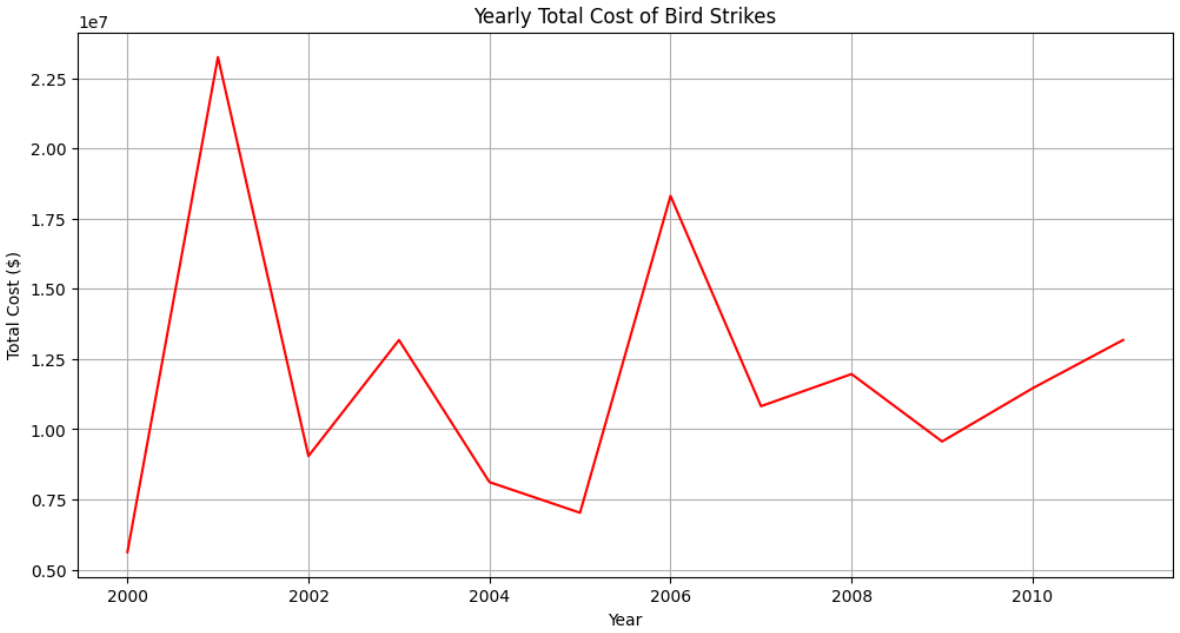
In [ ]:
```
# Grouping by 'Airport: Name' and sum the number of bird strikes
top_airports = DS.groupby('Airport: Name')['Wildlife: Number Struck Actual'].sum().nlargest(50)

# Plotting the bar chart
fig, ax = plt.subplots(figsize=(15, 8))
top_airports.plot(kind='bar',
                  title='Top 50 Airports by Total Bird Strikes',
                  xlabel='Airport',
                  ylabel='Total Bird Strikes',
                  color='purple',
                  ax=ax)


ax.set_xticklabels(top_airports.index, rotation=90)

plt.show()
```



Yearly Cost Incurred due to Bird Strikes

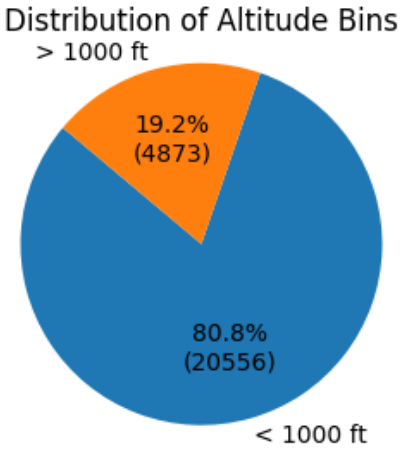
In [6]:
```
DF1['Cost: Total $'] = DF1['Cost: Total $'].replace('[\$,]', '', regex=True)
DF1['Cost: Total $'] = pd.to_numeric(DF1['Cost: Total $'], errors='coerce')
DF1['Cost: Total $'] = DF1['Cost: Total $'].fillna(0)
DF1['Incident Year'] = pd.to_datetime(DF1['FlightDate']).dt.year

#Grouping by 'Incident Year' and calculate the total cost per year
yearly_cost = DF1.groupby('Incident Year')['Cost: Total $'].sum()

# Plotting the line chart
fig, ax = plt.subplots(figsize=(12, 6))
yearly_cost.plot(kind='line',
                 title='Yearly Total Cost of Bird Strikes',
                 xlabel='Year',
                 ylabel='Total Cost ($)',
                 color='red',
                 ax=ax)

plt.grid(True)
plt.show()
```

```
<ipython-input-6-cde0eda95704>:4: UserWarning: Could not infer format, so each element will be
parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expecte
d, please specify a format.
  DF1['Incident Year'] = pd.to_datetime(DF1['FlightDate']).dt.year
```



Altitude of aeroplanes at the time of strike

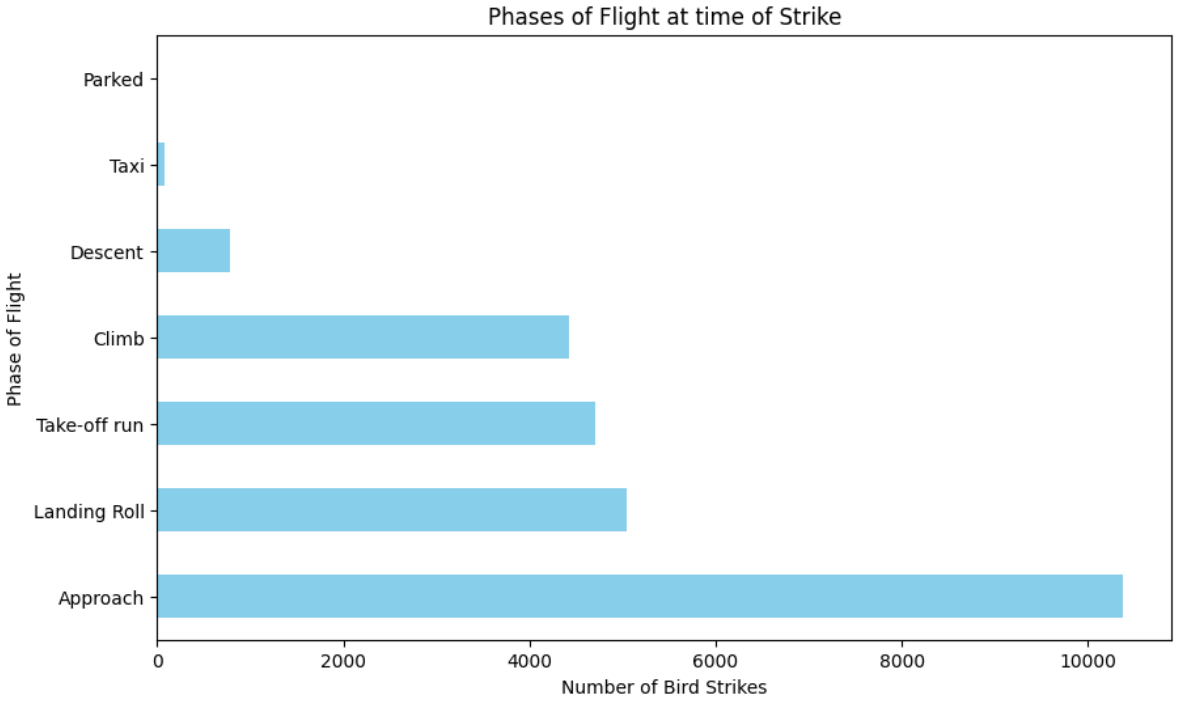
In [ ]:
```
altitude_distribution = DS['Altitude bin'].value_counts()

def autopct_format(values):
    def my_format(pct):
        total = sum(values)
        val = int(round(pct * total / 100.0))
        return '{:.1f}%\n({:d})'.format(pct, val)
    return my_format

# Plotting the pie chart
plt.figure(figsize=(6, 3))
plt.pie(altitude_distribution,
        labels=altitude_distribution.index,
        autopct=autopct_format(altitude_distribution),
        startangle=140)
plt.axis('equal')
plt.title('Distribution of Altitude Bins')
plt.show()
```



Phase of Flight

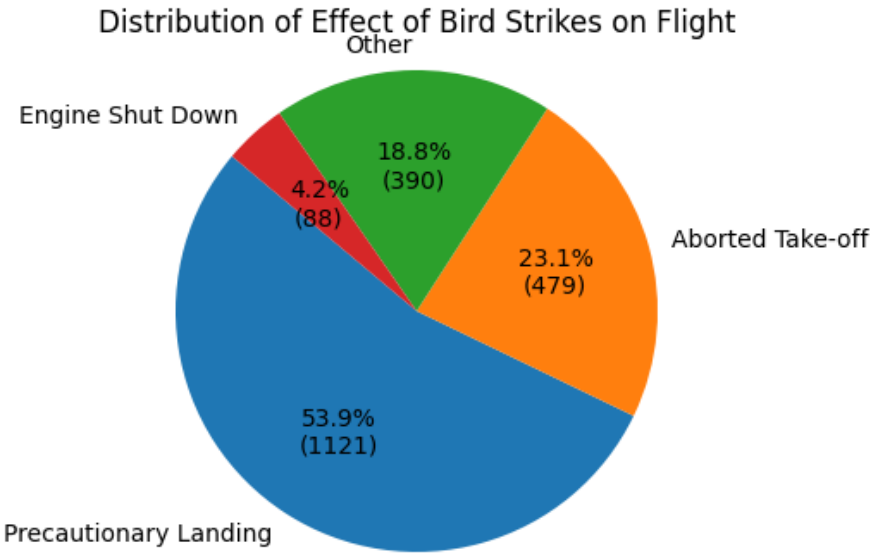
In [ ]:
```
# Calculating the distribution of 'Phase of Flight'
phase_of_flight = DS['When: Phase of flight'].value_counts()
print(phase_of_flight)
# Plotting the horizontal bar chart
plt.figure(figsize=(10, 6))
phase_of_flight.plot(kind='barh', color='skyblue')
plt.xlabel('Number of Bird Strikes')
plt.ylabel('Phase of Flight')
plt.title('Phases of Flight at time of Strike')
plt.show()
```

```
When: Phase of flight
Approach        10382
Landing Roll     5047
Take-off run     4711
Climb            4429
Descent           776
Taxi               74
Parked             10
Name: count, dtype: int64
```



Impact to Flight

In [ ]:
```
effect_of_flight_impact = DS['Effect: Impact to flight'].value_counts()

def autopct_format(values):
    def my_format(pct):
        total = sum(values)
        val = int(round(pct * total / 100.0))
        return '{:.1f}%\n({:d})'.format(pct, val)
    return my_format

# Plotting the pie chart
plt.figure(figsize=(8, 4))
plt.pie(effect_of_flight_impact,
        labels=effect_of_flight_impact.index,
        autopct=autopct_format(effect_of_flight_impact),
        startangle=140)
plt.axis('equal')
plt.title('Distribution of Effect of Bird Strikes on Flight')
plt.show()
```



Pilot warning vs Impact to Flight

In [ ]:
```
pilot_warning_effect_relation = DS.groupby(['Pilot warned of birds or wildlife?', 'Effect: Impac
plt.figure(figsize=(6, 4))
sns.heatmap(pilot_warning_effect_relation, annot=True, cmap='coolwarm', fmt='d')
plt.title('Relationship between Pilot Warning and Effect of Bird Strikes on Flight')
plt.xlabel('Effect: Impact to flight')
plt.ylabel('Pilot warned of birds or wildlife?')
plt.show()
```