Phase 1

Data Cleaning

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter("ignore")
```

In [2]:
```
df =pd.read_csv("AviationData.csv",encoding = "latin1")
```

In [3]:
```
df.head()
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States

5 rows × 31 columns

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

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States

5 rows × 31 columns

In [5]:
```
df.sample(5)
```

Out[5]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country
56400	20040331X00405	Accident	LAX04LA170	2004-03-25	Planada, CA	United States
40906	20001208X07000	Accident	CHI97FA030	1996-11-18	GRAND RAPIDS, MI	United States
84170	20191107X55010	Accident	WPR20FA019	2019-11-07	Upland, CA	United States
27909	20001212X16974	Accident	DEN91FA067	1991-05-06	HARTLEY, TX	United States
57183	20040830X01320	Accident	CHI04CA210	2004-07-31	Plymouth, MI	United States

5 rows × 31 columns

In [6]:
```
df.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
```

In [7]:
```
df.describe()
```

Out[7]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.U
count	82805.000000	77488.000000	76379.000000	76956.000000	8297
mean	1.146585	0.647855	0.279881	0.357061	
std	0.446510	5.485960	1.544084	2.235625	2
min	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	0.000000	0.000000	
max	8.000000	349.000000	161.000000	380.000000	69

In [8]:
```
df.describe(include = "O")
```

Out[8]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Co
count	88889	88889	88889	88889	88837	8
unique	87951	2	88863	14782	27758	
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	U S
freq	3	85015	2	25	434	8

4 rows × 26 columns

In [9]:
```
#column name
df.columns = df.columns.str.lower().str.replace(r'\.', '_', regex=True)
```

In [10]:
```
df.columns
```

Out[10]:
```
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_descripti
on',
       'schedule', 'purpose_of_flight', 'air_carrier', 'total_fatal_injuri
es',
       'total_serious_injuries', 'total_minor_injuries', 'total_uninjure
d',
       'weather_condition', 'broad_phase_of_flight', 'report_status',
       'publication_date'],
      dtype='object')
```

In [11]:
```
df.dtypes
```

Out[11]:
```
event_id                    object
investigation_type          object
accident_number             object
event_date                  object
location                    object
country                     object
latitude                    object
longitude                   object
airport_code                object
airport_name                object
injury_severity             object
aircraft_damage             object
aircraft_category           object
registration_number         object
make                        object
model                       object
amateur_built               object
number_of_engines          float64
engine_type                 object
far_description             object
schedule                    object
purpose_of_flight           object
air_carrier                 object
total_fatal_injuries       float64
total_serious_injuries     float64
total_minor_injuries       float64
total_uninjured            float64
weather_condition           object
broad_phase_of_flight       object
report_status               object
publication_date            object
dtype: object
```

Checking Missing Values and Correcting It.

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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
```

In [13]:
```
df.columns
```

Out[13]:
```
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_descripti
on',
       'schedule', 'purpose_of_flight', 'air_carrier', 'total_fatal_injuri
es',
       'total_serious_injuries', 'total_minor_injuries', 'total_uninjure
d',
       'weather_condition', 'broad_phase_of_flight', 'report_status',
       'publication_date'],
      dtype='object')
```

In [14]:
```
df = df[['event_id', 'event_date',
        'location', 'country','airport_code',
        'airport_name', 'injury_severity', 'aircraft_damage',
        'aircraft_category','make', 'model',
        'amateur_built', 'number_of_engines', 'engine_type',
        'purpose_of_flight', 'air_carrier', 'total_fatal_injuries',
        'total_serious_injuries', 'total_minor_injuries', 'total_uninjured',
        'weather_condition', 'broad_phase_of_flight',
         ]]
```

In [15]:
```
df.columns
```

Out[15]:
```
Index(['event_id', 'event_date', 'location', 'country', 'airport_code',
       'airport_name', 'injury_severity', 'aircraft_damage',
       'aircraft_category', 'make', 'model', 'amateur_built',
       'number_of_engines', 'engine_type', 'purpose_of_flight', 'air_carri
er',
       'total_fatal_injuries', 'total_serious_injuries',
       'total_minor_injuries', 'total_uninjured', 'weather_condition',
       'broad_phase_of_flight'],
      dtype='object')
```

In [16]:
```
# Forward fill, then backward fill
df['event_date'] = df['event_date'].fillna(method='ffill').fillna(method='b
```

In [17]:
```
# Impute using mode (for categorical data)
for col in df.select_dtypes(include = ["O"]):
    mode_value = df[col].mode()[0]
    df[col].fillna(mode_value, inplace=True)
```

In [18]:
```
for col in df.select_dtypes(include = ["number"]):
    mean_value = df[col].mean()
    df[col].fillna(mean_value, inplace=True)
```

In [19]:
```
df.isnull().sum()
```

Out[19]:
```
event_id                  0
event_date                0
location                  0
country                   0
airport_code              0
airport_name              0
injury_severity           0
aircraft_damage           0
aircraft_category         0
make                      0
model                     0
amateur_built             0
number_of_engines         0
engine_type               0
purpose_of_flight         0
air_carrier               0
total_fatal_injuries      0
total_serious_injuries    0
total_minor_injuries      0
total_uninjured           0
weather_condition         0
broad_phase_of_flight     0
dtype: int64
```

In [20]:
```
for col in df.select_dtypes(include = ["O"]):df[col] = df[col].str.lower()
```

In [21]:
```
df.duplicated().sum()
```

Out[21]:
```
25
```

In [22]:
```
df.drop_duplicates(inplace=True)
```

In [23]:
```
df.duplicated().sum()
```

Out[23]:
```
0
```

In [24]:
```
numeric_df = df.select_dtypes(include=[np.number])
numeric_df
```

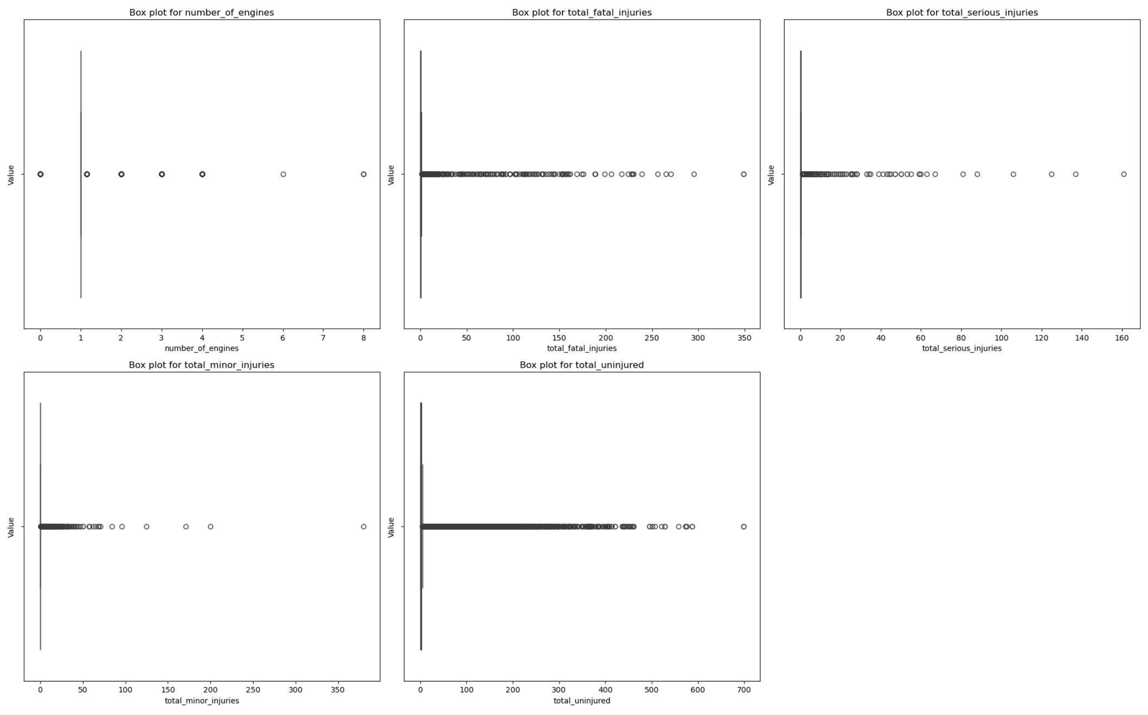
Out[24]:

	number_of_engines	total_fatal_injuries	total_serious_injuries	total_minor_injuries	tota
0	1.000000	2.0	0.000000	0.000000	
1	1.000000	4.0	0.000000	0.000000	
2	1.000000	3.0	0.279881	0.357061	
3	1.000000	2.0	0.000000	0.000000	
4	1.146585	1.0	2.000000	0.357061	
...	...	...	...	...	
88884	1.146585	0.0	1.000000	0.000000	
88885	1.146585	0.0	0.000000	0.000000	
88886	1.000000	0.0	0.000000	0.000000	
88887	1.146585	0.0	0.000000	0.000000	
88888	1.146585	0.0	1.000000	0.000000	

88864 rows × 5 columns

Checking for Outliers

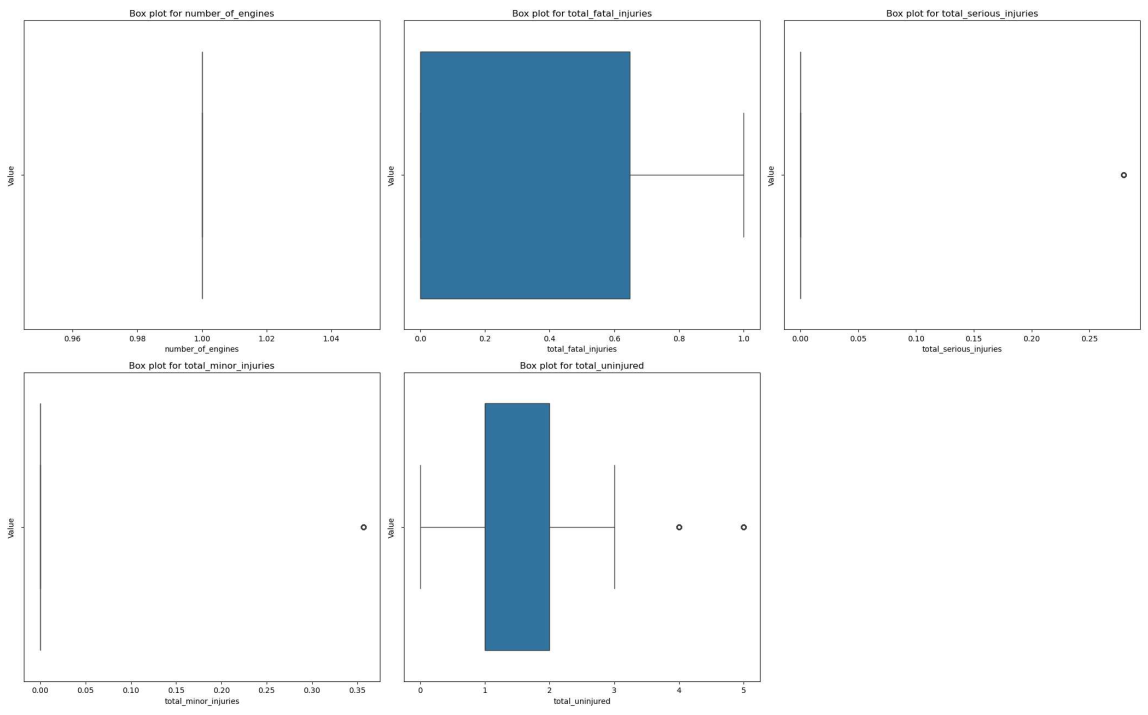
In [25]:
```
# Grid layout
rows, cols = 2, 3
fig, axes = plt.subplots(rows, cols, figsize=(21, 13))


# Flatten
axes = axes.flatten()

for i, column in enumerate(numeric_df):
    sns.boxplot(x=df[column], ax = axes[i])
    axes[i].set_title(f"Box plot for {column}")
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Value')

# Hide empty subplots
for j in range(i + 1, rows * cols):
    axes[j].axis('off')

plt.tight_layout()
plt.show()
```



In [26]:
```
# Select only numeric columns
numeric_data = df.select_dtypes(include=['number'])

# Calculate q1, q3, and IQR for numeric data only
q1 = numeric_data.quantile(0.25)  # .25
q3 = numeric_data.quantile(0.75)  # .75
IQR = q3 - q1

# Lower and upper bounds for identifying outliers
lower_bound = q1 - (1.5 * IQR)
upper_bound = q3 + (1.5 * IQR)

# Identify outliers in numeric columns
outliers_ = set()
for col in numeric_data.columns:
    outliers = numeric_data[(numeric_data[col] < lower_bound[col]) |(numeri
    outliers_.update(outliers.index)

# Count number of rows before removing outliers
num_rows_before = len(df)

# Remove rows with outliers from the original DataFrame (important!)
df = df.drop(index=outliers_)

# Num of rows after removing outliers
num_rows_after = len(df)

# Number of rows removed
rows_removed = num_rows_before - num_rows_after

# Print the results
print(f"Number of rows before removing outliers: {num_rows_before}")
print(f"Number of rows after removing outliers: {num_rows_after}")
print(f"Number of rows removed: {rows_removed}")
```

```
Number of rows before removing outliers: 88864
Number of rows after removing outliers: 42604
Number of rows removed: 46260
```

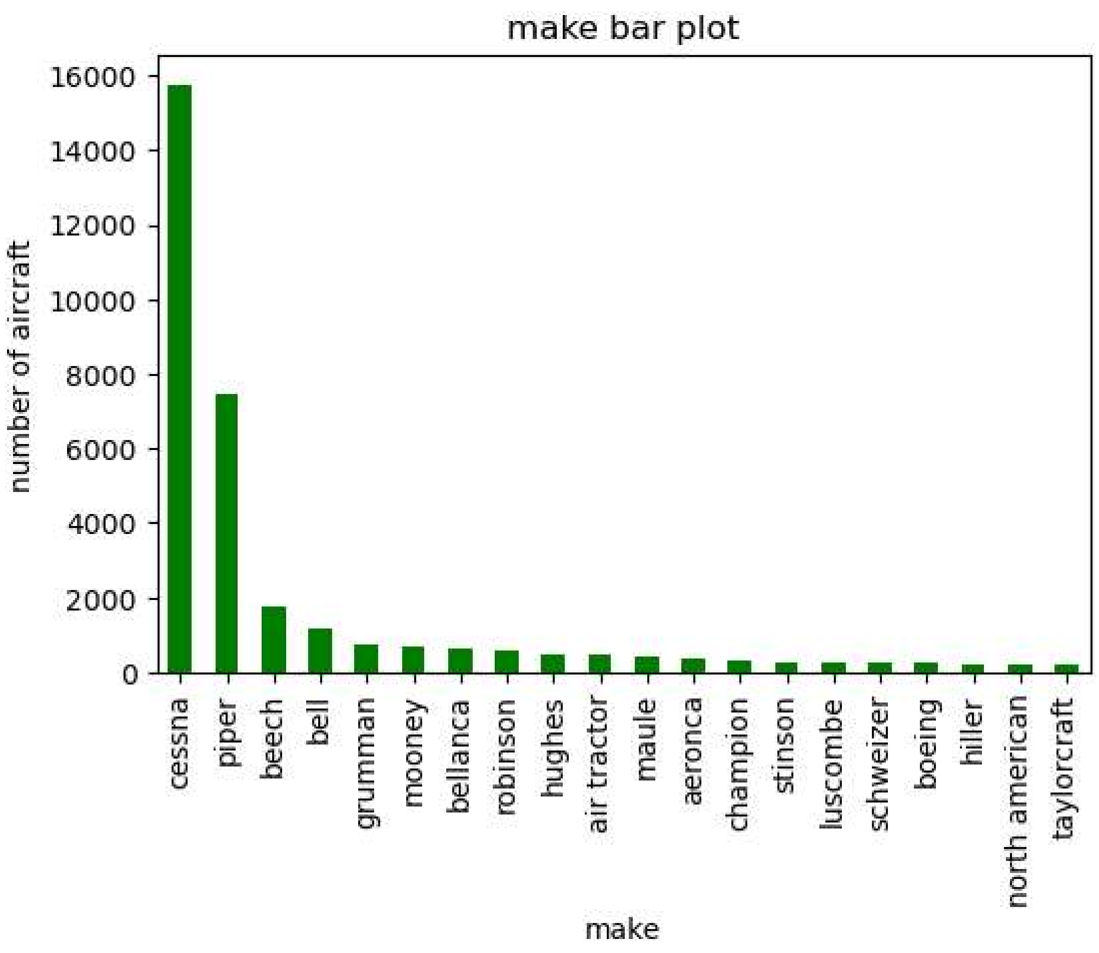
In [27]:
```
rows, cols = 2, 3
fig, axes = plt.subplots(rows, cols, figsize=(21, 13))


# Flatten
axes = axes.flatten()

for i, column in enumerate(numeric_df):
    sns.boxplot(x=df[column], ax = axes[i])
    axes[i].set_title(f"Box plot for {column}")
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Value')

# Hide empty subplots
for j in range(i + 1, rows * cols):
    axes[j].axis('off')

plt.tight_layout()
plt.show()
```



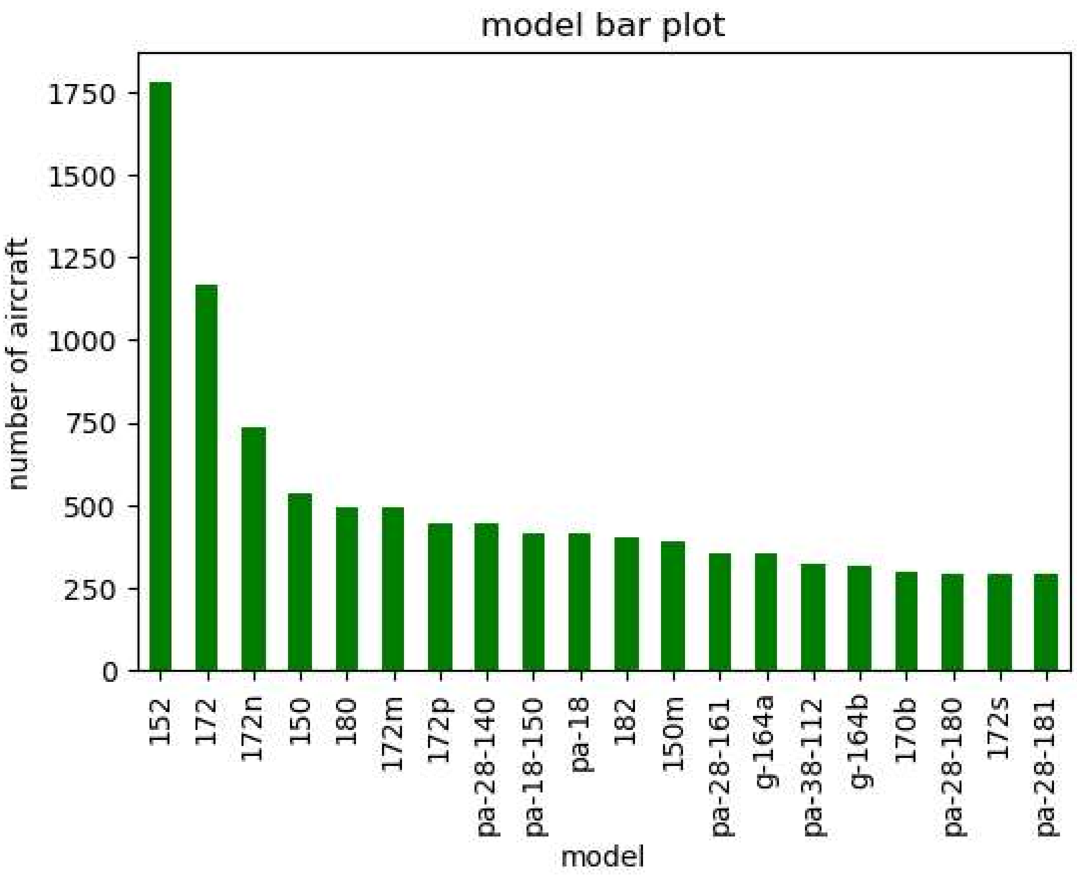
Saving the cleaned data.

In [28]:
```
df.to_csv("Project_1R.csv",index = False)
```

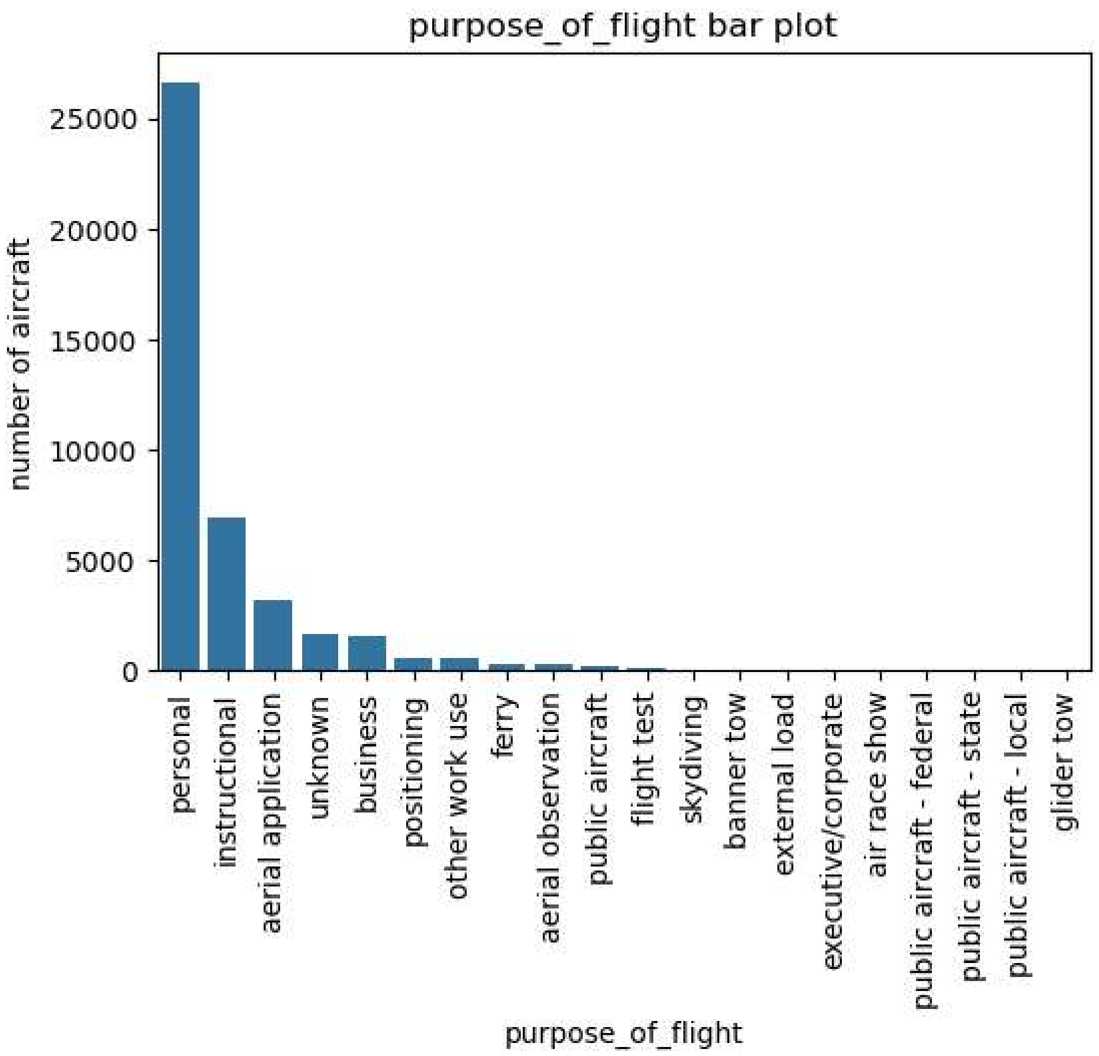
In [31]:
```
make_count = df["make"].value_counts().iloc[:20]
make_count
#visual using matplotlib bar
plt.figure(figsize=(6,4))
make_count.plot(kind="bar", color='g')
plt.title("make bar plot")
plt.ylabel("number of aircraft")
plt.show()
```



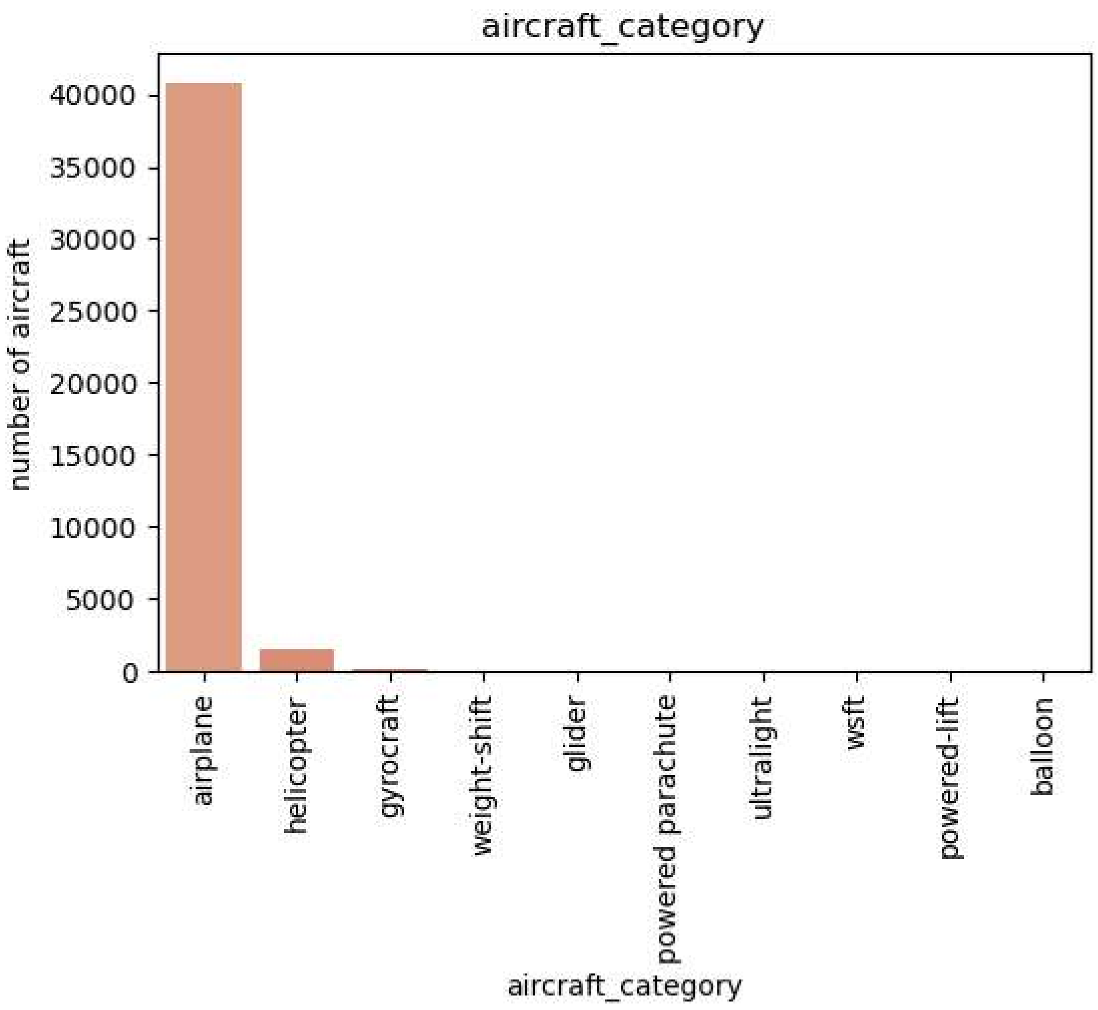
In [32]:
```
model_count = df["model"].value_counts().iloc[:20]
model_count
#visual using matplotlib bar
plt.figure(figsize=(6,4))
model_count.plot(kind="bar", color='g')
plt.title("model bar plot")
plt.ylabel("number of aircraft")
plt.show()
```



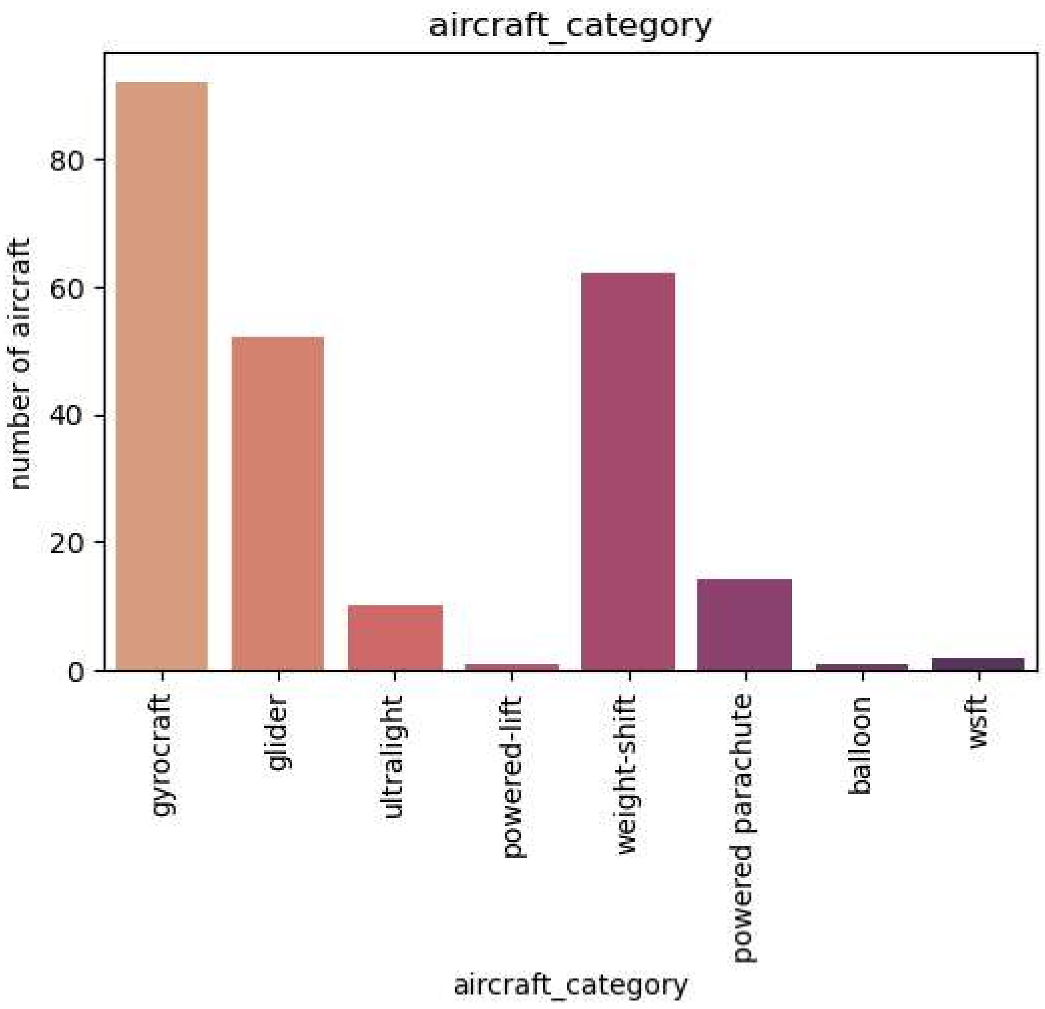
In [38]:
```
purpose_of_flight_count = df["purpose_of_flight"].value_counts().reset_inde
purpose_of_flight_count
#visual using matplotlib bar
plt.figure(figsize=(6,4))
sns.barplot(x = purpose_of_flight_count["purpose_of_flight"],y = purpose_of
plt.title("purpose_of_flight bar plot")
plt.ylabel("number of aircraft")
plt.xticks(rotation = 90)
plt.show()
```



In [59]:
```
df = df[df['aircraft_category'] != 'unknown']
aircraft_category = df["aircraft_category"].value_counts().reset_index().il
aircraft_category
#visual using matplotlib bar
plt.figure(figsize=(6,4))
sns.barplot(data = aircraft_category, x = "aircraft_category",y = "count",p
plt.title("aircraft_category")
plt.ylabel("number of aircraft")
plt.xticks(rotation = 90)
plt.show()
```



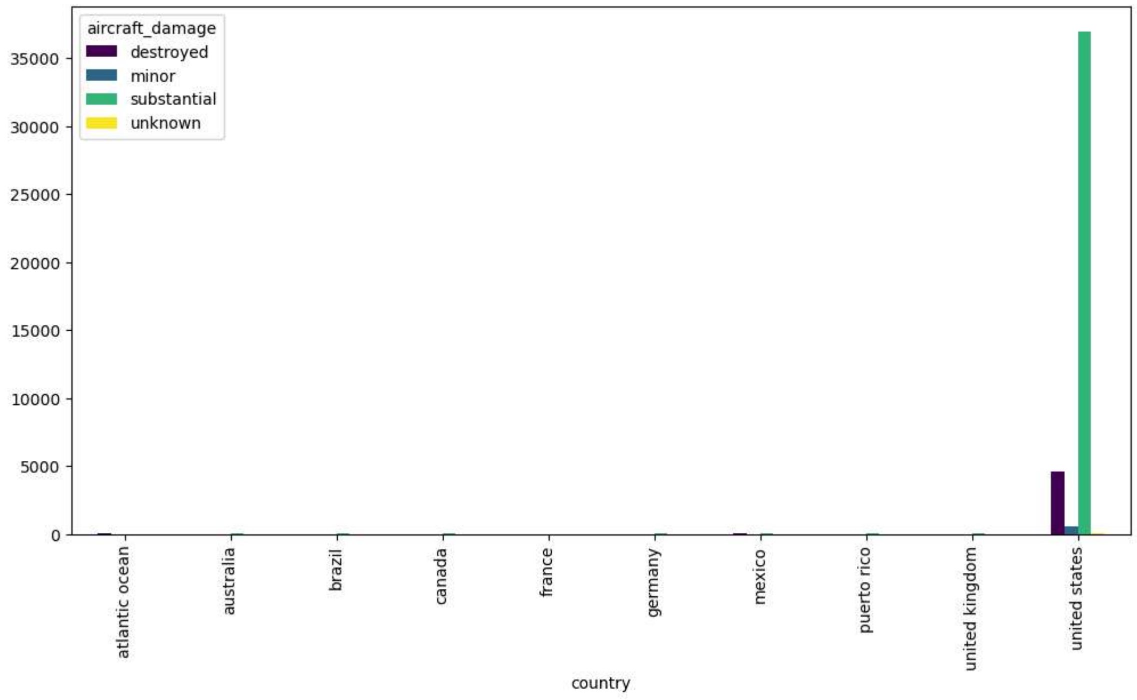
In [60]:
```
df_filtered = df[(df['aircraft_category'] != 'airplane') & (df['aircraft_ca

plt.figure(figsize=(6,4))
sns.countplot(data = df_filtered, x = "aircraft_category",dodge=False, pale
plt.title("aircraft_category")
plt.ylabel("number of aircraft")
plt.xticks(rotation = 90)
plt.show()
```



In [70]:
```
# Remove 'unknown', 'unk' to exclude
df = df[df['aircraft_damage'] != 'UNK']
df = df[df['aircraft_damage'] != 'Unknown']
# Aggregating the data by aircraft category
df1= df['country'].value_counts().head(10).index
#df1 = df[df['country'] != 'Unknown']
df1 = df[df['country'].isin(df1)]
df2= df1['country'].value_counts().tail(9).index
#df = df[df['country'] != 'Unknown']
df2 = df[df['country'].isin(df2)]
```

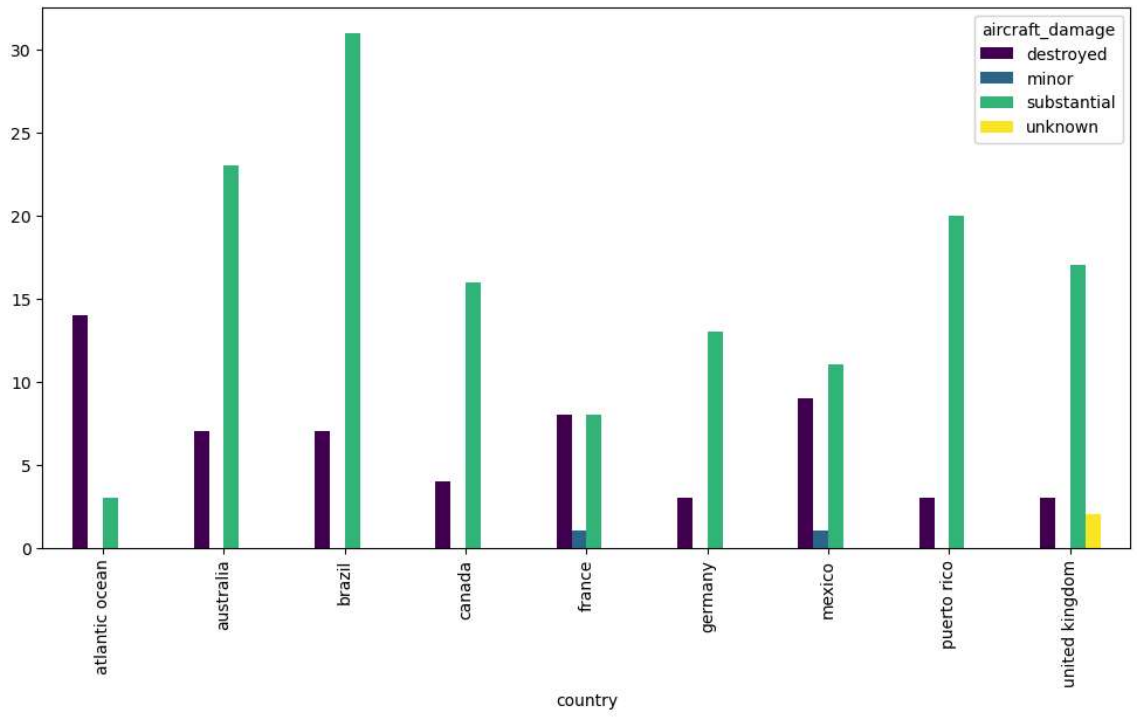
In [69]:
```
# Plot the bar plot
damage_data = df1.groupby(['country', 'aircraft_damage']).size().unstack().

# Plot grouped bar chart
damage_data.plot(kind='bar', figsize=(12, 6), colormap='viridis')
```

Out[69]:
```
<Axes: xlabel='country'>
```



In [71]:
```
# Plot the bar plot
damage_data = df2.groupby(['country', 'aircraft_damage']).size().unstack().

# Plot grouped bar chart
damage_data.plot(kind='bar', figsize=(12, 6), colormap='viridis')
```

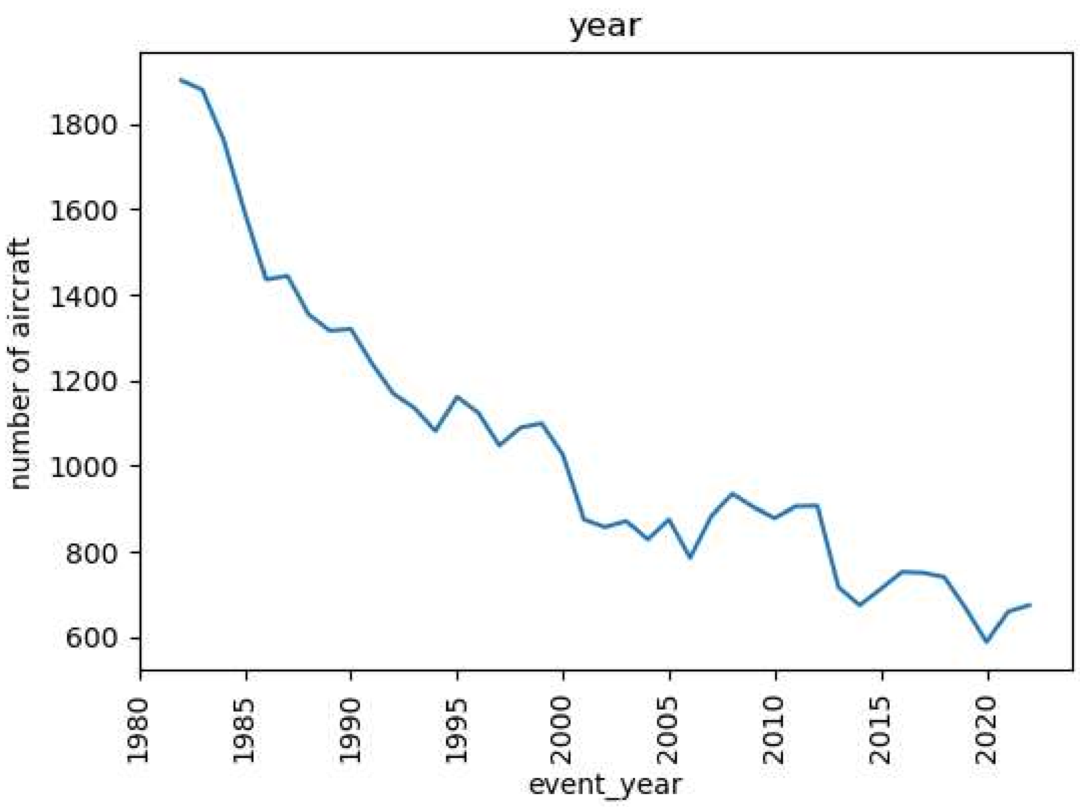
Out[71]:
```
<Axes: xlabel='country'>
```



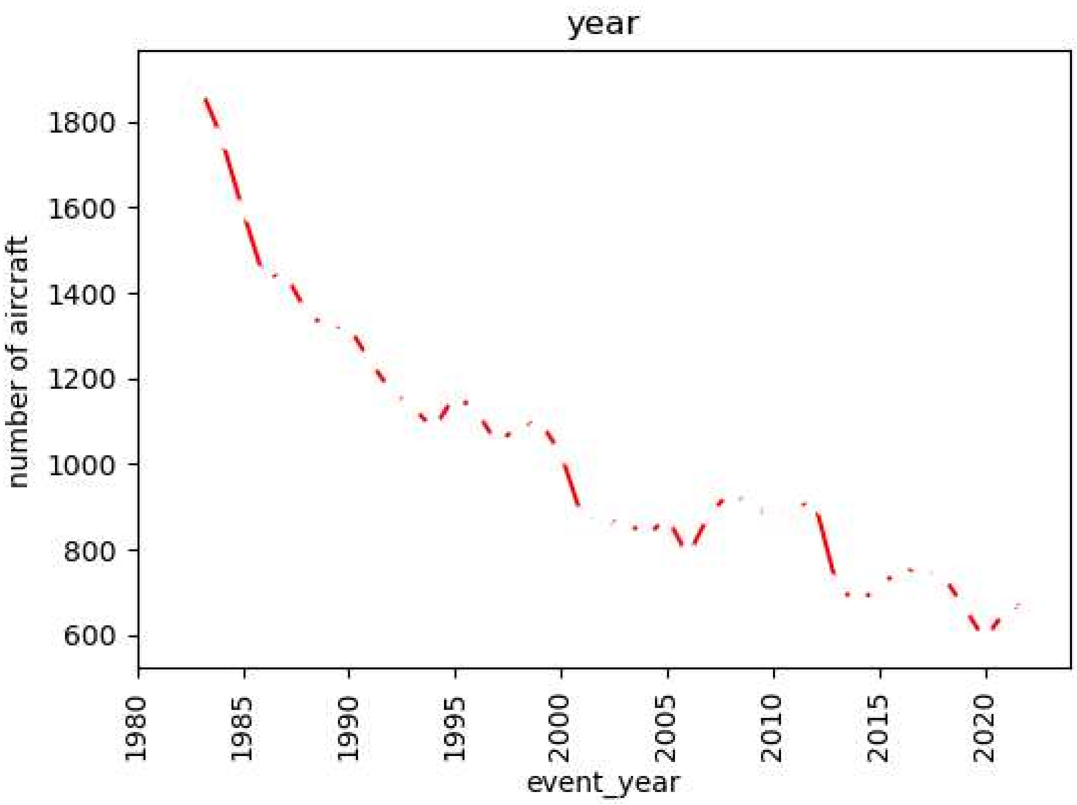
In [74]:
```
# Extract the year from the event.date column for temporal analysis.
df['event_year'] = pd.to_datetime(df['event_date']).dt.year
df['event_year'].head()
```

Out[74]:
```
7     1982
10    1982
11    1982
13    1982
14    1982
Name: event_year, dtype: int32
```

In [78]:
```
year = df['event_year'].value_counts().reset_index()
year
plt.figure(figsize=(6,4))
sns.lineplot(data = year, x = "event_year",y = "count",palette='flare')
plt.title("year")
plt.ylabel("number of aircraft")
plt.xticks(rotation = 90)
plt.show()
```



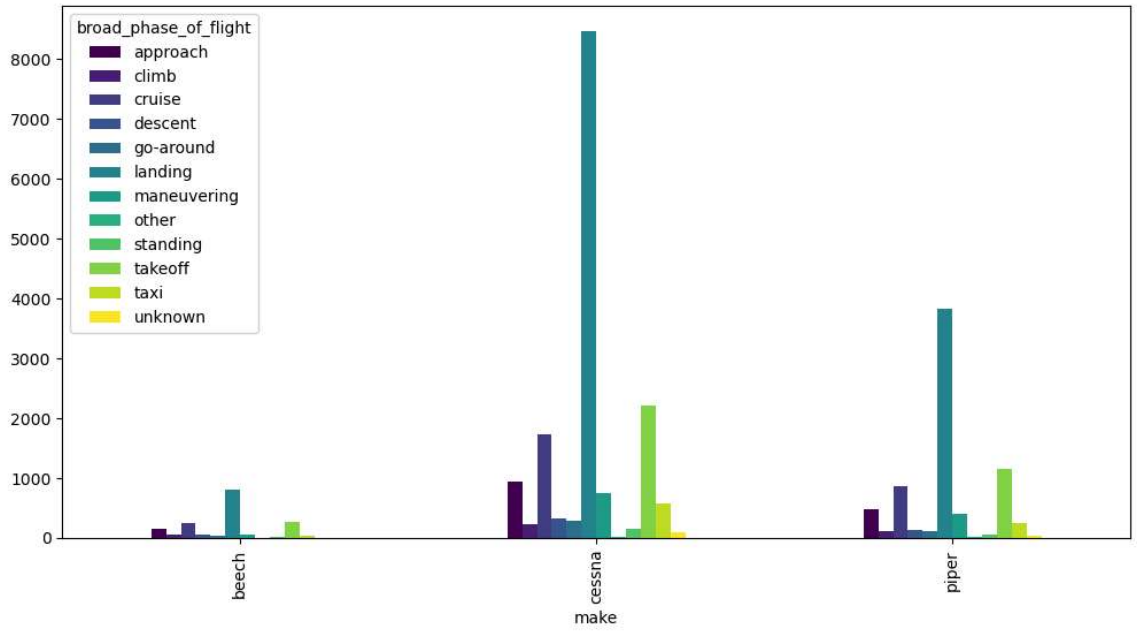
In [81]:
```
year = df['event_year'].value_counts().reset_index()
year
plt.figure(figsize=(6,4))
sns.lineplot(data = year, x = "event_year",y = "count",marker = "x", marker
plt.title("year")
plt.ylabel("number of aircraft")
plt.xticks(rotation = 90)
plt.show()
```



In [ ]:
```
from wordcloud import WordCloud
# Remove rows where airport.name contains 'Unknown' or 'None'
df = df[~df['airport.name'].str.contains('Unknown|None', case=False, na=Fal

# Filter data for high-risk airports
high_risk_airports = df[df['risk.category'] == 'High Risk']['airport.name']

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').gene

# Plot
plt.figure(figsize=(12, 8))
plt.imshow(wordcloud, interpolation='lanczos')
plt.axis('off')
plt.title('High-Risk Airports', fontsize=16, fontweight='bold')
plt.show()
```

In [82]:
```
df1= df['make'].value_counts().head(3).index
#df1 = df[df['make'] != 'Unknown']
df1 = df[df['make'].isin(df1)]
damage_data = df1.groupby(['make', 'broad_phase_of_flight']).size().unstack

# Plot grouped bar chart
damage_data.plot(kind='bar', figsize=(12, 6), colormap='viridis')
```

Out[82]:
```
<Axes: xlabel='make'>
```



In [ ]: