In [3]:
```
# importing the library that is needed
# import pandas, numpy, matplotlib and seaborn


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

In [4]:
```
# uploading the csv data and create a dataframe

df = pd.read_csv ("AviationData.csv", encoding="ISO-8859-1")
df
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3276\3640526320.py:3: DtypeWar
ning: Columns (6,7,28) have mixed types. Specify dtype option on import o
r set low_memory=False.
  df = pd.read_csv ("AviationData.csv", encoding="ISO-8859-1")
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Co
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	U S
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	U S
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	U S
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	U S
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	U S
...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	U S
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	U S
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	U S
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	U S
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	U S

88889 rows × 31 columns

Understanding the Data

In [5]:
```
# preview the data first five rows
df.head ()
```

Out[5]:

	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 [6]:
```
# preview the data last five rows
df.tail()
```

Out[6]:

	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 [7]:
```
# we need to identify the columns name
df.columns
```

Out[7]:
```
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.Descript
ion',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injur
ies',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [8]:
```
# checking the indexes
df.index
```

Out[8]:
```
RangeIndex(start=0, stop=88889, step=1)
```

In [9]:
```
# identifing the different data type of each column
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 [10]:
```
# Convert the 'Date' column to datetime format
df['Event.Date'] = pd.to_datetime(df['Event.Date'])
# now run the df.info to see the type of the variables
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  datetime64[ns]
 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: datetime64[ns](1), float64(5), object(25)
memory usage: 21.0+ MB
```

In [11]:
```
# to get the rows and columns
df.shape[0]
df.shape[1]

print(f"This data has {df.shape[0]} rows and {df.shape[1]} columns")
```

```
This data has 88889 rows and 31 columns
```

In [12]:
```
# the descriptive statistics of the numeric variables
df.describe()
# 5 variables are Numeric
```

Out[12]:

	Event.Date	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Tota
count	88889	82805.000000	77488.000000	76379.000000	
mean	1999-09-17 17:13:39.354475904	1.146585	0.647855	0.279881	
min	1948-10-24 00:00:00	0.000000	0.000000	0.000000	
25%	1989-01-15 00:00:00	1.000000	0.000000	0.000000	
50%	1998-07-18 00:00:00	1.000000	0.000000	0.000000	
75%	2009-07-01 00:00:00	1.000000	0.000000	0.000000	
max	2022-12-29 00:00:00	8.000000	349.000000	161.000000	
std	NaN	0.446510	5.485960	1.544084	

In [13]:
```
# the 'descriptive' analysis of the characters or objects
df.describe(include="object")
 # the objects columns are 26
```

Out[13]:

	Event.Id	Investigation.Type	Accident.Number	Location	Country	Latit
count	88889	88889	88889	88837	88663	34
unique	87951	2	88863	27758	219	25
top	20001212X19172	Accident	CEN22LA149	ANCHORAGE, AK	United States	3327
freq	3	85015	2	434	82248	

4 rows × 25 columns

Data Cleaning

Dropping Columns

Drop the insignificant and irrelevant Variable from the dataset

In [14]:
```
# first we star with dropping specific variabes that are needed
# select the column manually
df1 = df.drop(columns=[ 'Schedule',
            #'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'
            ])
```

In [16]:
```
# look for the new shape
df1.shape[0]
df1.shape[1]

print(f"This data has {df1.shape[0]} rows and {df1.shape[1]} columns")
```

```
This data has 88889 rows and 18 columns
```

Dropping Rows

Dropping anything that is not an Airplane

In [17]:
```
# Since our focus is on Airplane the most important column is Aircraft cat
# we need to check the unique values in the column aircraft category

df1["Aircraft.Category"].unique()
```

Out[17]:
```
array([nan, 'Airplane', 'Helicopter', 'Glider', 'Balloon', 'Gyrocraft',
       'Ultralight', 'Unknown', 'Blimp', 'Powered-Lift', 'Weight-Shift',
       'Powered Parachute', 'Rocket', 'WSFT', 'UNK', 'ULTR'], dtype=objec
t)
```

In [18]:
```
# The company main focus is purchasing and operating airplanes
# Therefore we remove everything that is not an airplane
df2 = df1.loc[df['Aircraft.Category']=='Airplane']
df2.head(5)
# Transpose
```

Out[18]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Ai
5	20170710X52551	Accident	1979-09-17	BOSTON, MA	United States	Non-Fatal	
7	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	
8	20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	
12	20020917X02148	Accident	1982-01-02	HOMER, LA	United States	Non-Fatal	
13	20020917X02134	Accident	1982-01-02	HEARNE, TX	United States	Fatal(1)	

In [19]:
```
# new shape
df2.shape [0]
df2.shape [1]
# the rows were initially 88889

print(f"This data has {df2.shape[0]} rows and {df2.shape[1]} columns")
```

```
This data has 27617 rows and 18 columns
```

Drop the rows with missing values

In [20]:
```
# to find how many missing values are in the variables that remained
df2.isna().sum().sort_values(ascending=False)
```

Out[20]:
```
Engine.Type                4226
Purpose.of.flight          3739
Total.Serious.Injuries     3224
Total.Fatal.Injuries       3165
Weather.Condition          3053
Total.Minor.Injuries       2878
Number.of.Engines          2754
Aircraft.damage            1282
Total.Uninjured             900
Injury.Severity             814
Model                        31
Make                          9
Country                       7
Location                      7
Investigation.Type            0
Aircraft.Category             0
Event.Date                    0
Event.Id                      0
dtype: int64
```

In [21]:
```
# the following variable have very little missing values thus drop the row
df3 = df2.dropna(subset=["Model", "Make", "Location", "Country", "Injury.S
df3.head(5)
```

Out[21]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	A
7	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	
8	20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	
12	20020917X02148	Accident	1982-01-02	HOMER, LA	United States	Non-Fatal	
13	20020917X02134	Accident	1982-01-02	HEARNE, TX	United States	Fatal(1)	
14	20020917X02119	Accident	1982-01-02	CHICKASHA, OK	United States	Fatal(1)	

In [22]:
```
# to check the shape of the new data frame
df3.shape[0]
df3.shape[1]

print(f"This data has {df3.shape[0]} rows and {df3.shape[1]} columns")
```

```
This data has 21365 rows and 18 columns
```

In [23]:
```
# to find how many missing values are in the variables that have remained
df3.isna().sum().sort_values(ascending=False)
```

Out[23]:
```
Total.Serious.Injuries    2920
Total.Fatal.Injuries      2903
Total.Minor.Injuries      2550
Total.Uninjured            756
Number.of.Engines          480
Event.Id                     0
Investigation.Type           0
Purpose.of.flight            0
Engine.Type                  0
Model                        0
Make                         0
Aircraft.Category            0
Aircraft.damage              0
Injury.Severity              0
Country                      0
Location                     0
Event.Date                   0
Weather.Condition            0
dtype: int64
```

In [24]:
```
# to find the mean and the median of the numeric variables
df3 [["Number.of.Engines", "Total.Uninjured", "Total.Fatal.Injuries", "Tot
```

Out[24]:

	Number.of.Engines	Total.Uninjured	Total.Fatal.Injuries	Total.Serious.Injuries	Total.M
mean	1.096481	1.597021	0.396219	0.249173	
median	1.000000	1.000000	0.000000	0.000000	

In [27]:

```
# to check the total missing values of the the variables Total.Uninjured,

df3 ["Total.Uninjured"].value_counts().sum()  # to check the total counts
df3 ["Total.Fatal.Injuries"].value_counts().sum()  # to check the total co
df3 ["Total.Serious.Injuries"].value_counts().sum() # to check the total c
df3 ["Total.Minor.Injuries"].value_counts().sum() # to check the total cou

# to find the medium of the Total.Uninjured, Total.Fatal.Injuries, Total.S
median_total_uninjured = df3 ["Total.Uninjured"].median()
median_total_fatal_injuries = df3 ["Total.Fatal.Injuries"].median()
median_total_serious_injuries  = df3 ["Total.Serious.Injuries"].median()
median_total_minor_injuries = df3["Total.Minor.Injuries"].median()
median_number_of_engines = df3["Number.of.Engines"].median()

# to replace the missing values of median of the Total.Uninjured, Total.Fa
df3 ["Total.Uninjured"].fillna(median_total_uninjured, inplace=True)
df3 ["Total.Fatal.Injuries"].fillna(median_total_fatal_injuries, inplace=T
df3 ["Total.Serious.Injuries"].fillna(median_total_serious_injuries, inpla
df3 ["Total.Minor.Injuries"].fillna(median_total_minor_injuries, inplace=T
df3 ["Number.of.Engines"].fillna(median_number_of_engines, inplace= True)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2486130080.py:16: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df3 ["Total.Uninjured"].fillna(median_total_uninjured, inplace=True)
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2486130080.py:17: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df3 ["Total.Fatal.Injuries"].fillna(median_total_fatal_injuries, inplac
e=True)
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2486130080.py:18: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df3 ["Total.Serious.Injuries"].fillna(median_total_serious_injuries, in
place=True)
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2486130080.py:19: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df3 ["Total.Minor.Injuries"].fillna(median_total_minor_injuries, inplac
e=True)
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2486130080.py:20: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df3 ["Number.of.Engines"].fillna(median_number_of_engines, inplace= Tru
e);
```

In [28]:
```
df3.isna().sum().sort_values(ascending=False)
```

Out[28]:
```
Event.Id                  0
Investigation.Type        0
Total.Uninjured           0
Total.Minor.Injuries      0
Total.Serious.Injuries    0
Total.Fatal.Injuries      0
Purpose.of.flight         0
Engine.Type               0
Number.of.Engines         0
Model                     0
Make                      0
Aircraft.Category         0
Aircraft.damage           0
Injury.Severity           0
Country                   0
Location                  0
Event.Date                0
Weather.Condition         0
dtype: int64
```

Drop and Edit Duplicates

In [29]:
```
# to check duplicates
df3.duplicated().sum()

# there are 3 rows that are duplicaed

# Thus remove the rows that are duplicated
df4 = df3.drop_duplicates()
```

In [30]:
```
#to verify the duplicate have been removed
df4.duplicated().sum()
```

Out[30]:
```
0
```

In [32]:

```
# to check the duplicates
df4["Make"].duplicated().any() #True

# in Make variable remove duplicates

df4["Make"].value_counts().head(20) # this shows the output has duplicates

# thus
df4["Make"] = df4['Make'].str.lower()
df4["Make"] = df4["Make"].str.capitalize()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2605200923.py:9: SettingW
ithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df4["Make"] = df4['Make'].str.lower()
C:\Users\user\AppData\Local\Temp\ipykernel_3276\2605200923.py:10: Setting
WithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df4["Make"] = df4["Make"].str.capitalize()
```

In [33]:
```
# to confirm that the data has no duplicate
df4["Make"].value_counts().head(20)
```

Out[33]:
```
Make
Cessna                7022
Piper                 3981
Beech                 1340
Mooney                 361
Bellanca               267
Grumman                223
Maule                  212
Aeronca                205
Boeing                 188
Air tractor inc        185
Air tractor            183
Cirrus design corp     170
Champion               158
Luscombe               152
Stinson                135
Taylorcraft            104
North american         101
Cirrus                  89
Aero commander          86
Vans                    81
Name: count, dtype: int64
```

In [34]:
```
# to get the rows and columns of df4
df4.shape[0]
df4.shape[1]

print(f"This data has {df4.shape[0]} rows and {df4.shape[1]} columns")
```

```
This data has 21362 rows and 18 columns
```

Slitting and Spilling

In [35]:

```
# Form the abbreviation Column from location colum by splitting and
df4['Abbreviation'] = df4['Location'].str.split(',').str[1].str.strip()
# to create an abbreviation column

df4.head(5)
# the abbreviation column is at the end of the columns
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3276\60246515.py:2: SettingWit
hCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df4['Abbreviation'] = df4['Location'].str.split(',').str[1].str.strip()
```

Out[35]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	A
7	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	
8	20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	
12	20020917X02148	Accident	1982-01-02	HOMER, LA	United States	Non-Fatal	
13	20020917X02134	Accident	1982-01-02	HEARNE, TX	United States	Fatal(1)	
14	20020917X02119	Accident	1982-01-02	CHICKASHA, OK	United States	Fatal(1)	

In [36]:
```
# the codes given are the US states code which shows taht the location of
df5 = df4.loc[df["Country"] == "United States"]

df5.head(5)
```

Out[36]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	A
7	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	
8	20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	
12	20020917X02148	Accident	1982-01-02	HOMER, LA	United States	Non-Fatal	
13	20020917X02134	Accident	1982-01-02	HEARNE, TX	United States	Fatal(1)	
14	20020917X02119	Accident	1982-01-02	CHICKASHA, OK	United States	Fatal(1)	

In [37]:
```
# Calculate the average and round to 2 decimal places
df5["Sum_Total_Injuries"] = df5["Total.Fatal.Injuries"] + df5["Total.Minor
df5.head()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3276\1813218833.py:2: SettingW
ithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df5["Sum_Total_Injuries"] = df5["Total.Fatal.Injuries"] + df5["Total.Mi
nor.Injuries"] + df5["Total.Serious.Injuries"]
```

Out[37]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	A
7	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	
8	20020909X01561	Accident	1982-01-01	EAST HANOVER, NJ	United States	Non-Fatal	
12	20020917X02148	Accident	1982-01-02	HOMER, LA	United States	Non-Fatal	
13	20020917X02134	Accident	1982-01-02	HEARNE, TX	United States	Fatal(1)	
14	20020917X02119	Accident	1982-01-02	CHICKASHA, OK	United States	Fatal(1)	

In [38]:
```
# after adding a new column check the shape of df5
df5.shape[0]
df5.shape[1]

print(f"This data has {df5.shape[0]} rows and {df5.shape[1]} columns")
```

```
This data has 21083 rows and 20 columns
```

Merging Data

In [39]:
```
# upload the data set
US_state_code = pd.read_csv("USState_Codes.csv")
US_state_code
```

Out[39]:

	US_State	Abbreviation
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA
...	...	...
57	Virgin Islands	VI
58	Washington_DC	DC
59	Gulf of mexico	GM
60	Atlantic ocean	AO
61	Pacific ocean	PO

62 rows × 2 columns

In [40]:
```
# to merge the data from AviationData and USStates_Codes.csv
df6 = df5.merge(US_state_code, on = "Abbreviation")
df6.head (2)
```

Out[40]:

	Event.Id	Investigation.Type	Event.Date	Location	Country	Injury.Severity	Airc
0	20020909X01562	Accident	1982-01-01	PULLMAN, WA	United States	Non-Fatal	
1	20020917X02574	Accident	1982-01-08	PULLMAN, WA	United States	Non-Fatal	

2 rows × 21 columns

In [72]:
```
# to make the Injury.Severity unique values uniform
df6['Injury.Severity'] = df6['Injury.Severity'].str.split('(').str[0]
df6['Injury.Severity'].unique()
```

Out[72]:
```
array(['Non-Fatal', 'Fatal', 'Serious', 'Incident', 'Minor'], dtype=objec
t)
```

In [73]:
```
# after adding a new column check the shape of df6
df6.shape[0]
df6.shape[1]

print(f"This data has {df6.shape[0]} rows and {df6.shape[1]} columns")
```

```
This data has 21057 rows and 21 columns
```

In [74]:
```
df6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21057 entries, 0 to 21056
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                21057 non-null  object
 1   Investigation.Type      21057 non-null  object
 2   Event.Date              21057 non-null  datetime64[ns]
 3   Location                21057 non-null  object
 4   Country                 21057 non-null  object
 5   Injury.Severity         21057 non-null  object
 6   Aircraft.damage         21057 non-null  object
 7   Aircraft.Category       21057 non-null  object
 8   Make                    21057 non-null  object
 9   Model                   21057 non-null  object
 10  Number.of.Engines       21057 non-null  float64
 11  Engine.Type             21057 non-null  object
 12  Purpose.of.flight       21057 non-null  object
 13  Total.Fatal.Injuries    21057 non-null  float64
 14  Total.Serious.Injuries  21057 non-null  float64
 15  Total.Minor.Injuries    21057 non-null  float64
 16  Total.Uninjured         21057 non-null  float64
 17  Weather.Condition       21057 non-null  object
 18  Abbreviation            21057 non-null  object
 19  Sum_Total_Injuries      21057 non-null  float64
 20  US_State                21057 non-null  object
dtypes: datetime64[ns](1), float64(6), object(14)
memory usage: 3.4+ MB
```

Discriptive and Summary Statistics

In [91]:
```
# the descriptive statistics of the numeric variables
df6.describe()
# 7 variables are Numeric
```

Out[91]:

	Event.Date	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total
count	21057	21057.000000	21057.000000	21057.000000	
mean	2008-05-22 04:01:24.114546176	1.092938	0.288408	0.212423	
min	1982-01-01 00:00:00	0.000000	0.000000	0.000000	
25%	2007-01-09 00:00:00	1.000000	0.000000	0.000000	
50%	2011-07-02 00:00:00	1.000000	0.000000	0.000000	
75%	2016-05-05 00:00:00	1.000000	0.000000	0.000000	
max	2022-11-09 00:00:00	8.000000	228.000000	26.000000	
std	NaN	0.305652	1.784100	0.605149	

In [93]:
```
# the 'descriptive' analysis of the characters or objects
df6.describe(include="object")
```

Out[93]:

	Event.Id	Investigation.Type	Location	Country	Injury.Severity	Aircraft.dar
count	21057	21057	21057	21057	21057	2
unique	21004	2	9776	1	5	
top	20020917X03442	Accident	Anchorage, AK	United States	Non-Fatal	Subst
freq	2	20842	83	21057	17629	1

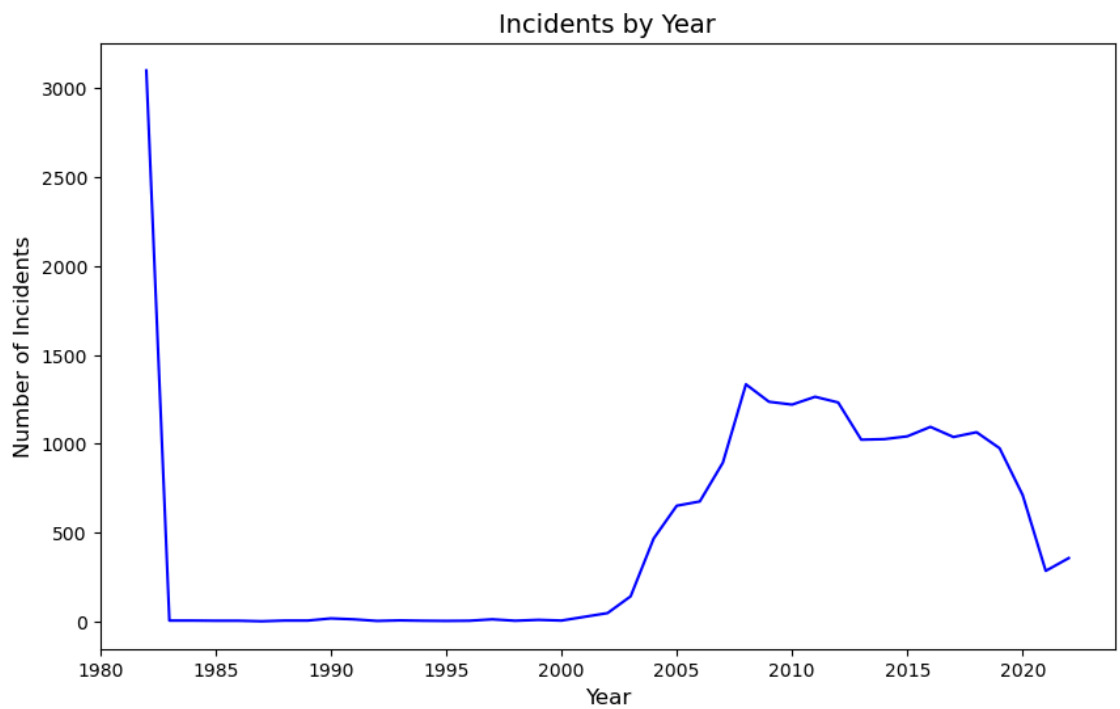
In [44]:

```
# Plot: Incidents by Year as a line graph
plt.figure(figsize=(10, 6))
plt.plot(df6['Event.Date'].dt.year.value_counts().sort_index(), color='b')

# labels and title
plt.title('Incidents by Year', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Incidents', fontsize=12)

# Show plot

plt.show()
```



In [47]:

```
# To calculate percentage and counts for incidents by location
location_incidents = df6['Location'].value_counts()

# Calculate percentage of incidents by location
location_percentage = (location_incidents / location_incidents.sum()) * 10

# Combine counts and percentages into a summary DataFrame
location_summary = pd.DataFrame({
    'Count': location_incidents,
    'Percentage': location_percentage
})
location_summary.sort_values(by=['Percentage'], ascending=False).head(5)
```

Out[47]:

Location	Count	Percentage
Anchorage, AK	83	0.394168
Palmer, AK	66	0.313435
Fairbanks, AK	55	0.261196
Talkeetna, AK	54	0.256447
Phoenix, AZ	52	0.246949

In [46]:

```
location_summary.sort_values(by=['Percentage'], ascending=True).head(5)
```

Out[46]:

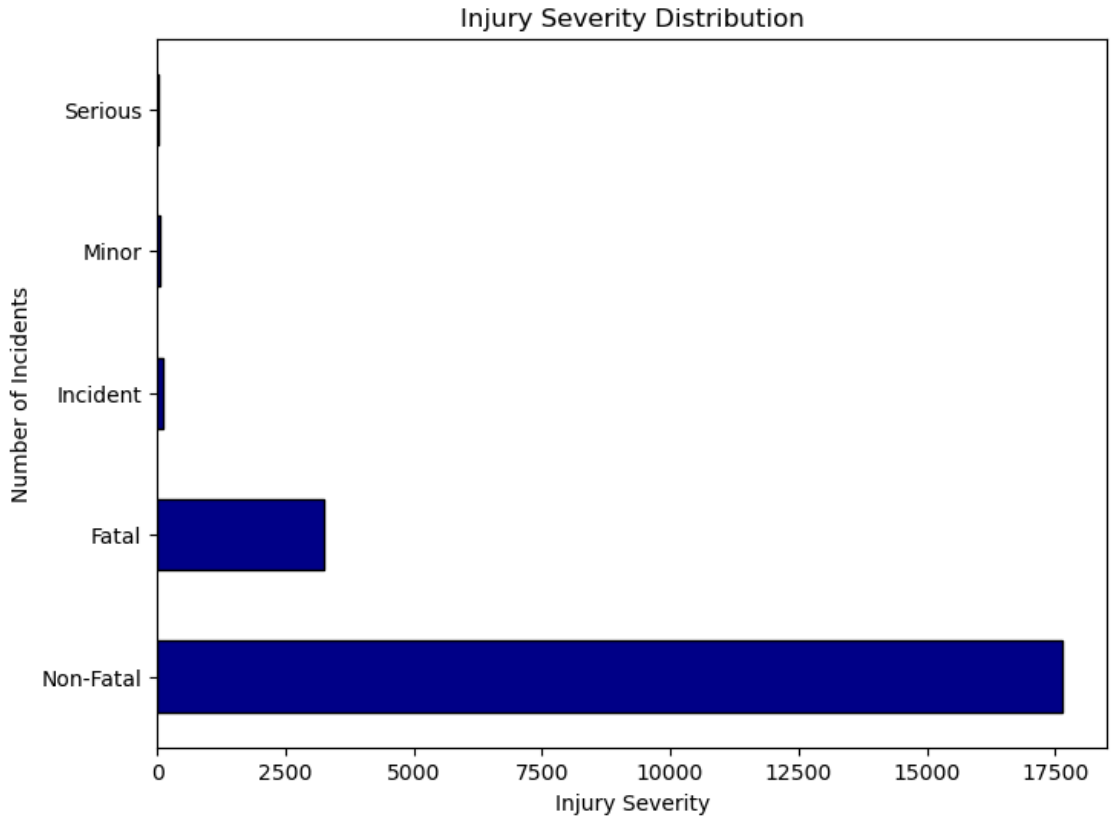
Location	Count	Percentage
Trussville, AL	1	0.004749
EDGERTON, WI	1	0.004749
ABBEYVILLE, LA	1	0.004749
Walworth, WI	1	0.004749
MONROE, LA	1	0.004749

In [81]:

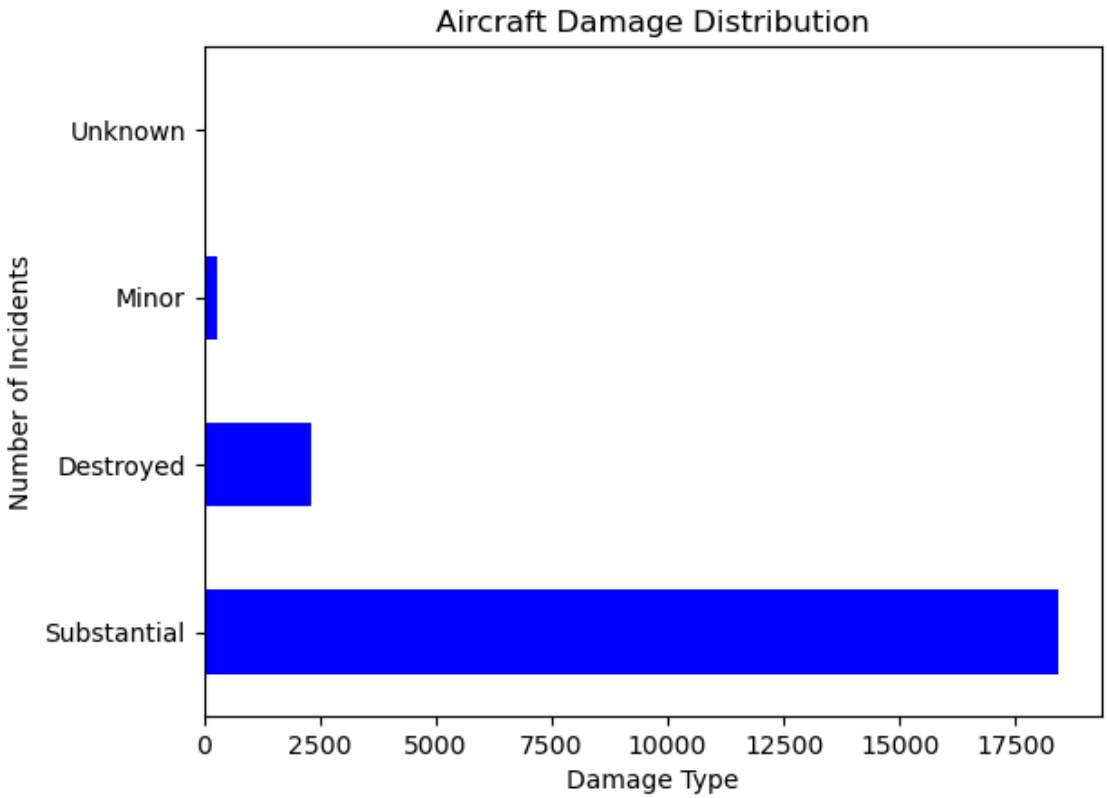
```
# Calculate the distribution of Injury Severity
injury_severity_counts = df6['Injury.Severity'].value_counts()

# Plot a bar chart
plt.figure(figsize=(8, 6))
colors = ['darkblue']
injury_severity_counts.plot(kind='barh', color=colors, edgecolor='black')
plt.title('Injury Severity Distribution')
plt.xlabel('Injury Severity')
plt.ylabel('Number of Incidents')

plt.show()
```



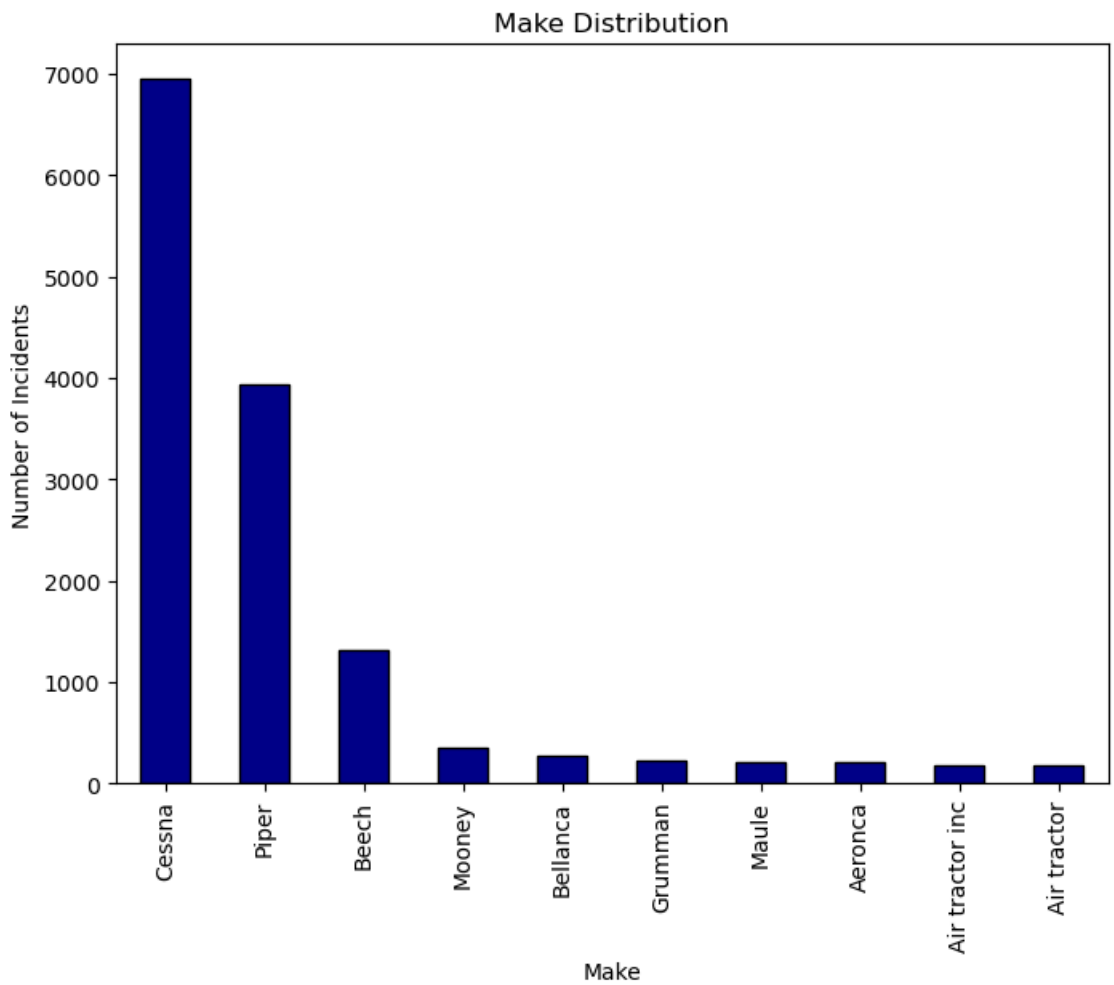
In [68]:
```
# Plot: Aircraft Damage Distribution
plt.subplot(1, 1, 1)
df6['Aircraft.damage'].value_counts().plot(kind='barh', color='blue')
plt.title('Aircraft Damage Distribution')
plt.xlabel('Damage Type')
plt.ylabel('Number of Incidents');
```



In [90]:

```
# Calculate the distribution of Make
Make_counts = df6['Make'].value_counts().head(10)

# Plot a bar chart
plt.figure(figsize=(8, 6))
Make_counts.plot(kind='bar', color='darkblue', edgecolor='black')
plt.title('Make Distribution')
plt.xlabel('Make')
plt.ylabel('Number of Incidents');
```



In [86]:

```
# To calculate percentage and counts for incidents by Model
Model_incidents = df6['Model'].value_counts()

# Calculate percentage of incidents by Model
Model_percentage = (Model_incidents / Model_incidents.sum()) * 100

# Combine counts and percentages into a summary DataFrame
Model_summary = pd.DataFrame({
    'Count': Model_incidents,
    'Percentage': Model_percentage
})
# indicate the highest 10 values
Model_summary.sort_values(by=['Percentage'], ascending=False).head(10)
```

Out[86]:

Model	Count	Percentage
172	696	3.305314
152	410	1.947096
172N	288	1.367716
182	269	1.277485
172S	248	1.177756
180	221	1.049532
PA28	211	1.002042
150	207	0.983046
PA-28-140	197	0.935556
172M	195	0.926058

In [65]:

```
# to indicate the lowest 10 values
Model_summary.sort_values(by=['Percentage'], ascending=True).head(10)
```

Out[65]:

Model	Count	Percentage
PA-28-200R	1	0.004749
Sportstar-Evektor	1	0.004749
SKYSTAR KITFOX 4	1	0.004749
65C	1	0.004749
AVID FLYER MK IV	1	0.004749
PA-22-125	1	0.004749
Benoist Type XIV	1	0.004749
PRECEPTOR ULTRA PUP	1	0.004749
AVID AMPHIBIAN	1	0.004749
2S	1	0.004749

In [94]:
```
# To calculate percentage and counts for incidents by Model
Number_of_engines_incidents = df6['Number].value_counts()

# Calculate percentage of incidents by Model
Model_percentage = (Model_incidents / Model_incidents.sum()) * 100

# Combine counts and percentages into a summary DataFrame
Model_summary = pd.DataFrame({
    'Count': Model_incidents,
    'Percentage': Model_percentage
})
# indicate the highest 10 values
Model_summary.sort_values(by=['Percentage'], ascending=False).head(10)
```

Out[94]:
```
array([1., 2., 3., 4., 8., 0.])
```

In [118]:
```
# Pivot table to count the number of occurrences of each injury severity p
severity_counts = df6.pivot_table(index= 'df6['Make'].value_counts().head(

# Plot a stacked bar chart
severity_counts.plot(kind='bar', stacked=True, color=['red', 'orange', 'ye
```

```
  Cell In[118], line 2
    severity_counts = df6.pivot_table(index='(df6['Make'].value_counts().
head(10))', columns='Injury.Severity', aggfunc='size', fill_value=0)
                                                    ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

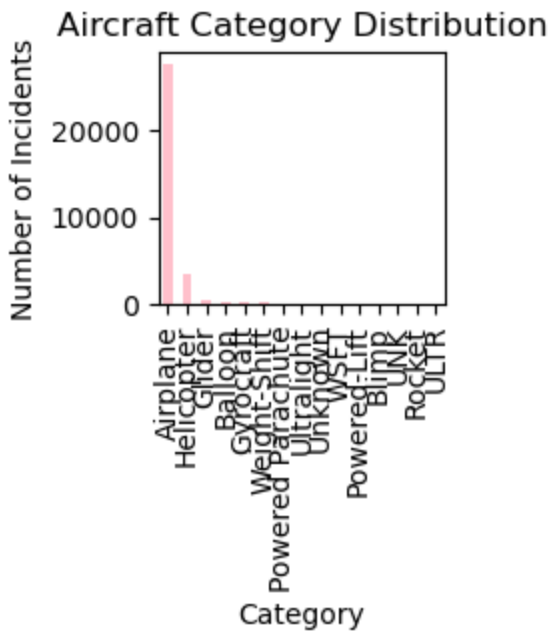
In [99]:
```
df6.columns
```

Out[99]:
```
Index(['Event.Id', 'Investigation.Type', 'Event.Date', 'Location', 'Count
ry',
       'Injury.Severity', 'Aircraft.damage', 'Aircraft.Category', 'Make',
       'Model', 'Number.of.Engines', 'Engine.Type', 'Purpose.of.flight',
       'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Abbreviation', 'Sum_Total_Injuries', 'US_State'],
      dtype='object')
```

In [56]:

```
# Plot 6: Aircraft Category Distribution
plt.subplot(2, 3, 6)
df['Aircraft.Category'].value_counts().plot(kind='bar', color='pink')
plt.title('Aircraft Category Distribution')
plt.xlabel('Category')
plt.ylabel('Number of Incidents')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



In [57]:

```
# Plot 3: Incidents by Location
plt.subplot(2, 3, 3)
df6['Location'].value_counts().plot(kind='bar', color='coral')
plt.title('Incidents by Location')
plt.xlabel('Location')
plt.ylabel('Number of Incidents');
```

Out[57]:

```
Text(0, 0.5, 'Number of Incidents')

Error in callback <function flush_figures at 0x00000259C14DC360> (for
post_execute), with arguments args (),kwargs {}:

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call
last)
File c:\Users\user\anaconda3\Lib\site-packages\matplotlib_inline\backe
nd_inline.py:126, in flush_figures()
    123 if InlineBackend.instance().close_figures:
    124     # ignore the tracking, just draw and close all figures
    125     try:
--> 126         return show(True)
    127     except Exception as e:
    128         # safely show traceback if in IPython, else raise
    129         ip = get_ipython()

File c:\Users\user\anaconda3\Lib\site-packages\matplotlib_inline\backe
```

In [1]:

```
df6.column
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
Cell In[1], line 1
----> 1 df6.column

NameError: name 'df6' is not defined
```

In [ ]:
```
# Descriptive analysis
descriptive_stats = {
    'Incidents by Year': df['Event.Date'].dt.year.value_counts().sort_inde
    'Incidents by Location': df['Location'].value_counts(),
    'Incidents by Country': df['Country'].value_counts(),
    'Injury Severity Distribution': df['Injury.Severity'].value_counts(),
    'Aircraft Damage Distribution': df['Aircraft.damage'].value_counts(),
    'Aircraft Category Distribution': df['Aircraft.Category'].value_counts
    'Make Distribution': df['Make'].value_counts(),
    'Model Distribution': df['Model'].value_counts(),
    'Engine Count Distribution': df['Number.of.Engines'].value_counts(),
    'Weather Condition Distribution': df['Weather.Condition'].value_counts
}
# Displaying the results
print(descriptive_stats)
```

```
Incidents by Year:
Event.Date
1948       1
1962       1
1974       1
1977       1
1979       2
1981       1
1982    3593
1983    3556
1984    3457
1985    3096
1986    2880
1987    2828
1988    2730
1989    2544
1990    2518
1991    2462
```

In [ ]:

Continue from Here

In [551]:
```
df3["Engine.Type"].value_counts()
df3["Purpose.of.flight"].value_counts()
df3["Weather.Condition"].value_counts()
df3["Aircraft.damage"].value_counts()
df6["Injury "].value_counts()
```

Out[551]:
```
Aircraft.damage
Substantial    18559
Destroyed       2457
Minor            343
Unknown            6
Name: count, dtype: int64
```

In [553]:
```
df5 = df4.groupby(["Make", "Model"])["Injury.Severity"].value_counts().sor
```

In [ ]:
```

```

In [ ]:
```
cleaned_df = df
cleaned_df.to_csv('cleaned_data.csv', index = False)
```

In [ ]:
```
# assigning safety weightage
Fatal = 6
Serious = 3
Minor = 1

# calculating safety score..
df2['Safety_Score'] =   (df2['Total.Fatal.Injuries'] * Fatal
                                 + df2['Total.Serious.Injuries'] * Seriou
                                 + df2['Total.Minor.Injuries'] * Minor)
df2.head()
```

In [600]:
```
(df4["Total.Fatal.Injuries"] + df4["Total.Minor.Injuries"] + df4["Total.Se
```

Out[600]:
```
7        0.000000
8        0.000000
12       0.333333
13       0.333333
14       0.333333
           ...
88639    0.000000
88647    0.000000
88661    0.000000
88735    0.333333
88767    0.000000
Length: 21362, dtype: float64
```

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

Out[67]:
```
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.Descript
ion',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injur
ies',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [125]:
```
df5.shape
```

Out[125]:
```
(21083, 19)
```