Data Understanding

The high level goal of analyzing the Aviation Accident Database & Synopses, up to 2023 Data set is to determine which aircraft has the lowest risk to ensure the company starts a new business endevour. I'll do that by determining which models causes the most accidents per year. When the Dataset was being collected in 2023, there were six fatal accidents globally in 2023, with these resulting in 115 onboard deaths, and this triggered a need to look for a solution.

Data Understanding

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

In [ ]:
```
import zipfile
```

In [ ]:
```
#Reading the Dataset
df = pd.read_csv("AviationData.csv (1).zip", encoding="iso-8859-1", low_memory=Fals
```

In [ ]:
```
#Checking for number of rows and columns
df.shape
```

Out[ ]:
```
(88889, 31)
```

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

Out[ ]:

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

5 rows × 31 columns

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

Out[ ]:

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

5 rows × 31 columns

Checking for summary statistics

In [ ]:
```
#Checking for summary statistics
df.describe()
```

Out[ ]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Tot
count	82805.000000	77488.000000	76379.000000	76956.000000	8
mean	1.146585	0.647855	0.279881	0.357061	
std	0.446510	5.485960	1.544084	2.235625	
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	

Checking for Data types

In [ ]:

```
#Checking for the Data types
df.dtypes
```

```
Out[ ]: 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 for missing values

```
In [ ]: #Checking for missing values
        df.isna().sum()
```

```
Out[ ]: 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
```

Checking for Duplicates

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

```
Out[ ]: np.int64(0)
```

```
In [ ]: 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 [ ]:
```
df.index
```

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

In [ ]:
```
df.value_counts()
```

Out[ ]:
```
Series([], Name: count, dtype: int64)
```

In [ ]:
```
df.values
```

```
Out[ ]: array([['20001218X45444', 'Accident', 'SEA87LA080', ..., 'Cruise',
                'Probable Cause', nan],
               ['20001218X45447', 'Accident', 'LAX94LA336', ..., 'Unknown',
                'Probable Cause', '19-09-1996'],
               ['20061025X01555', 'Accident', 'NYC07LA005', ..., 'Cruise',
                'Probable Cause', '26-02-2007'],
               ...,
               ['20221227106497', 'Accident', 'WPR23LA075', ..., nan, nan,
                '27-12-2022'],
               ['20221227106498', 'Accident', 'WPR23LA076', ..., nan, nan, nan],
               ['20221230106513', 'Accident', 'ERA23LA097', ..., nan, nan,
                '30-12-2022']], dtype=object)
```

Data Cleaning

```
In [ ]: #Checking if there are duplicates
        df.duplicated().sum()
```

```
Out[ ]: np.int64(0)
```

```
In [ ]: df.isna().sum()
```

```
Out[ ]: 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 [ ]: df.isna().sum()
```

```
Out[ ]: 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 [ ]: df.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
df.dropna()
```

Out[ ]:

Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude

0 rows × 31 columns

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

Out[ ]:
```
np.int64(0)
```

In [ ]:
```
print(df.columns)
```

```
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.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'],
      dtype='object')
```

Calculate the percentage of values being NaN for each column

In [ ]:
```
# Calculate the percentage of values being NaN for each column
rows = len(df)
missing = df.isna().sum()
percentage_missing = missing / rows
```

In [ ]:
```
# Put the data in a DataFrame and sort it
percentage_missing_df = pd.DataFrame({'Missing' : percentage_missing})
percentage_missing_df.sort_values('Missing', ascending = False, inplace = True)
```

In [ ]:
```
#printing columns with more than 10% missing values
print(percentage_missing_df[percentage_missing_df['Missing']>0.1])
```

```
                         Missing
Schedule                0.858453
Air.carrier             0.812710
FAR.Description         0.639742
Aircraft.Category       0.636772
Longitude               0.613304
Latitude                0.613203
Airport.Code            0.436016
Airport.Name            0.407081
Broad.phase.of.flight   0.305606
Publication.Date        0.154924
Total.Serious.Injuries  0.140737
Total.Minor.Injuries    0.134246
Total.Fatal.Injuries    0.128261
```

Drop columns with over 50% missing values

In [ ]:
```
# Drop columns with over 50% missing values
cols_to_drop = list(percentage_missing_df[percentage_missing_df['Missing'] > 0.5].i
df.drop(columns = cols_to_drop, axis = 1, inplace = True)
print(cols_to_drop)
```

```
['Schedule', 'Air.carrier', 'FAR.Description', 'Aircraft.Category', 'Longitude', 'La
titude']
```

Drop records not Accidents in United States

```
In [ ]: # Drop records not Accidents in United States
        before = len(df)
        df = df[(df['Investigation.Type'] == 'Accident') & (df['Country'] == 'United States
        dropped = before - len(df)
        print(str(dropped) + ' rows dropped.')
```

```
8983 rows dropped.
```

```
In [ ]: # Convert Date to a datetime, add a Year & Month column and remove data before 1982
        df['Event.Date'] = pd.to_datetime(df['Event.Date'])
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 79906 entries, 0 to 88888
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                79906 non-null  object
 1   Investigation.Type      79906 non-null  object
 2   Accident.Number         79906 non-null  object
 3   Event.Date              79906 non-null  datetime64[ns]
 4   Location                79895 non-null  object
 5   Country                 79906 non-null  object
 6   Airport.Code            47449 non-null  object
 7   Airport.Name            49918 non-null  object
 8   Injury.Severity         79854 non-null  object
 9   Aircraft.damage         78782 non-null  object
 10  Registration.Number     79838 non-null  object
 11  Make                    79894 non-null  object
 12  Model                   79877 non-null  object
 13  Amateur.Built           79891 non-null  object
 14  Number.of.Engines       78147 non-null  float64
 15  Engine.Type             76988 non-null  object
 16  Purpose.of.flight       78025 non-null  object
 17  Total.Fatal.Injuries    69641 non-null  float64
 18  Total.Serious.Injuries  68921 non-null  float64
 19  Total.Minor.Injuries    69551 non-null  float64
 20  Total.Uninjured         74911 non-null  float64
 21  Weather.Condition       79345 non-null  object
 22  Broad.phase.of.flight   59297 non-null  object
 23  Report.Status           77341 non-null  object
 24  Publication.Date        67649 non-null  object
dtypes: datetime64[ns](1), float64(5), object(19)
memory usage: 15.9+ MB
```

```
In [ ]: #Add a day, month & year column
        df['Year'] = df['Event.Date'].dt.year
        df['Month.Abbr'] = df['Event.Date'].dt.month_name().str[:3]
        df['Day.Name.Abbr'] = df['Event.Date'].dt.day_name().str[:3]

        # Add a weekend column
        df.loc[(df['Day.Name.Abbr'] == 'Sat') | (df['Day.Name.Abbr'] == 'Sun'), 'Weekend']
        df.loc[(df['Day.Name.Abbr'] != 'Sat') & (df['Day.Name.Abbr'] != 'Sun'), 'Weekend']

        # Remove data before 1982
        df = df[df['Year'] >= 1982]
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 79899 entries, 7 to 88888
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                79899 non-null  object
 1   Investigation.Type      79899 non-null  object
 2   Accident.Number         79899 non-null  object
 3   Event.Date              79899 non-null  datetime64[ns]
 4   Location                79888 non-null  object
 5   Country                 79899 non-null  object
 6   Airport.Code            47449 non-null  object
 7   Airport.Name            49918 non-null  object
 8   Injury.Severity         79847 non-null  object
 9   Aircraft.damage         78775 non-null  object
 10  Registration.Number     79831 non-null  object
 11  Make                    79887 non-null  object
 12  Model                   79870 non-null  object
 13  Amateur.Built           79884 non-null  object
 14  Number.of.Engines       78141 non-null  float64
 15  Engine.Type             76982 non-null  object
 16  Purpose.of.flight       78019 non-null  object
 17  Total.Fatal.Injuries    69635 non-null  float64
 18  Total.Serious.Injuries  68916 non-null  float64
 19  Total.Minor.Injuries    69546 non-null  float64
 20  Total.Uninjured         74905 non-null  float64
 21  Weather.Condition       79338 non-null  object
 22  Broad.phase.of.flight   59290 non-null  object
 23  Report.Status           77334 non-null  object
 24  Publication.Date        67643 non-null  object
 25  Year                    79899 non-null  int32
 26  Month.Abbr              79899 non-null  object
 27  Day.Name.Abbr           79899 non-null  object
 28  Weekend                 79899 non-null  object
dtypes: datetime64[ns](1), float64(5), int32(1), object(22)
memory usage: 18.0+ MB
```

In [ ]:
```
# Merge same airport names together
df['Airport.Name'].replace(to_replace = '(?i)^.*private.*$', value = 'PRIVATE', inp
df['Airport.Name'].replace(to_replace = '(?i)none', value = 'NONE', inplace = True,
df['Airport.Name'].value_counts().nlargest(10)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_5972\2540906905.py:2: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Airport.Name'].replace(to_replace = '(?i)^.*private.*$', value = 'PRIVATE', in
place = True, regex = True)
C:\Users\Admin\AppData\Local\Temp\ipykernel_5972\2540906905.py:3: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Airport.Name'].replace(to_replace = '(?i)none', value = 'NONE', inplace = Tru
e, regex = True)
```

Out[ ]:

```
Airport.Name
PRIVATE          1204
NONE              143
MERRILL FIELD      83
VAN NUYS           79
MUNICIPAL          75
CENTENNIAL         74
UNKNOWN            68
CHINO              53
BIRCHWOOD          49
SEDONA             47
Name: count, dtype: int64
```

In [ ]:

```
# Merge same registration numbers together
df['Registration.Number'].replace(to_replace = '(?i)none', value = 'NONE', inplace
df['Registration.Number'].value_counts().nlargest(10)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_5972\2380449474.py:2: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Registration.Number'].replace(to_replace = '(?i)none', value = 'NONE', inplace
= True, regex = True)
```

Out[ ]:

```
Registration.Number
NONE      341
UNREG     114
N20752      7
N4101E      6
N11VH       6
N8402K      6
N53893      6
N5408Y      6
N121CC      6
N3331R      5
Name: count, dtype: int64
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 79899 entries, 7 to 88888
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                79899 non-null  object
 1   Investigation.Type      79899 non-null  object
 2   Accident.Number         79899 non-null  object
 3   Event.Date              79899 non-null  datetime64[ns]
 4   Location                79888 non-null  object
 5   Country                 79899 non-null  object
 6   Airport.Code            47449 non-null  object
 7   Airport.Name            49918 non-null  object
 8   Injury.Severity         79847 non-null  object
 9   Aircraft.damage         78775 non-null  object
 10  Registration.Number     79831 non-null  object
 11  Make                    79887 non-null  object
 12  Model                   79870 non-null  object
 13  Amateur.Built           79884 non-null  object
 14  Number.of.Engines       78141 non-null  float64
 15  Engine.Type             76982 non-null  object
 16  Purpose.of.flight       78019 non-null  object
 17  Total.Fatal.Injuries    69635 non-null  float64
 18  Total.Serious.Injuries  68916 non-null  float64
 19  Total.Minor.Injuries    69546 non-null  float64
 20  Total.Uninjured         74905 non-null  float64
 21  Weather.Condition       79338 non-null  object
 22  Broad.phase.of.flight   59290 non-null  object
 23  Report.Status           77334 non-null  object
 24  Publication.Date        67643 non-null  object
 25  Year                    79899 non-null  int32
 26  Month.Abbr              79899 non-null  object
 27  Day.Name.Abbr           79899 non-null  object
 28  Weekend                 79899 non-null  object
dtypes: datetime64[ns](1), float64(5), int32(1), object(22)
memory usage: 18.0+ MB
```

Merge different capitalizations of Make together

In [ ]:
```
# Merge different capitalizations of Make together
df['Make'] = df['Make'].str.title()
df['Make'].value_counts().nlargest(10)
```

```
Out[ ]: Make
        Cessna      25566
        Piper       14008
        Beech        4892
        Bell         2236
        Mooney       1272
        Grumman      1131
        Bellanca     1036
        Boeing        931
        Robinson      916
        Hughes        868
        Name: count, dtype: int64
```

```
In [ ]: # Transform Amateur Built to boolean
        df['Amateur.Built'].replace(to_replace = ['Yes', 'Y'], value = True, inplace = True
        df['Amateur.Built'].replace(to_replace = ['No', 'N'], value = False, inplace = True
        df['Amateur.Built'].value_counts()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_5972\3362052493.py:2: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Amateur.Built'].replace(to_replace = ['Yes', 'Y'], value = True, inplace = Tru
e, regex = False)
```

```
Out[ ]: Amateur.Built
        False    71589
        True      8295
        Name: count, dtype: int64
```

Split location in city and state

```
In [ ]: # Split location in city and state
        df['City'] = df['Location'].str.split(',').str[0]
        df['State'] = df['Location'].str.split(',').str[1]
        df[['City', 'State']].head(10)
```

Out[ ]:

	City	State
7	PULLMAN	WA
8	EAST HANOVER	NJ
9	JACKSONVILLE	FL
10	HOBBS	NM
11	TUSKEGEE	AL
12	HOMER	LA
13	HEARNE	TX
14	CHICKASHA	OK
15	LITTLE ROCK	AR
16	MIDWAY	UT

Categorize the amount of injuries as this is already in another column

In [ ]:

```
# Remove amount of injuries as this is already in another column
df['Injury.Severity'] = df['Injury.Severity'].str.split('(').str[0]
df['Injury.Severity'].value_counts()
```

Out[ ]:

```
Injury.Severity
Non-Fatal      64457
Fatal          15019
Minor            203
Serious          153
Unavailable       15
Name: count, dtype: int64
```

In [ ]:

```
# Merge weather condition unknowns
df['Weather.Condition'].replace(to_replace = ['Unk', 'UNK'], value = 'Unknown', inp
df['Weather.Condition'].value_counts()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_5972\1600600250.py:2: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Weather.Condition'].replace(to_replace = ['Unk', 'UNK'], value = 'Unknown', in
place = True, regex = False)
```

Out[ ]:
```
Weather.Condition
VMC        73340
IMC         5387
Unknown      611
Name: count, dtype: int64
```

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

Out[ ]:
```
(79899, 31)
```

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

Out[ ]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Airport.Code', 'Airport.Name',
       'Injury.Severity', 'Aircraft.damage', 'Registration.Number', 'Make',
       'Model', 'Amateur.Built', 'Number.of.Engines', 'Engine.Type',
       'Purpose.of.flight', 'Total.Fatal.Injuries', 'Total.Serious.Injuries',
       'Total.Minor.Injuries', 'Total.Uninjured', 'Weather.Condition',
       'Broad.phase.of.flight', 'Report.Status', 'Publication.Date', 'Year',
       'Month.Abbr', 'Day.Name.Abbr', 'Weekend', 'City', 'State'],
      dtype='object')
```

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

Out[ ]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Cou
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA	Un St
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ	Un St
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL	Un St
10	20020909X01559	Accident	FTW82DA034	1982-01-01	HOBBS, NM	Un St
11	20020909X01558	Accident	ATL82DKJ10	1982-01-01	TUSKEGEE, AL	Un St

5 rows × 31 columns

In [ ]:
```
injury_data = df[df['Injury.Severity'] != 'Unavailable']
```

Number of accidents per year

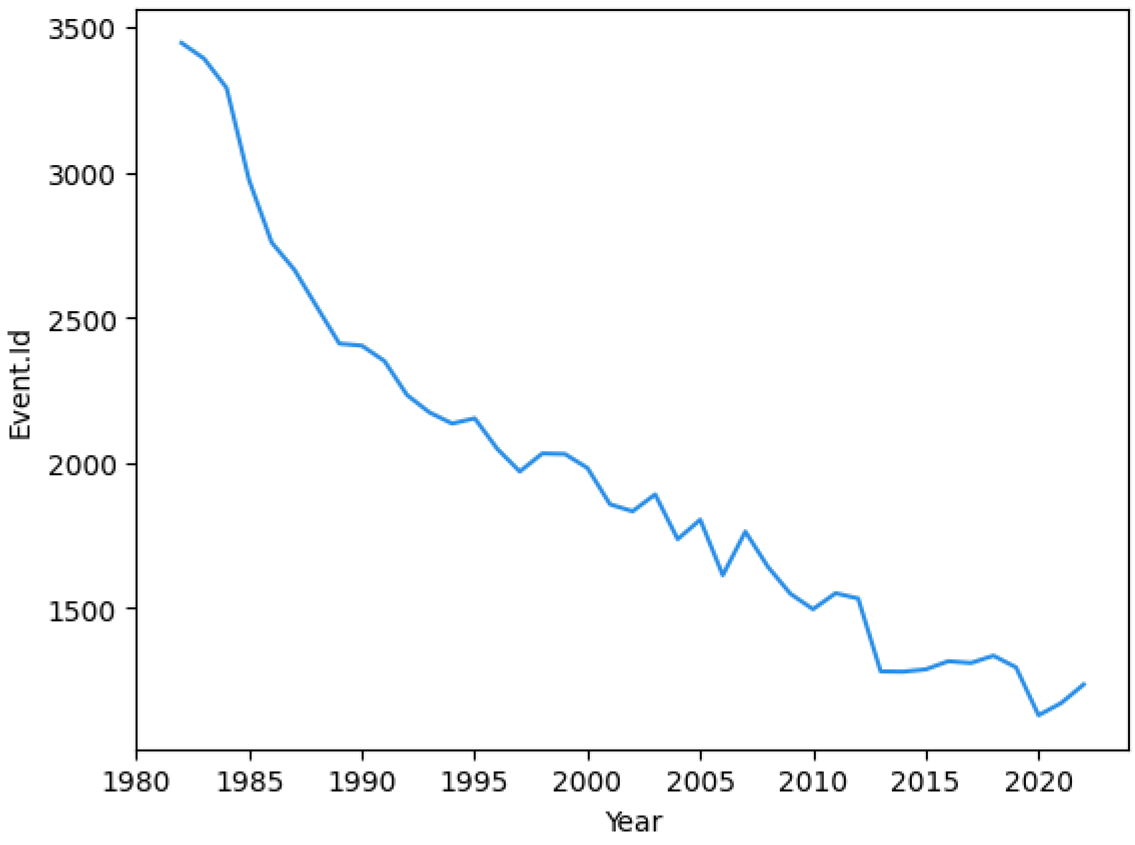
In [ ]:
```
# Number of accidents per year
accidents_per_year = df.groupby(['Year'], as_index = False)['Event.Id'].count()
accidents_per_year
```

Out[ ]:

	Year	Event.Id
0	1982	3445
1	1983	3391
2	1984	3290
3	1985	2972
4	1986	2758
5	1987	2665
6	1988	2537
7	1989	2410
8	1990	2403
9	1991	2350
10	1992	2233
11	1993	2173
12	1994	2135
13	1995	2153
14	1996	2048
15	1997	1970
16	1998	2032
17	1999	2030
18	2000	1982
19	2001	1857
20	2002	1833
21	2003	1891
22	2004	1737
23	2005	1804
24	2006	1613
25	2007	1763
26	2008	1642
27	2009	1549
28	2010	1496
29	2011	1551

	Year	Event.Id
30	2012	1533
31	2013	1282
32	2014	1281
33	2015	1289
34	2016	1317
35	2017	1311
36	2018	1336
37	2019	1296
38	2020	1131
39	2021	1173
40	2022	1237

In [ ]:
```
plot = sns.lineplot(x = 'Year', y = 'Event.Id', data = accidents_per_year, color =
```



Number of fatal accidents per year

In [ ]:
```
# Number of fatal accidents per year
fatal_accidents_per_year = df[df['Injury.Severity'] == 'Fatal'].groupby(['Year'], a
fatal_accidents_per_year
```

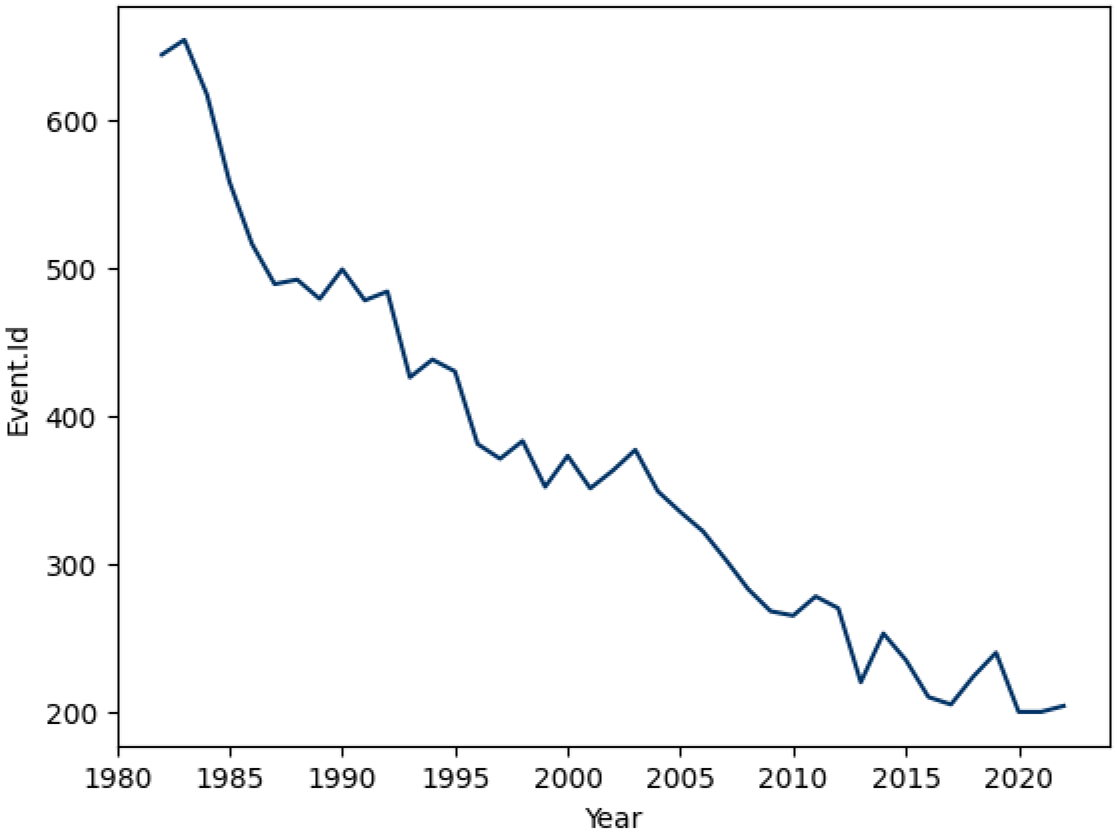
Out[ ]:

	Year	Event.Id
0	1982	644
1	1983	654
2	1984	617
3	1985	558
4	1986	516
5	1987	489
6	1988	492
7	1989	479
8	1990	499
9	1991	478
10	1992	484
11	1993	426
12	1994	438
13	1995	430
14	1996	381
15	1997	371
16	1998	383
17	1999	352
18	2000	373
19	2001	351
20	2002	363
21	2003	377
22	2004	349
23	2005	335
24	2006	322
25	2007	303
26	2008	283
27	2009	268
28	2010	265
29	2011	278

	Year	Event.Id
30	2012	270
31	2013	220
32	2014	253
33	2015	235
34	2016	210
35	2017	205
36	2018	224
37	2019	240
38	2020	200
39	2021	200
40	2022	204

In [ ]:
```
sns.lineplot(x = 'Year', y = 'Event.Id', data = fatal_accidents_per_year, color = '
```

Out[ ]:
```
<Axes: xlabel='Year', ylabel='Event.Id'>
```



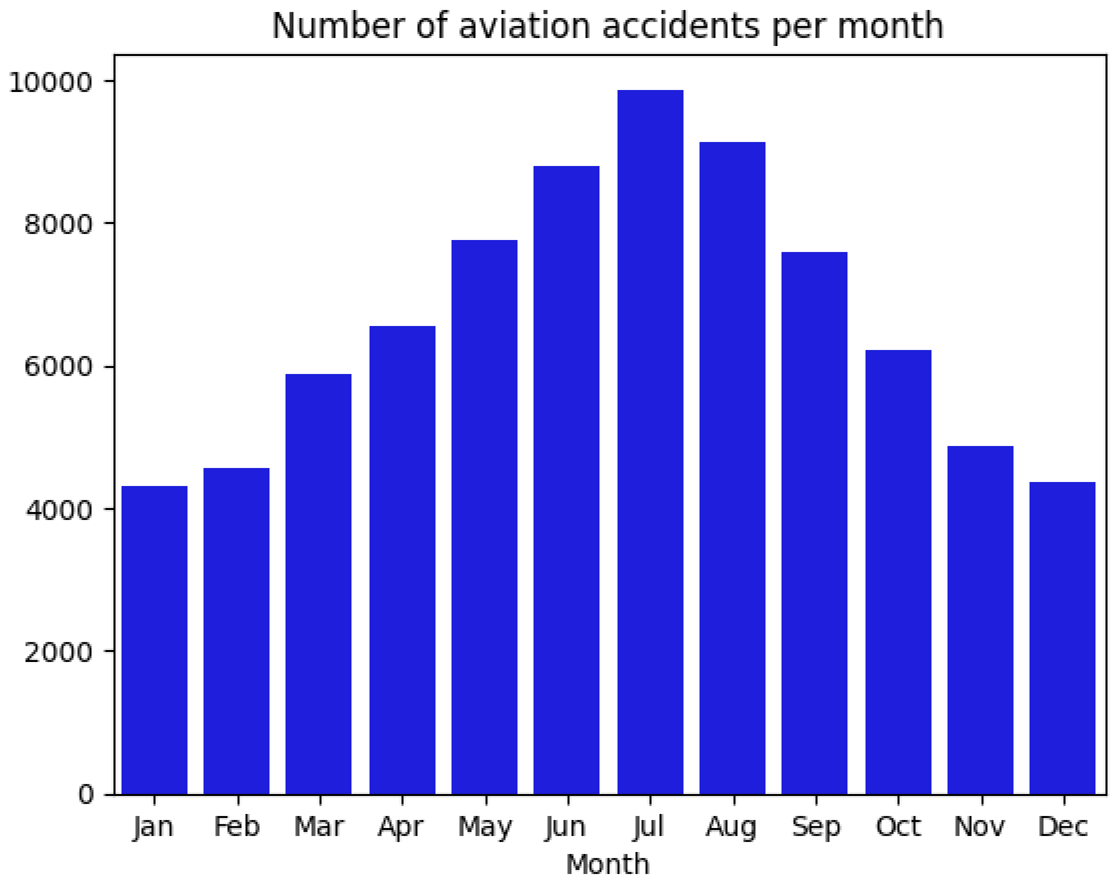
In [ ]:
```
# Calculate average fatality rate
averagefatal = len(injury_data[injury_data['Injury.Severity'] == 'Fatal'].index) /
print("Average fatality rate: " + str(round(averagefatal * 100, 2)) + '%')
```

```
Average fatality rate: 18.8%
```

Months with the most accidents

In [ ]:
```python
# Months with the most accidents
plot = sns.countplot(x = 'Month.Abbr', color = 'b', data = df)
plot.set(xlabel = 'Month', ylabel = None, title = 'Number of aviation accidents per
```

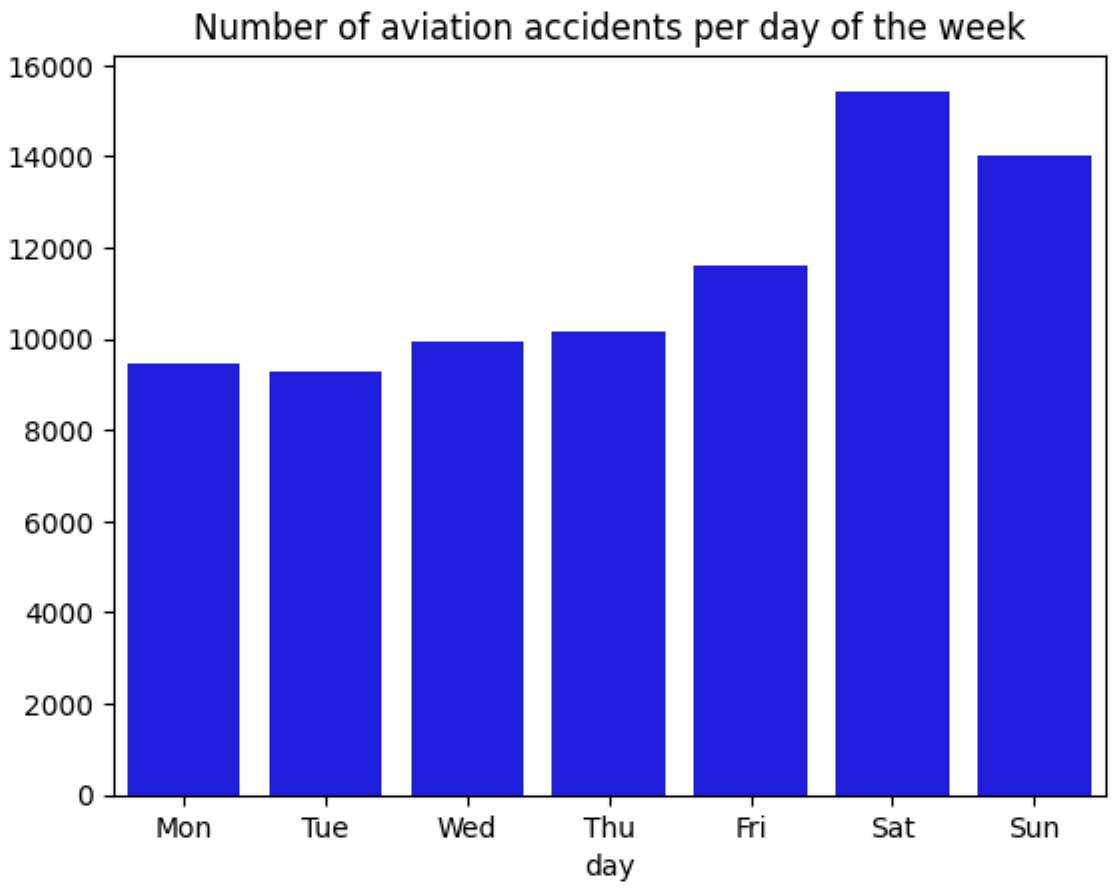
Out[ ]:
```
[Text(0.5, 0, 'Month'),
 Text(0, 0.5, ''),
 Text(0.5, 1.0, 'Number of aviation accidents per month')]
```



Days with the most accidents

In [ ]:
```python
# Days with the most accidents
plot = sns.countplot(x = 'Day.Name.Abbr', order = ['Mon','Tue','Wed','Thu','Fri','S
plot.set(xlabel = 'day', ylabel = None, title = 'Number of aviation accidents per d
```

Out[ ]:
```
[Text(0.5, 0, 'day'),
 Text(0, 0.5, ''),
 Text(0.5, 1.0, 'Number of aviation accidents per day of the week')]
```



Number of fatal accidents per year agaist the model.

With accdents per year against the model will help us know which model caused more accidents at a particular year.

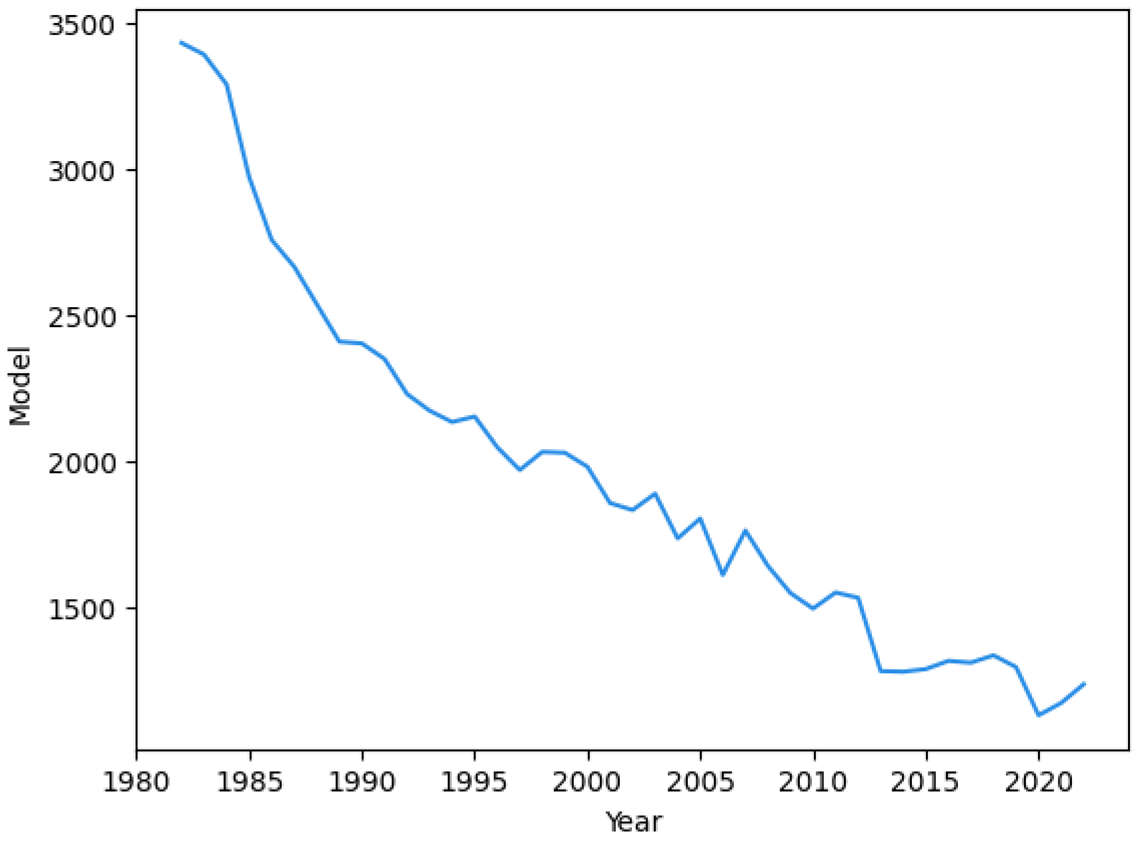
```
In [ ]: #Number of fatal accidents per year agaist the model
        accidents_per_years = df.groupby(['Year'], as_index = False)['Model'].count()
        accidents_per_years
```

Out[ ]:

	Year	Model
0	1982	3431
1	1983	3391
2	1984	3289
3	1985	2972
4	1986	2757
5	1987	2665
6	1988	2537
7	1989	2409
8	1990	2403
9	1991	2350
10	1992	2230
11	1993	2173
12	1994	2134
13	1995	2153
14	1996	2048
15	1997	1970
16	1998	2032
17	1999	2029
18	2000	1981
19	2001	1857
20	2002	1833
21	2003	1889
22	2004	1736
23	2005	1804
24	2006	1611
25	2007	1763
26	2008	1642
27	2009	1549
28	2010	1496
29	2011	1551

	Year	Model
30	2012	1533
31	2013	1282
32	2014	1280
33	2015	1289
34	2016	1317
35	2017	1311
36	2018	1336
37	2019	1296
38	2020	1131
39	2021	1173
40	2022	1237

```
In [ ]: plot = sns.lineplot(x = 'Year', y = 'Model', data = accidents_per_years, color = '#
```



```
In [ ]: df
```

Out[ ]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location
7	20020909X01562	Accident	SEA82DA022	1982-01-01	PULLMAN, WA
8	20020909X01561	Accident	NYC82DA015	1982-01-01	EAST HANOVER, NJ
9	20020909X01560	Accident	MIA82DA029	1982-01-01	JACKSONVILLE, FL
10	20020909X01559	Accident	FTW82DA034	1982-01-01	HOBBS, NM
11	20020909X01558	Accident	ATL82DKJ10	1982-01-01	TUSKEGEE, AL
...	...	...	...	...	...
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA

79899 rows × 31 columns

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