Data Insights & Analytics Preparation Task

The NTSB Aviation Accident Database contains information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters.

Exploration

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('Aviation_Data.csv', encoding='cp437')
df.shape
```

```
(87507, 31)
```

```
df.head()
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.of.flight	Air.carrier	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries
0	20001218X45444	Accident	SEA087LA094	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	2.0	0.0	0.0
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	4.0	0.0	0.0
2	20061025X01555	Accident	NYC07LA044	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	NaN	...	Personal	NaN	3.0	0.0	0.0
3	20061025X01462	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	2.0	0.0	0.0
4	20061025X01464	Accident	CHI79FA064	1979-09-17	Canton, OH	United States	NaN	NaN	NaN	NaN	...	Personal	NaN	1.0	2.0	0.0

5 rows × 31 columns

```
df.info()
```

```
<class: pandas.core.frame.DataFrame>
RangeIndex: 87507 entries, 0 to 87506.
Data columns (total 31 columns):
# Column                  Non-Null Count  Dtype
---  ------                --------------  -----
0  Event.Id                87507 non-null  object
1  Investigation.Type      87507 non-null  object
2  Accident.Number         87507 non-null  object
3  Event.Date              87507 non-null  object
4  Location                87507 non-null  object
5  Country                 87503 non-null  object
6  Latitude                34067 non-null  object
7  Longitude               34064 non-null  object
8  Airport.Code            50132 non-null  object
9  Airport.Name            52704 non-null  object
10 Injury.Severity         87021 non-null  object
11 Aircraft.damage         85772 non-null  object
12 Aircraft.Category       32287 non-null  object
13 Registration.Number     87067 non-null  object
14 Make                    87558 non-null  object
15 Model                   87500 non-null  object
16 Amateur.Built           87546 non-null  object
17 Number.of.Engines       82991 non-null  float64
18 Engine.Type             77488 non-null  object
19 FAR.Description         32049 non-null  object
20 Schedule                12822 non-null  object
21 Purpose.of.flight       82697 non-null  object
22 Air.carrier             18732 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         82928 non-null  float64
27 Weather.Condition       84397 non-null  object
28 Broad.phase.of.flight   61724 non-null  object
29 Report.Status           82508 non-null  object
30 Publication.Date        73659 non-null  object
dtypes: float64(5), object(26)
memory usage: 20.5 MB
```

```
df.isnull.sum()
```

```
# Percentage of null values by column
percentage_df = df.isnull().sum() / len(df) * 100

print("Percentage of null values:")
print(percentage_df.sort_values(ascending=False), sep='\n')

# Optional: Filter and show only columns with more than 10% missing values
print("\nColumns with more than 10% missing values:")
percentage_df[percentage_df > 10].sort_values(ascending=False), inplace=True)
```

```
Percentage of null values by column:
Variable                Percentage
0   Schedule            85.07270
1   Accident            55.78208
2   FAR.Description     63.19353
3   Aircraft.Category   63.09000
4   Longitude           61.05000
5   Latitude            61.07990
6   Airport.Code        42.69143
7   Airport.Name        39.80000
8   Broad.phase.of.flight  30.20000
9   Publication.Date    15.84000
10  Publication.Date    15.84000
dtype: float64
```

5 rows × 31 columns

```
df.describe()
```

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	82991.000000	77488.000000	76379.000000	76956.000000	82928.000000
mean	1.146585	0.660753	0.279881	0.357061	5.325440
std	0.446785	5.485289	1.544084	2.235625	27.913034
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	1.000000	0.000000	0.000000	2.000000
max	8.000000	349.000000	161.000000	380.000000	699.000000

Finally, checking the differents options each column can be

```
num_values = 2
values = 0
for i in range(df.columns):
    unique_values = df.iloc[:, i].unique()
    print(f"{df.columns[i]}: {len(unique_values)} unique values")
    if len(unique_values) <= 10:
        print(f"Values: {unique_values}")
    else:
        print(f"Sample values: {unique_values[:5]}...")
    print()
```

Variable	Values
Investigation.Type	[Accident, Incident]
Accident.Number	[SEA087LA094, NYC07LA044, CHI79FA064, CHI79FA064, LAX96LA321]...
Event.Date	[1948-10-24, 1962-07-19, 1974-08-30, 1977-06-19, 1979-09-17]...
Location	[MOOSE CREEK, ID, BRIDGEPORT, CA, Saltville, VA, EUREKA, CA, Canton, OH]...
Country	[United States, Mexico, Brazil, Canada, United Kingdom]...
Latitude	[NaN, 36.922223, 47.975556, 41.422222, 38.054722]...
Longitude	[NaN, -81.878056, -122.136111, -123.372222, -82.821667]...
Airport.Code	[NaN, MC77, BRI, EUREKA, NaN]...
Airport.Name	[NaN, Merrill Creek, Bridgeport, Eureka, NaN]...
Injury.Severity	[Fatal(2), Fatal(4), Fatal(3), Fatal(2), Fatal(1)]...
Aircraft.damage	[Destroyed, Substantial, Minor, NaN]...
Aircraft.Category	[NaN, Airplane, Helicopter, Glider, Balloon, Gyrocopter, Weight-Shift]...
Registration.Number	[NC9324E, NC1066N, NC5291N, NC5290N, NC2427N]...
Make	[Stinson, Piper, Cessna, Rockwell, Mooney]...
Model	[108-3, PA-18 Super Cub, 172, 172RG, 172N, 172F, 172, 172, 172, 172]...
Amateur.Built	[No, No, No, No, No]...
Number.of.Engines	[1.0, 1.0, 1.0, 1.0, 1.0]...
Engine.Type	[Reciprocating, Reciprocating, Reciprocating, Reciprocating, Reciprocating]...
FAR.Description	[Part 91: General Aviation, Public, CFI, NaN, Part 137: Agricultural]...
Schedule	[NaN, NaN, NaN, NaN, NaN]...
Purpose.of.flight	[Personal, Personal, Personal, Personal, Personal]...
Air.carrier	[NaN, NaN, NaN, NaN, NaN]...
Total.Fatal.Injuries	[2.0, 4.0, 3.0, 2.0, 1.0]...
Total.Serious.Injuries	[0.0, 0.0, 0.0, 0.0, 2.0]...
Total.Minor.Injuries	[0.0, 0.0, 0.0, 0.0, 0.0]...
Total.Uninjured	[0.0, 1.0, 2.0, 1.0, 2.0]...
Weather.Condition	[UNK, UNK, UNK, UNK, UNK]...
Broad.phase.of.flight	[Cruise, Unknown, Cruise, Cruise, Cruise]...
Report.Status	[Probable Cause, Probable Cause, Probable Cause, Probable Cause, Probable Cause]...
Publication.Date	[NaN, 04/01/1980, 08/12/1980, 01/01/1980, 01/01/1980]...

Cleaning

Removing duplicates

```
sum(df['Event.Id'].duplicated())
```

```
0L
```

```
df.duplicates = pd.concat(for _ in df.groupby('Event.Id') if len(g) > 1)
```

It looks like the duplicates are due to updates and not to order. We can see in the Accident.Number column 4, 8 or C depending on the amount of updates done

```
df.duplicates.sort_values(by='Event.Id')
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.of.flight	Air.carrier	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries
40982	20001205X00560	Accident	LAX96LA056	1995-01-05	WASCO, CA	United States	NaN	NaN	SAN	MONTEREY PENINSULA AIRPORT	...	Personal	NaN	0.0	0.0	0.0
40984	20001205X00560	Accident	LAX96LA056	1995-01-05	WASCO, CA	United States	NaN	NaN	SAN	MONTEREY PENINSULA AIRPORT	...	Personal	NaN	0.0	0.0	0.0
40985	20001205X00560	Accident	CHI96SA056	1995-01-05	Sun Lake, IN	United States	NaN	NaN	NaN	NaN	...	Unknown	NaN	0.0	0.0	0.0
40986	20001205X00560	Accident	CHI96SA056	1995-01-05	Sun Lake, IN	United States	NaN	NaN	NaN	NaN	...	Unknown	NaN	0.0	0.0	0.0
40978	20001205X00560	Accident	CHI96SA056	1995-01-05	DEVONS, MN	United States	NaN	NaN	MSP	MINNEAPOLIS-ST PAUL INTL/WOLD-CHAMBERLAIN	...	Unknown	NaN	0.0	0.0	0.0

5 rows × 31 columns

```
df.sort_values(by='Event.Id', keepdim=True)
df.drop_duplicates(subset='Event.Id', keep='last', inplace=True)
```

```
df[df['Event.Id'].isin(['20001205X00560'])]
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Name	...	Purpose.of.flight	Air.carrier	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries
40978	20001205X00560	Accident	LAX96LA056	1995-01-05	WASCO, CA	United States	NaN	NaN	SAN	MONTEREY PENINSULA AIRPORT	...	Personal	NaN	0.0	0.0	0.0

5 rows × 31 columns

Correcting values

Some values resist the same but are written differently. Is this, CANCELLED, capital letters and transforming them. Also for some columns with few I will try to impute else I will fill with No

```
df['Weather.Condition'] = df['Weather.Condition'].replace('UNK', 'Unknown')
df['Weather.Condition'] = df['Weather.Condition'].replace('Unk', 'Unknown')
df['Registration.Number'] = df['Registration.Number'].replace(np.nan, 'Unknown')
df['Registration.Number'] = df['Registration.Number'].replace('Unknown', 'Unknown')

df['Amateur.Built'] = df['Amateur.Built'].replace('N', 'No')
df['Amateur.Built'] = df['Amateur.Built'].replace('Y', 'Yes')
```

Lowercase FIXED-Wing, NONE Helos, PINSTRIPE Wings...

```
df['Make'] = df['Make'].str.lower()
df['Registration.Number'] = df['Registration.Number'].str.lower()
df['Injury.Severe'] = df['Injury.Severe'].str.lower()
```

Na values

```
df.missing_values[df.sum()]
```

Variable	Percentage
Schedule	85.07270
Accident	55.78208
FAR.Description	63.19353
Aircraft.Category	63.75000
Longitude	61.05000
Latitude	61.07990
Airport.Code	42.69143
Airport.Name	39.80000
Broad.phase.of.flight	30.20000
Publication.Date	15.84000

I will remove columns I will not use because they have to many null values. I want to keep Longitude and Latitude. Perhaps it can be used for a great visualization.

```
df.drop(columns = ['Schedule', 'Accident', 'FAR.Description', 'Aircraft.Category', 'Airport.Code', 'Airport.Name'], axis = 1, inplace = True)
```

We can fill Injury Severity with information from Total.Fatal.Injuries and vice-versa

```
df[df['Total.Fatal.Injuries'].isnull()].groupby('Injury.Severity')['Total.Fatal.Injuries'].head()
```

Injury.Severity	Total.Fatal.Injuries
#0!0!	Non-Fatal
#0!0!	Non-Fatal
#0!0!	Non-Fatal
#0!0!	Non-Fatal
#0!0!	Non-Fatal

```
index_nan = df[df['Injury.Severity'].isna()].index
index_non_fatal = df[df['Injury.Severity'].str.lower() == 'non-fatal'].iloc[:df['Injury.Severity'].str.lower().isna().sum()].index
df.loc[index_nan, 'Injury.Severity'] = 'Non-Fatal'
df.loc[index_nan, 'Total.Fatal.Injuries'] = 0.0
```

```
index_nan = df[df['Injury.Severity'].isna()].index
index_accident = df[df['Injury.Severity'].str.lower() == 'fatal(1)'].iloc[:df['Injury.Severity'].str.lower().isna().sum()].index
df.loc[index_nan, 'Injury.Severity'] = 'Fatal(1)'
df.loc[index_nan, 'Total.Fatal.Injuries'] = 1.0
```

```
index_nan_fatal = df[df['Total.Fatal.Injuries'].isna()]
index_fatal = df[df['Injury.Severity'].str.lower() == 'fatal(1)'].iloc[:df['Injury.Severity'].str.lower().isna().sum()].index
df.loc[index_nan_fatal, 'Injury.Severity'] = 'Non-Fatal'
df.loc[index_nan_fatal, 'Injury.Severity'] = 'Fatal(1)'
```

Now we can also transform all the df['Injury.Severity'] in df['Injury.Severity'] and vice versa

```
df['Injury.Severity'] = df['Injury.Severity'].str.split('(').str[0]
```

Fill with Unknown

```
df['Location'].fillna('Unknown', inplace=True)
df['Country'].fillna('Unknown', inplace=True)
df['Weather.Condition'].fillna('Unknown', inplace=True)
df['Injury.Severity'].fillna('Unknown', inplace=True)
df['Location'].fillna('Unknown', inplace=True)
df['Aircraft.Damage'].fillna('Unknown', inplace=True)
df['Make'].fillna('Unknown', inplace=True)
df['Model'].fillna('Unknown', inplace=True)
df['Amateur.Built'].fillna('Unknown', inplace=True)
df['Purpose.of.flight'].fillna('Unknown', inplace=True)
df['Broad.phase.of.flight'].fillna('Unknown', inplace=True)
df['Broad.phase.of.flight'].fillna('Unknown', inplace=True)
df['Report.Status'].fillna('Unknown', inplace=True)
df['Publication.Date'].fillna('Unknown', inplace=True)
df['Total.Fatal.Injuries'].fillna(0.0, inplace=True)
df['Total.Fatal.Injuries'].fillna(0.0, inplace=True)
```

For the Number.of.Engines I will take the most common number of engines done by the Make (company). If no entries I put 1 (most common value for Number.of.Engines)

```
index_make = df[df['Make'].isna()].index
most_common = df.groupby('Make')['Number.of.Engines'].mode()
index = df[df[df.index.isin(index_make)]['Number.of.Engines'].isna()].index
```

```
for i in tq.update:
    try:
        df.loc[i, 'Number.of.Engines'] = df['Number.of.Engines'][df['Make'].isin(df.loc[i]['Make'])].value_counts().index[0]
    except:
        df.loc[i, 'Number.of.Engines'] = 1.0 # Most common value
```

Add columns for date

```
df['Event.Date'] = pd.to_datetime(df['Event.Date'])
df['Year'] = df['Event.Date'].dt.year
df['Month'] = df['Event.Date'].dt.month
df['Day'] = df['Event.Date'].dt.day
```

Analysis

When do accidents happen?

I sort our accidents before 2000 and after 2001 because there is no little data

```
df_time = df[df['Year'] < 2000].df[df['Year'] > 1980]
accidents_by_year = df.groupby('Year').value_counts().sort_index()
accidents_fatal = df[df['Total.Fatal.Injuries'] > 0].groupby('Year')['Total.Fatal.Injuries'].sum()
accidents_fatal = df[df['Total.Fatal.Injuries'] > 0].groupby('Year')['Total.Fatal.Injuries'].count()
accidents_fatal = df.groupby('Year')['Total.Fatal.Injuries'].sum()
```

```
plt.plot(accidents_by_year, label='Accidents', color='blue')
plt.plot(accidents_non_fatal, label='Non-Fatal Accidents', color='green')
plt.plot(accidents_fatal, label='Fatal Accidents', color='red')
plt.legend(labels=['Total', 'Non-Fatal', 'Fatal Accidents'])
```

```
df_time['Month'] = df_time['Month'].int.clip(lower=1, upper=12)
sns.countplot(x = 'Month', hue = 'Injury.Severe', data = df_time)
```

```
df_time['Day'] = df_time['Month'].int.clip(lower=1, upper=31)
sns.countplot(x = 'Day', hue = 'Injury.Severe', data = df_time, order=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])
```

Number of accidents reduced over time

Most accidents during summer (May-Sep)

Most accidents during weekends

At what phase do accidents happen?

```
df.phase = df[df'Broad.phase.of.flight'] != 'Unknown']
df.phase.shape
df.phase[phase = 'Broad.phase.of.flight', color = 'red', color_palette='Pastel', H=10], data = df.phase, order=df.phase['Broad.phase.of.flight'].value_counts().index)
```

```
<matplotlib.axes.AxesSubplot at 0x7f>
```


Where do accidents happen?

60% of the recorded accidents are in USA. So I am plotting only them.

```
import plotly.express as px
```

```
states = df[df['Country'] == 'United States'].groupby('Location').sort_values(ascending=False)
states = states.reset_index()
states = states[states > 15]

fig = px.choropleth(df,
                    locations=states.index,
                    locationmode="USA-states",
                    color=states,
                    scope="usa",
                    color_continuous_scale=px.colors.sequential.Plasma,
                    labels={'color':'Amount'}
                    )

fig.show()
```

Correlations

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), annot = True)
```

```
<matplotlib.c
```


Conclusion

Accidents decreased during the years by 2%

Number of Fatal Injuries by year seems unstable

Most of the Accidents happened between May-Sep (Summer Time)

Most of the Accidents happened on Saturday or Sunday (Weekend)

Most of the Accidents where while Landing or Takeoff

60% of the dataset is from accidents in USA

The state with the most accidents is California

There seems to be a correlation between Number of Engines and Total Uninjured