DATA EXPLORATION AND PREPARATION

In [ ]:
```
#Importing Packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn  as sns
```

In [ ]:
```
#Loading the Data
#Handling the encoding
df=pd.read_csv("AviationData.csv",encoding='latin1',low_memory=False)
```

In [ ]:
```
# Display all the columns
pd.set_option('display.max_columns',None)
```

In [ ]:
```
# Display the first rows
df.head()
```

In [ ]:
```
# Check the number of rows and columns in the data
df.shape
```

In [ ]:
```
# Summary of DataFrame
df.info()
```

In [ ]:
```
#columns names
df.columns.tolist()
```

In [ ]:
```
# Statistical Summary
df.describe()
```

DATA CLEANING

Handling Missing Values

In [19]:
```
#checking for the missing values in each column
df.isnull().sum()
```

Out[19]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   13
Country                   115
Latitude                44181
Longitude               44191
Airport.Code            30370
Airport.Name            28091
Injury.Severity            19
Aircraft.damage             0
Aircraft.Category       48240
Registration.Number       443
Make                        0
Model                       0
Amateur.Built              29
Number.of.Engines        2192
Engine.Type              3167
FAR.Description         48192
Schedule                70345
Purpose.of.flight           0
Air.carrier             64028
Total.Fatal.Injuries        0
Total.Serious.Injuries      0
Total.Minor.Injuries        0
Total.Uninjured             0
Weather.Condition         792
Broad.phase.of.flight   20649
Report.Status            2777
Publication.Date        11197
Year                        0
Total.Injuries              0
Weather.condition           0
dtype: int64
```

In [20]:
```
#1. clean 'EventDate'  by droppinfg missing value
df = df.dropna(subset=['Event.Date'])
#convert to datetime format
df['Event.Date'] = pd.to_datetime(df['Event.Date'], errors='coerce')
#Create a new 'Year' column
df['Year'] = df['Event.Date'].dt.year
```

In [21]:
```
#2. clean the 'injuries column'
# step 1. Fill the missing values with zero  and convert to an int
df['Total.Fatal.Injuries']=df['Total.Fatal.Injuries'].fillna(0).astype(int)
df['Total.Serious.Injuries']=df['Total.Serious.Injuries'].fillna(0).astype(int)
df['Total.Minor.Injuries']=df['Total.Minor.Injuries'].fillna(0).astype(int)
#step 2. Total Injuries
df['Total.Injuries']=df[['Total.Fatal.Injuries','Total.Serious.Injuries','Total.Min
```

In [22]:
```
# 3. clean the the 'total uninjured'
df['Total.Uninjured']=df['Total.Uninjured'].fillna(0).astype(int)
```

In [23]:
```
#4. clean the 'model' & 'make
```

```
# convert make and model to uppercase

df['Make'] = df['Make'].str.strip().str.upper()
df['Model'] = df['Model'].str.strip().str.upper()

# drop the missing values

df = df.dropna(subset=['Make', 'Model'])
```

In [24]:
```
#5.cleaning the ,purpose of flight' column

df['Purpose.of.flight']=df['Purpose.of.flight'].astype(str).str.strip().str.title()
df['Purpose.of.flight']=df['Purpose.of.flight'].replace(['','nan','Nan','NaN'], 'Un
df=df[df['Purpose.of.flight'] != 'Unknown']
```

In [25]:
```
#6. cleaning the 'Aircraft Damage column
df['Aircraft.damage']=df['Aircraft.damage'].astype(str).str.strip().str.title()
df['Aircraft.damage']=df['Aircraft.damage'].replace(['','nan','Nan','NAN'], 'Unknow
df=df[df['Aircraft.damage'] != 'Unknown']
```

In [26]:
```
#7. Clean 'Weather Condition'

df['Weather.condition']=df['Weather.Condition'].astype(str).str.strip().str.title()
df['Weather.Condition']=df['Weather.Condition'].replace(['','UNK','Unk','Nan'], 'Un

#filter out the 'unknown' from the column

df=df[df['Weather.Condition'] != 'Unknown']
```

Handling Duplicates

In [27]:
```
 # check for duplicates

duplicates=df[df.duplicated(subset=['Event.Id','Accident.Number'], keep=False)]

# drop the duplicate and keep the first row

df=df.drop_duplicates(subset=['Event.Id','Accident.Number'] , keep= 'first')
```

In [29]:
```
# save the clean data

df.to_csv('Aviation_clean_Data.csv',index=False)
df.to_csv(r'C:\Users\YourName\Desktop\Aviation_clean_Data.csv', index=False)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[29], line 4
      1 # save the clean data
      3 df.to_csv('Aviation_clean_Data.csv',index=False)
----> 4 df.to_csv(r'C:\Users\YourName\Desktop\Aviation_clean_Data.csv', index=False)

File ~\GRAPHISOFT\Publisher log files\Lib\site-packages\pandas\core\generic.py:3902,
in NDFrame.to_csv(self, path_or_buf, sep, na_rep, float_format, columns, header, ind
ex, index_label, mode, encoding, compression, quoting, quotechar, lineterminator, ch
unksize, date_format, doublequote, escapechar, decimal, errors, storage_options)
   3891 df = self if isinstance(self, ABCDataFrame) else self.to_frame()
   3893 formatter = DataFrameFormatter(
   3894     frame=df,
   3895     header=header,
   (...)
   3899     decimal=decimal,
   3900 )
-> 3902 return DataFrameRenderer(formatter).to_csv(
   3903     path_or_buf,
   3904     lineterminator=lineterminator,
   3905     sep=sep,
   3906     encoding=encoding,
   3907     errors=errors,
   3908     compression=compression,
   3909     quoting=quoting,
   3910     columns=columns,
   3911     index_label=index_label,
   3912     mode=mode,
   3913     chunksize=chunksize,
   3914     quotechar=quotechar,
   3915     date_format=date_format,
   3916     doublequote=doublequote,
   3917     escapechar=escapechar,
   3918     storage_options=storage_options,
   3919 )

File ~\GRAPHISOFT\Publisher log files\Lib\site-packages\pandas\io\formats\format.p
y:1152, in DataFrameRenderer.to_csv(self, path_or_buf, encoding, sep, columns, index
_label, mode, compression, quoting, quotechar, lineterminator, chunksize, date_forma
t, doublequote, escapechar, errors, storage_options)
   1131     created_buffer = False
   1133 csv_formatter = CSVFormatter(
   1134     path_or_buf=path_or_buf,
   1135     lineterminator=lineterminator,
   (...)
   1150     formatter=self.fmt,
   1151 )
-> 1152 csv_formatter.save()
   1154 if created_buffer:
   1155     assert isinstance(path_or_buf, StringIO)

File ~\GRAPHISOFT\Publisher log files\Lib\site-packages\pandas\io\formats\csvs.py:24
7, in CSVFormatter.save(self)
    243 """
    244 Create the writer & save.
```

```
    245 """
    246 # apply compression and byte/text conversion
--> 247 with get_handle(
    248     self.filepath_or_buffer,
    249     self.mode,
    250     encoding=self.encoding,
    251     errors=self.errors,
    252     compression=self.compression,
    253     storage_options=self.storage_options,
    254 ) as handles:
    255     # Note: self.encoding is irrelevant here
    256     self.writer = csvlib.writer(
    257         handles.handle,
    258         lineterminator=self.lineterminator,
   (...)
    263         quotechar=self.quotechar,
    264     )
    266     self._save()

File ~\GRAPHISOFT\Publisher log files\Lib\site-packages\pandas\io\common.py:739, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    737 # Only for write methods
    738 if "r" not in mode and is_path:
--> 739     check_parent_directory(str(handle))
    741 if compression:
    742     if compression != "zstd":
    743         # compression libraries do not like an explicit text-mode

File ~\GRAPHISOFT\Publisher log files\Lib\site-packages\pandas\io\common.py:604, in check_parent_directory(path)
    602 parent = Path(path).parent
    603 if not parent.is_dir():
--> 604     raise OSError(rf"Cannot save file into a non-existent directory: '{parent}'")

OSError: Cannot save file into a non-existent directory: 'C:\Users\YourName\Desktop'
```

In [30]:
```
#check the cleaned data

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

Out[30]:
```
Event.Id                    0
Investigation.Type          0
Accident.Number             0
Event.Date                  0
Location                   13
Country                   115
Latitude                44181
Longitude               44191
Airport.Code            30370
Airport.Name            28091
Injury.Severity            19
Aircraft.damage             0
Aircraft.Category       48240
Registration.Number       443
Make                        0
Model                       0
Amateur.Built              29
Number.of.Engines        2192
Engine.Type              3167
FAR.Description         48192
Schedule                70345
Purpose.of.flight           0
Air.carrier             64028
Total.Fatal.Injuries        0
Total.Serious.Injuries      0
Total.Minor.Injuries        0
Total.Uninjured             0
Weather.Condition         792
Broad.phase.of.flight   20649
Report.Status            2777
Publication.Date        11197
Year                        0
Total.Injuries              0
Weather.condition           0
dtype: int64
```

Data Analysis

1.Plot Trends over Time

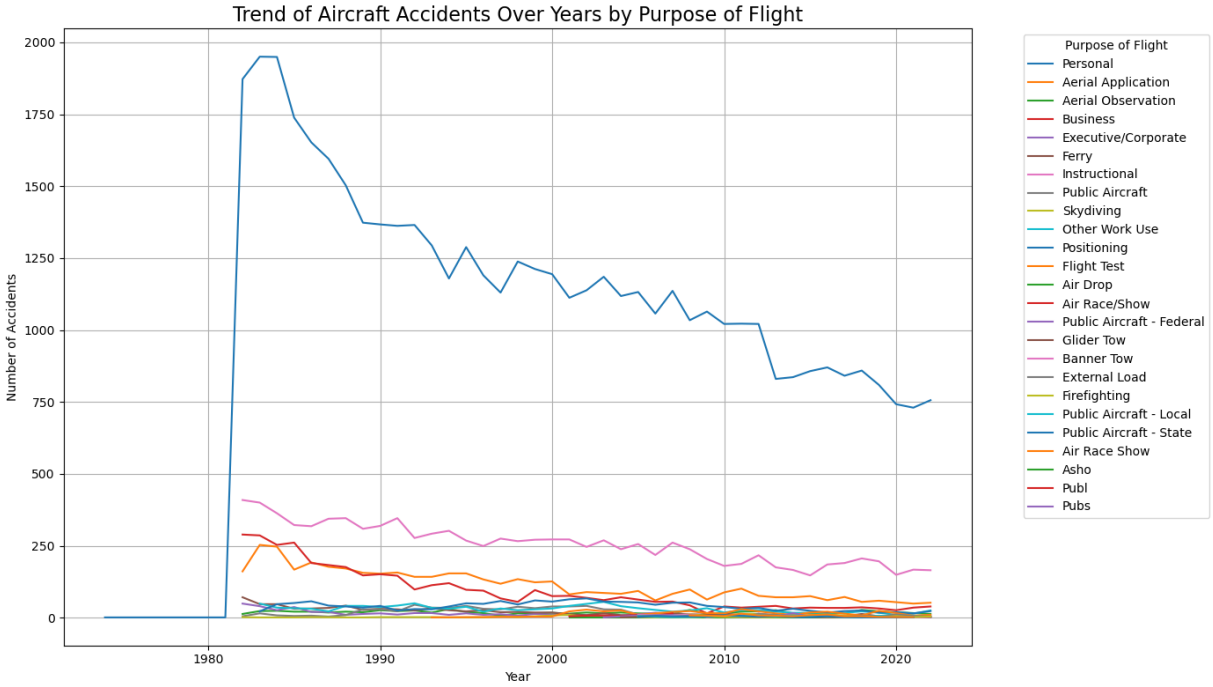
In [31]:
```
#Group by year and purpose,
accident_counts = df.groupby(['Year', 'Purpose.of.flight']).size().reset_index(name

#. Plot using a loop — no pivot table
plt.figure(figsize=(14, 8))

# Get unique purposes
purposes = accident_counts['Purpose.of.flight'].unique()

# Plot each purpose separately
for purpose in purposes:
    yearly_data = accident_counts[accident_counts['Purpose.of.flight'] == purpose]
    plt.plot(yearly_data['Year'], yearly_data['AccidentCount'], label=purpose)
```

```
# 5. Customize plot
plt.title('Trend of Aircraft Accidents Over Years by Purpose of Flight', fontsize=1
plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.legend(title='Purpose of Flight', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```



3.Analyzing the Impact of Weather Conditions on Aircraft Accident Frequency and Severity

In [32]:

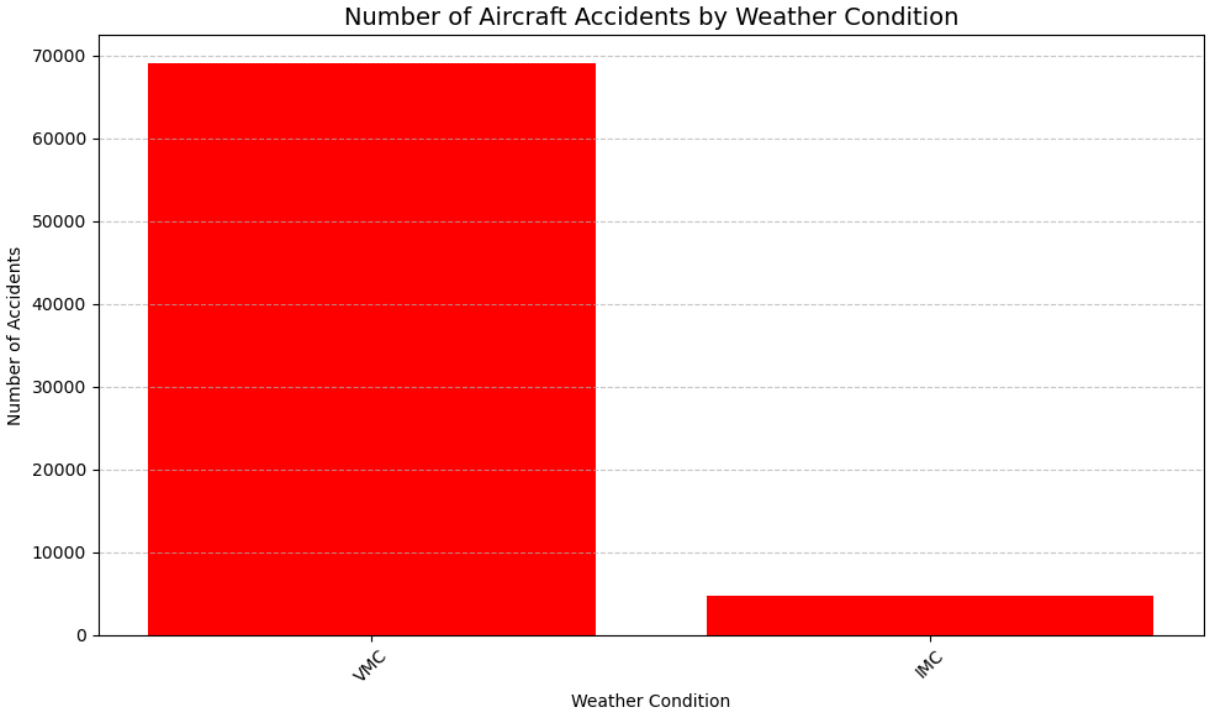
```
# Group by weather condition and count number of accidents
weather_counts =df['Weather.Condition'].value_counts().reset_index()
weather_counts.columns = ['Weather Condition', 'Number of Accidents']

#  Plot bar chart
plt.figure(figsize=(10, 6))
plt.bar(weather_counts['Weather Condition'], weather_counts['Number of Accidents'],

plt.title('Number of Aircraft Accidents by Weather Condition', fontsize=14)
plt.xlabel('Weather Condition')

plt.ylabel('Number of Accidents')

plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



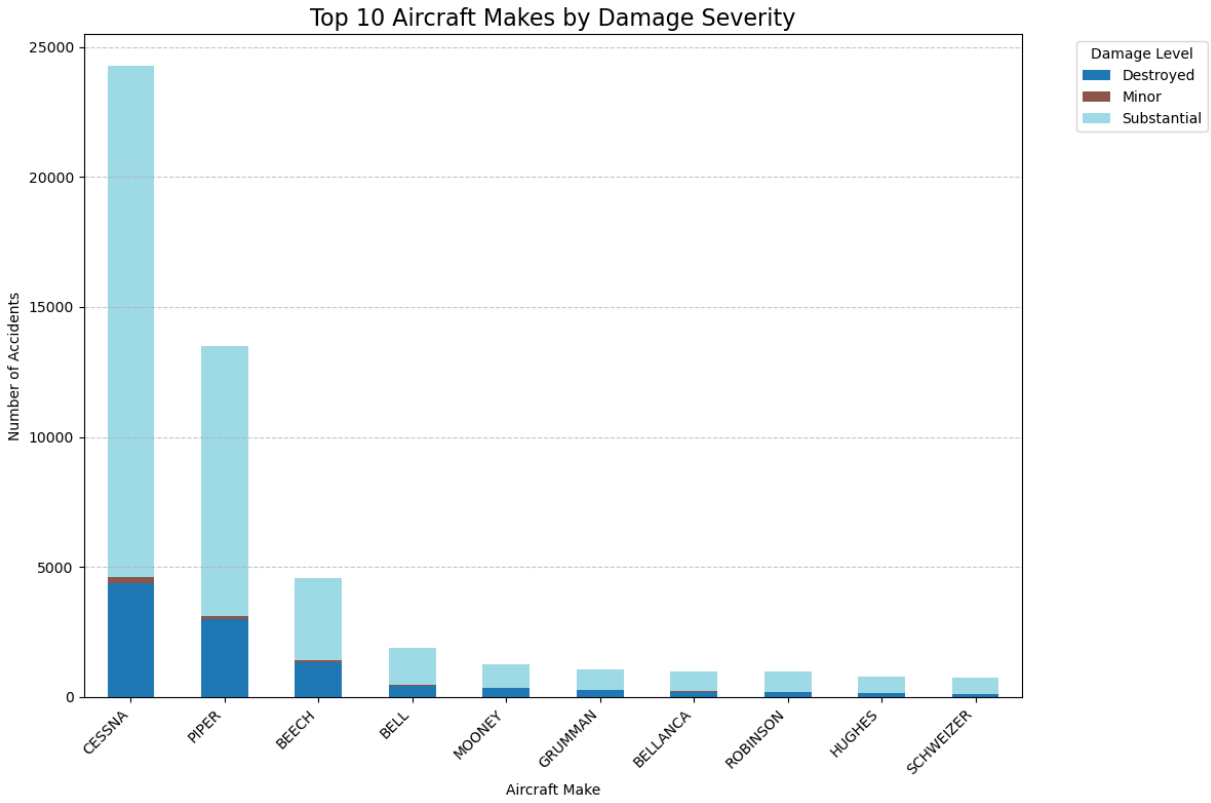
3.Evaluating Damage Outcomes by Aircraft Make in Aviation Accidents

In [33]:
```
damage_by_make = pd.crosstab(df['Make'], df['Aircraft.damage'])

# 2. Select top 10 makes by total accidents
top_makes = damage_by_make.sum(axis=1).nlargest(10).index
damage_top10 = damage_by_make.loc[top_makes]

# 3. Plot stacked bar chart
ax = damage_top10.plot(
    kind='bar',
    stacked=True,
    figsize=(12, 8),
    colormap='tab20'
)

# 4. Customize plot
ax.set_title('Top 10 Aircraft Makes by Damage Severity', fontsize=16)
ax.set_xlabel('Aircraft Make')
ax.set_ylabel('Number of Accidents')
ax.legend(title='Damage Level', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```


Top 10 Aircraft Makes by Damage Severity
Damage Level
Destroyed
Minor
Substantial
Number of Accidents
25000
20000
15000
10000
5000
0
CESSNA
PIPER
BEECH
BELL
MOONEY
GRUMMAN
BELLANCA
ROBINSON
HUGHES
SCHWEIZER
Aircraft Make

In [ ]:

In [ ]:

In [ ]:

In [ ]: