Overview

This dataset consists of aviation accident survey data, which includes information on whether the aircraft was destroyed, the type of injuries sustained by individuals, the location of the accidents along with the year and month, the type of engine involved, weather conditions, the purpose of the flight, and other relevant details.

Business Problem

This analysis aims to identify patterns and trends in aviation accidents to enhance safety measures. By understanding the relationship between weather conditions, engine types, injury severity, and flight purpose, we can develop insights to prevent future accidents. The findings will support data-driven decision-making in aviation safety policies and operational practices. Ultimately, this research seeks to improve overall flight safety and reduce accident-related risks

Data Understanding

In this data analysis, the first step involves importing the necessary libraries and loading the dataset to begin the analysis. Next, I will handle any missing values to ensure the data is complete and reliable. Following this, I will transform the data to enhance its interpretability, such as grouping related categories into broader groups for better clarity. The data will then be explored through various visualizations to uncover relationships and patterns between key variables, such as weather conditions, injury severity, and flight purposes. Finally, I will summarize the findings and provide conclusions that offer insights into aviation safety and accident prevention.

DATA AND LIBRARY IMPORTATION

In [85]:

```
#import libraries
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.cm as cm
```

In [86]:

```
# Loading dataset
df = pd.read_csv('AviationData.csv', encoding='latin-1',low_memory=False)
df.head()
```

Out[86]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.C
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	N
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	N
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	N
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	N
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	N

5 rows × 31 columns

DATA UNDERSTANDING

In [87]:

```
# understanding basic information of the dataset
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 [88]:

```
# creating a copy of my original data so as not to bring any modification
df1 = df.copy()
```

HANDLING MISSING VALUES

In [89]:

```
# dropping columns that have a lot of missing values
df1.drop(columns=['Latitude'], inplace=True)
df1.drop(columns=['Longitude'],inplace=True)
df1.drop(columns=['Airport.Code'],inplace=True)
df1.drop(columns=['Airport.Name'],inplace=True)
df1.drop(columns=['Aircraft.Category'],inplace=True)
df1.drop(columns=['FAR.Description'],inplace=True)
df1.drop(columns=['Schedule'],inplace=True)
df1.drop(columns=['Air.carrier'],inplace=True)
```

In [90]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 23 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   Injury.Severity        87889 non-null  object
 7   Aircraft.damage        85695 non-null  object
 8   Registration.Number    87507 non-null  object
 9   Make                   88826 non-null  object
 10  Model                  88797 non-null  object
 11  Amateur.Built          88787 non-null  object
 12  Number.of.Engines      82805 non-null  float64
 13  Engine.Type            81793 non-null  object
 14  Purpose.of.flight      82697 non-null  object
 15  Total.Fatal.Injuries   77488 non-null  float64
 16  Total.Serious.Injuries 76379 non-null  float64
 17  Total.Minor.Injuries   76956 non-null  float64
 18  Total.Uninjured        82977 non-null  float64
 19  Weather.Condition      84397 non-null  object
 20  Broad.phase.of.flight  61724 non-null  object
 21  Report.Status          82505 non-null  object
 22  Publication.Date       75118 non-null  object
dtypes: float64(5), object(18)
memory usage: 15.6+ MB
```

In [91]:

```
# checking the number of missing value
df1.isnull().sum()
```

Out[91]:

```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Injury.Severity            1000
Aircraft.damage            3194
Registration.Number        1382
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7096
Purpose.of.flight          6192
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
```

DATA CLEANING

In [92]:

```
# further columns removal
columns_name = ['Publication.Date','Broad.phase.of.flight','Engine.Type']
df1.drop(columns=columns_name,inplace=True)
```

In [93]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 20 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   Injury.Severity         87889 non-null  object 
 7   Aircraft.damage         85695 non-null  object 
 8   Registration.Number     87507 non-null  object 
 9   Make                    88826 non-null  object 
 10  Model                   88797 non-null  object 
 11  Amateur.Built           88787 non-null  object 
 12  Number.of.Engines       82805 non-null  float64
 13  Purpose.of.flight       82697 non-null  object 
 14  Total.Fatal.Injuries    77488 non-null  float64
 15  Total.Serious.Injuries  76379 non-null  float64
 16  Total.Minor.Injuries    76956 non-null  float64
 17  Total.Uninjured         82977 non-null  float64
 18  Weather.Condition       84397 non-null  object 
 19  Report.Status           82505 non-null  object 
dtypes: float64(5), object(15)
memory usage: 13.6+ MB
```

In [94]:

```
# creating a subset with only numerical columns
df1_num = df1.select_dtypes(include=['number'])
```

In [95]:

```
df1_num.corr()
```

Out[95]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
Number.of.Engines	1.000000	0.098505	0.046157	0.098162	0.406058
Total.Fatal.Injuries	0.098505	1.000000	0.135724	0.073559	-0.015214
Total.Serious.Injuries	0.046157	0.135724	1.000000	0.326849	0.052869
Total.Minor.Injuries	0.098162	0.073559	0.326849	1.000000	0.147770
Total.Uninjured	0.406058	-0.015214	0.052869	0.147770	1.000000

In [96]:

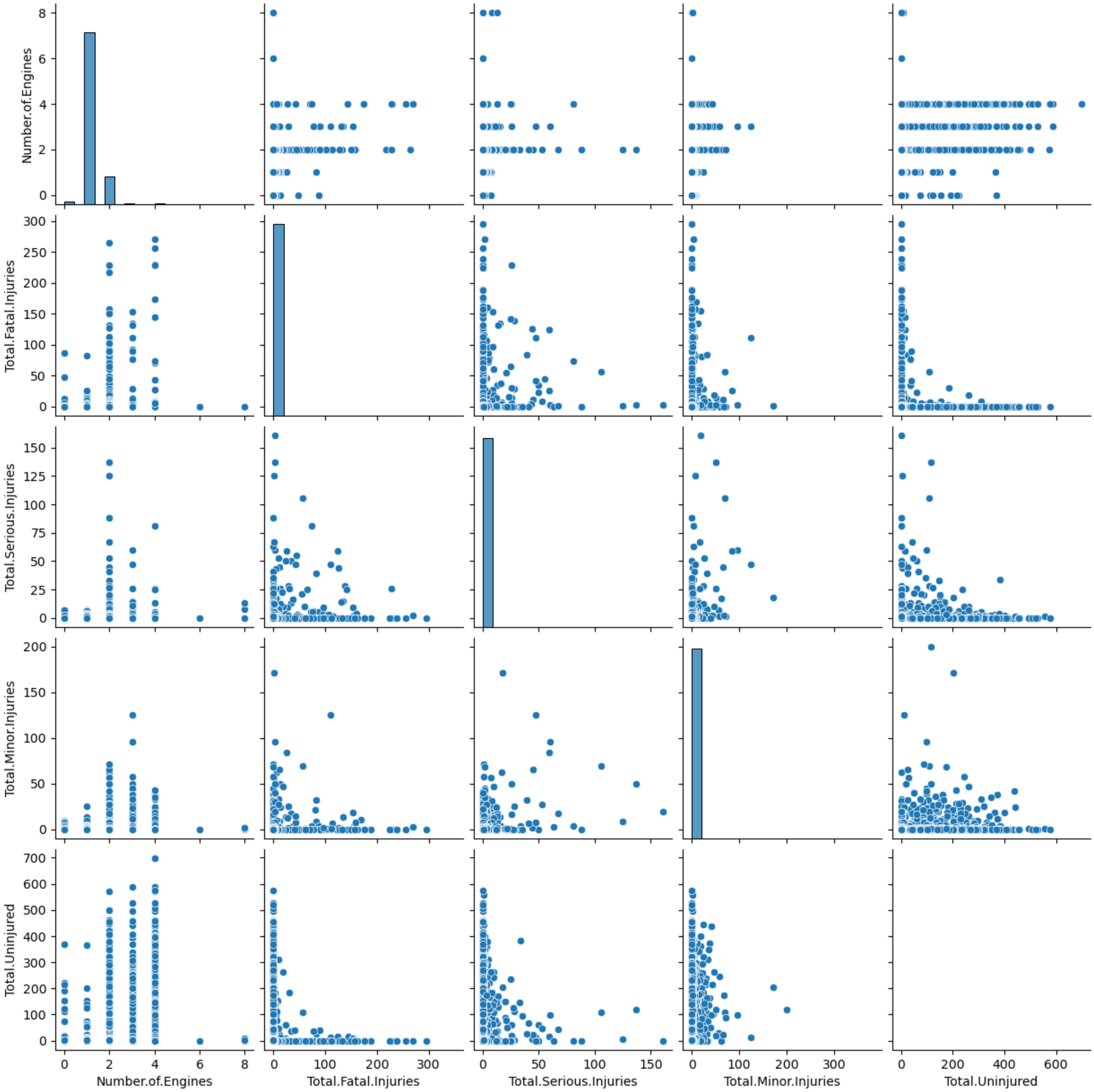
```
# ploting the correlation between the different columns
sns.pairplot(df1_num)
```

```
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
```

```
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[96]:

```
<seaborn.axisgrid.PairGrid at 0x1eda217a510>
```



From this plots i can understand that the correlation betweeen the columns is poor,so the correlation can not be used to handle missing values

In [97]:

```
# in order to handle the missing values, the mean,median and mode for each column is comp
uted so as to understand the distribution of the data

mean1=df1_num['Total.Fatal.Injuries'].mean()
median1=df1_num['Total.Fatal.Injuries'].median()
mode1=df1_num['Total.Fatal.Injuries'].mode()[0]
print(f"Mean: {mean1}, Median: {median1}, Mode: {mode1}")
```

```
Mean: 0.6478551517654346, Median: 0.0, Mode: 0.0
```

In [98]:

```
mean2=df1_num['Total.Serious.Injuries'].mean()
median2=df1_num['Total.Serious.Injuries'].median()
mode2=df1_num['Total.Serious.Injuries'].mode()[0]
print(f"Mean: {mean2}, Median: {median2}, Mode: {mode2}")
```

```
Mean: 0.27988059545162935, Median: 0.0, Mode: 0.0
```

In [99]:

```
mean3=df1_num['Total.Minor.Injuries'].mean()
median3=df1_num['Total.Minor.Injuries'].median()
mode3=df1_num['Total.Minor.Injuries'].mode()[0]
print(f"Mean: {mean3}, Median: {median3}, Mode: {mode3}")
```

```
Mean: 0.3570611778158948, Median: 0.0, Mode: 0.0
```

In [100]:

```
mean4=df1_num['Total.Uninjured'].mean()
median4=df1_num['Total.Uninjured'].median()
mode4=df1_num['Total.Uninjured'].mode()[0]
print(f"Mean: {mean4}, Median: {median4}, Mode: {mode4}")
```

```
Mean: 5.325439579642552, Median: 1.0, Mode: 0.0
```

In [101]:

```
df1_num['Total.Fatal.Injuries'].fillna(median1, inplace=True)
df1_num['Total.Serious.Injuries'].fillna(median2, inplace=True)
df1_num['Total.Minor.Injuries'].fillna(median3, inplace=True)
df1_num['Total.Uninjured'].fillna(median4, inplace=True)
```

In [102]:

```
df1_num.isnull().sum()
```

Out[102]:

```
Number.of.Engines         6084
Total.Fatal.Injuries         0
Total.Serious.Injuries       0
Total.Minor.Injuries         0
Total.Uninjured              0
dtype: int64
```

In [103]:

```
# I am returning back to the data frame the numerical columns,where the missing value has
just been handled

columns_to_replace = ['Total.Fatal.Injuries', 'Total.Serious.Injuries','Total.Minor.Injur
ies','Total.Uninjured']
df1[columns_to_replace] = df1_num[columns_to_replace]
```

In [104]:

```
df1.isnull().sum()
```

Out[104]:

```
Event.Id                     0
Investigation.Type           0
Accident.Number              0
Event.Date                   0
Location                    52
Country                    226
Injury.Severity           1000
Aircraft.damage           3194
Registration.Number       1382
Make                        63
Model                       92
Amateur.Built              102
```

```
Number.of.Engines         6084
Purpose.of.flight         6192
Total.Fatal.Injuries         0
Total.Serious.Injuries       0
Total.Minor.Injuries         0
Total.Uninjured              0
Weather.Condition         4492
Report.Status             6384
dtype: int64
```

In [105]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 20 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   Injury.Severity         87889 non-null  object 
 7   Aircraft.damage         85695 non-null  object 
 8   Registration.Number     87507 non-null  object 
 9   Make                    88826 non-null  object 
 10  Model                   88797 non-null  object 
 11  Amateur.Built           88787 non-null  object 
 12  Number.of.Engines       82805 non-null  float64
 13  Purpose.of.flight       82697 non-null  object 
 14  Total.Fatal.Injuries    88889 non-null  float64
 15  Total.Serious.Injuries  88889 non-null  float64
 16  Total.Minor.Injuries    88889 non-null  float64
 17  Total.Uninjured         88889 non-null  float64
 18  Weather.Condition       84397 non-null  object 
 19  Report.Status           82505 non-null  object 
dtypes: float64(5), object(15)
memory usage: 13.6+ MB
```

In [106]:

```
# Am checking missing value in the columns if they share the same row

num_nulls = df1.isnull().sum(axis=1)
rows_with_nulls = df1[num_nulls == 9].index
rows_with_nulls
df1= df1.drop(index=rows_with_nulls )
df1.isnull().sum()
```

Out[106]:

```
Event.Id                     0
Investigation.Type           0
Accident.Number              0
Event.Date                   0
Location                    52
Country                    226
Injury.Severity            998
Aircraft.damage           3192
Registration.Number       1380
Make                        61
Model                       90
Amateur.Built              102
Number.of.Engines         6082
Purpose.of.flight         6190
Total.Fatal.Injuries         0
Total.Serious.Injuries       0
Total.Minor.Injuries         0
Total.Uninjured              0
```

```
Weather.Condition          4490
Report.Status              6382
dtype: int64
```

DATA HANDLING

In [107]:

```
# i want to convert the Event.Date to seperate columns year,month,day. So am doing this t
o compare it year wise,month wise and day wise

df1['Event.Date'] = pd.to_datetime(df['Event.Date'])
df1['Event.Year'] = df1['Event.Date'].dt.year
df1['Event.Month'] = df1['Event.Date'].dt.month
df1['Event.Day'] = df1['Event.Date'].dt.day
```

In [108]:

```
# I want to change the Injury.Severity column. When the value is fetal,the number of peop
le injured is written with it so am going to seperate them.

df1['num_injured'] = df1['Injury.Severity'].str.extract(r'\((\d+)\)').astype(float)
df1['Injury.Severity'] = df1['Injury.Severity'].str.replace(r'\(\d+\)', '', regex=True).
str.strip()
df1['Injury.Severity'].unique()
```

Out[108]:

```
array(['Fatal', 'Non-Fatal', 'Incident', 'Unavailable', nan, 'Minor',
       'Serious'], dtype=object)
```

In [109]:

```
# So i decided to change the 'Serious' injuries to Fatal while the 'Minor' and 'Incident'
to NOn-Fetal. Incident was considered as minor because the corrisponding value in Aircraf
t.damage is minor.
# The purpuse of this is to reduce the categories.

df1['Injury.Severity'] = df1['Injury.Severity'].replace({
    'Serious': 'Fatal',
    'Incident': 'Non-Fatal',
    'Minor': 'Non-Fatal',
    'Unavailable': np.nan
})
df1['Injury.Severity'].unique()
```

Out[109]:

```
array(['Fatal', 'Non-Fatal', nan], dtype=object)
```

In [110]:

```
df1['Aircraft.damage'].unique()
```

Out[110]:

```
array(['Destroyed', 'Substantial', 'Minor', nan, 'Unknown'], dtype=object)
```

In [111]:

```
# I am going to consider the Substaintial and Minor as Non destroyed,Unknown is going to
be considered as nan.

df1["Aircraft.damage"] = df1["Aircraft.damage"].replace({
    'Minor': 'Non-Destroyed',
    'Substantial': 'Non-Destroyed',
    'Unknown': np.nan
})
df1['Aircraft.damage'].unique()
```

Out[111]:

```
array(['Destroyed', 'Non-Destroyed', nan], dtype=object)
```

In [112]:

```
df1['Purpose.of.flight'].unique()
```

Out[112]:

```
array(['Personal', nan, 'Business', 'Instructional', 'Unknown', 'Ferry',
       'Executive/corporate', 'Aerial Observation', 'Aerial Application',
       'Public Aircraft', 'Skydiving', 'Other Work Use', 'Positioning',
       'Flight Test', 'Air Race/show', 'Air Drop',
       'Public Aircraft - Federal', 'Glider Tow',
       'Public Aircraft - Local', 'External Load',
       'Public Aircraft - State', 'Banner Tow', 'Firefighting',
       'Air Race show', 'PUBS', 'ASHO', 'PUBL'], dtype=object)
```

In [113]:

```
# Since Purpose of has many categorical values, i am going to arrenge them in to 7 catego
ries
category_mapping = {
    'Personal': 'Personal/Business',
    'Business': 'Personal/Business',
    'Executive/corporate': 'Personal/Business',
    'Other Work Use': 'Personal/Business',
    'Positioning': 'Ferry/Positioning',
    'Instructional': 'Flight Training/Testing',
    'Flight Test': 'Flight Training/Testing',
    'Unknown': np.nan,
    'Ferry': 'Ferry/Positioning',
    'Aerial Observation': 'Aerial Work',
    'Aerial Application': 'Aerial Work',
    'Public Aircraft': 'Public Aircraft',
    'Skydiving': 'Recreational/Sport',
    'Air Race/show': 'Recreational/Sport',
    'Air Race show': 'Recreational/Sport',
    'Air Drop': 'Aerial Work',
    'Public Aircraft - Federal': 'Public Aircraft',
    'Glider Tow': 'Aerial Work',
    'Public Aircraft - Local': 'Public Aircraft',
    'External Load': 'Aerial Work',
    'Public Aircraft - State': 'Public Aircraft',
    'Banner Tow': 'Aerial Work',
    'Firefighting': 'Aerial Work',
    'ASHO': 'Recreational/Sport',
    'PUBS': 'Public Aircraft',
    'PUBL': 'Public Aircraft'
}
df1['Purpose.of.flight'] = df1['Purpose.of.flight'].replace(category_mapping)
df1['Purpose.of.flight'].unique()
```

Out[113]:

```
array(['Personal/Business', nan, 'Flight Training/Testing',
       'Ferry/Positioning', 'Aerial Work', 'Public Aircraft',
       'Recreational/Sport'], dtype=object)
```

In [114]:

```
df1['Weather.Condition'].unique()
```

Out[114]:

```
array(['UNK', 'IMC', 'VMC', nan, 'Unk'], dtype=object)
```

In [115]:

```
# Am changing the UNK value in nan in Weather Condition column

df1['Weather.Condition'] = df1['Weather.Condition'].replace({
    'Unk': np.nan,
    'UNK': np.nan ,
```

```
    'Unavailable': np.nan
})
df1['Weather.Condition'].unique()
```

Out[115]:

```
array([nan, 'IMC', 'VMC'], dtype=object)
```

In [116]:

```
# The Report.status column is removed because it has entry that has long sentances.

df1.drop(['Report.Status'],axis=1 ,inplace=True)
```

In [117]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 88887 entries, 0 to 88888
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   Event.Id                88887 non-null  object        
 1   Investigation.Type      88887 non-null  object        
 2   Accident.Number         88887 non-null  object        
 3   Event.Date              88887 non-null  datetime64[ns]
 4   Location                88835 non-null  object        
 5   Country                 88661 non-null  object        
 6   Injury.Severity         87793 non-null  object        
 7   Aircraft.damage         85576 non-null  object        
 8   Registration.Number     87507 non-null  object        
 9   Make                    88826 non-null  object        
 10  Model                   88797 non-null  object        
 11  Amateur.Built           88785 non-null  object        
 12  Number.of.Engines       82805 non-null  float64       
 13  Purpose.of.flight       75895 non-null  object        
 14  Total.Fatal.Injuries    88887 non-null  float64       
 15  Total.Serious.Injuries  88887 non-null  float64       
 16  Total.Minor.Injuries    88887 non-null  float64       
 17  Total.Uninjured         88887 non-null  float64       
 18  Weather.Condition       83279 non-null  object        
 19  Event.Year              88887 non-null  int32         
 20  Event.Month             88887 non-null  int32         
 21  Event.Day               88887 non-null  int32         
 22  num_injured             12564 non-null  float64       
dtypes: datetime64[ns](1), float64(6), int32(3), object(13)
memory usage: 15.3+ MB
```

In [118]:

```
# i am filling all the nan values with UNK
df1.fillna("UNK", inplace=True)
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_23872\1537370911.py:2: FutureWarning: Setting
an item of incompatible dtype is deprecated and will raise in a future error of pandas. V
alue 'UNK' has dtype incompatible with float64, please explicitly cast to a compatible dt
ype first.
  df1.fillna("UNK", inplace=True)
```

In [119]:

```
df1.drop(['num_injured'],axis=1, inplace=True)
```

In [120]:

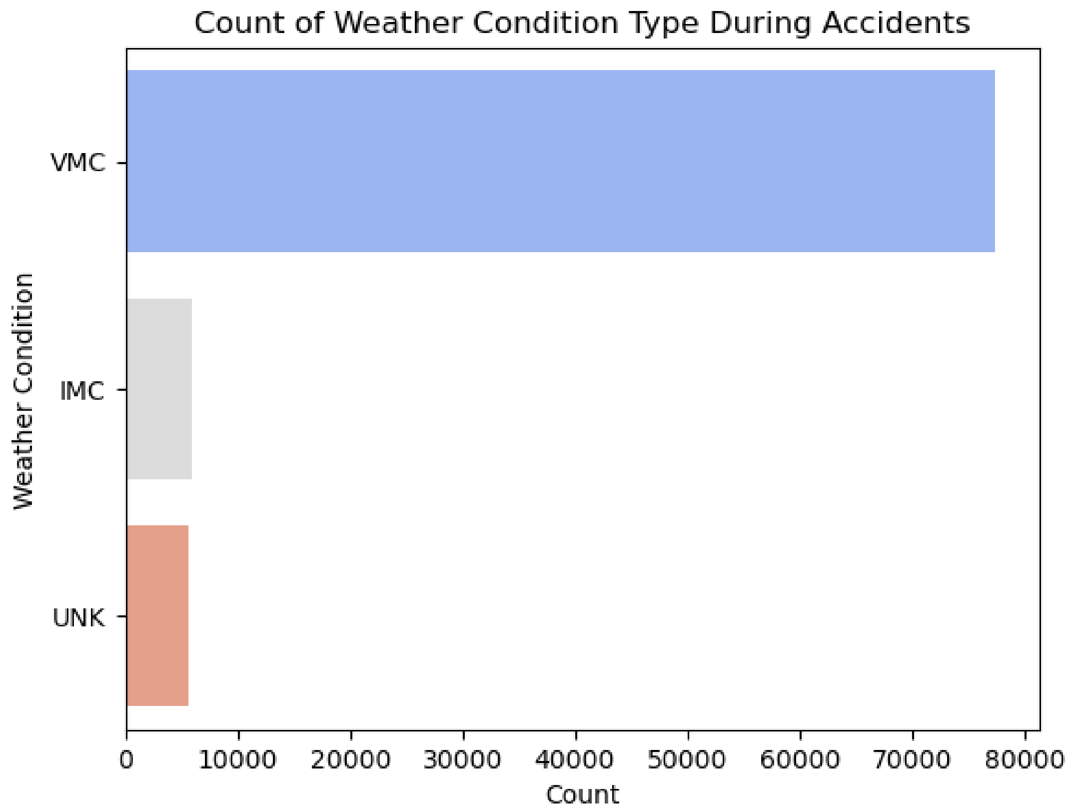
```
# The value for location is seperated into city and state

df1['City'] = df['Location'].str.split(',').str[0]
df1['State'] = df['Location'].str.split(',').str[1]
```

DATA VISUALIZATION

DATA VISUALIZATION

In [134]:

```
#In this code i am trying to understand what is the weather like during accidents

weather_condition_counts = df1['Weather.Condition'].value_counts()
sns.barplot(
    x=weather_condition_counts.values,
    y=weather_condition_counts.index,
    palette="coolwarm"
)
plt.xlabel('Count')
plt.ylabel('Weather Condition')
plt.title('Count of Weather Condition Type During Accidents')
plt.show()
```

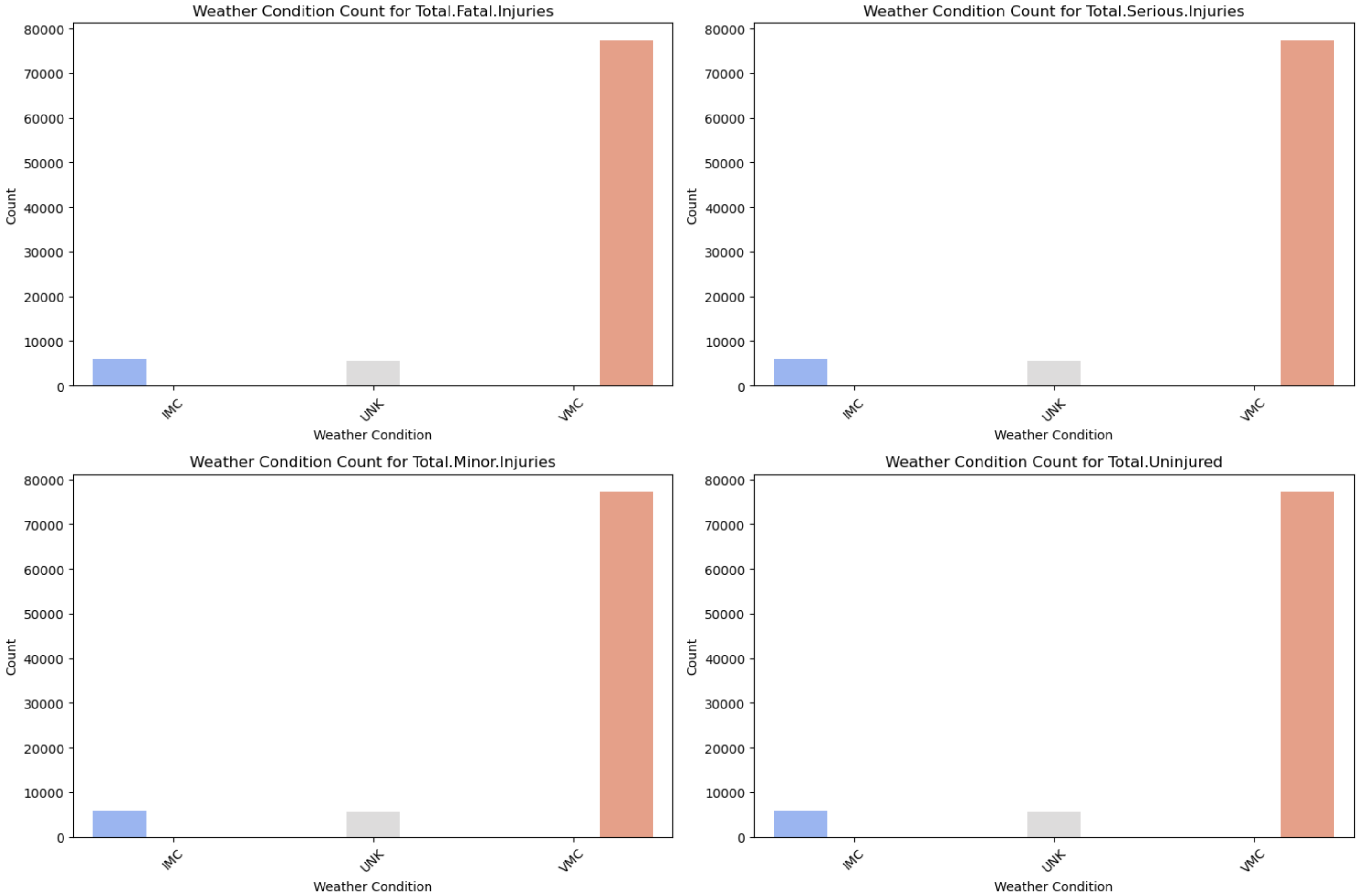


conclusion:i observed that most of the accident occur when the weather condition is VMC meaning the visibility is good.

In [138]:

```
# in this plot i will count the type of weather condition for each type of injury so as t
o understand what kind of weather prevails for different degree of injury

injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries
', 'Total.Uninjured']

fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, column in enumerate(injury_columns):
    injury_counts_by_weather = df1.groupby('Weather.Condition')[column].count()
    sns.barplot(
        x=injury_counts_by_weather.index,
        y=injury_counts_by_weather.values,
        ax=axes[i // 2, i % 2],
        palette="coolwarm",
        hue=injury_counts_by_weather.index
    )
    axes[i // 2, i % 2].set_title(f'Weather Condition Count for {column}')
    axes[i // 2, i % 2].set_xlabel('Weather Condition')
    axes[i // 2, i % 2].set_ylabel('Count')
    axes[i // 2, i % 2].tick_params(axis='x', rotation=45)
```

```
    axes[i // 2, i % 2].legend([], [], frameon=False)

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



conclusion: it can be seen that for all the injury the weather condition is VMC

In [125]:

```
df1['Weather.Condition'].unique()
```

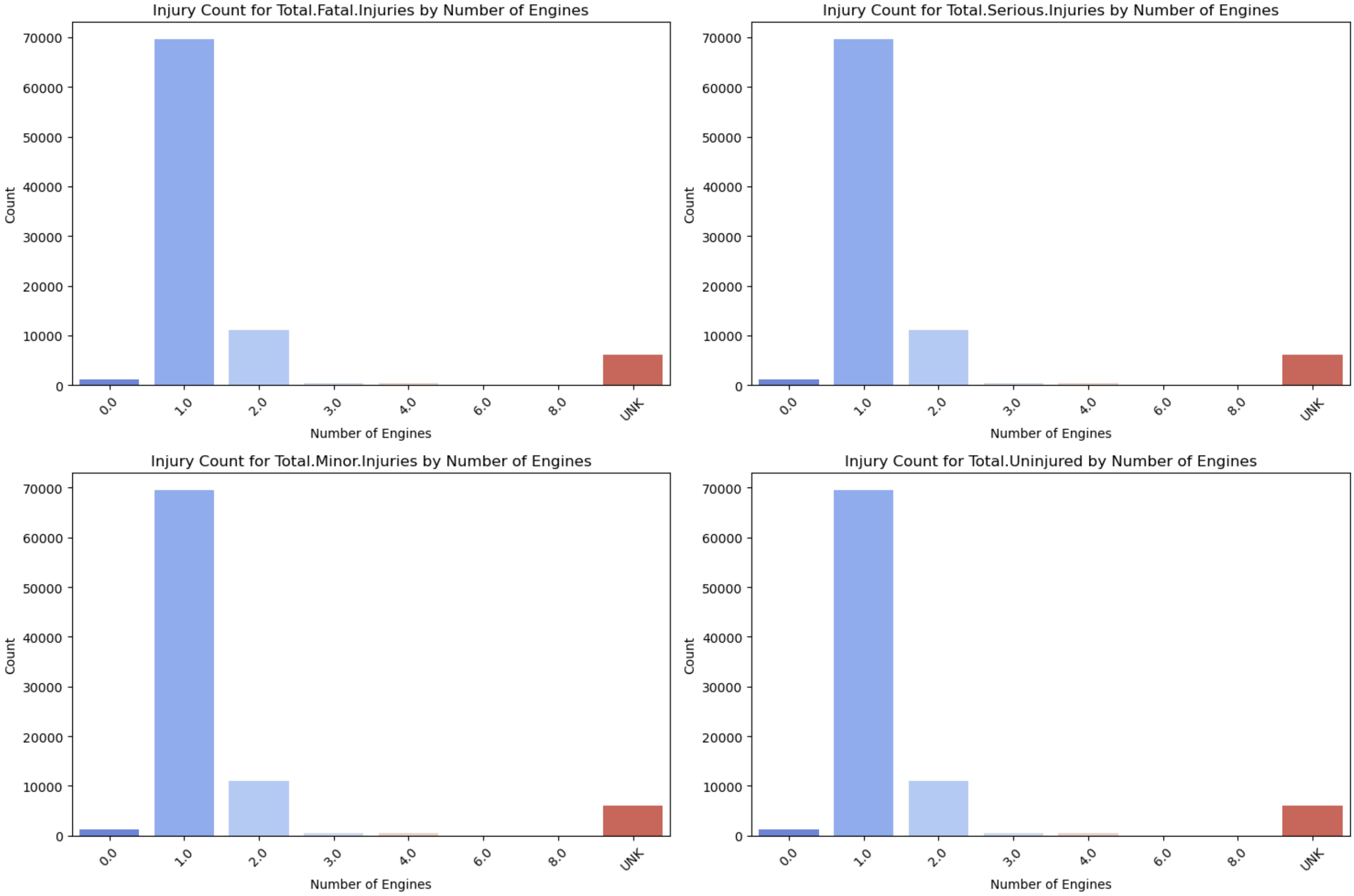
Out[125]:

```
array(['UNK', 'IMC', 'VMC'], dtype=object)
```

In [141]:

```
#i want to see the relationship of engine type with the injury type, so as to understand
which engine type result in major

injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.Injuries
', 'Total.Uninjured']

fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, column in enumerate(injury_columns):
    injury_counts_by_engines = df1.groupby('Number.of.Engines')[column].size()
    sns.barplot(
        x=injury_counts_by_engines.index,
        y=injury_counts_by_engines.values,
        ax=axes[i // 2, i % 2],
        palette="coolwarm"
    )
    axes[i // 2, i % 2].set_title(f'Injury Count for {column} by Number of Engines')
    axes[i // 2, i % 2].set_xlabel('Number of Engines')
    axes[i // 2, i % 2].set_ylabel('Count')
    axes[i // 2, i % 2].tick_params(axis='x', rotation=45)

plt.tight_layout()
```

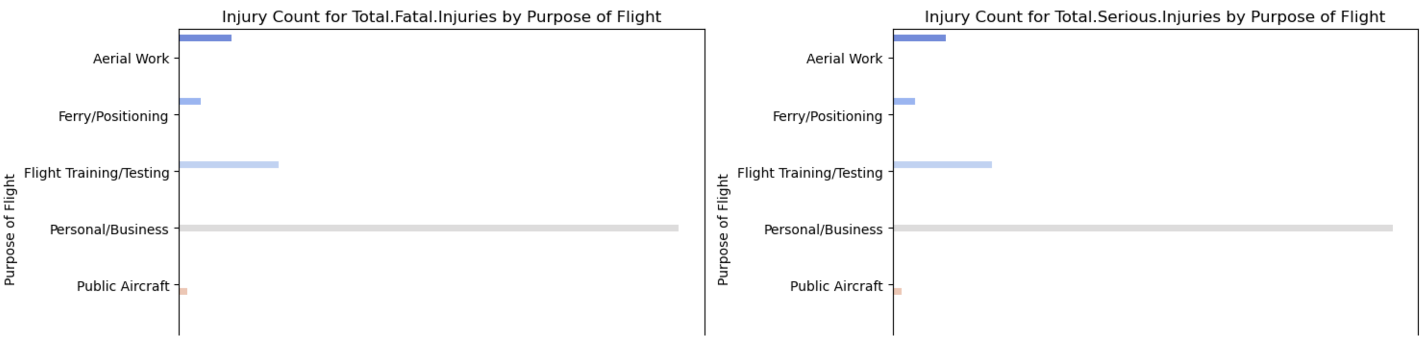
```
plt.show()
```

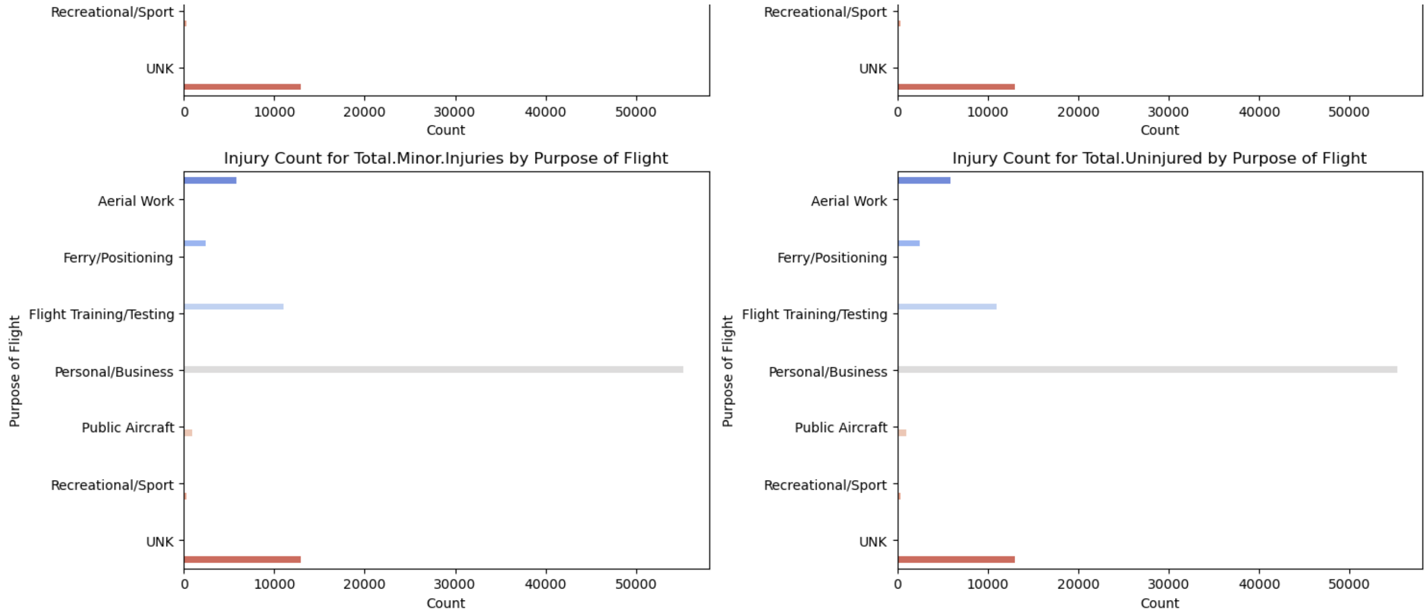


conclusion:engine 1 has the highest number of accidents for the different injuries.

In [142]:

```
#i want to see the relation between type ofinjury and the purpose of flight, to see in wh
at kind of purposes the accident is fatal and in which is just minor
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, column in enumerate(injury_columns):
    injury_counts_by_purpose = df1.groupby('Purpose.of.flight')[column].count()
    sns.barplot(
        x=injury_counts_by_purpose.values,
        y=injury_counts_by_purpose.index,
        ax=axes[i // 2, i % 2],
        palette="coolwarm",
        hue=injury_counts_by_purpose.index,

    )
    axes[i // 2, i % 2].set_title(f'Injury Count for {column} by Purpose of Flight')
    axes[i // 2, i % 2].set_xlabel('Count')
    axes[i // 2, i % 2].set_ylabel('Purpose of Flight')
    axes[i // 2, i % 2].tick_params(axis='y', rotation=0)
    axes[i // 2, i % 2].legend([], [], frameon=False)

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

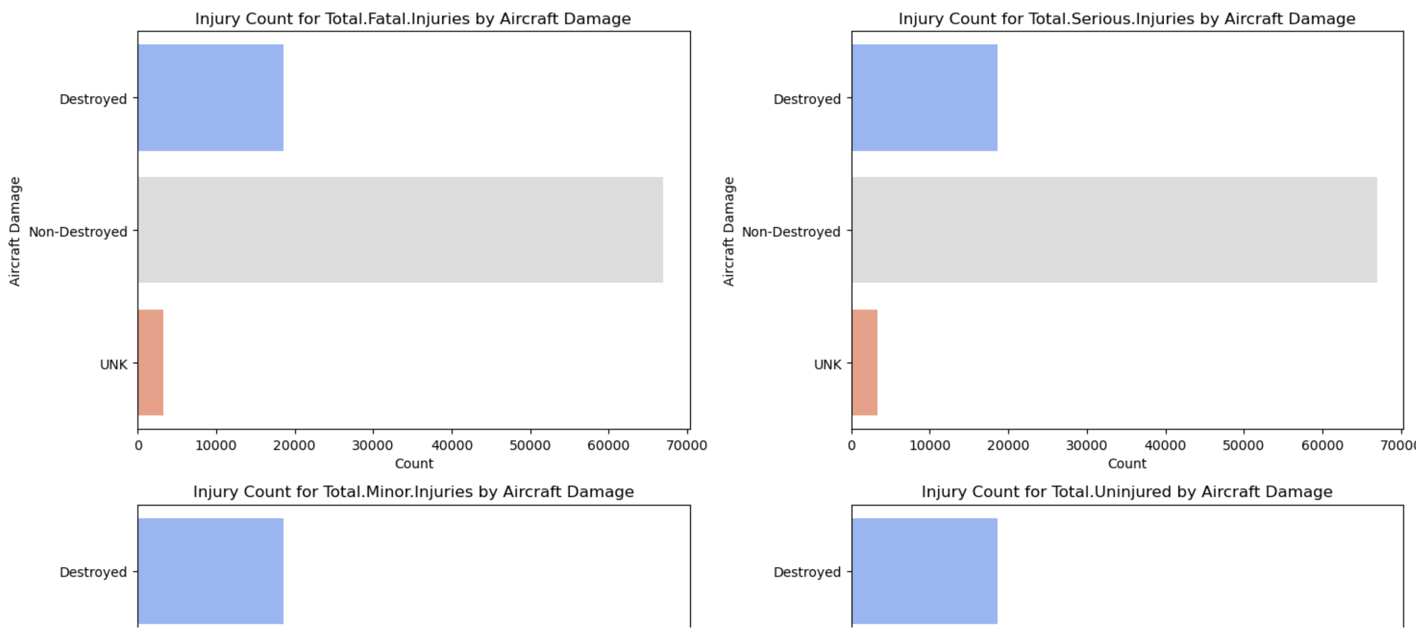


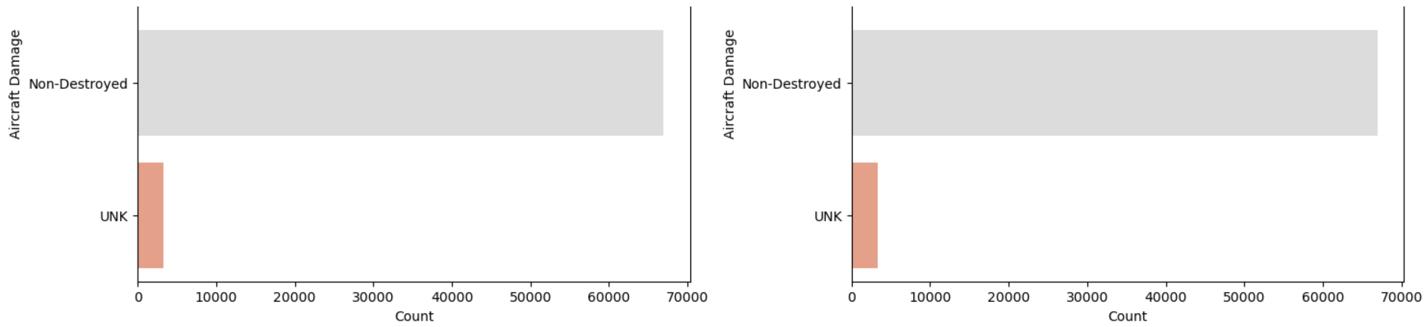


conclusion: most of the injury occur for personal or business. substatial injuries also occur for flight training or testing purpose.

In [143]:

```
#I want to see the relation between the injury type and the degree of damage of the aircr
aft, so as to see if there is a relation between fatal injury and destroyed aircraft.

fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, column in enumerate(injury_columns):
    injury_counts_by_damage = df1.groupby('Aircraft.damage')[column].count()
    sns.barplot(
        x=injury_counts_by_damage.values,
        y=injury_counts_by_damage.index,
        ax=axes[i // 2, i % 2],
        palette="coolwarm",
        hue=injury_counts_by_damage.index,
        dodge=False,

    )
    axes[i // 2, i % 2].set_title(f'Injury Count for {column} by Aircraft Damage')
    axes[i // 2, i % 2].set_xlabel('Count')
    axes[i // 2, i % 2].set_ylabel('Aircraft Damage')
    axes[i // 2, i % 2].tick_params(axis='y', rotation=0)
    axes[i // 2, i % 2].legend([], [], frameon=False)

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

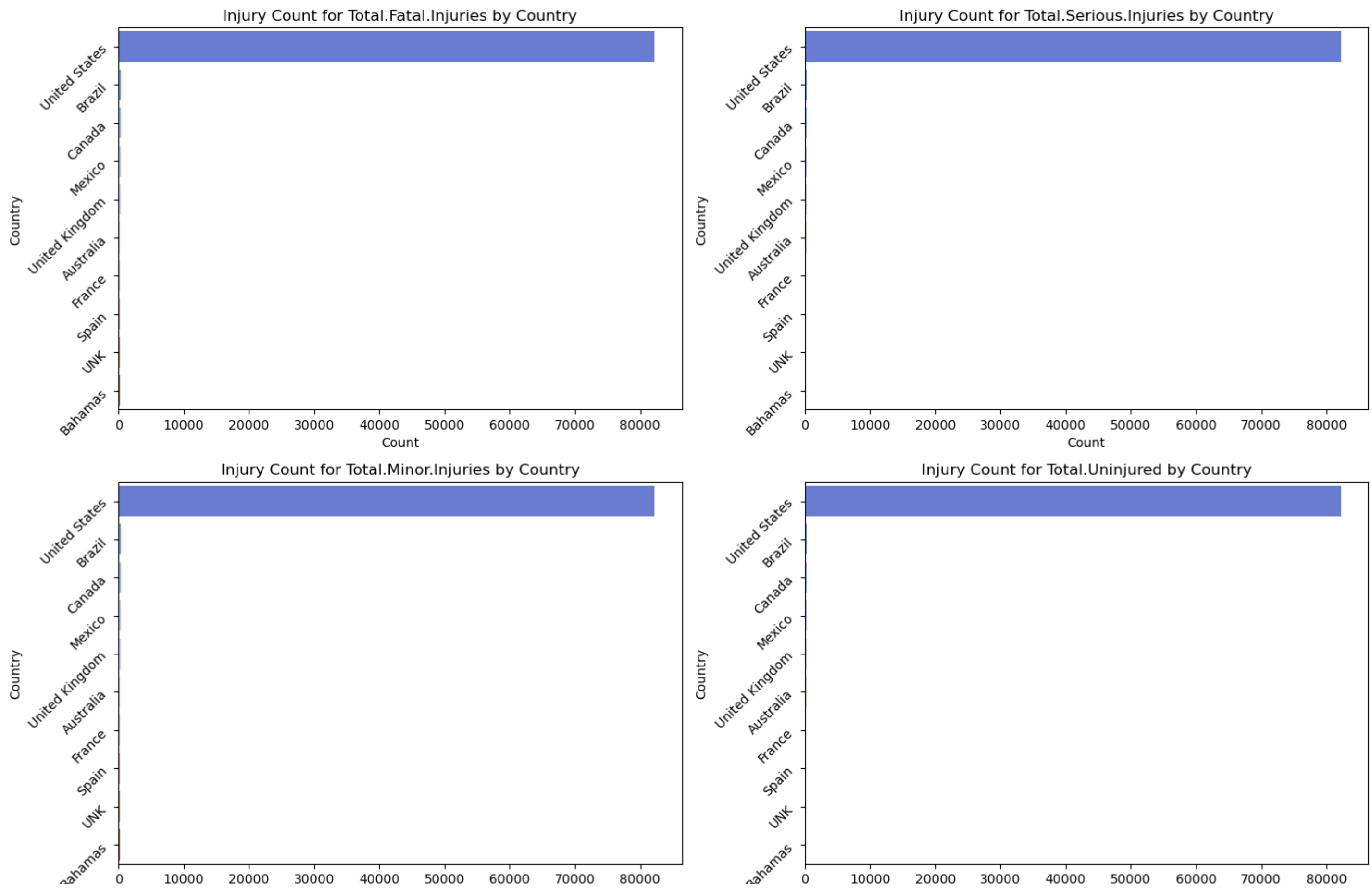




conclusion: most of the aircraft were not destroyed in relation to the different injury that occured.

In [144]:

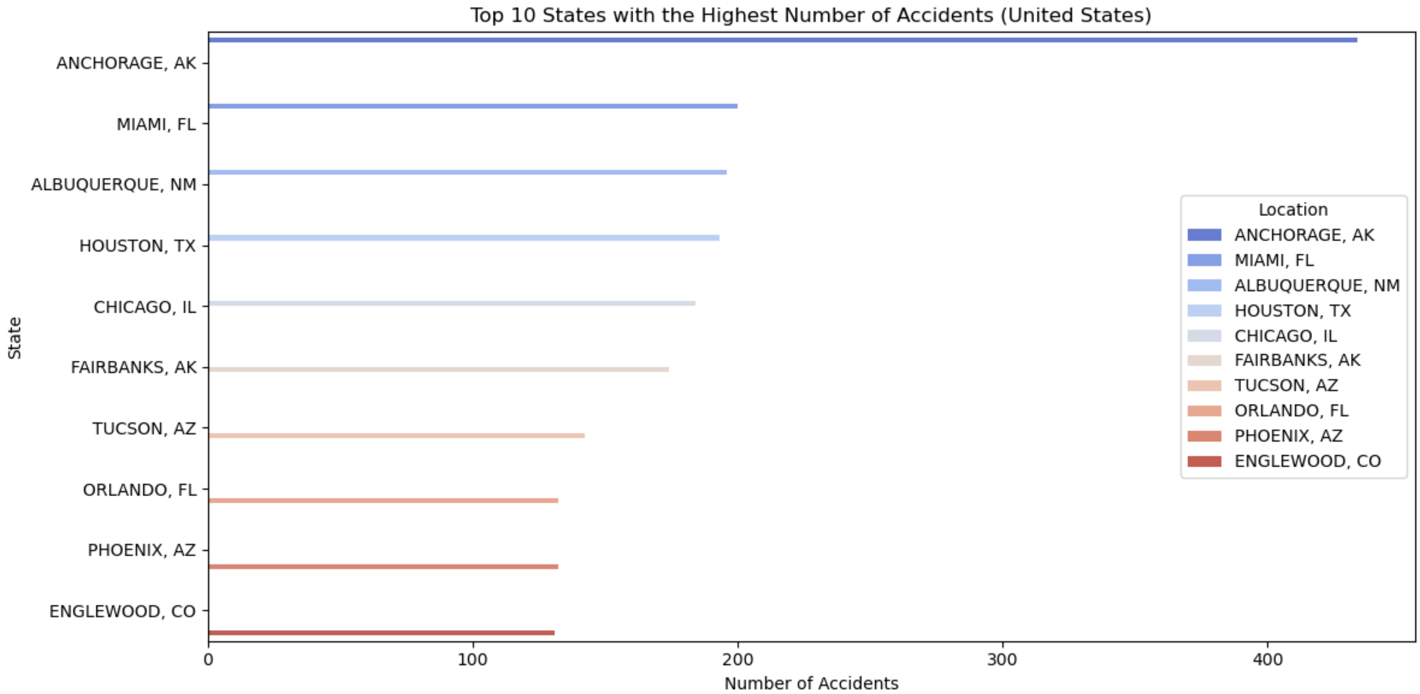
```
# i want to see the relationship between top 10 country and type of injury, so as to unde
rstand which country has highest number of different type of injury

fig, axes = plt.subplots(2, 2, figsize=(15, 10))

for i, column in enumerate(injury_columns):
    injury_counts_by_country = df1.groupby('Country')[column].count()
    top_countries = injury_counts_by_country.nlargest(10)
    sns.barplot(
        x=top_countries.values,
        y=top_countries.index,
        ax=axes[i // 2, i % 2],
        palette="coolwarm",
        hue=top_countries.index,
        dodge=False,

    )
    axes[i // 2, i % 2].set_title(f'Injury Count for {column} by Country')
    axes[i // 2, i % 2].set_xlabel('Count')
    axes[i // 2, i % 2].set_ylabel('Country')
    axes[i // 2, i % 2].tick_params(axis='y', rotation=45)
    axes[i // 2, i % 2].legend([], [], frameon=False)

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



conclusion: it can be seen that USA has the highest count in number of accidents for different injury types.

In [145]:

```
# i want to see the number of accidents for different states in the USA since the country
has the highest accidents
usa_accidents = df1[df1['Country'] == 'United States']
accidents_by_state = usa_accidents.groupby('Location').size().nlargest(10)

plt.figure(figsize=(12, 6))
sns.barplot(x=accidents_by_state.values, y=accidents_by_state.index, palette="coolwarm",
hue=accidents_by_state.index)

plt.xlabel('Number of Accidents')
plt.ylabel('State')
plt.title('Top 10 States with the Highest Number of Accidents (United States)')
plt.tight_layout()
plt.show()
```

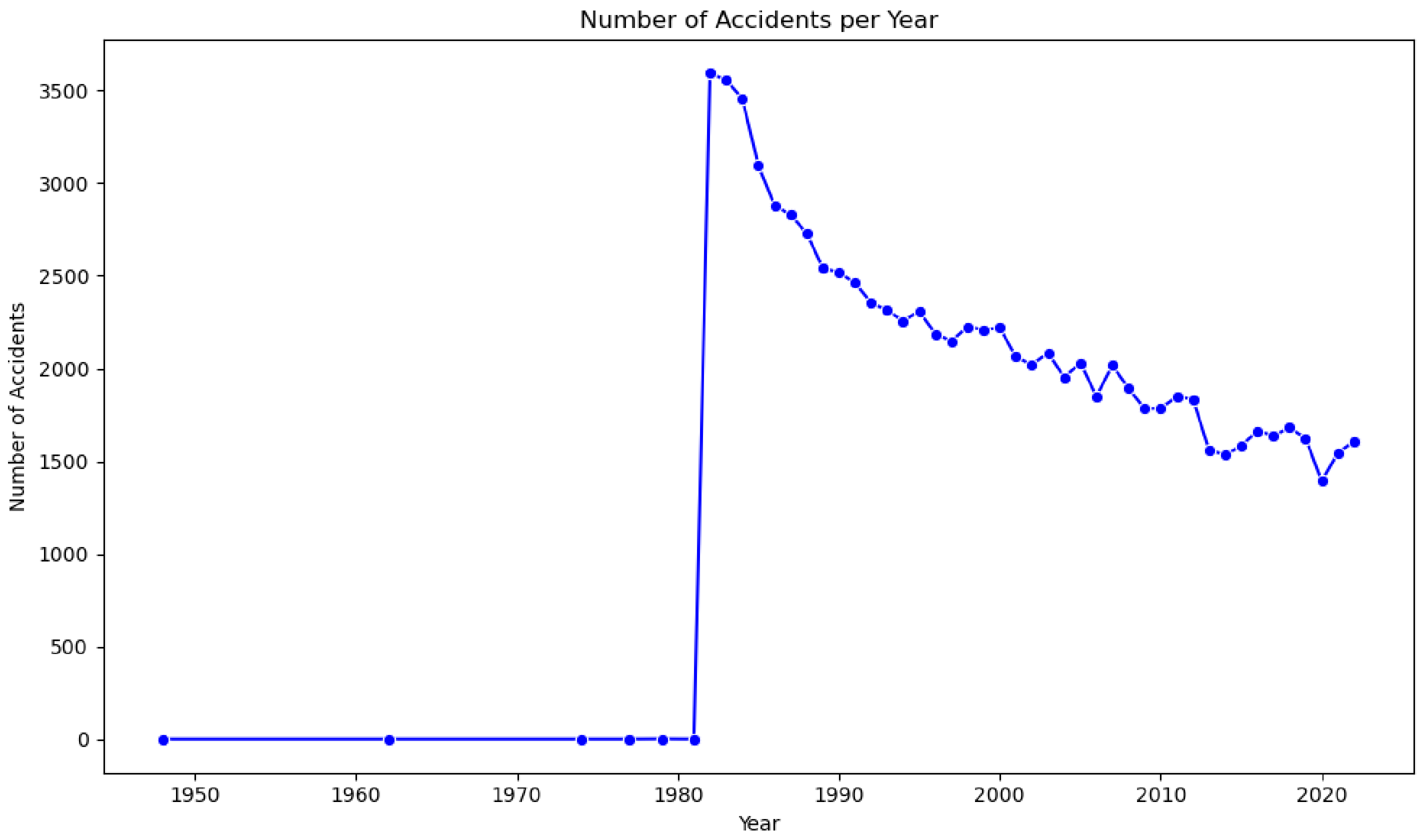


conclusion: Anchorage state has the highest accidents

In [146]:

```
# i want to see the number of accident per year so as to know which years have high value
accidents_per_year = df1.groupby('Event.Year').size()

plt.figure(figsize=(10, 6))
sns.lineplot(x=accidents_per_year.index, y=accidents_per_year.values, marker='o', color='
b')

plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents per Year')

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

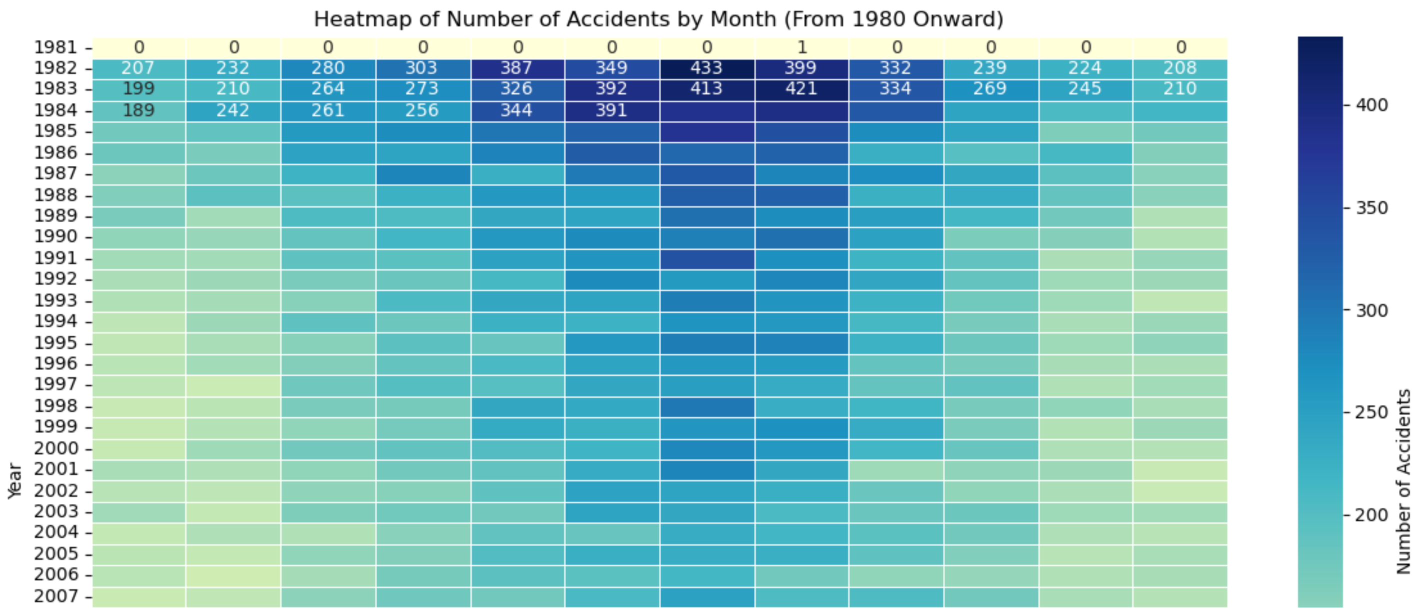
```
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADMIN\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_i
nf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

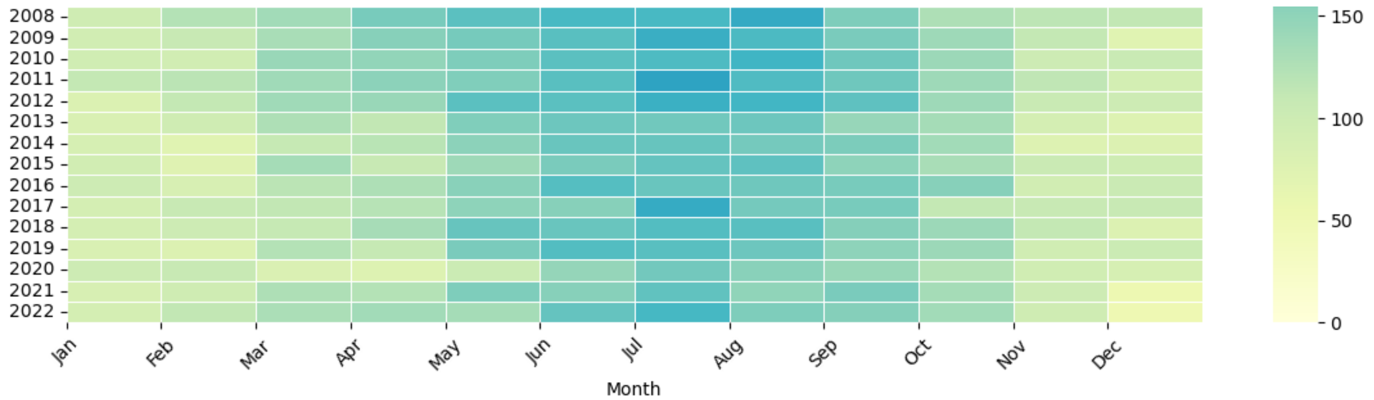


conclusion: there are a lot of accidents after 1980

In [147]:

```
#To visualize the number of accidents by month from the year 1980 onward, to know which m
onth have high accidents
df_filtered = df1[df1['Event.Year'] >= 1980]
accidents_by_month = df_filtered.groupby(['Event.Year', 'Event.Month']).size().unstack(f
ill_value=0)

plt.figure(figsize=(12, 8))
sns.heatmap(accidents_by_month, cmap='YlGnBu', annot=True, fmt='d', cbar_kws={'label': 'N
umber of Accidents'}, linewidths=0.5)

plt.xlabel('Month')
plt.ylabel('Year')
plt.title('Heatmap of Number of Accidents by Month (From 1980 Onward)')
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Au
g', 'Sep', 'Oct', 'Nov', 'Dec'], rotation=45)
plt.tight_layout()
plt.show()
```

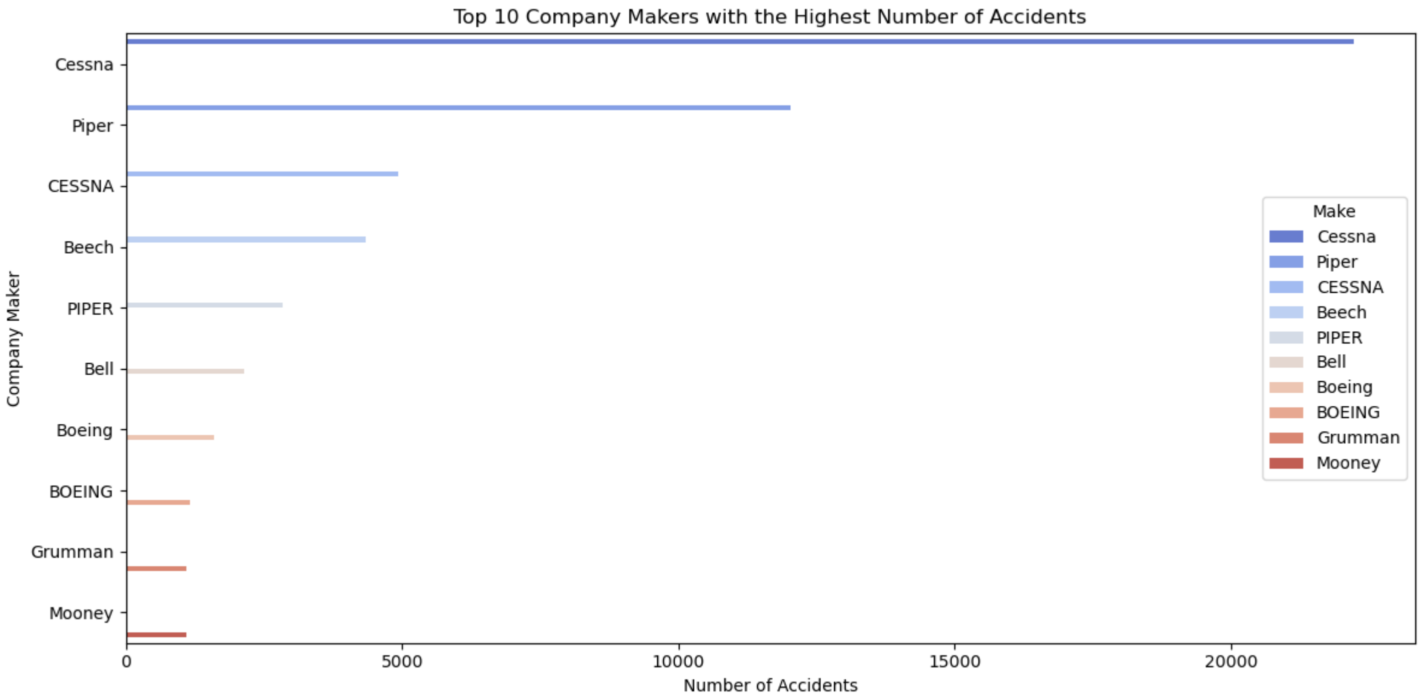




conclusion: summer months show more accidents form 1980 onward. It is especially high for late years than recent ones.

In [148]:

```
# i want to see the number of accidents per company maker so as to know which company is
not doing a good job
accidents_by_make = df1.groupby('Make').size().nlargest(10)

plt.figure(figsize=(12, 6))
sns.barplot(x=accidents_by_make.values, y=accidents_by_make.index, palette="coolwarm", h
ue=accidents_by_make.index)

plt.xlabel('Number of Accidents')
plt.ylabel('Company Maker')
plt.title('Top 10 Company Makers with the Highest Number of Accidents')
plt.tight_layout()
plt.show()
```



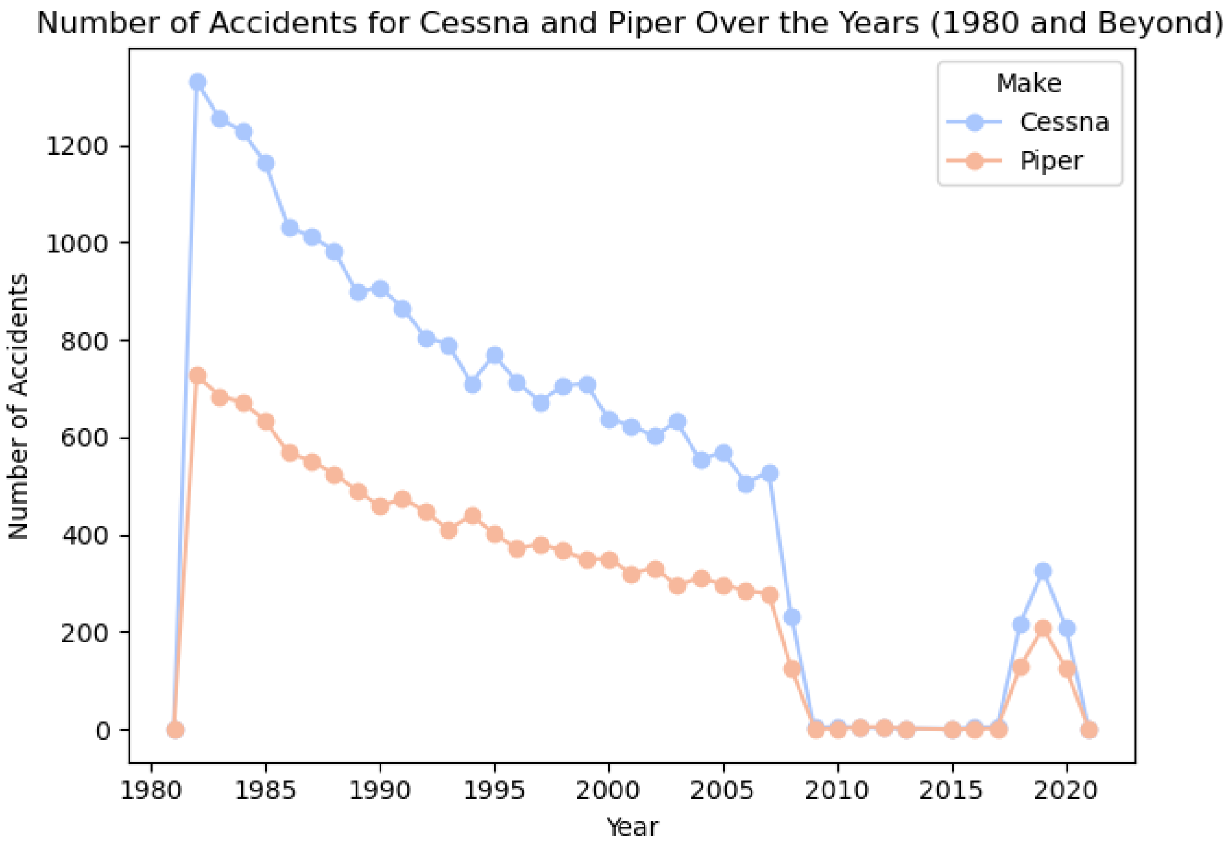
conclusion: Cessna has the highest record of accidents followed by Piper.

In [149]:

```
#i want to see the relationship between the number of accidents for Cessna and Piper, and
how this varies over the years from 1980 onwards
filtered_data = df1[(df1['Make'].isin(['Cessna', 'Piper'])) & (df1['Event.Year'] >= 1980
)]

accidents_by_year = filtered_data.groupby(['Event.Year', 'Make']).size().unstack().filln
a(0)

plt.figure(figsize=(12, 6))
```

```
accidents_by_year.plot(kind='line', marker='o', color=sns.color_palette("coolwarm", 2))

plt.xlabel('Year')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents for Cessna and Piper Over the Years (1980 and Beyond)')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



conclusion: it can be seen that the accidents were slowly decreasing from 1980 onward, but around 2016 a suddent increase occured which slowly decreased till 2022

Conclusion

Weather Condition: VMC (Visual Meteorological Conditions) is the most common weather condition for accidents across all injury types.

Engine Type: Single-engine aircraft (1 engine) are involved in the majority of accidents.

Purpose of Flight: Most accidents occur during personal or business flights.

Aircraft Damage: The majority of accidents do not result in severe aircraft destruction.

Geographic Location: The USA has the highest number of recorded accidents, with Anchorage having the highest accident rate among states.

Trends Over Time: A significant number of accidents have occurred since 1980, with a descending trend in recent years.

Seasonal Pattern: The highest number of accidents occur during the summer months.

Aircraft Make: Cessna is the maker with the highest number of accidents, followed by Piper.