Business Understanding

Business problem:

My company is expanding in to new industries to diversify its portfolio. They are interested in purchasing and operating airplanes for commercial and private enterprises, I am charged to determine anything about the potential risks of aircraft. I am charged with determining which aircraft are the lowest risk for the company to start this new business endeavor.I am must then translate my findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase.

Project objectives:

Main objective

To help decide which aircraft to purchase and operate airplanes for commercial and private enterprises.

Specific objectives

- To find correlation and visualize between

    1. make and mode and their risks
    2. the type of aircraft category and their risks.
    3. The type of engine and no of engines and their risks..

Data Understanding

Data collection

Data (https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses) was collected from the National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

In [1]:
```
#importing libraries for data manipulation (pandas, numpy) and visualizati
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# set the maximum number of columns to 40 to display all columns
pd.set_option('display.max_columns', 40)
```

Reading Files

Using pandas and load using

```
ntsb_df = pd.read_csv('AviationData.csv', encoding='latin1')
```

Then describing ntsb_df

In [3]:
```
#Load National Transportation Safety Board(ntsb) dataset to ntsb_df
#set encoding to latin1 to handle encoding error
ntsb_df = pd.read_csv('AviationData.csv', encoding='latin1')
```

```
C:\Users\Windows User\anaconda3\lib\site-packages\IPython\core\interactiv
eshell.py:3444: DtypeWarning: Columns (6,7,28) have mixed types.Specify d
type option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [4]:
```
#getting top 5 rows
ntsb_df.head()
```

Out[4]:

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

In [5]:
```
#getting bottom 5 rows
ntsb_df.tail()
```

Out[5]:

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

In [6]:
```
#getting sample 5 rows
ntsb_df.tail()
```

Out[6]:

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

In [7]:
```
#checking data info
ntsb_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            50249 non-null  object 
 9   Airport.Name            52790 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     87572 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             81812 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           82508 non-null  object 
 30  Publication.Date        75118 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

In [8]:
```
#describing numerical columns
ntsb_df.describe()
```

Out[8]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total
count	82805.000000	77488.000000	76379.000000	76956.000000	82
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	

In [9]:
```
#describing numerical columns
ntsb_df.describe(include='O')
```

Out[9]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Co
count	88889	88889	88889	88889	88837	
unique	87951	2	88863	14782	27758	
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	
freq	3	85015	2	25	434	

In [12]:
```
# Checking shape of ntsb_df
ntsb_df.shape
```

Out[12]: (88889, 31)

In [13]:
```
#filtering top 30 Make
Make_df = ntsb_df.Make.value_counts().reset_index().iloc[:10]
ntsb_df = ntsb_df[ntsb_df.Make.isin(Make_df['index'])]
```

In [14]:
```
ntsb_df.shape
```

Out[14]: (53414, 31)

Data Preparation

Cleaning

1. Changing Columns Data Types and/or Values
2. Filling Missing Values

In [15]:
```
ntsb_df.columns
```

Out[15]:
```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type', 'FAR.Descript
ion',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injur
ies',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjure
d',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [16]:
```
#Changing columns name by replace . with _ and lowercasing
ntsb_df.columns = ntsb_df.columns.str.lower()
ntsb_df.columns = ntsb_df.columns.str.replace(".","_")
ntsb_df.columns
```

```
C:\Users\WINDOW~1\AppData\Local\Temp/ipykernel_2596/257064276.py:3: Futur
eWarning: The default value of regex will change from True to False in a
future version. In addition, single character regular expressions will *n
ot* be treated as literal strings when regex=True.
  ntsb_df.columns = ntsb_df.columns.str.replace(".","_")
```

Out[16]:
```
Index(['event_id', 'investigation_type', 'accident_number', 'event_date',
       'location', 'country', 'latitude', 'longitude', 'airport_code',
       'airport_name', 'injury_severity', 'aircraft_damage',
       'aircraft_category', 'registration_number', 'make', 'model',
       'amateur_built', 'number_of_engines', 'engine_type', 'far_descript
ion',
       'schedule', 'purpose_of_flight', 'air_carrier', 'total_fatal_injur
ies',
       'total_serious_injuries', 'total_minor_injuries', 'total_uninjure
d',
       'weather_condition', 'broad_phase_of_flight', 'report_status',
       'publication_date'],
      dtype='object')
```

In [17]:
```
#checking columns
for col in ntsb_df.columns:
    #unique values and no of unique values
    unique_val = str(ntsb_df[col].unique())
    no_unique_val = str(ntsb_df[col].nunique())

    print(col + " | No of unique = "+ no_unique_val +"\n" +unique_val+"\n"
```

```
event_id | No of unique = 52978
['20001218X45447' '20061025X01555' '20041105X01764' ... '2022122710649
1'
 '20221227106498' '20221230106513']

investigation_type | No of unique = 2
['Accident' 'Incident']

accident_number | No of unique = 53406
['LAX94LA336' 'NYC07LA005' 'CHI79FA064' ... 'ERA23LA093' 'WPR23LA076'
 'ERA23LA097']

event_date | No of unique = 13807
['1962-07-19' '1974-08-30' '1979-08-02' ... '2022-12-21' '2022-12-26'
 '2022-12-29']

location | No of unique = 19028
['BRIDGEPORT, CA' 'Saltville, VA' 'Canton, OH' ...
 'Happy Valley-Goose Bay, OF' 'San Manual, AZ' 'Auburn Hills, MI']
```

In [18]:

```
#Fixing values which are written in different cases
for col in set(ntsb_df.select_dtypes(include = ['O'])):
    #original values contained within a column
    org_unique_val = set(ntsb_df[col].unique())
    no_org_unique_val = ntsb_df[col].nunique()

    if col in ['location','engine_type','make','model','airport_name','air
        ntsb_df[col] = ntsb_df[col].str.title()
    elif col in ['weather_condition','airport_code','longitude','registrat
        ntsb_df[col] = ntsb_df[col].str.upper()

    #Values contained within a column after formatting(capitalizing/titlin
    no_fmr_unique_val =  ntsb_df[col].nunique()

    #check the value of difference before any computation
    diff = no_org_unique_val - no_fmr_unique_val

    if diff > 0:
        fmr_unique_val =  set(ntsb_df[col].unique())
        #difference btw values
        print(col+"\n No of similar values " + str(no_org_unique_val- no_f
        #to check the similar values
        print( "Sample Values\n"+str(list(org_unique_val - fmr_unique_val)
```

```
air_carrier
 No of similar values 150
Sample Values
['(dba: Buffalo Express)', 'NEWGREN STEVEN A', 'CITADEL AVIATION LLC', 'J
OHANNES SCOTT', 'Rocky Mountain Intensive (dba: Air Care, Inc.)', 'BAKER
GEORGE', 'TROY MINSKE', 'McElwain Sprayers, LLC', "Moe's Crop Dusting Ser
vice Inc.", 'WENCK KENT W', 'Peninsula Airways Inc. (dba: Penair)', '(db
a: Sundance Aviation)', 'DOOLEY CHARLES R', 'FLIGHT TRAINING OF MOBILE LL
C', 'SKINNER LEASING LLC']
model
 No of similar values 7
Sample Values
['PA-22', 'PA 28-180', 'PA32', 'PA22-160', 'PA-18-95', '177XP', 'MD', 'PA
-32-260\x7f', 'BE-18', 'PA-22-150', 'PA-46-310', '152II', '412HP', 'KINGA
IR 200', 'PA60-601P']
location
 No of similar values 3442
Sample Values
['Monroe, GA', 'ADEL, GA', 'Clayton, OK', 'CEDARVILLE, OH', 'GLEN BURNIE,
MD', 'Ponca City, OK', 'Koliganek, AK', 'KINTNERSVILLE, PA', 'GUAYAQUIL,
Ecuador', 'Greenwater, WA', 'DEFIANCE, OH', 'BRAITHWAITE, LA', 'Forest Fa
lls, CA', 'Shallowater, TX', 'W. BRATTLEBORO, VT']
airport_code
 No of similar values 19
Sample Values
['55m', 'Hot', 'cdg', 'pbi', 'cdw', 'cxl', 'Unkn', 'psp', 'pvt', 'ipl',
'xxx', 'na', 'sna', 'jjc', 'None']
make
 No of similar values 3
Sample Values
['BOEING', 'CESSNA', 'PIPER']
registration_number
 No of similar values 7
Sample Values
['Ex-009', 'N5759t', 'N29cf', 'None', 'Vh-XMO', 'unknown', 'N5464n', 'N91
33m', 'N5929t', 'none', 'N7821k', 'N738Ej', 'N4418c', 'N2242r', 'N2995v']
airport_name
 No of similar values 1985
Sample Values
['SEARCY', 'MARSHALL MEM', 'CANTON', 'BARBOUR ISLAND', 'MCCOMAS', 'RICHAR
D B HELGESON', 'JAMISON STRIP', 'CLEVELAND-HOPKINS', 'ST. GEORGE', 'REMOT
E AIRSTRIP', 'PANAMINT SPRINGS RESORT', 'SAIPAN INTERNATIONAL', 'JOHNSON
COUTY EXEC.ARPT.', 'SUMMERLAND KEY', 'WEATHERFORD']
longitude
 No of similar values 5
Sample Values
[-88.355555, -85.663611, -81.878056, -113.8875, -173.24]
weather_condition
 No of similar values 1
Sample Values
['Unk']
```

In [19]:

```
#Checking if there is any similar values after the manipulation
total_diff = 0
for col in set(ntsb_df.select_dtypes(include = ['O'])):
    #original values contained within a column
    no_org_unique_val = ntsb_df[col].nunique()

    if col in ['Location','Engine.Type','Make','Airport.Name','Air.carrier
        ntsb_df[col] = ntsb_df[col].str.title()
    elif col in ['Weather.Condition','Airport.Code','Longitude','Registrat
        ntsb_df[col] = ntsb_df[col].str.upper()

    #Values contained within a column after formatting(capitalizing/titlin
    no_fmr_unique_val =  ntsb_df[col].nunique()

    #check the value of difference before any computation
    diff = no_org_unique_val - no_fmr_unique_val

    total_diff+=diff

print("Similar values = "+str(total_diff))
```

```
Similar values = 0
```

In [20]:
```
#checking missing values
ntsb_df.isna().sum()
```

Out[20]:
```
event_id                    0
investigation_type          0
accident_number             0
event_date                  0
location                   37
country                   155
latitude                36741
longitude               36746
airport_code            22176
airport_name            20463
injury_severity           581
aircraft_damage          1595
aircraft_category       37533
registration_number       819
make                        0
model                      12
amateur_built              65
number_of_engines        2726
engine_type              3198
far_description         37631
schedule                46330
purpose_of_flight        3167
air_carrier             45462
total_fatal_injuries     6706
total_serious_injuries   7519
total_minor_injuries     7136
total_uninjured          3280
weather_condition        2287
broad_phase_of_flight   12484
report_status            2993
publication_date         9532
dtype: int64
```

In [21]:
```
#checking injury_severity
ntsb_df['injury_severity'].unique()
```

Out[21]:
```
array(['Fatal(4)', 'Fatal(3)', 'Fatal(1)', 'Non-Fatal', 'Fatal(2)',
       'Incident', 'Fatal(8)', 'Fatal(78)', 'Fatal(6)', 'Fatal(5)',
       'Fatal(7)', 'Fatal(153)', 'Fatal(14)', 'Fatal(10)', 'Fatal(17)',
       'Fatal(29)', 'Fatal(9)', 'Unavailable', 'Fatal(25)', 'Fatal(82)',
       'Fatal(18)', 'Fatal(270)', 'Fatal(144)', 'Fatal(11)', 'Fatal(13
1)',
       'Fatal(73)', 'Fatal(34)', 'Fatal(13)', 'Fatal(47)', 'Fatal(12)',
       'Fatal(132)', 'Fatal(54)', 'Fatal(65)', 'Fatal(72)', 'Fatal(20)',
       'Fatal(160)', 'Fatal(189)', 'Fatal(123)', 'Fatal(33)',
       'Fatal(230)', 'Fatal(70)', 'Fatal(349)', 'Fatal(125)', 'Fatal(3
5)',
       'Fatal(228)', 'Fatal(97)', 'Fatal(52)', 'Fatal(87)', 'Fatal(15)',
       'Fatal(16)', 'Fatal(80)', 'Fatal(217)', 'Fatal(60)', 'Fatal(83)',
       'Fatal(44)', 'Fatal(64)', 'Fatal(92)', 'Fatal(118)', 'Fatal(138)',
       'Fatal(206)', 'Fatal(71)', 'Fatal(23)', 'Fatal(21)', 'Fatal(102)',
       'Fatal(115)', 'Fatal(141)', 'Fatal(104)', 'Fatal(121)',
       'Fatal(45)', 'Fatal(145)', 'Fatal(117)', 'Fatal(154)', 'Fatal(9
6)',
       'Fatal(114)', 'Fatal(89)', nan, 'Fatal', 'Minor', 'Serious'],
      dtype=object)
```

In [22]:
```
#changing Fatal(x) injury_severity to Fatal
ntsb_df['injury_severity'] = ntsb_df['injury_severity'].map(lambda x: 'Fat
ntsb_df['injury_severity'].unique()
```

Out[22]:
```
array(['Fatal', 'Incident', nan, 'Minor', 'Serious'], dtype=object)
```

In [23]:
```
#changing nan injury_severity to Unknown
ntsb_df['injury_severity'] = ntsb_df['injury_severity'].map(lambda x: 'Unk
ntsb_df['injury_severity'].unique()
```

Out[23]:
```
array(['Fatal', 'Incident', 'Unknown', 'Minor', 'Serious'], dtype=object)
```

In [24]:

```
# Checking missing values before
print(ntsb_df['country'].isna().sum())

# Create location_make_df with unique combinations of 'country', 'location
location_make_df = ntsb_df[['country', 'location', 'make', 'model']].dropn

# First attempt: Update missing 'country' based on 'location'
ntsb_df['country'] = ntsb_df.apply(
    lambda row: location_make_df.loc[
        location_make_df['location'] == row['location'], 'country'].values
    if pd.isna(row['country']) and row['location'] in location_make_df['lo
    else row['country'],
    axis=1
)

# Second attempt: Update missing 'country' based on 'make' and 'model' if
def fill_country_based_on_make_model(row):
    if pd.isna(row['country']):
        matching_rows = location_make_df[
            (location_make_df['make'] == row['make']) & (location_make_df[

        if not matching_rows.empty:
            return matching_rows['country'].values[0]
        else:
            return row['country']
    return row['country']

ntsb_df['country'] = ntsb_df.apply(fill_country_based_on_make_model, axis=

# Checking missing values after
print(ntsb_df['country'].isna().sum())
```

```
155
4
```

In [25]:

```
# Checking missing values before
print(ntsb_df['location'].isna().sum())

# Create location_make_df with unique combinations of 'country', 'location
location_make_df = ntsb_df[['country', 'location', 'make', 'model']].dropn

# First attempt: Update missing 'location' based on 'country'
ntsb_df['location'] = ntsb_df.apply(
    lambda row: location_make_df.loc[
        location_make_df['country'] == row['country'], 'location'].values[
    if pd.isna(row['location']) and row['country'] in location_make_df['co
    else row['location'],
    axis=1
)

# Second attempt: Update missing 'location' based on 'make' and 'model' if
ntsb_df['location'] = ntsb_df.apply(
    lambda row: location_make_df.loc[
        (location_make_df['make'] == row['make']) & (location_make_df['mod
    if pd.isna(row['location']) and row['make'] in location_make_df['make'
    else row['location'],
    axis=1
)

# Checking missing values after
print(ntsb_df['location'].isna().sum())
```

```
37
0
```

In [26]:

```
# Checking missing values before
print(ntsb_df[['make', 'model']].isna().sum())

# Create location_make_df with unique combinations of 'country', 'location
location_make_df = ntsb_df[['country', 'location', 'make', 'model']].dropn

# First attempt: Update missing 'make' based on 'country' and 'location'
ntsb_df['make'] = ntsb_df.apply(
    lambda row: location_make_df.loc[
        (location_make_df['country'] == row['country']) & (location_make_d
    if pd.isna(row['make']) and row['country'] in location_make_df['countr
    else row['make'],
    axis=1
)

# Second attempt: Update missing 'make' based on 'model' if 'country' and
ntsb_df['make'] = ntsb_df.apply(
    lambda row: location_make_df.loc[
        (location_make_df['model'] == row['model']), 'make'].values[0]
    if pd.isna(row['make']) and row['model'] in location_make_df['model'].
    else row['make'],
    axis=1
)

# Second attempt: Update missing 'model' based on 'make' and 'country' if
def safe_fill_model(row):
    if pd.isna(row['model']):
        matching_rows = location_make_df[
            (location_make_df['make'] == row['make']) & (location_make_df[

        if not matching_rows.empty:
            return matching_rows['model'].values[0]
        else:
            return row['model']  # If no match found, return the original
    return row['model']

ntsb_df['model'] = ntsb_df.apply(safe_fill_model, axis=1)

# Checking missing values after
print(ntsb_df[['make', 'model']].isna().sum())
```

```
make      0
model    12
dtype: int64
make     0
model    1
dtype: int64
```

In [27]:

```
# Checking missing values before
print(ntsb_df['aircraft_category'].isna().sum())

# Create location_make_df with unique combinations of 'aircraft_category',
location_make_df = ntsb_df[['aircraft_category', 'location', 'make', 'mode

# First attempt: Update missing 'aircraft_category' based on 'location'
ntsb_df['aircraft_category'] = ntsb_df.apply(
    lambda row: location_make_df.loc[
        location_make_df['location'] == row['location'], 'aircraft_categor
    if pd.isna(row['aircraft_category']) and row['location'] in location_m
    else row['aircraft_category'],
    axis=1
)

# Second attempt: Update missing 'aircraft_category' based on 'make' and '
def fill_aircraft_category_based_on_make_model(row):
    if pd.isna(row['aircraft_category']):
        matching_rows = location_make_df[
            (location_make_df['make'] == row['make']) & (location_make_df[

        if not matching_rows.empty:
            return matching_rows['aircraft_category'].values[0]
        else:
            return row['aircraft_category']
    return row['aircraft_category']

ntsb_df['aircraft_category'] = ntsb_df.apply(fill_aircraft_category_based_

# Checking missing values after
print(ntsb_df['aircraft_category'].isna().sum())
```

```
37533
875
```

In [28]:
```
ntsb_df.isna().sum()
```

Out[28]:
```
event_id                    0
investigation_type          0
accident_number             0
event_date                  0
location                    0
country                     4
latitude                36741
longitude               36746
airport_code            22176
airport_name            20463
injury_severity             0
aircraft_damage          1595
aircraft_category         875
registration_number       819
make                        0
model                       1
amateur_built              65
number_of_engines        2726
engine_type              3198
far_description         37631
schedule                46330
purpose_of_flight        3167
air_carrier             45462
total_fatal_injuries     6706
total_serious_injuries   7519
total_minor_injuries     7136
total_uninjured          3280
weather_condition        2287
broad_phase_of_flight   12484
report_status            2993
publication_date         9532
dtype: int64
```

In [29]:
```
#ntsb_df[['make','injury_severity']].value_counts().reset_index()
```

In [30]:
```
#ntsb_df[['country','location','make']].value_counts().reset_index().iloc[
```

In [31]:
```
#drop all missing values
#ntsb_df.dropna(inplace=True)
```

Feature Engineering

In [32]:
```
 #Adding year from event_date
ntsb_df['year'] = ntsb_df['event_date'].map(lambda x: x.split("-")[0].stri
ntsb_df['year'].unique()[:50]
```

Out[32]:
```
array(['1962', '1974', '1979', '1981', '1982', '1983', '1984', '1985',
       '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
       '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001',
       '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009',
       '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017',
       '2018', '2019', '2020', '2021', '2022'], dtype=object)
```

In [33]:
```
#Adding states
states = pd.read_csv('USState_Codes.csv')
```

In [34]:
```
states.head()
```

Out[34]:

	US_State	Abbreviation
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA

In [35]:
```
ntsb_df.head()
```

Out[35]:

	event_id	investigation_type	accident_number	event_date	location	country	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	Bridgeport, Ca	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, Va	United States	3(
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, Oh	United States	
6	20001218X45446	Accident	CHI81LA106	1981-08-01	Cotton, Mn	United States	
7	20020909X01562	Accident	SEA82DA022	1982-01-01	Pullman, Wa	United States	

In [36]:
```
 #Adding state from location
ntsb_df['state'] = ntsb_df['location'].map(lambda x: x.split(",")[-1].uppe
ntsb_df['state'] = ntsb_df['state'].map(lambda x: states.loc[states['Abbre
ntsb_df['state'].unique()[:50]
```

Out[36]:
```
array(['California', 'Virginia', 'Ohio', 'Minnesota', 'Washington',
       'New Jersey', 'New Mexico', 'Alabama', 'Texas', 'Arkansas',
       'Alaska', 'Pennsylvania', 'Michigan', 'Georgia', 'Florida',
       'Oregon', 'Nevada', 'Indiana', 'SAINT CROIX', 'Arizona',
       'New York', 'North Carolina', 'Missouri', 'Wyoming', 'Illinois',
       'South Carolina', 'Montana', 'Maryland', 'Hawaii', 'Colorado',
       'Mississippi', 'Washington_DC', 'Oklahoma', 'Utah', 'Vermont',
       'Louisiana', 'Kansas', 'New Hampshire', 'Iowa', 'Idaho',
       'Wisconsin', 'Massachusetts', 'Connecticut', 'Kentucky',
       'GULF OF MEXICO', 'Tennessee', 'SAN JUAN', 'South Dakota',
       'Nebraska', 'Rhode Island'], dtype=object)
```

In [37]:
```
 #Adding town from location
ntsb_df['town'] = ntsb_df['location'].map(lambda x: x.split(",")[0].strip(
ntsb_df['town'].unique()[:5]
```

Out[37]:
```
array(['Bridgeport', 'Saltville', 'Canton', 'Cotton', 'Pullman'],
      dtype=object)
```

In [38]:
```
# Creating a function to determine the severity based on the greatest valu
def calculate_severity(row):
    # Extract the values for each injury type
    fatal = row['total_fatal_injuries']
    serious = row['total_serious_injuries']
    minor = row['total_minor_injuries']
    uninjured = row['total_uninjured']

    # Determine the category with the greatest number
    max_value = max(fatal, serious, minor, uninjured)

    # Classify based on the greatest value
    if max_value == fatal:
        return 'Fatal'
    elif max_value == serious:
        return 'Serious'
    elif max_value == minor:
        return 'Minor'
    else:
        return 'Uninjured'

# Apply the function to create the new 'calculated_severity' column
ntsb_df['calculated_severity'] = ntsb_df.apply(calculate_severity, axis=1)
```

In [39]:
```
ntsb_df['calculated_severity'].unique()
```

Out[39]:
```
array(['Fatal', 'Serious', 'Uninjured', 'Minor'], dtype=object)
```

In [40]:
```
#taking companies based on us
ntsb_main_df = ntsb_df[ntsb_df['country'] == 'United States']
ntsb_main_df['country'].unique()
```

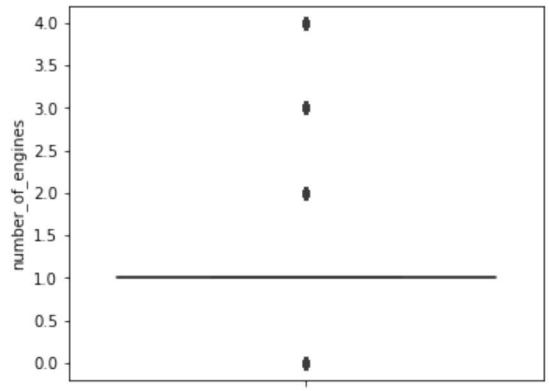
Out[40]:
```
array(['United States'], dtype=object)
```

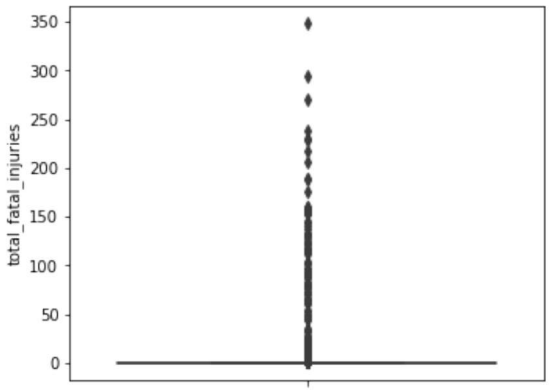
Checking Outliers

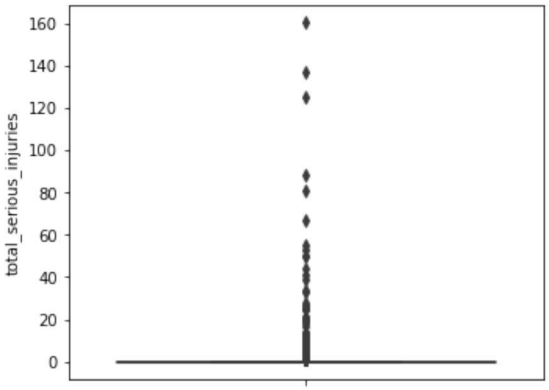
In [41]:
```
numeric_df =ntsb_main_df.select_dtypes(include = ['number'])
```

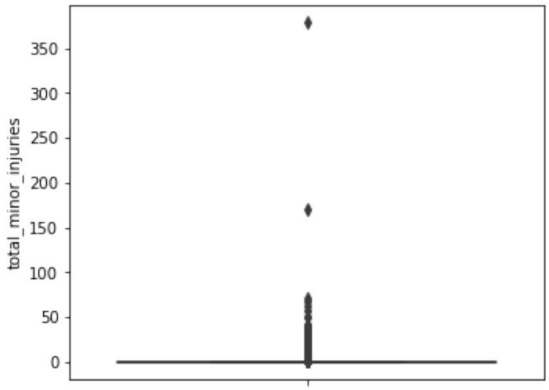
In [42]:

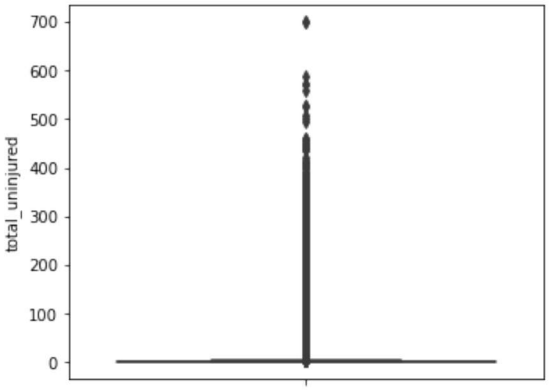
```
#calculate the number of fig to fit height
grid=(numeric_df.shape[1]+1)//2
#allocating each plot a height of 5
plt.figure(figsize=(12, grid * 5))

count=0
for col in numeric_df:
    count += 1
    plt.subplot(grid,2,count)
    sns.boxplot(y=ntsb_df[col])
```







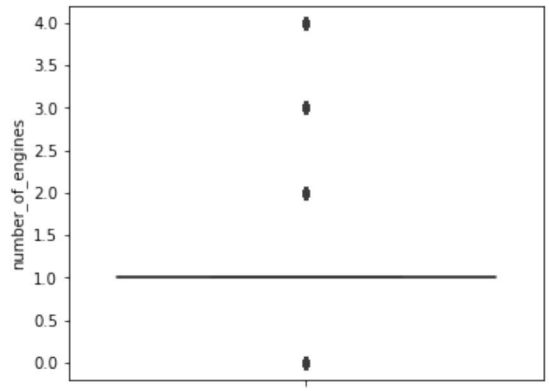


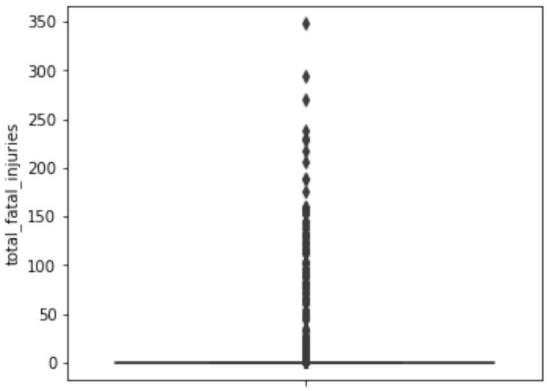


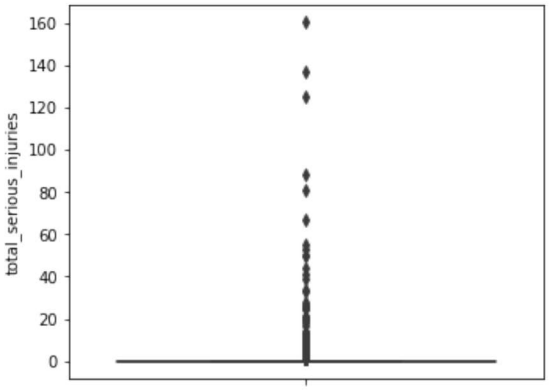
In [43]:

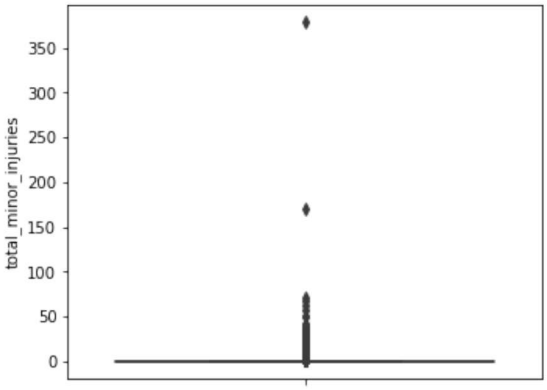
```
for col in numeric_df:
    q1= ntsb_main_df[col].quantile(0.25)
    q3 = ntsb_main_df[col].quantile(0.90) #because of large amount of outl

    # calculating iqr
    iqr = q3 - q1

    #defining lower and upper bound
    upper= q3 + 1.5 * iqr
    lower = q1 - 1.5 * iqr

    # removing outliers
    ntsb_main_df_cleaned = ntsb_main_df[(ntsb_main_df[col] >= lower) & (nt
```

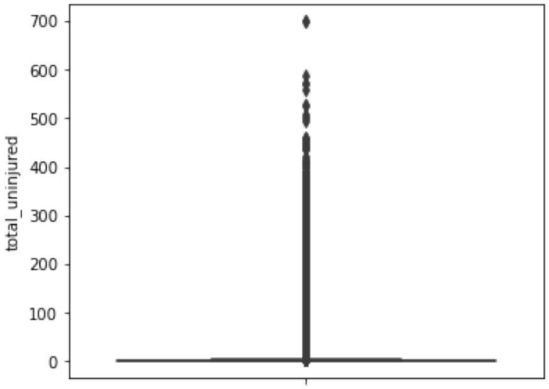
In [44]:
```
#allocating each plot a height of 5
plt.figure(figsize=(12, grid * 5))

count=0
for col in numeric_df:
    count += 1
    plt.subplot(grid,2,count)
    sns.boxplot(y=ntsb_df[col])
```











In [45]:
```
#taking no of engine above 0
ntsb_df = ntsb_df[ntsb_df.number_of_engines > 0]
```

Checking Duplicates

In [46]:
```
ntsb_df.duplicated()
```

Out[46]:
```
1        False
2        False
6        False
7        False
8        False
         ...
88858    False
88861    False
88865    False
88869    False
88877    False
Length: 50546, dtype: bool
```

In [47]:
```
ntsb_df.duplicated().sum()
```

Out[47]:
```
0
```

In [48]:
```
ntsb_df.drop_duplicates(inplace=True)
```

In [49]:
```
#checking shape
ntsb_main_df.shape
```

Out[49]:
```
(49852, 35)
```

In [50]:
```
#taking companies based on us
ntsb_main_df = ntsb_df[ntsb_df['country'] == 'United States']
ntsb_main_df['country'].unique()
```

Out[50]:
```
array(['United States'], dtype=object)
```

In [51]:
```
#select columns to use
ntsb_main_df = ntsb_df[['event_id', 'investigation_type', 'accident_number
       'location', 'country','injury_severity', 'aircraft_damage',
       'aircraft_category', 'registration_number', 'make', 'model',
       'amateur_built', 'number_of_engines', 'engine_type', 'far_descripti
       'weather_condition', 'broad_phase_of_flight',
       'publication_date', 'state','year','calculated_severity','town','to
       'total_serious_injuries', 'total_minor_injuries', 'total_uninjured'
```

In [52]:
```
ntsb_main_df.shape
```

Out[52]:
```
(50546, 28)
```

In [53]:
```
ntsb_main_df.isna().sum()
```

Out[53]:
```
event_id                    0
investigation_type          0
accident_number             0
event_date                  0
location                    0
country                     3
injury_severity             0
aircraft_damage           977
aircraft_category         782
registration_number        34
make                        0
model                       1
amateur_built              12
number_of_engines           0
engine_type              1077
far_description         36327
purpose_of_flight        1575
weather_condition         739
broad_phase_of_flight    9700
publication_date         9046
state                       0
year                        0
calculated_severity         0
town                        0
total_fatal_injuries     6219
total_serious_injuries   6801
total_minor_injuries     6384
total_uninjured          2714
dtype: int64
```

In [54]:
```
ntsb_main_df.dropna(inplace=True)
```

```
C:\Users\Windows User\anaconda3\lib\site-packages\pandas\util\_decorator
s.py:311: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  return func(*args, **kwargs)
```

In [55]:
```
ntsb_main_df.shape
```

Out[55]:
```
(2629, 28)
```

In [56]:
```
ntsb_main_df.isna().sum()
```

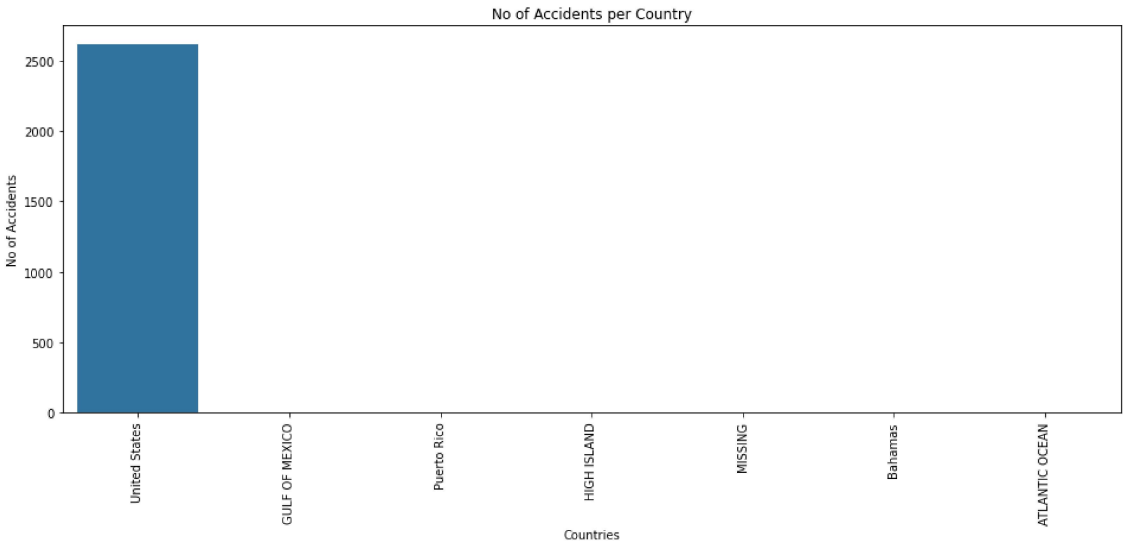
Out[56]:
```
event_id                0
investigation_type      0
accident_number         0
event_date              0
location                0
country                 0
injury_severity         0
aircraft_damage         0
aircraft_category       0
registration_number     0
make                    0
model                   0
amateur_built           0
number_of_engines       0
engine_type             0
far_description         0
purpose_of_flight       0
weather_condition       0
broad_phase_of_flight   0
publication_date        0
state                   0
year                    0
calculated_severity     0
town                    0
total_fatal_injuries    0
total_serious_injuries  0
total_minor_injuries    0
total_uninjured         0
dtype: int64
```

In [57]:
```
ntsb_main_df.to_csv("ntsb_clean.csv", index=False)
```

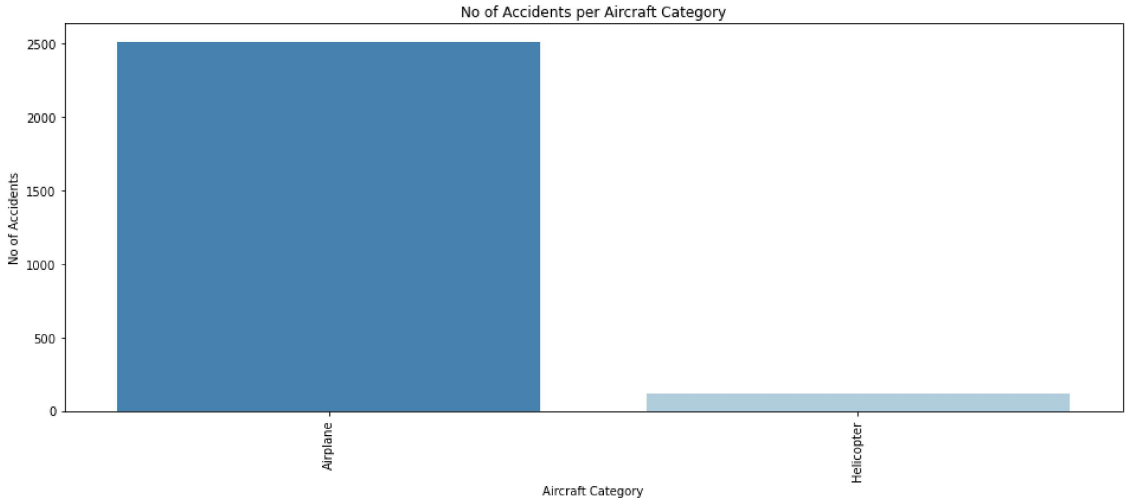
Explanatory Data Analysis

Univariate Analysis

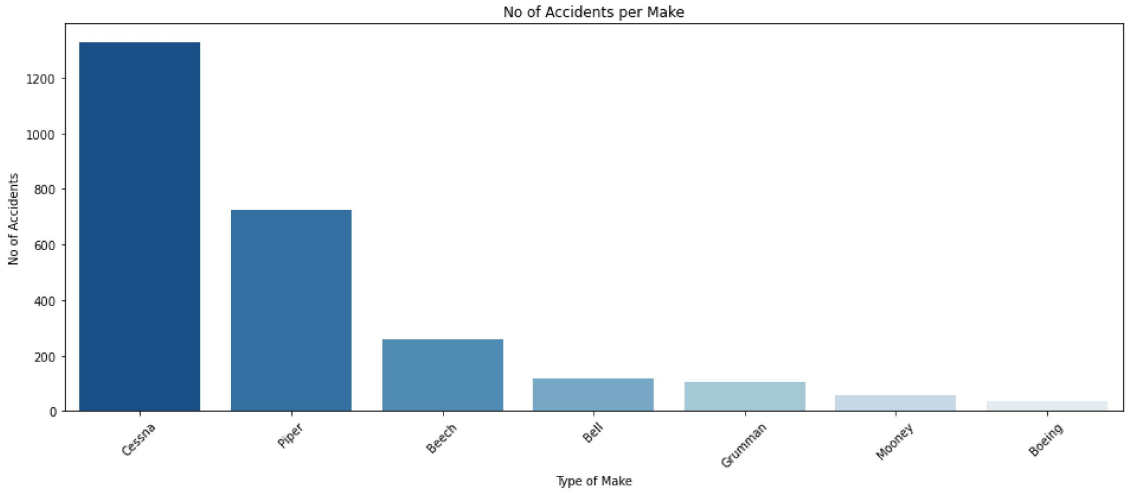
In [58]:
```
#plotting countries with no of accidents
countries_df = ntsb_main_df['country'].value_counts().reset_index().iloc[:
plt.figure(figsize=(16,6))
sns.barplot(x=countries_df['index'],y=countries_df['country'])
plt.xlabel('Countries')
plt.xticks(rotation=90)
plt.ylabel('No of Accidents')
plt.title("No of Accidents per Country")
plt.show()
#Observation: US is leading in accidents
```



In [60]:
```
#plotting aircraft category with no of accidents
aircraft_category_df = ntsb_main_df['aircraft_category'].value_counts().re
plt.figure(figsize=(16,6))
sns.barplot(x=aircraft_category_df['index'],y=aircraft_category_df['aircra
plt.xlabel('Aircraft Category')
plt.xticks(rotation=90)
plt.ylabel('No of Accidents')
plt.title("No of Accidents per Aircraft Category")
plt.show()
#Observation: airplane category has most accidents
```

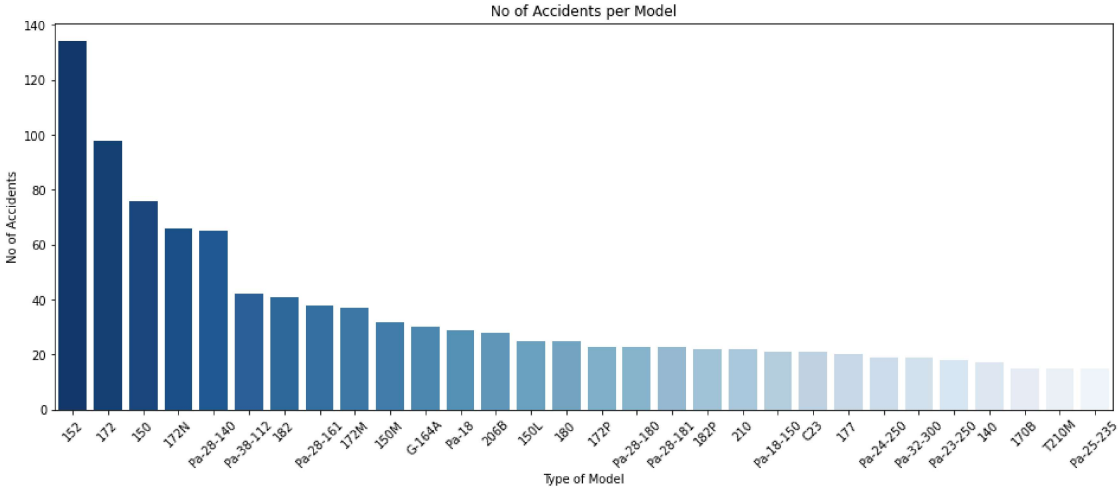


In [61]:
```
#plotting make with no of accidents
make_df = ntsb_main_df['make'].value_counts().reset_index()
plt.figure(figsize=(16,6))
sns.barplot(x=make_df['index'],y=make_df['make'],palette='Blues_r')
plt.xlabel('Type of Make')
plt.xticks(rotation=45)
plt.ylabel('No of Accidents')
plt.title("No of Accidents per Make")
plt.savefig('No_of_Accidents_per_Make.png')
plt.show()
```



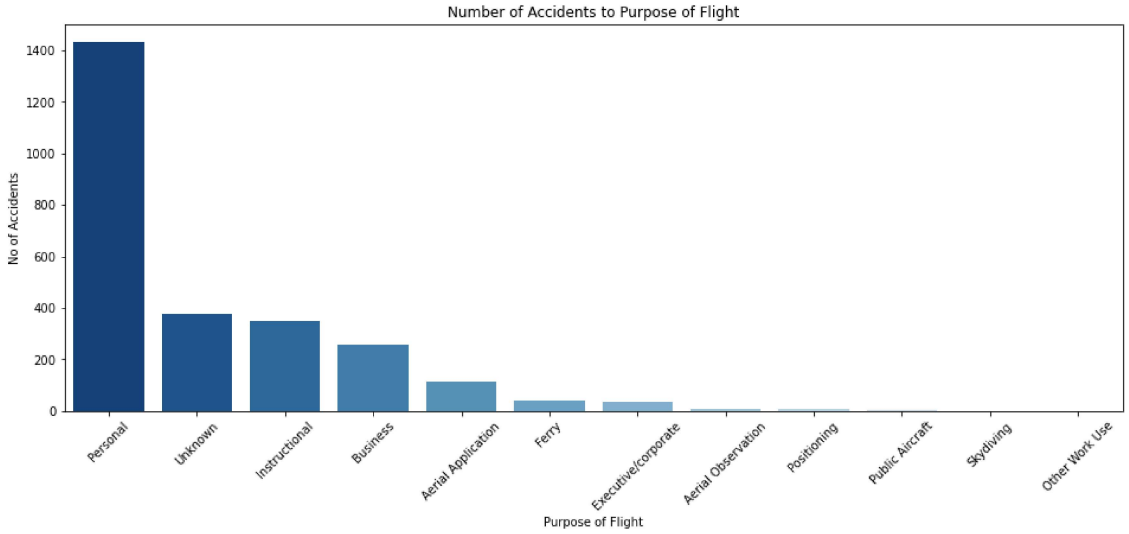
Observation: Cessna has most accidents while Boeing has least

In [62]:
```
#plotting model with no of accidents
model_df = ntsb_main_df['model'].value_counts().reset_index().iloc[:30]
plt.figure(figsize=(16,6))
sns.barplot(x=model_df['index'],y=model_df['model'],palette='Blues_r')
plt.xlabel('Type of Model')
plt.xticks(rotation=45)
plt.ylabel('No of Accidents')
plt.title("No of Accidents per Model")
plt.savefig('No_of_Accidents_per_Model.png')
plt.show()
```



Observation: 152 is the most common model from all models

In [63]:
```
#comparison btw number of engines and purpose of flight
plt.figure(figsize=(16,6))
purpose_count =  ntsb_main_df['purpose_of_flight'].value_counts()
sns.countplot(x=ntsb_main_df['purpose_of_flight'],order=purpose_count.inde
plt.ylabel('No of Accidents')
plt.xticks(rotation=45)
plt.xlabel('Purpose of Flight')
plt.title("Number of Accidents to Purpose of Flight")
plt.show()

#Observation: Personal, Unknown and Instructionsl are  most common purpose
```

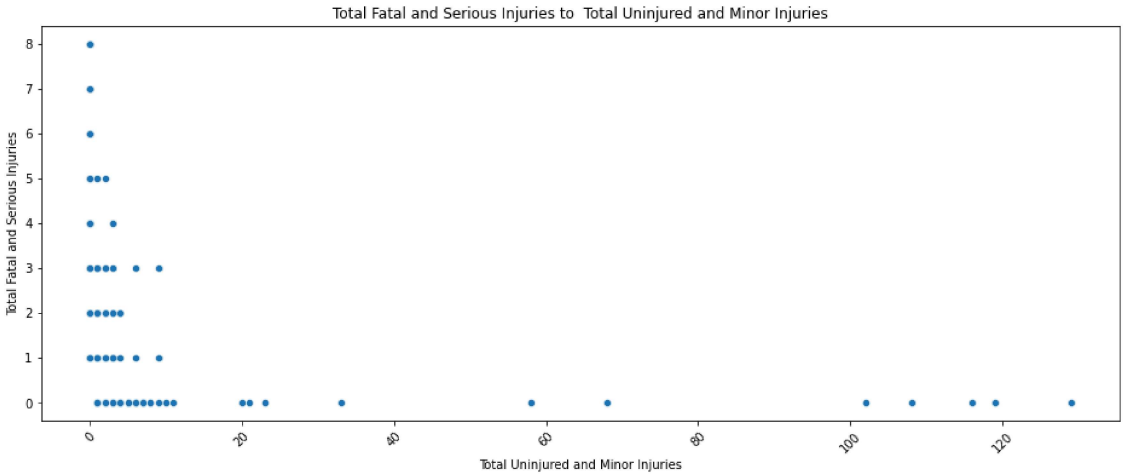


Summary

The country to look into is United States and how top make and their respectively models and their aircraft category accessed to their potential risks. And which specific purpose of flight has lower risk.

Bivariate Analysis

In [64]:
```
#type of make and country
plt.figure(figsize=(16,6))
sns.scatterplot(x=ntsb_main_df['total_uninjured']+ntsb_main_df['total_mino
plt.xlabel('Total Uninjured and Minor Injuries')
plt.xticks(rotation=45)
plt.ylabel('Total Fatal and Serious Injuries')
plt.title("Total Fatal and Serious Injuries to  Total Uninjured and Minor
plt.show()
```

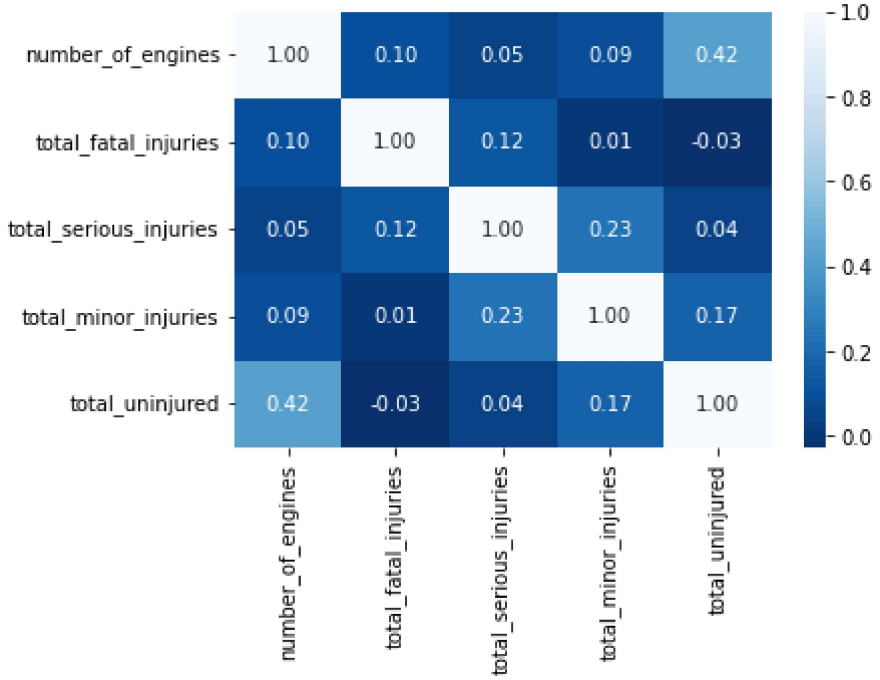


In [65]:
```
ntsb_main_df['total_fatal_injuries'].unique()
```

Out[65]:
```
array([0., 1., 2., 3., 8., 4., 6., 5., 7.])
```

In [66]:
```
#correlation of numeric columns
corr = numeric_df.corr()
sns.heatmap(corr,annot=True,fmt=".2f", cmap="Blues_r")
```

Out[66]: <AxesSubplot:>



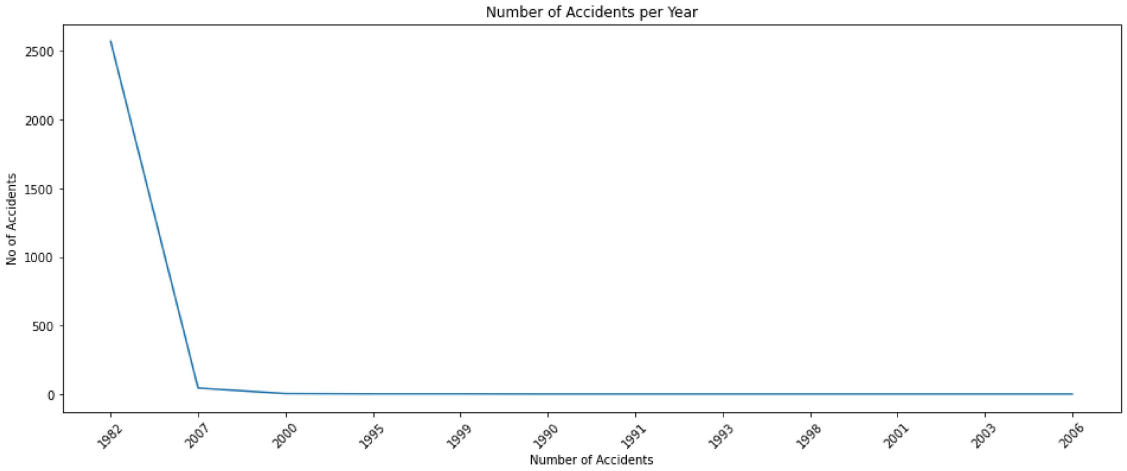
Observation: number_of_engines and 'total_injured' have positive correlation

In [67]:
```
year_df =  ntsb_main_df['year'].value_counts().reset_index()
year_df
```

Out[67]:

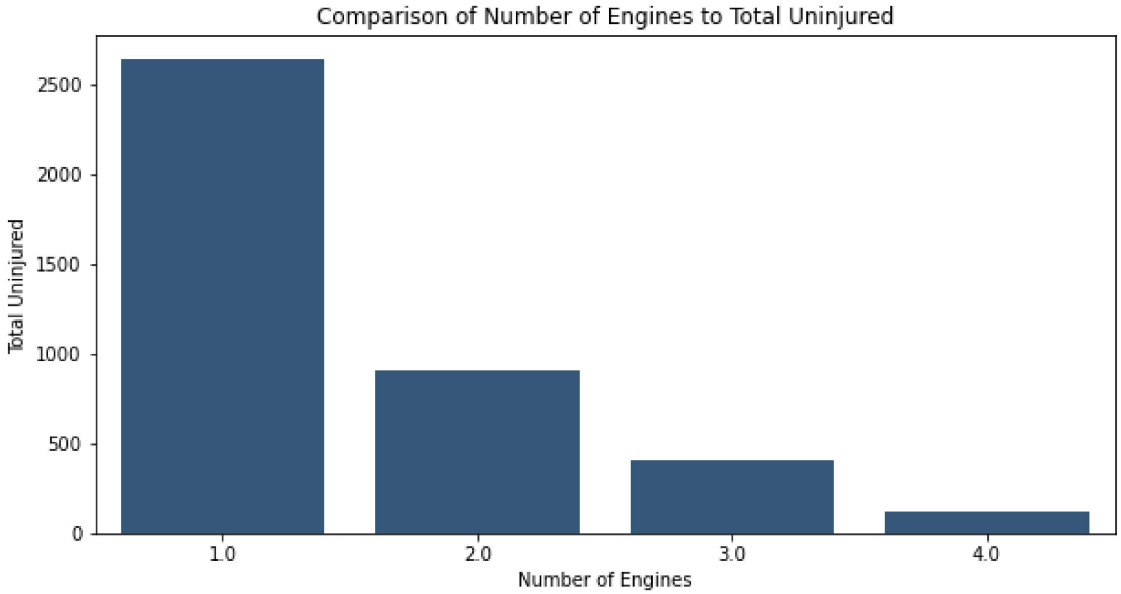
	index	year
0	1982	2570
1	2007	44
2	2000	4
3	1995	2
4	1999	2
5	1990	1
6	1991	1
7	1993	1
8	1998	1
9	2001	1
10	2003	1
11	2006	1

In [68]:
```
#comparison btw number of accidents and purpose of flight
plt.figure(figsize=(16,6))
year_df =  ntsb_main_df['year'].value_counts().reset_index()
sns.lineplot(x=year_df['index'],y=year_df['year'])
plt.ylabel('No of Accidents')
plt.xticks(rotation=45)
plt.xlabel('Number of Accidents')
plt.title("Number of Accidents per Year")
plt.show()
```

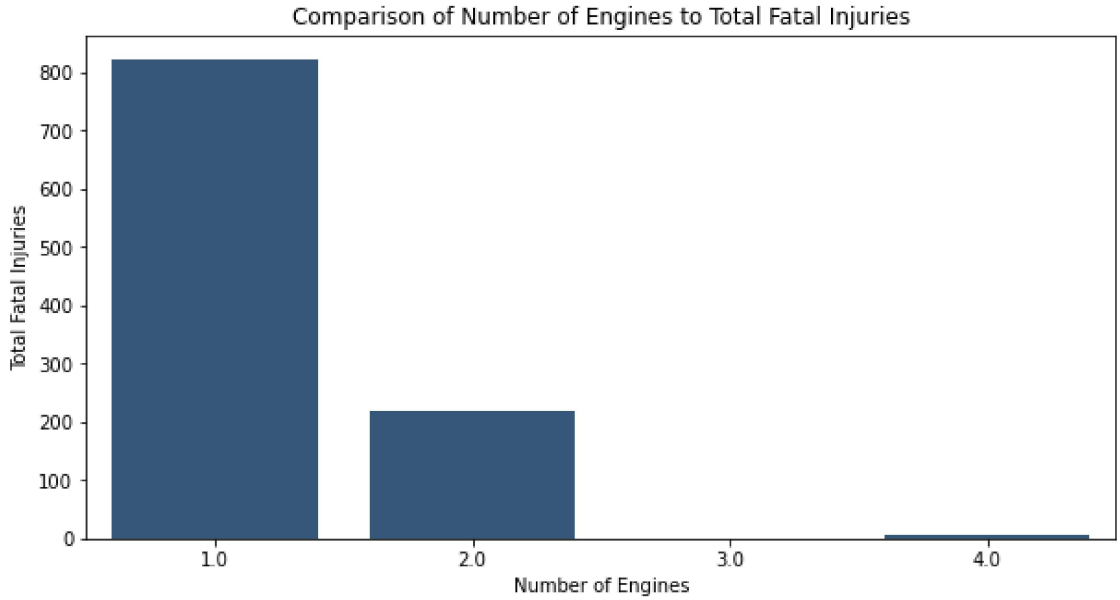


Observation: There is be decline of accidents over the years

In [69]:
```
# Creating pivot table to aggregate total_uninjured by number_of_engines
engine_uninjured = ntsb_main_df.pivot_table(index='number_of_engines', val

# Plotting data
plt.figure(figsize=(10, 5))
sns.barplot(x=engine_uninjured.index, y=engine_uninjured['total_uninjured'
plt.xlabel('Number of Engines')
plt.ylabel('Total Uninjured')
plt.title('Comparison of Number of Engines to Total Uninjured');
```



In [70]:
```
# Creating pivot table to aggregate total_uninjured by number_of_engines
engine_fatal = ntsb_main_df.pivot_table(index='number_of_engines', values=

# Plotting data
plt.figure(figsize=(10, 5))
sns.barplot(x=engine_fatal.index, y=engine_fatal['total_fatal_injuries'],
plt.xlabel('Number of Engines')
plt.ylabel('Total Fatal Injuries')
plt.title('Comparison of Number of Engines to Total Fatal Injuries')
plt.savefig('No_of_Engines_to_Total_Fatal_Injuries.png')
```

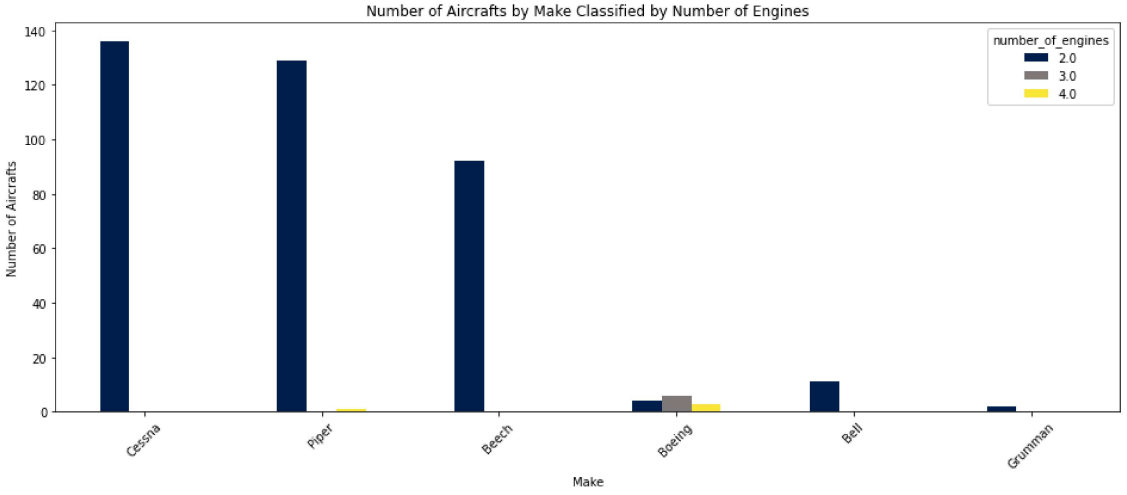


Observation: Aircraft with `number_of_engines` = 1 is high risk

Multivariate Analysis

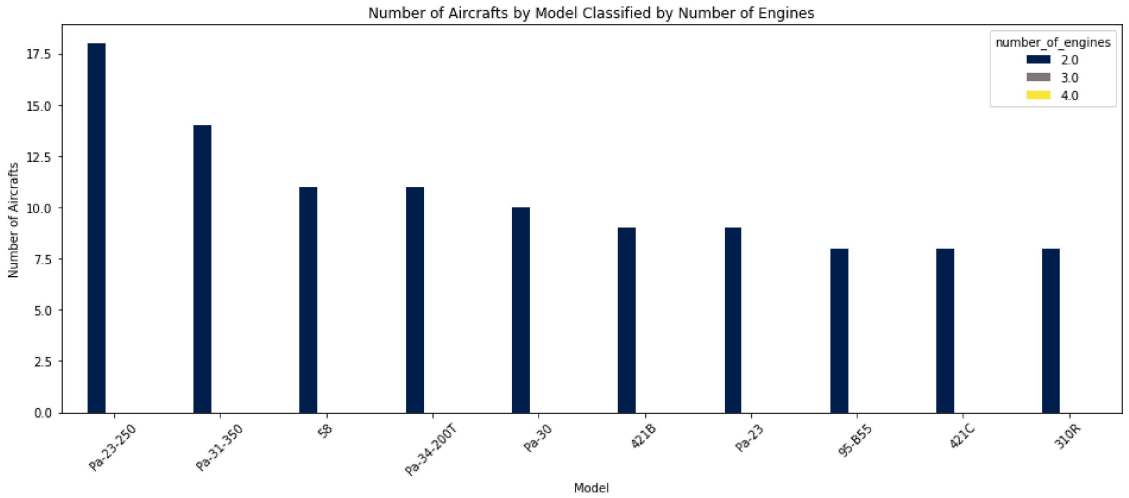
In [71]:

```
# Creating pivot table by count of number_of_engines by make
make_engine = ntsb_main_df[ ntsb_main_df.number_of_engines > 1].pivot_tabl

#Creating total column for each row to help sort
make_engine['total'] = make_engine.sum(axis=1)
make_engine = make_engine.sort_values('total', ascending=False)
plt.figure(figsize=(16, 6))
make_engine.drop('total', axis=1).plot(kind='bar', stacked=False, ax=plt.g

# Plotting the data
plt.ylabel('Number of Aircrafts')
plt.xlabel('Make')
plt.xticks(rotation=45)
plt.title("Number of Aircrafts by Make Classified by Number of Engines")
plt.savefig('Number_of_Aircrafts_per_Make.png')
plt.show()
```



Observation: Aircraft with `make` recommended is either Cessna, Piper or Beech

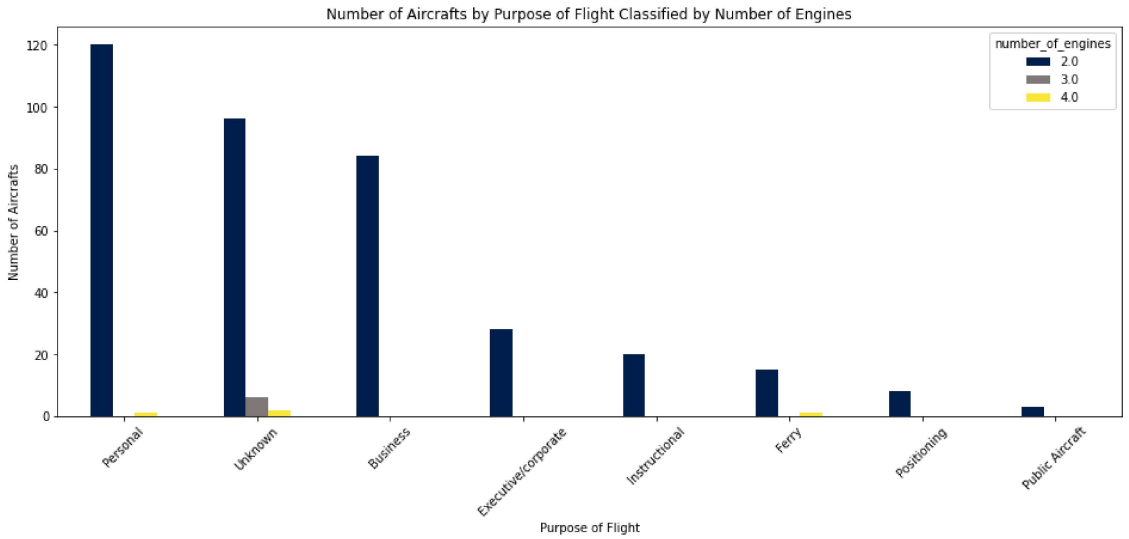
In [72]:
```
# Creating pivot table by count of number_of_engines by model
model_engine = ntsb_main_df[ ntsb_main_df.number_of_engines > 1].pivot_tab

#Creating total column for each row to help sort
model_engine['total'] = model_engine.sum(axis=1)
model_engine = model_engine.sort_values('total', ascending=False).iloc[:10
plt.figure(figsize=(16, 6))
model_engine.drop('total', axis=1).plot(kind='bar', stacked=False, ax=plt.

# Plotting the data
plt.ylabel('Number of Aircrafts')
plt.xlabel('Model')
plt.xticks(rotation=45)
plt.title("Number of Aircrafts by Model Classified by Number of Engines")
plt.show()
```



Observation: Aircraft `model` recommended is Pa=23-250, Pa-31-350 and 58

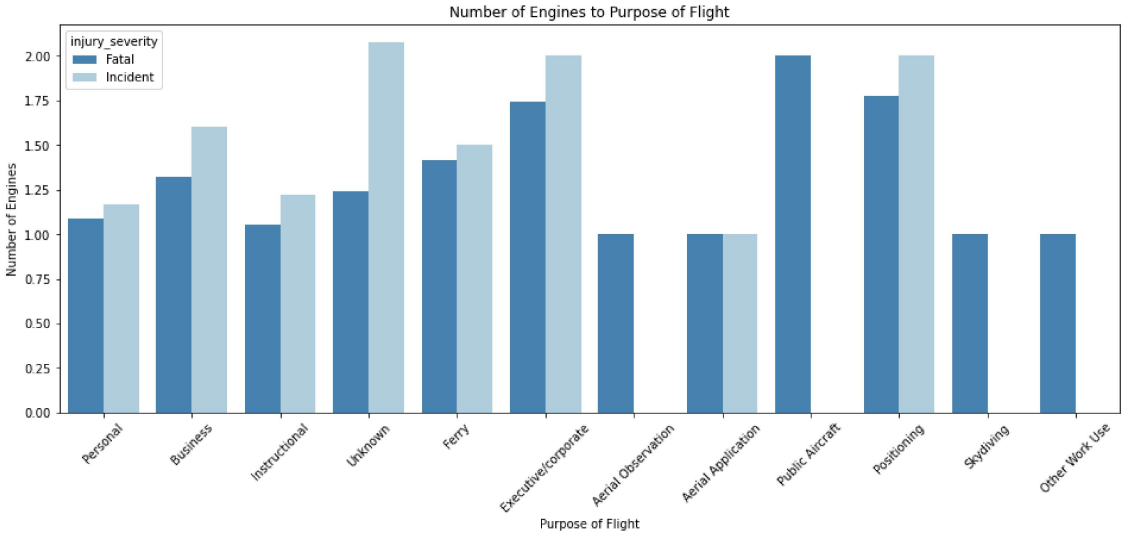
In [73]:

```
# Creating pivot table by count of number_of_engines by purpose_of_flight
model_engine = ntsb_main_df[ ntsb_main_df.number_of_engines > 1].pivot_tab

#Creating total column for each row to help sort
model_engine['total'] = model_engine.sum(axis=1)
model_engine = model_engine.sort_values('total', ascending=False).iloc[:10
plt.figure(figsize=(16, 6))
model_engine.drop('total', axis=1).plot(kind='bar', stacked=False, ax=plt.

# Plotting the data
# Plotting the data
plt.ylabel('Number of Aircrafts')
plt.xlabel('Purpose of Flight')
plt.xticks(rotation=45)
plt.title("Number of Aircrafts by Purpose of Flight Classified by Number o
plt.show()
```



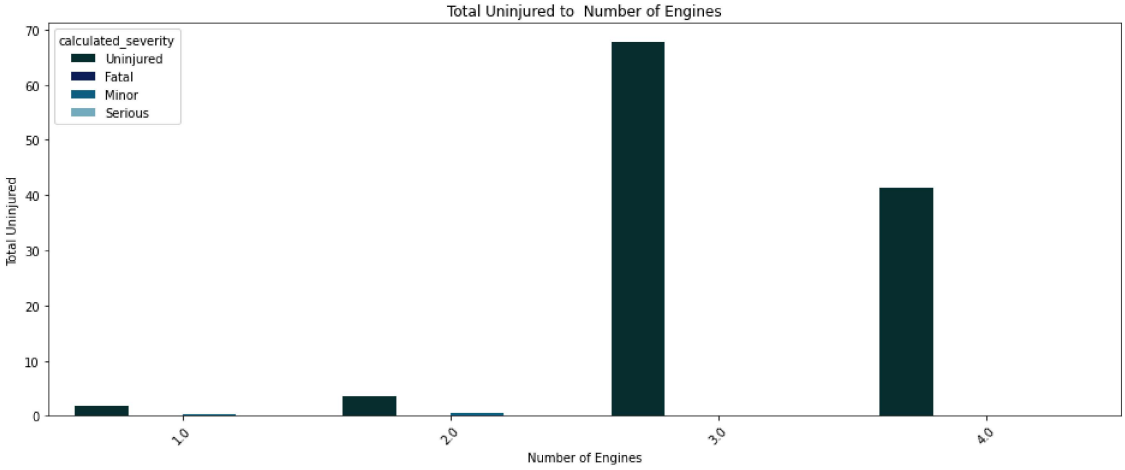
Observation: Aircraft for Personal, Unknown, Business dominates with `number_of_engines` greater than 1

In [74]:
```
#comparison btw number of engines, purpose of flight and severity
plt.figure(figsize=(16,6))
sns.barplot(y=ntsb_main_df['number_of_engines'],x=ntsb_main_df['purpose_of
plt.ylabel('Number of Engines')
plt.xticks(rotation=45)
plt.xlabel('Purpose of Flight')
plt.title("Number of Engines to Purpose of Flight")
plt.show()
```



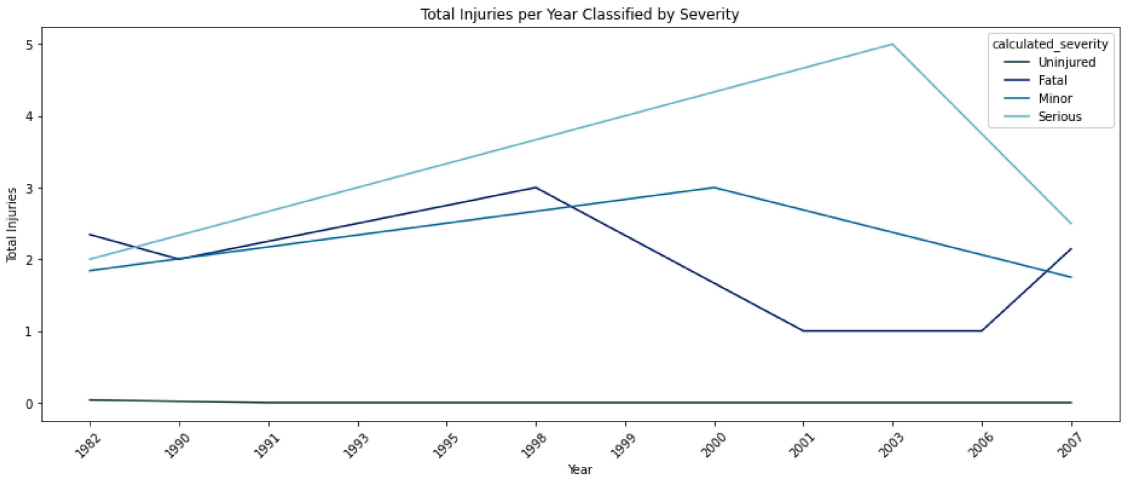
Observation: Aircraft for Personal, Unknown, Business,Instructional, Ferry, Executive/Corporate,Positioning has `severity` for incident greater than fatal

In [75]:
```
#comparison btw number of engines, total injured  and severity
plt.figure(figsize=(16,6))
sns.barplot(x=ntsb_main_df['number_of_engines'],y=ntsb_main_df['total_unin
plt.ylabel('Total Uninjured')
plt.xticks(rotation=45)
plt.xlabel('Number of Engines')
plt.title("Total Uninjured to  Number of Engines")
plt.savefig('Total_Uninjured_to_No_of_Engines.png')
plt.show()
```



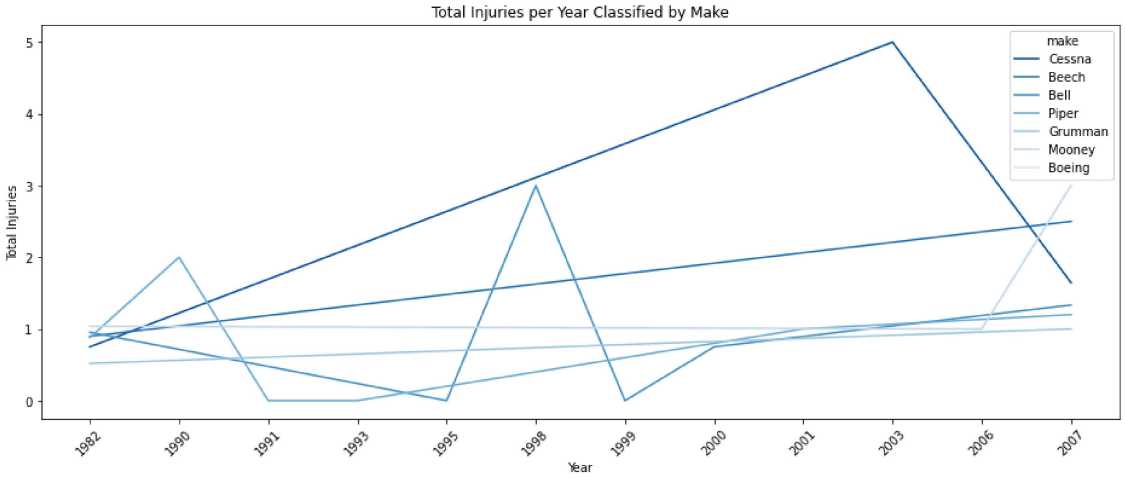
Observation: Engines with 2 injuries caused less harm

In [76]:
```
#comparison btw year, total injured and injury_severity
plt.figure(figsize=(16,6))
total_injuries = ntsb_main_df['total_fatal_injuries']+ntsb_main_df['total_
sns.lineplot(x=ntsb_main_df['year'],y=total_injuries, hue=ntsb_main_df['ca
plt.ylabel('Total Injuries')
plt.xticks(rotation=45)
plt.xlabel('Year')
plt.title("Total Injuries per Year Classified by Severity")
plt.savefig('Total_Injuries_per_Year.png')
plt.show()
```



Observation There is increase of accidents over the years with severity getting worse

In [77]:
```
#comparison btw year, total injured and injury_severity
plt.figure(figsize=(16,6))
total_injuries = ntsb_main_df['total_fatal_injuries']+ntsb_main_df['total_
sns.lineplot(x=ntsb_main_df['year'],y=total_injuries, hue=ntsb_main_df['ma
plt.ylabel('Total Injuries')
plt.xticks(rotation=45)
plt.xlabel('Year')
plt.title("Total Injuries per Year Classified by Make")
plt.savefig('Total_Injuries_per_Year_by_make.png')
plt.show()
```



Observation: There is increase of accidents for all the aircraft make. Cessna has however from around 2003 has experienced a decline.