jtjohn172 / aviation-industry-risk-analysis-project Public

Code Issues Pull requests Actions Projects Wiki Security Insights Se

main aviation-industry-risk-analysis-project / Final Notebook.ipynb

Go to file t

jtjohn172 Final notebook completed 22 minutes ago

2943 lines (2943 loc) · 780 KB

Preview Code Blame Raw

Peanut Butter Inc Aviation Risk Data Analysis

Overview

This project analyzes which aircrafts have the lowest risk for Peanut Butter INC. to enter the commercial and private enterprises industry. We are making our recommendation based on 90,000 incident records over the past 70 years.

Business Problem

Peanut Butter Inc is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating airplanes for commerical and private enterprises, but do not know anything about the potential risks of aircraft. We will be analyzing the NTSB Aviation Accident data to determine which aircraft are the lowest risk, and the risk associated with operating in our South, West, Midwest, and North East regions for Peanut Butter Inc's new business endeavor.

We will define risk as loss of life,injury, and damage to aircraft

We will use this analysis to recommend:

- 1. Make and Model of Commercial Airplane based on risk
- 2. Make and Model of Private Aircraft
- 3. Risk associated with region of operation

Data Understanding

In the data folder is a dataset 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.

Importing Packages

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Project Data

In [2]:

```
df = pd.read_csv('data/Aviation_Data.csv', encoding='latin-1', low_memory=False)
```

Checking the data:

- .head()
- .tail()
- .info()

In [3]:

```
df.head()
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latit
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	36.922
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	

5 rows × 31 columns

In [4]:

```
df.tail()
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latit
90343	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	
90344	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	
90345	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	3415
90346	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	
90347	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	

5 rows × 31 columns

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90348 entries, 0 to 90347
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Event.Id                88889 non-null  object 
 1   Investigation.Type      90348 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        73659 non-null  object 
dtypes: float64(5), object(26)
memory usage: 21.4+ MB
```

In [ ]:

Data Preparation

Data Cleaning

- We dropped 10 columns because they were missing a large amounts of data and/or were not relevant for our analysis.
- We chose to Event.Date range to start on 11/19/2001 which was when TSA was established. (https://www.tsa.gov/timeline#:~:text=Jackson%2C%20who%20was%20the%20Deputy,Bush%20on%2

In [6]:

```python
# Clean column names, replacing . to _ and making them lowercase
df = df.rename(columns={c: c.lower().replace('.', '_') for c in df.columns})

# Formatting all object columns lowercase
df['make'] = df['make'].str.lower()
df['model'] = df['model'].str.lower()
df['location'] = df['location'].str.lower()
df['investigation_type'] = df['investigation_type'].str.lower()
df['country'] = df['country'].str.lower()
df['injury_severity'] = df['injury_severity'].str.lower()
df['aircraft_category'] = df['aircraft_category'].str.lower()
df['engine_type'] = df['engine_type'].str.lower()
df['amateur_built'] = df['amateur_built'].str.lower()
```

```
#dropping the columns we will not be using
df = df[['location','investigation_type','event_date','country',
         'injury_severity','aircraft_category','make',
         'model','number_of_engines','engine_type','total_fatal_injuries',
         'total_uninjured','total_serious_injuries','total_minor_injuries',
         'latitude','longitude','amateur_built','aircraft_damage']]

# Convert event_date column to datetime format
df['event_date'] = pd.to_datetime(df['event_date'])

# We will be looking at data from 2001 to 2022
df = df[df['event_date'] > '2001-11-19']

# Creating a new dataframe with data from the US
df = df[df['country'] == 'united states']

# Split location column into city and state columns + Cleaning format
df[['city', 'state']] = df['location'].str.split(', ', n=1, expand=True)
df['city'] = df['city'].str.lower()

# Droping 7 missing null values in location
df.dropna(subset=['location'], inplace=True)

# populating injury_severity based on fatalities =/or/!= 0
df.loc[(df['total_fatal_injuries'] == 0) & (df['injury_severity'].isna()), 'injury_sever
df.loc[(df['total_fatal_injuries'] != 0) & (df['injury_severity'].isna()), 'injury_sever

# Cleaning amateur_built formatting + filtering to NOT amateur built
df = df[df['amateur_built'] == 'no']

# Droping 9 missing values in Make/Model + cleaning data
df.dropna(subset=['make'], inplace=True)
df.dropna(subset=['model'], inplace=True)
df['make']=df['make'].str.replace('-', ' ')

# Adding placeholder in 'state' for missing values = 'unknown'
df['state'] = df['state'].fillna('Unknown')

# Dropping 883 missing values in number_of_engines + number_of_engines >= 1
df.dropna(subset=['number_of_engines'], inplace=True)
df = df[df['number_of_engines'] >= 1]

# Cleaning 'aircraft_category' with null values
engine_types = ['reciprocating', 'turbo prop', 'turbo fan', 'turbo jet']
df.loc[(df['aircraft_category'].isnull()) & (df['engine_type'].isin(engine_types)), 'air

# Dropping everything except 'Airplane' in engine_type
df.loc[~df['engine_type'].isin(engine_types), 'engine_type'] = np.nan
df.dropna(subset=['engine_type'], inplace=True)

# Filling missing value in total_fatal_injuries, total_serious_injuries, and total_minor
df['total_fatal_injuries'] = df['total_fatal_injuries'].fillna(0)
df['total_serious_injuries'] = df['total_serious_injuries'].fillna(0)
df['total_minor_injuries'] = df['total_minor_injuries'].fillna(0)
df['total_uninjured'] = df['total_minor_injuries'].fillna(0)
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23335 entries, 51802 to 90226
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   location                23335 non-null  object
 1   investigation_type      23335 non-null  object
 2   event_date              23335 non-null  datetime64[ns]
 3   country                 23335 non-null  object
 4   injury_severity         23335 non-null  object
```

```
 4   injury_severity         23335 non-null  object
 5   aircraft_category       23335 non-null  object
 6   make                    23335 non-null  object
 7   model                   23335 non-null  object
 8   number_of_engines       23335 non-null  float64
 9   engine_type             23335 non-null  object
 10  total_fatal_injuries    23335 non-null  float64
 11  total_uninjured         23335 non-null  float64
 12  total_serious_injuries  23335 non-null  float64
 13  total_minor_injuries    23335 non-null  float64
 14  latitude                22912 non-null  object
 15  longitude               22904 non-null  object
 16  amateur_built           23335 non-null  object
 17  aircraft_damage         22842 non-null  object
 18  city                    23335 non-null  object
 19  state                   23335 non-null  object
dtypes: datetime64[ns](1), float64(5), object(14)
memory usage: 3.7+ MB
```

We mapped the state with its assocaited region for visualizations

In [8]:

```python
#Creating a dictionary of state abbreviation and their corresponding region with all low
state_region_dict = {'ct': 'North East', 'de': 'North East', 'me': 'North East', 'md': '
                     'il': 'Midwest', 'in': 'Midwest', 'ia': 'Midwest', 'ks': 'Midwest',
                     'mo': 'Midwest', 'ne': 'Midwest', 'nd': 'Midwest', 'oh': 'Midwest',
                     'al': 'South', 'ar': 'South', 'fl': 'South', 'ga': 'South', 'ky': '
                     'nc': 'South', 'ok': 'South', 'sc': 'South', 'tn': 'South', 'tx': '
                     'ak': 'West', 'az': 'West', 'ca': 'West', 'co': 'West', 'hi': 'West

# Add a new column 'region' to the dataframe and map the state to its corresponding regi
df['region'] = df['state'].map(state_region_dict)

#Only including the 50 United States
df[df['state'].isin(state_region_dict.keys())]
```

Out[8]:

	location	investigation_type	event_date	country	injury_severity	aircraft_category	m
51802	fairhope, al	accident	2001-11-20	united states	non-fatal	airplane	ces
51803	stuart, fl	accident	2001-11-20	united states	non-fatal	airplane	ces
51804	evans, ga	accident	2001-11-20	united states	non-fatal	airplane	p
51805	crystal river, fl	accident	2001-11-20	united states	non-fatal	airplane	ces
51806	memphis, tn	accident	2001-11-20	united states	non-fatal	airplane	boe
...	...	...	...	...	...	...	
90089	navasota, tx	accident	2022-10-05	united states	non-fatal	airplane	ces
90098	iola, tx	accident	2022-10-06	united states	non-fatal	airplane	ces
90106	dacula, ga	accident	2022-10-08	united states	non-fatal	airplane	ces
90120	ardmore, ok	accident	2022-10-13	united states	non-fatal	airplane	be
90226	bridgeport, tx	accident	2022-11-09	united states	non-fatal	airplane	luscor

23186 rows × 21 columns

Private Plane Risk Assessment

In this portion of the project we will define private flights as those which carry less than 20 passengers. This number comes from our independent research of the difference between commercial and private flights.(https://www.internationaljet.com/how-many-passengers-can-a-private-jethold.html#:~:text=Similar%20to%20commercial%20planes%2C%20large,flights%20seat%20closer%20t

In [9]:

```
#create a new column for the estimated total number of passengers on board each flight
df['passengers'] = df['total_uninjured']+df['total_minor_injuries']+df['total_serious_in
```

In [10]:

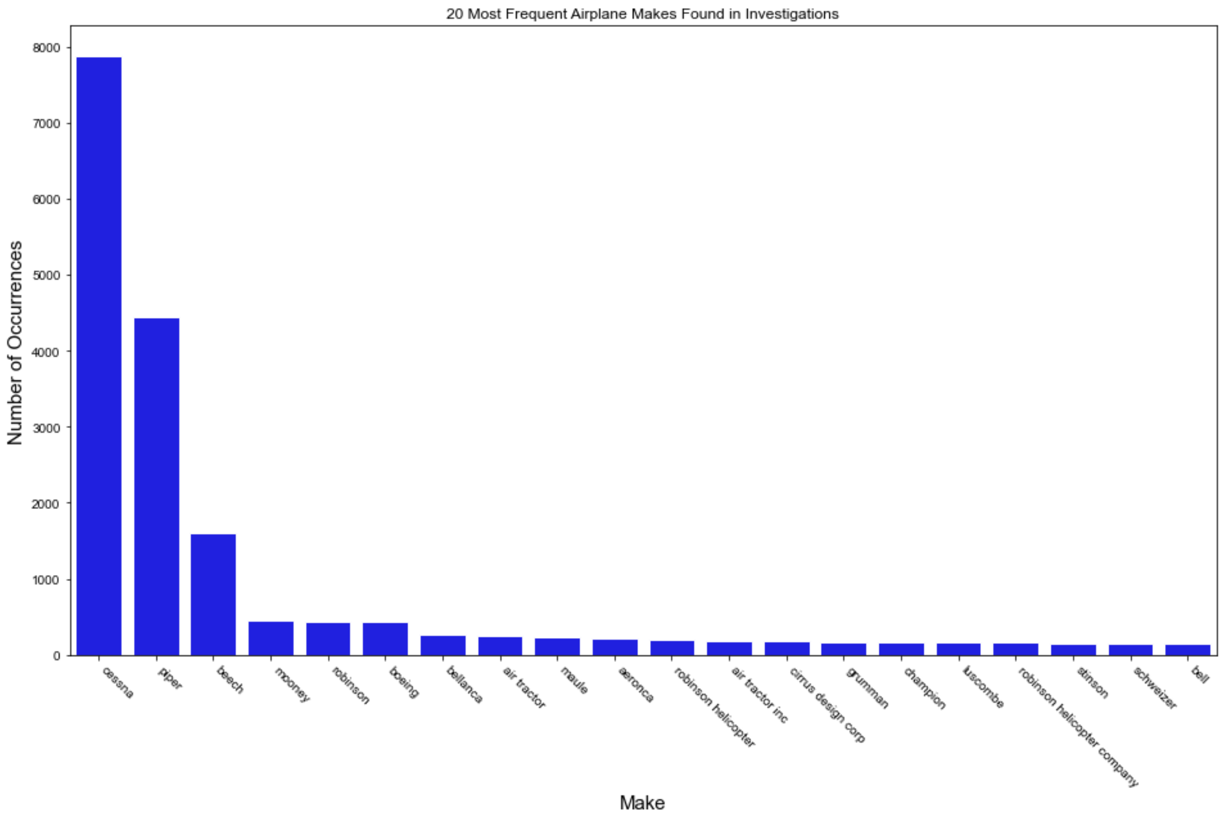
```
#create a new dataframe focusing on private planes
private_planes = df.loc[df['passengers'] <20]
```

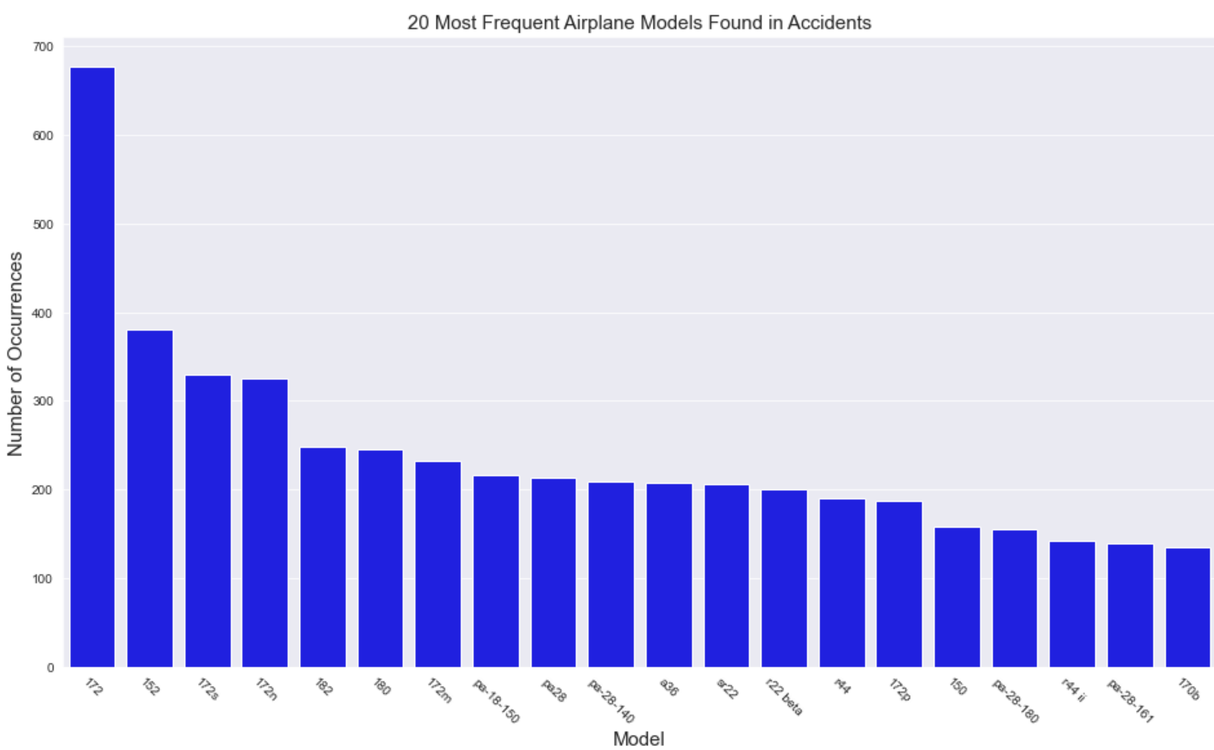
We now have a subset of the data that focuses on planes estimated to be carrying less than 20 passengers. We will examine the distributions of make and model to determine what recommendations should be made.

Visualization

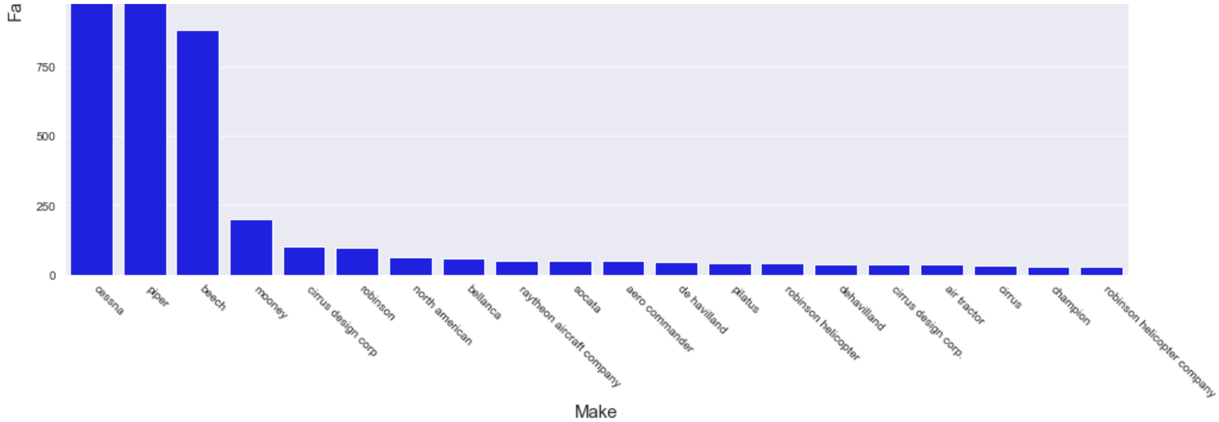
In [11]:

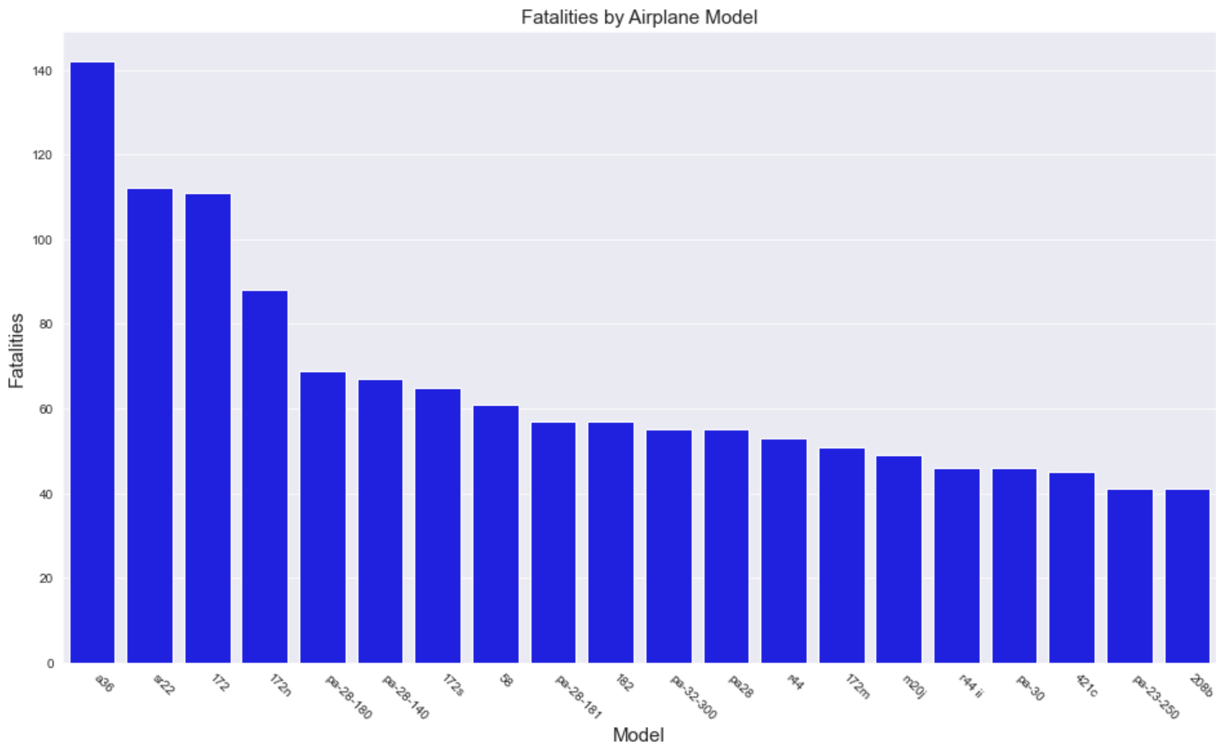
```
#Creating a figure showing investigation occurance by Private Airplane Make
fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
makes=sns.barplot(data=private_planes, x=private_planes['make'].value_counts().index[:20
makes.set_title('20 Most Frequent Airplane Makes Found in Investigations')
makes.set_xlabel('Make', fontsize=15)
makes.set_ylabel('Number of Occurrences', fontsize=15)
makes.set_xticklabels(private_planes['make'].value_counts().index[:20], rotation=-45, ha

#Creating a figure showing investigation frequency by Private Airplane Model
fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
makes=sns.barplot(data=private_planes, x=private_planes['model'].value_counts().index[:2
makes.set_title('20 Most Frequent Airplane Models Found in Accidents', fontsize=15)
makes.set_xlabel('Model', fontsize=15)
makes.set_ylabel('Number of Occurrences', fontsize=15)
makes.set_xticklabels(private_planes['model'].value_counts().index[:20], rotation=-45, h

#Showing fatalities by Private Airplane Make
fatalities=private_planes.groupby('make')['total_fatal_injuries'].sum().sort_values(asce

fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
f=sns.barplot(data=private_planes, x=fatalities.index[:20], y=fatalities.values[:20], co
f.set_title('Fatalities by Airplane Make', fontsize=15)
f.set_xlabel('Make', fontsize=15)
f.set_ylabel('Fatalities', fontsize=15)
f.set_xticklabels(fatalities.index[:20], rotation=-45, ha='left');

#Showing fatalities by Private Airplane Model
fatalities=private_planes.groupby('model')['total_fatal_injuries'].sum().sort_values(asc

fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
f=sns.barplot(data=private_planes, x=fatalities.index[:20], y=fatalities.values[:20], co
f.set_title('Fatalities by Airplane Model', fontsize=15)
f.set_xlabel('Model', fontsize=15)
f.set_ylabel('Fatalities', fontsize=15)
f.set_xticklabels(fatalities.index[:20], rotation=-45, ha='left');
```


20 Most Frequent Airplane Makes Found in Investigations
Number of Occurrences
0
1000
2000
3000
4000
5000
6000
7000
8000
cessna
piper
beech
mooney
robinson
boeing
bellanca
air tractor
maule
aeronca
robinson helicopter
air tractor inc
cirrus design corp
grumman
champion
luscombe
robinson helicopter company
stinson
schweizer
bell
Make


20 Most Frequent Airplane Models Found in Accidents
Number of Occurrences
0
100
200
300
400
500
600
700
172
152
172s
172n
182
180
172m
pa-18-150
pa28
pa-28-140
a36
sr22
r22 beta
r44
172p
150
pa-28-180
r44 ii
pa-28-161
170b
Model


Fatalities by Airplane Make
2000
1750
1500
1250
1000





We can see that although Cessna has nearly twice as many investigations as Piper, the difference in fatalities is less stark. As for models, the 172 was the most common model investigated but accounted for only the 3rd most deaths. The A36 model accounted for the most fatalities, despite being only the 9th most common model involved in investigations.

Ultimately a plane cannot be several types of makes and models -it can only be one. Therefore the next step in our analysis will be to combine make and model into one column, and use this column to make our final recommendations. Specifically, we will look to see which models have the lowest percentage of deaths and injuries out of their total passengers.

We will look at all planes which flew more than 100 passengers total (to ensure we have a significant sample size). This comes out to 91 total make/models. From these 91 we will select those which tend to be the safest. The strategy will be to examine the lowest 20 death rates, lowest 20 serious injury rates, and lowest 20 minor injury rates, and then see which planes appear in all 3. We will also see which of those planes tended to have less damage to the aircraft.

In [12]:

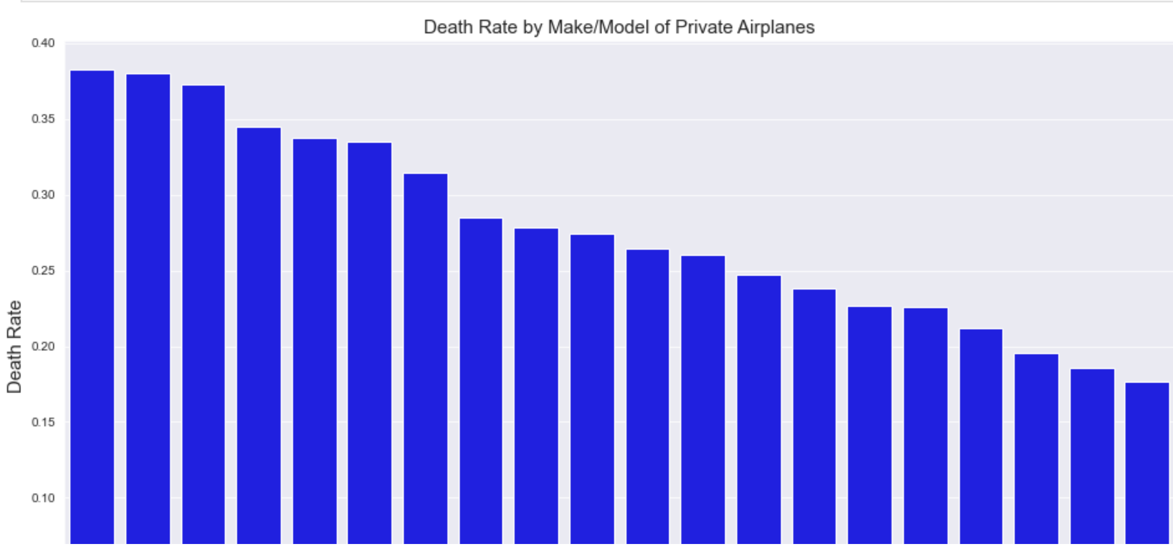
```
#creating a new column combining make and model
private_planes['plane']=private_planes['make'].str[0:] + ' ' + private_planes['model'].s

#grouping the data by planes which carried over 100 passengers in total
most_common_private_planes=private_planes.groupby('plane')['passengers'].sum().sort_valu
```

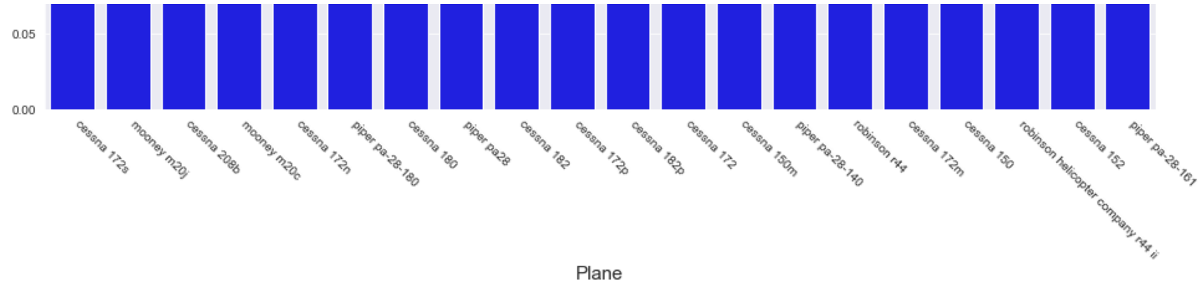
```
top_private_planes=private_planes.loc[private_planes['plane'].isin(most_common_private_p

#finding the fatality rate for each plane
death_rates=top_private_planes.groupby('plane')['total_fatal_injuries'].sum()/top_privat
death_rates=death_rates.sort_values(ascending=False)

#finding the serious rate for each plane
serious_injury_rates=top_private_planes.groupby('plane')['total_serious_injuries'].sum()
serious_injury_rates=serious_injury_rates.sort_values(ascending=False)

#finding the minor rate for each plane
minor_injury_rates=top_private_planes.groupby('plane')['total_minor_injuries'].sum()/top
minor_injury_rates=serious_injury_rates.sort_values(ascending=False)
```

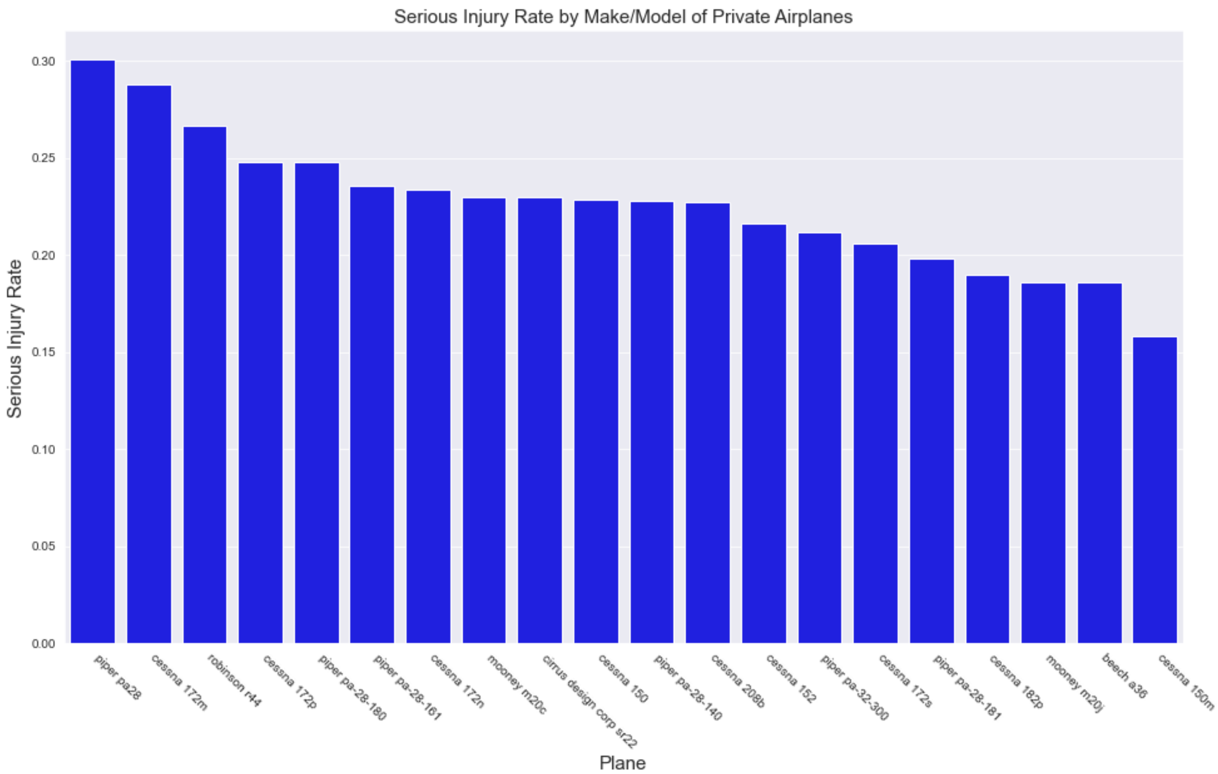
```
<ipython-input-12-c6fa3f7eeaa8>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  private_planes['plane']=private_planes['make'].str[0:] + ' ' + private_planes['model'].s
tr[0:]
```

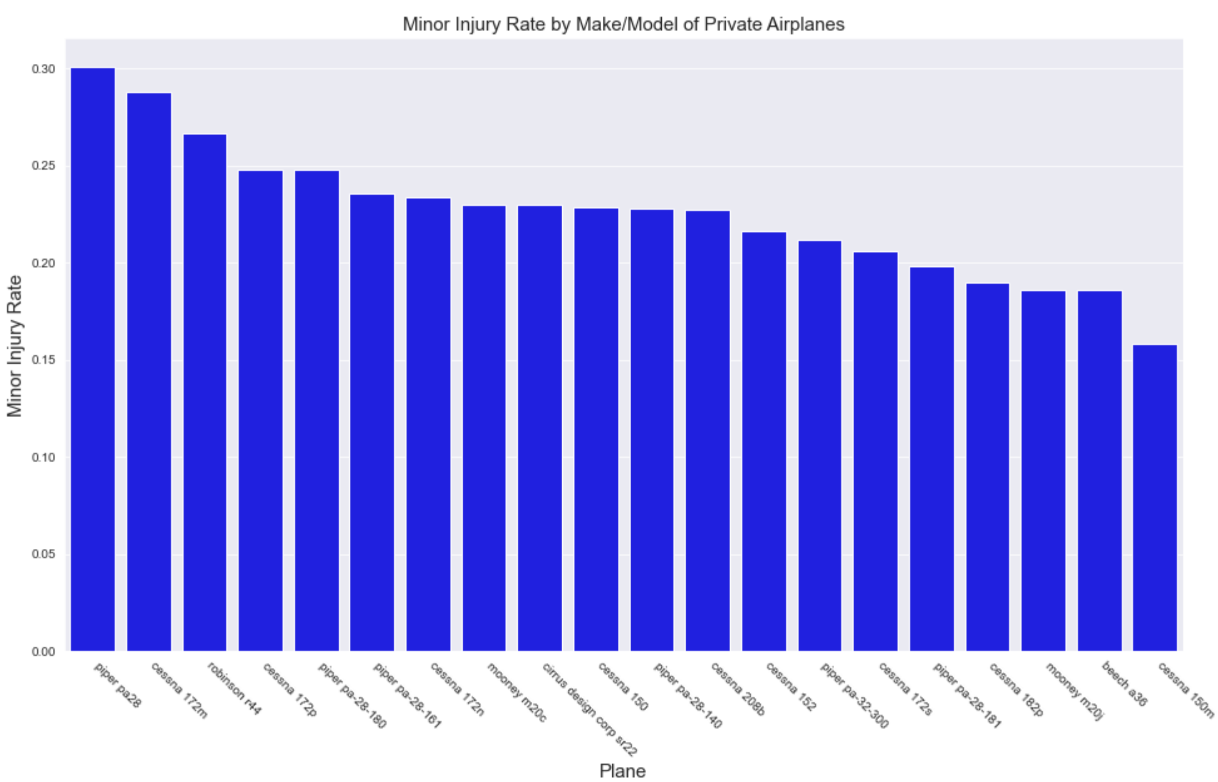
In [13]:

```
#Viewing lowest death rates by airplane make/model
fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
f=sns.barplot(data=top_private_planes, x=death_rates.index[-20:], y=death_rates.values[-
f.set_title('Death Rate by Make/Model of Private Airplanes', fontsize=15)
f.set_xlabel('Plane', fontsize=15)
f.set_ylabel('Death Rate', fontsize=15)
f.set_xticklabels(death_rates.index[-20:], rotation=-45, ha='left');

#Viewing lowest Serious injury rates by airplane make/model
fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
f=sns.barplot(data=top_private_planes, x=serious_injury_rates.index[-20:], y=serious_inj
f.set_title('Serious Injury Rate by Make/Model of Private Airplanes', fontsize=15)
f.set_xlabel('Plane', fontsize=15)
f.set_ylabel('Serious Injury Rate', fontsize=15)
f.set_xticklabels(serious_injury_rates.index[-20:], rotation=-45, ha='left');

#Viewing Lowest Minor injury rates by airplane make/model
fig, ax=plt.subplots(figsize=(16,9))
sns.set_style('darkgrid')
f=sns.barplot(data=top_private_planes, x=minor_injury_rates.index[-20:], y=minor_injury_
f.set_title('Minor Injury Rate by Make/Model of Private Airplanes', fontsize=15)
f.set_xlabel('Plane', fontsize=15)
f.set_ylabel('Minor Injury Rate', fontsize=15)
f.set_xticklabels(minor_injury_rates.index[-20:], rotation=-45, ha='left');
```









Next we will look at the breakdown of damage to the airplane. We will create binary columns to represent the level of damage.

In [14]:
```
top_private_planes['aircraft_damage'].value_counts()
```

Out[14]:
```
Substantial    4206
Destroyed       405
Minor            42
Unknown           1
Name: aircraft_damage, dtype: int64
```

In [15]:
```
#creating binary columns to represent the level of damage
top_private_planes['substantial']=top_private_planes['aircraft_damage'].apply(lambda x:
top_private_planes['destroyed']=top_private_planes['aircraft_damage'].apply(lambda x: 1
top_private_planes['minor']=top_private_planes['aircraft_damage'].apply(lambda x: 1 if x

#grouping the data to look at which planes experienced which level of damage
substantial_damage=top_private_planes.groupby('plane')['substantial'].sum()
destroyed=top_private_planes.groupby('plane')['destroyed'].sum().sort_values(ascending=F
minor_damage=top_private_planes.groupby('plane')['minor'].sum().sort_values(ascending=Fa
```

```
<ipython-input-15-eb8040da91a5>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  top_private_planes['substantial']=top_private_planes['aircraft_damage'].apply(lambda x:
1 if x=='Substantial' else 0)
<ipython-input-15-eb8040da91a5>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  top_private_planes['destroyed']=top_private_planes['aircraft_damage'].apply(lambda x: 1
if x=='Destroyed' else 0)
<ipython-input-15-eb8040da91a5>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  top_private_planes['minor']=top_private_planes['aircraft_damage'].apply(lambda x: 1 if x
=='Minor' else 0)
```

In [16]:
```
#creating a table to show the breakdown of damage by each type of plane
minor_damage_table=pd.merge(minor_damage, top_private_planes['plane'].value_counts(), le
substantial_damage_table=pd.merge(substantial_damage, top_private_planes['plane'].value_
destroyed_table=pd.merge(destroyed, top_private_planes['plane'].value_counts(), left_ind

damage_table=minor_damage_table.merge(substantial_damage_table, how='inner', left_index=
damage_table=damage_table.rename({'plane': 'planes'}, axis=1)
damage_table=damage_table.drop(columns=['plane_x', 'plane_y'], axis=1)

damage_table['unknown']=damage_table['planes']-(damage_table['minor']+damage_table['subs
damage_table
```

Out[16]:

	minor	substantial	destroyed	planes	unknown
cirrus design corp sr22	6	88	18	113	1
cessna 152	4	354	20	381	3
piper pa-28-181	4	111	11	126	0
cessna 172s	4	298	22	330	6
cessna 182	3	222	22	248	1

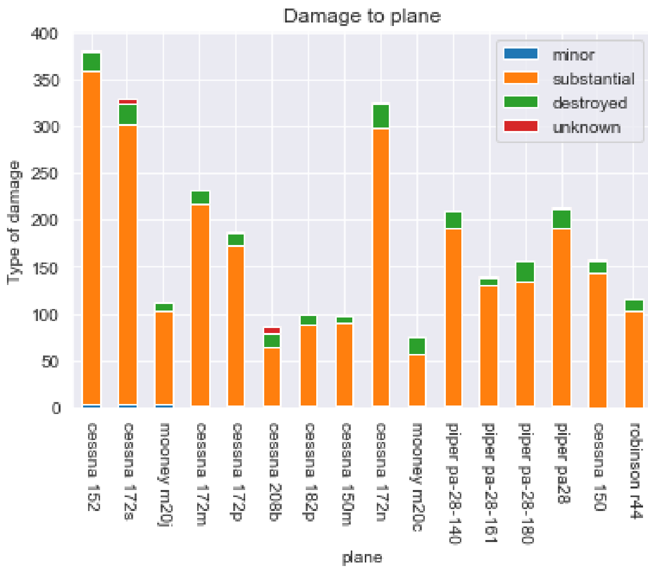
cessna 172	3	635	33	676	5
mooney m20j	3	99	11	113	0
cessna 172m	2	215	14	232	1
cessna 172p	2	171	13	187	1
cessna 208b	2	63	14	86	7
cessna 182p	1	87	12	100	0
cessna 150m	1	89	8	98	0
cessna 172n	1	297	25	325	2
beech a36	1	140	37	180	2
mooney m20c	1	56	19	76	0
piper pa-28-140	1	190	18	209	0
piper pa-28-161	1	130	7	139	1
piper pa-28-180	1	134	21	156	0
piper pa28	1	191	20	214	2
cessna 180	0	236	10	246	0
robinson helicopter company r44 ii	0	107	7	115	1
piper pa-32-300	0	47	17	64	0
cessna 150	0	144	12	158	2
robinson r44	0	102	14	116	0

In [17]:
```
#finding planes that had low rates of death, serious injury, and minor injury
safest_planes=[x for x in minor_injury_rates.index[-20:] if x in serious_injury_rates.in

#filtering the damage table to those planes which we determined are the safest
damage_table_safe=damage_table.loc[damage_table.index.isin(safest_planes)]

safest_planes
```

Out[17]:
```
['piper pa28',
 'cessna 172m',
 'robinson r44',
 'cessna 172p',
 'piper pa-28-180',
 'piper pa-28-161',
 'cessna 172n',
 'mooney m20c',
 'cessna 150',
 'piper pa-28-140',
 'cessna 208b',
 'cessna 152',
 'cessna 172s',
 'cessna 182p',
 'mooney m20j',
 'cessna 150m']
```

We will create a visualization to show the breakdown of damage incurred by each type of plane.

In [18]:
```
#resettting the index so we can make a plot
damage_table_safe_reset=damage_table_safe.reset_index().drop(columns='planes').rename(co
damage_table_safe_reset.plot(x='plane', kind='bar', stacked=True)
plt.title('Damage to plane')
plt.ylabel('Type of damage')
plt.xticks(rotation=-90);
```



Based on the visualization, a few models such as the Mooney m20j, Cessna 150m, and the Piper pa-28-161 stand out as having low rates of destruction. The Cessna 152 also has a higher percentage of the damage being minor than substantial. These models would be our recommendations for safest private planes if the company did decide to go in that direction.

Commercial Plane Risk Assessment

The second portin will asses what risk is associated with Commerical Airplanes

We will start by examining the number_of_engines in the dataset

In [19]:
```
commercial_df = df
```

In [20]:
```
## Research:
# Reciprocating = Yes; some are / aren't
# Turbo Prop = No
# Turbo Fan = Yes; some are / aren't
# Delete: Unknown, Turbo Shaft, Electric, UNK


# Identifying which **engine_type** is used for commercial planes and filtering accordin
#Only using **Reciprocating** and **Turbo Fan**:
commercial_df = commercial_df[(commercial_df['engine_type'] == 'reciprocating') | (comme
commercial_df['engine_type'].value_counts()

# Filtering to planes with 2 or more engines:
commercial_df = commercial_df[commercial_df['number_of_engines'] >= 2]
commercial_df['number_of_engines'].value_counts()

# Top US mnaufacters of Commercial Planes
commercial_manufacturers = [
    "airbus",
    "boeing",
    "embraer",
    "comac",
    "atr",
    "mcdonnell douglas",
    "mcdonnell",
    "tupolev",
```

```
    "ilyushin",
]

# Filtering dataframe to Commercial Planes only
make = commercial_df['make'].isin(commercial_manufacturers)
filtered_df = commercial_df[make]
filtered_df['make'].value_counts()
commercial_df = commercial_df[commercial_df['make'].isin(commercial_manufacturers)]


# confirming all of the 'models' are indeed commercial planes:
not_commercial = ['a75', 'a75n1', 'b75n1', 'a75n1(pt17)', 'a75n1 (pt17)', 'b75', 'e75',

# the ~ in front of df is a negation operator to
# do the opposite of the following action:
commercial_df = commercial_df[~commercial_df['model'].isin(not_commercial)]
```

The most common commercial plane manufacturers are:

- boeing
- airbus
- embraer
- mcdonnell douglas (now owned by boeing)
- bombardier

In [21]:

```
commercial_df.head()
```

Out[21]:

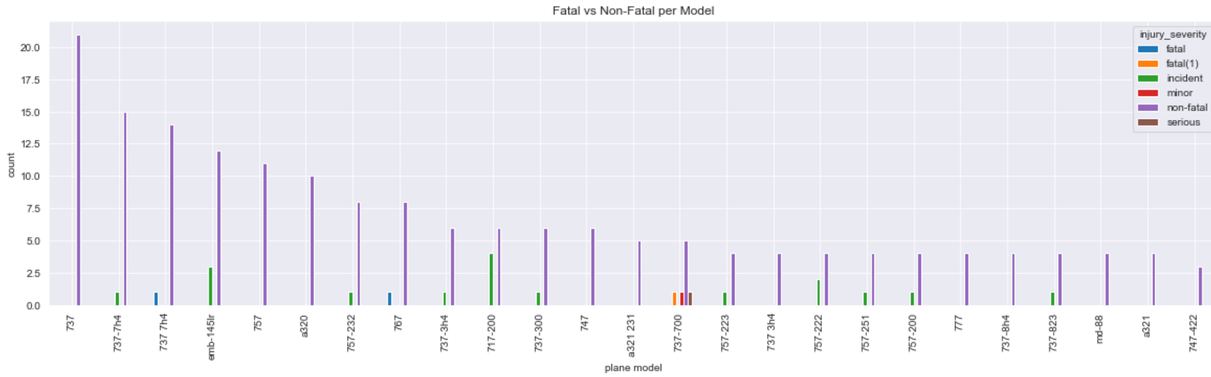
	location	investigation_type	event_date	country	injury_severity	aircraft_category	m
51806	memphis, tn	accident	2001-11-20	united states	non-fatal	airplane	bo
51850	romulus, mi	accident	2001-11-30	united states	non-fatal	airplane	mcdor dou
51905	chicago, il	incident	2001-12-13	united states	incident	airplane	bo
51944	anchorage, ak	accident	2001-12-28	united states	non-fatal	airplane	bo
51945	chicago, il	incident	2001-12-28	united states	incident	airplane	bo

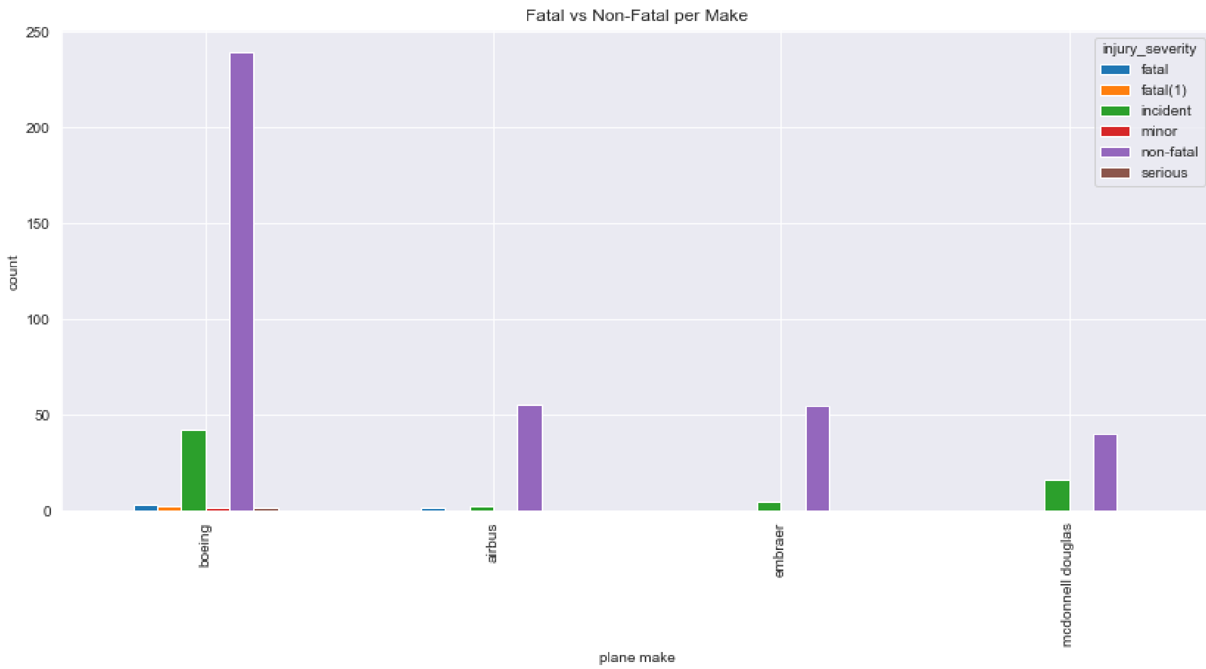
5 rows × 22 columns

Determining which models have had the most accident/incidents and if they were fatal/non-fatal:

In [22]:

```
#Showing fatal versus non-fatal per Model
grouped_1 = commercial_df.groupby(['model', 'injury_severity']).size().unstack().sort_va
grouped_1.plot(kind='bar', stacked=False, figsize=(20,5))
plt.xlabel('plane model')
plt.ylabel('count')
plt.title('Fatal vs Non-Fatal per Model')

plt.show()

#Showing fatal versus non-fatal per Make
grouped_2 = commercial_df.groupby(['make', 'injury_severity']).size().unstack().sort_val
grouped_2.plot(kind='bar', stacked=False, figsize=(14,6))
plt.xlabel('plane make')
plt.ylabel('count')
plt.title('Fatal vs Non-Fatal per Make')
plt.show()
```





In [23]:
```
commercial_df.loc[commercial_df['injury_severity'] == 'fatal']
```

Out[23]:

	location	investigation_type	event_date	country	injury_severity	aircraft_category	mal
74008	san francisco, ca	accident	2013-07-06	united states	fatal	airplane	boei
74252	birmingham, al	accident	2013-08-14	united states	fatal	airplane	airb
82061	philadelphia, pa	accident	2018-04-17	united states	fatal	airplane	boei
83727	trinity bay, tx	accident	2019-02-23	united states	fatal	airplane	boei

4 rows × 22 columns

Fatal Accident Context:

1.) boeing 777-200er (2013-07-06):

- .Pilot error; upon landing.
- .documentation: https://aviation-safety.net/database/record.php?id=20130706-0

2.) airbus a300 - f4 622r (2013-08-14):

- .Pilot error; failure to properly configure and verify the flight management computer for the profile approach
- .documentation: https://aviation-safety.net/database/record.php?id=20130814-0

3.) boeing 737 7h4 (2018-04-17):

- metal fatigue in the area where the blade broke in the engine.
- documentation: https://aviation-safety.net/database/record.php?id=20180417-0

4.) boeing 767 (2019-02-23):

- .Pilot error; inappropriate response by the first officer as the pilot flying to an inadvertent activation of the go-around mode, which led to his spatial disorientation.
- .documentation: https://aviation-safety.net/database/record.php?id=20190223-0

Operation Location Risk Assessment

For our location analysis we will want to look at the sum of total injuries for private and commercial planes in each region

In [24]:

```
# Group the data by state and sum the total injuries for private planes
injuries_by_state_south = private_planes[private_planes['region'] == 'South'].groupby('s
injuries_by_state_west = private_planes[private_planes['region'] == 'West'].groupby('sta
injuries_by_state_northeast = private_planes[private_planes['region'] == 'North East'].g
injuries_by_state_midwest =  private_planes[private_planes['region'] == 'Midwest'].group

# Group the data by state and sum the total injuries for commercial planes
commercial_injuries_by_state_south = commercial_df[commercial_df['region'] == 'South'].g
commercial_injuries_by_state_west = commercial_df[commercial_df['region'] == 'West'].gro
commercial_injuries_by_state_northeast = commercial_df[commercial_df['region'] == 'North
commercial_injuries_by_state_midwest =  commercial_df[commercial_df['region'] == 'Midwes
```

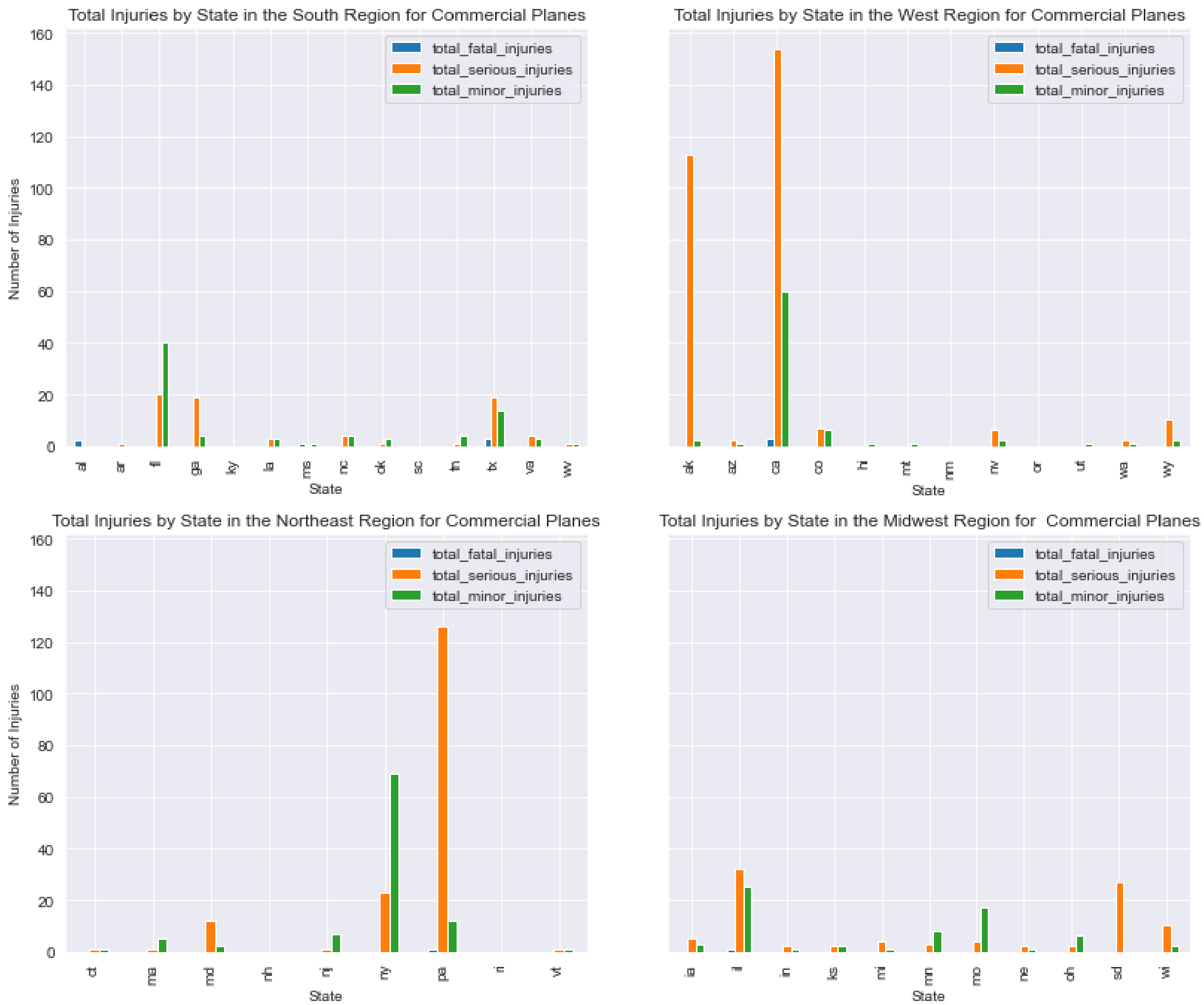
Visualizations

We will look at the injury data by state/region for commerical aircraft

In [25]:

```
# Set the figure size and layout
fig, ax = plt.subplots(2, 2, figsize=(12, 10), sharey=True)
fig.tight_layout(pad=4.0)

# Create a bar chart of the total injuries by state in the South Region for commercial p
commercial_injuries_by_state_south.plot(kind='bar', ax=ax[0,0])
ax[0,0].set_title('Total Injuries by State in the South Region for Commercial Planes')
ax[0,0].set_xlabel('State')
ax[0,0].set_ylabel('Number of Injuries')

# Create a bar chart of the total injuries by state in the West Region for commercial pl
commercial_injuries_by_state_west.plot(kind='bar', ax=ax[0,1])
ax[0,1].set_title('Total Injuries by State in the West Region for Commercial Planes')
ax[0,1].set_xlabel('State')
ax[0,1].set_ylabel('Number of Injuries')

# Create a bar chart of the total injuries by state in the North East Region for commerc
commercial_injuries_by_state_northeast.plot(kind='bar', ax=ax[1,0])
ax[1,0].set_title('Total Injuries by State in the Northeast Region for Commercial Planes
ax[1,0].set_xlabel('State')
ax[1,0].set_ylabel('Number of Injuries')
```

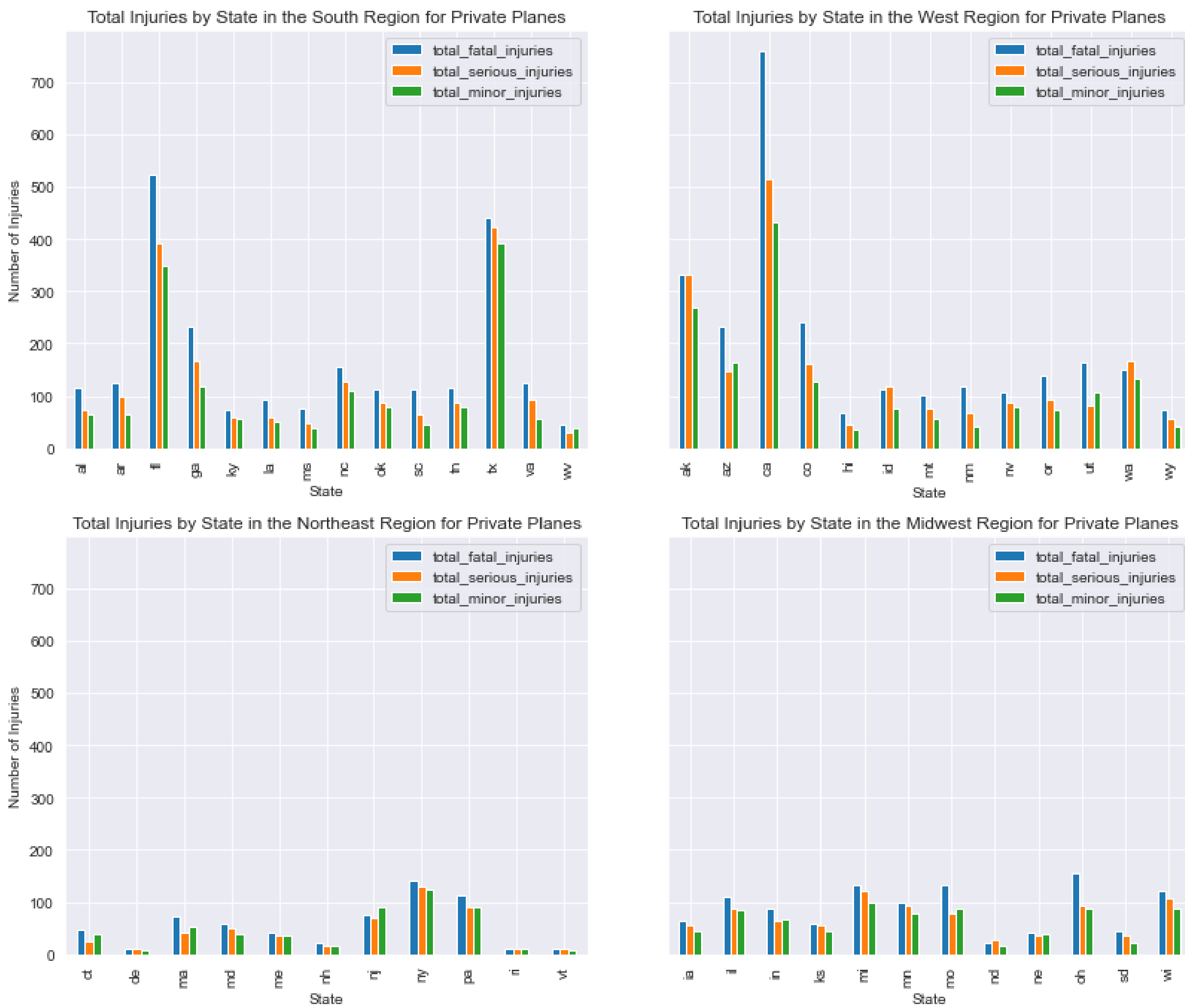
```
# Create a bar chart of the total injuries by state in the Midwest Region for commercial
commercial_injuries_by_state_midwest.plot(kind='bar', ax=ax[1,1])
ax[1,1].set_title('Total Injuries by State in the Midwest Region for  Commercial Planes'
ax[1,1].set_xlabel('State')
ax[1,1].set_ylabel('Number of Injuries')

plt.show()
```



Next, we will look at the total injuries by state/region for private aircraft

In [26]:
```
# Set the figure size and layout
fig, ax = plt.subplots(2, 2, figsize=(12, 10), sharey=True)
fig.tight_layout(pad=4.0)

# Create a bar chart of the total injuries by state in the South region for private plan
injuries_by_state_south.plot(kind='bar', ax=ax[0,0])
ax[0,0].set_title('Total Injuries by State in the South Region for Private Planes')
ax[0,0].set_xlabel('State')
ax[0,0].set_ylabel('Number of Injuries')

# Create a bar chart of the total injuries by state in the West region for private plane
injuries_by_state_west.plot(kind='bar', ax=ax[0,1])
ax[0,1].set_title('Total Injuries by State in the West Region for Private Planes')
ax[0,1].set_xlabel('State')
ax[0,1].set_ylabel('Number of Injuries')

# Create a bar chart of the total injuries by state in the North East region for private
injuries_by_state_northeast.plot(kind='bar', ax=ax[1,0])
ax[1,0].set_title('Total Injuries by State in the Northeast Region for Private Planes')
ax[1,0].set_xlabel('State')
```

```
ax[1,0].set_ylabel('Number of Injuries')

# Create a bar chart of the total injuries by state in the Midwest region for private pl
injuries_by_state_midwest.plot(kind='bar', ax=ax[1,1])
ax[1,1].set_title('Total Injuries by State in the Midwest Region for Private Planes')
ax[1,1].set_xlabel('State')
ax[1,1].set_ylabel('Number of Injuries')

plt.show()
```

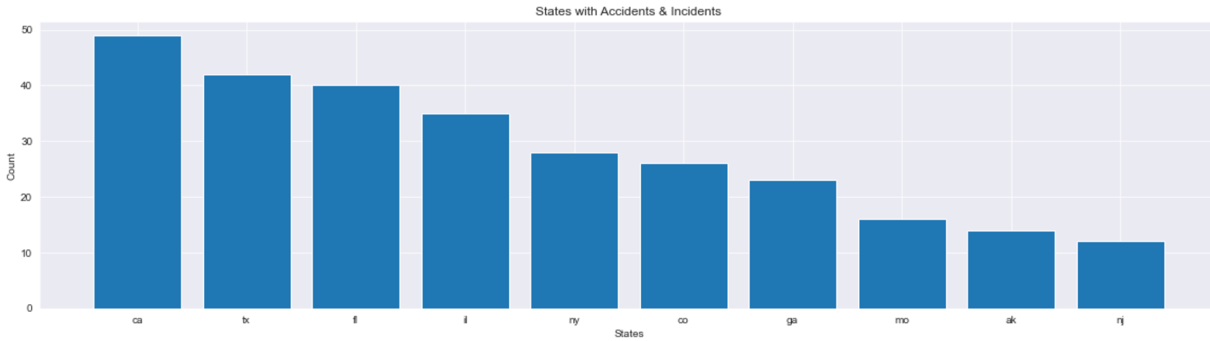


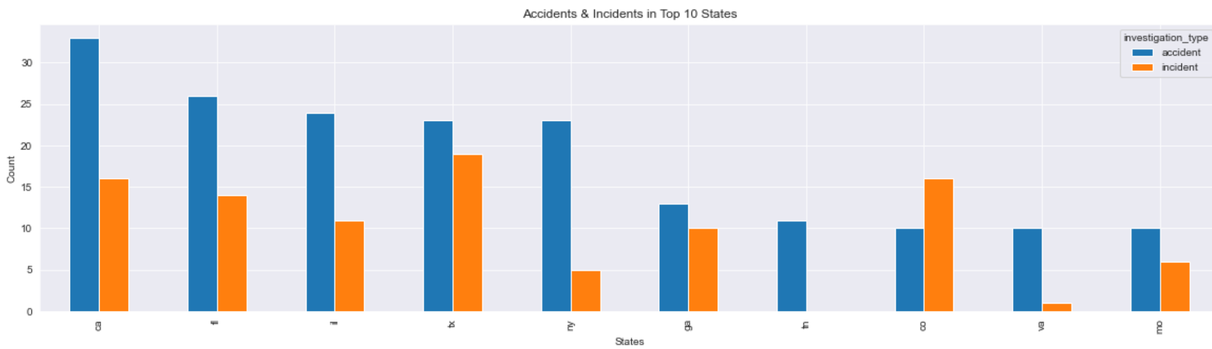
Next we will visualize the amount of accidents and incidents of the top 10 states with the highest count of investigations. Then we seperate accident and incident and view the counts.

In [27]:

```
#Identifying which states have the highest amount of Accidents/Incidents:
counts = commercial_df['state'].value_counts().sort_values(ascending=False).head(10)
x = counts.index
y = counts.values

fig, ax = plt.subplots(figsize=(20,5))
ax.bar(x, y)
plt.xlabel('States')
plt.ylabel('Count')
plt.title('States with Accidents & Incidents')
plt.show()

#Top 10 States & whether there was an Accident vs Incident:
grouped_1 = commercial_df.groupby(['state', 'investigation_type']).size().unstack().sort
grouped_1.plot(kind='bar', stacked=False, figsize=(20,5))
plt.xlabel('States')
plt.ylabel('Count')
```

```
plt.title('Accidents & Incidents in Top 10 States')

plt.show()
```





- From our analysis we see the majority of fatalities occur in Private planes across all regions. The lowest risk locations for private planes would be the Northeast, followed by the Midwest region.
- There is low risk associated with Commercial aircraft compared to private aircraft when viewing fatalities by region

---