Python Data Cleaning and Analysis

1a)Importing Libraries

In [2]:

```
#importing Libraries we need

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

```
c:\Users\Gmwende\anaconda3\envs\learn-env\lib\site-packages\scipy\__init__.py:138: UserWa
rning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detect
ed version 1.24.4)
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion} is required for
this version of "
```

b)Reading the Dataset from our CSV file

Dataset from National Transportation Safety Board that contains accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the united states and international waters

In [3]:

```
#lets read the data from the csv file and create a dataframe to be used
df = pd.read_csv('data\AviationData.csv',encoding='latin-1',low_memory=False)
```

c)Previewing our dataset

In [4]:

```
#increase rows and columns size
pd.set_option('display.max_rows',32)
pd.set_option('display.max_columns',32)
#lets preview the first five and last five rows by calling our dataframe
df
```

Out[4]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airp
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	
...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airp
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	

88889 rows × 31 columns

d)Accessing information about our dataset

In [5]:

```
#getting to know more about our dataset by accessing its information
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
```

e)Accessing summary statistics about our data

In [6]:

```
#statisctics for int and float objects
df.describe()
```

Out[6]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Total.Uninjured
count	82805.000000	77488.000000	76379.000000	76956.000000	82977.000000
mean	1.146585	0.647855	0.279881	0.357061	5.325440
std	0.446510	5.485960	1.544084	2.235625	27.913634
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	0.000000	0.000000	2.000000
max	8.000000	349.000000	161.000000	380.000000	699.000000

In [7]:

```
# statics for string objects
df.describe(include='O')
```

Out[7]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airpo
count	88889	88889	88889	88889	88837	88663	34382	34373	
unique	87951	2	88863	14782	27758	219	25589	27154	
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	United States	332739N	0112457W	
freq	3	85015	2	25	434	82248	19	24	

2) Cleaning Our Dataset

perrming data cleaning procedures below providing a documetation for our action and reasons.Will perform as amny data cleaning procedures as we think suitable for the various dimensions of data

In [8]:

```
#lets do a copy of our dataset first
aviation = df.copy()
#preview first 5 rows
aviation.head()
```

Out[8]:

	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

a)Selecting columns that will be useful for our analysis

In [9]:

```
#checking columns
aviation.columns
```

Out[9]:

```
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.Description',
       'Schedule', 'Purpose.of.flight', 'Air.carrier', 'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

In [10]:

```
#selecting columns we need for analysis
aviation = aviation[['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date',
'Country','Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model','Amateur.Built','Sche
dule', 'Purpose.of.flight','Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries', 'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight','Report.Status']]
#check first five rows
aviation.head()
```

Out[10]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Country	Injury.Severity	Aircraft.damage	Aircraft.Categor
0	20001218X45444	Accident	SEA87LA080	1948-10-24	United States	Fatal(2)	Destroyed	Na
1	20001218X45447	Accident	LAX94LA336	1962-07-19	United States	Fatal(4)	Destroyed	Na
2	20061025X01555	Accident	NYC07LA005	1974-08-30	United States	Fatal(3)	Destroyed	Na
3	20001218X45448	Accident	LAX96LA321	1977-06-19	United States	Fatal(2)	Destroyed	Na
4	20041105X01764	Accident	CHI79FA064	1979-08-02	United States	Fatal(1)	Destroyed	Na

In [11]:

```
#check form info the remaining no of columns
aviation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 21 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   Country                 88663 non-null  object
 5   Injury.Severity         87889 non-null  object
 6   Aircraft.damage         85695 non-null  object
 7   Aircraft.Category       32287 non-null  object
 8   Registration.Number     87572 non-null  object
 9   Make                    88826 non-null  object
 10  Model                   88797 non-null  object
 11  Amateur.Built           88787 non-null  object
 12  Schedule                12582 non-null  object
 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  Broad.phase.of.flight   61724 non-null  object
 20  Report.Status           82508 non-null  object
dtypes: float64(4), object(17)
memory usage: 14.2+ MB
```

b)cleaning our column names

strip of any spaces , replace the fullstops to underscores and capitalize the columns

In [12]:

```
#we use strip to remove trailing and leading spaces, replace  method to replace 'dot(.)'
with undescore(_) and upper to convert to uppercase/capitalize all
aviation.columns = aviation.columns.str.strip().str.replace('.','_',regex=True).str.upper
()
#check first five rows and if columns have been updated
aviation.head()
```

Out[12]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	COUNTRY	INJURY_SEVERITY	AIRCRAFT_DAI
0	20001218X45444	Accident	SEA87LA080	1948-10-24	United States	Fatal(2)	Dest
1	20001218X45447	Accident	LAX94LA336	1962-07-19	United States	Fatal(4)	Dest
2	20061025X01555	Accident	NYC07LA005	1974-08-30	United States	Fatal(3)	Dest
3	20001218X45448	Accident	LAX96LA321	1977-06-19	United States	Fatal(2)	Dest
4	20041105X01764	Accident	CHI79FA064	1979-08-02	United States	Fatal(1)	Dest

c)Check for missing values and how to handle them

In [13]:

```
#check the sum of missing values per column
aviation.isna().sum()
```

Out[13]:

```
EVENT_ID                   0
INVESTIGATION_TYPE         0
ACCIDENT_NUMBER            0
EVENT_DATE                 0
COUNTRY                  226
INJURY_SEVERITY         1000
AIRCRAFT_DAMAGE         3194
AIRCRAFT_CATEGORY      56602
REGISTRATION_NUMBER     1317
MAKE                      63
MODEL                     92
AMATEUR_BUILT            102
SCHEDULE               76307
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               6381
dtype: int64
```

In [14]:

```
#1. INJURY_SEVERITY column
aviation['INJURY_SEVERITY'].value_counts(dropna=False).head(20) # what do the numbers wit
h the Fatal mean?
```

Out[14]:

```
Non-Fatal      67357
Fatal(1)        6167
Fatal           5262
Fatal(2)        3711
Incident        2219
Fatal(3)        1147
NaN             1000
Fatal(4)         812
Fatal(5)         235
Minor            218
Serious          173
Fatal(6)         161
Unavailable       96
Fatal(7)          56
Fatal(8)          51
Fatal(10)         32
Fatal(9)          18
Fatal(14)         11
Fatal(11)         10
Fatal(13)          9
Name: INJURY_SEVERITY, dtype: int64
```

In [15]:

```
'''
We can see a pattern here that the number on fatal correspond to the total_fatal_injuries
eg Fatal(2) has fatal injuries as 2.
Since we have the number on another column we can strip the ones on the injury _severity_
column
'''
aviation[['INJURY_SEVERITY','TOTAL_FATAL_INJURIES']].head(10)
```

Out[15]:

	INJURY_SEVERITY	TOTAL_FATAL_INJURIES
0	Fatal(2)	2.0
1	Fatal(4)	4.0
2	Fatal(3)	3.0
3	Fatal(2)	2.0
4	Fatal(1)	1.0
5	Non-Fatal	NaN
6	Fatal(4)	4.0
7	Non-Fatal	0.0
8	Non-Fatal	0.0
9	Non-Fatal	0.0

In [16]:

```
aviation['INJURY_SEVERITY'] = aviation['INJURY_SEVERITY'].str.replace(r'\(.*\)','',regex
=True) #remove any text within the parentheses(including the parantheses themselves)
#we do a value count of this colun again
```

In [17]:

```
aviation['INJURY_SEVERITY'].value_counts(dropna=False).head(20) # we see all the fatal wi
th brackets are now just fatal
```

Out[17]:

```
Non-Fatal      67357
Fatal          17826
Incident        2219
NaN             1000
Minor            218
Serious          173
Unavailable       96
Name: INJURY_SEVERITY, dtype: int64
```

In [18]:

```
'''
since they are unavailable values lets fill all NaNs with this placeholder i dont want to
fill with mode since these are real accidents that happened and cant says they were Non-F
atal which is the mode
 whereas some could have been fatal thats why i used placeholders instead
'''
aviation['INJURY_SEVERITY'] = aviation['INJURY_SEVERITY'].fillna('Unavailable')
#we do isna for tha column
aviation['INJURY_SEVERITY'].isna().sum()
```

Out[18]:

```
0
```

In [19]:

```
# 2. AIRCRAFT_DAMAGE
aviation['AIRCRAFT_DAMAGE'].value_counts(dropna=False)
```

Out[19]:

```
Substantial    64148
Destroyed      18623
NaN             3194
Minor           2805
Unknown          119
Name: AIRCRAFT_DAMAGE, dtype: int64
```

In [20]:

```
#fillna with the existing placeholder unknown
aviation['AIRCRAFT_DAMAGE'] = aviation['AIRCRAFT_DAMAGE'].fillna('Unknown')
#check if  NaNs have been replaced
aviation['AIRCRAFT_DAMAGE'].isna().sum()
```

Out[20]:

```
0
```

In [21]:

```
# 3. AIRCRAFT_CATEGORY
aviation['AIRCRAFT_CATEGORY'].value_counts(dropna=False) #more than half of the data is m
issing(63.68%) but this column is very much needed for our analysis.Might another column
help us know these values?
#Yes from the make column but make columns has too many values so we will just keep this
column as is to help us know the category when we have the make
```

Out[21]:

```
NaN                 56602
Airplane            27617
Helicopter           3440
Glider                508
Balloon               231
Gyrocraft             173
Weight-Shift          161
```

```
Powered Parachute         91
Ultralight                30
Unknown                   14
WSFT                       9
Powered-Lift               5
Blimp                      4
UNK                        2
Rocket                     1
ULTR                       1
Name: AIRCRAFT_CATEGORY, dtype: int64
```

In [22]:

```
#fill with place holder unknown
aviation['AIRCRAFT_CATEGORY'].fillna('Unknown',inplace=True)
#convert UNK to unkwown as well
aviation['AIRCRAFT_CATEGORY'] = aviation['AIRCRAFT_CATEGORY'].replace('UNK','Unknown')
#confirm missing values removed
aviation['AIRCRAFT_CATEGORY'].isna().sum()
```

Out[22]:

```
0
```

In [23]:

```
aviation['AIRCRAFT_CATEGORY'].value_counts(dropna=False)
```

Out[23]:

```
Unknown                56618
Airplane               27617
Helicopter              3440
Glider                   508
Balloon                  231
Gyrocraft                173
Weight-Shift             161
Powered Parachute         91
Ultralight                30
WSFT                       9
Powered-Lift               5
Blimp                      4
Rocket                     1
ULTR                       1
Name: AIRCRAFT_CATEGORY, dtype: int64
```

In [ ]:

```

```

In [24]:

```
#4. REGISTRATION_NUMBER
aviation['REGISTRATION_NUMBER'].value_counts(dropna=False).head(30) #data entry isuess a
ll convert to UNK for unknown
```

Out[24]:

```
NaN        1317
NONE        344
UNREG       126
None         65
UNK          13
USAF          9
N20752        8
N53893        6
N121CC        6
N4101E        6
N8402K        6
unknown       6
N11VH         6
N5408Y        6
N75LE         5
```

```
N9957J        5
N8653Y        5
N32133        5
N93067        5
USN           5
N5246E        5
N420SB        5
N3331R        5
N8597D        5
N3125N        5
N99HV         5
N89ZC         4
N5291G        4
N99US         4
N99Y          4
Name: REGISTRATION_NUMBER, dtype: int64
```

In [25]:

```
aviation['REGISTRATION_NUMBER'] = aviation['REGISTRATION_NUMBER'].str.replace('NONE','UNK
').str.replace('None','UNK').str.replace('unknown','UNK')
#
#Below works as well
# replace_values = {
#     'unknown': 'UNK',
#     'NONE': 'UNK',
#     'None': 'UNK'
# }
# aviation['REGISTRATION_NUMBER'].replace(replace_values)
```

In [26]:

```
#Do value counts again
aviation['REGISTRATION_NUMBER'].value_counts(dropna=False)
```

Out[26]:

```
NaN       1317
UNK        428
UNREG      126
USAF         9
N20752       8
          ...
N93478       1
N519UA       1
N8840W       1
N21040       1
N9026P       1
Name: REGISTRATION_NUMBER, Length: 79103, dtype: int64
```

In [27]:

```
#fillna with placeholder UNK
aviation['REGISTRATION_NUMBER'] = aviation['REGISTRATION_NUMBER'].fillna('UNK')
aviation['REGISTRATION_NUMBER'].isna().sum()
```

Out[27]:

```
0
```

In [28]:

```
#5)MAKE column
aviation['MAKE'].value_counts(dropna=False).head(20)
```

Out[28]:

```
Cessna                22227
Piper                 12029
CESSNA                 4922
Beech                  4330
PIPER                  2841
Bell                   2134
```

```
Bell                        2134
Boeing                      1594
BOEING                      1151
Grumman                     1094
Mooney                      1092
BEECH                       1042
Robinson                     946
Bellanca                     886
Hughes                       795
Schweizer                    629
Air Tractor                  595
BELL                         588
Mcdonnell Douglas            526
Aeronca                      487
Maule                        445
Name: MAKE, dtype: int64
```

In [29]:

```
'''
we notice they are  data entry issues here.
1)we have both Cessna and CESSNA (only cases are different) , Boeing and BOEING etc
we can convert back to title cases
2) Cessna Aircraft ,Cessna Company    and cessna  are different (convert to one thing by
replcacing )
3)Robinson Helicopter Company,Robinson Helicopter,Robinson Helicopter Co. ,Robinson Helic
opter Co are different convert  to Robinson by replcaing

'''
#3b) convert make to title case
aviation['MAKE'] = aviation['MAKE'].str.title().str.strip('.')
#Replace
aviation['MAKE'] = aviation['MAKE'].str.replace('Robinson Helicopter Company','Robinson')
.str.replace('Robinson Helicopter','Robinson').str.replace('Robinson Helicopter Co','Robi
nson').str.replace('Robinson Co','Robinson').str.replace('Cessna Aircraft','Cessna').str.
replace('Cessna Company','Cessna')
#check value counts again
aviation['MAKE'].value_counts(dropna=False).head(20) # we see they are combined into a ca
tegory now
```

Out[29]:

```
Cessna                     27171
Piper                      14870
Beech                       5372
Boeing                      2745
Bell                        2722
Robinson                    1677
Mooney                      1334
Grumman                     1172
Bellanca                    1045
Hughes                       932
Schweizer                    773
Air Tractor                  691
Aeronca                      636
Mcdonnell Douglas            608
Maule                        589
Champion                     519
Stinson                      439
Aero Commander               429
De Havilland                 422
Luscombe                     414
Name: MAKE, dtype: int64
```

In [30]:

```
# #Check Piper make
# aviation.query('MAKE == "Piper" & AIRCRAFT_CATEGORY.notna()')['AIRCRAFT_CATEGORY'].valu
e_counts()# we see all piper are airplane category
# # do a function for above to prevent repetition
# def get_category(make):
#      return aviation.query('MAKE == @make & AIRCRAFT_CATEGORY.notna()')['AIRCRAFT_CATEG
```

```
ORY'].value_counts().index[0]
# get_category('Bell')##Cessna,Piper,Beech
# #Bell-helicopter
```

In [31]:

```
#recheck to see the missing values for the make column
aviation['MAKE'].isna().sum()/aviation.shape[0] #only 0.07% of data missing we can drop t
hem
```

Out[31]:

```
0.0007087491140636075
```

In [32]:

```
#drop missing MAKE values
aviation.dropna(subset=['MAKE'],inplace=True)
```

In [33]:

```
#)Recheck missing values for the dataset
aviation.isna().sum()
```

Out[33]:

```
EVENT_ID                      0
INVESTIGATION_TYPE            0
ACCIDENT_NUMBER               0
EVENT_DATE                    0
COUNTRY                     225
INJURY_SEVERITY               0
AIRCRAFT_DAMAGE               0
AIRCRAFT_CATEGORY             0
REGISTRATION_NUMBER           0
MAKE                          0
MODEL                        49
AMATEUR_BUILT               100
SCHEDULE                  76283
PURPOSE_OF_FLIGHT          6150
TOTAL_FATAL_INJURIES      11394
TOTAL_SERIOUS_INJURIES    12500
TOTAL_MINOR_INJURIES      11922
TOTAL_UNINJURED            5901
WEATHER_CONDITION          4454
BROAD_PHASE_OF_FLIGHT     27113
REPORT_STATUS              6349
dtype: int64
```

In [34]:

```
# 6. Check Model
aviation['MODEL'].value_counts(dropna=False)
```

Out[34]:

```
152                       2367
172                       1756
172N                      1164
PA-28-140                  932
150                        829
                          ...
SNJ-5C                       1
QUICKSILVER SPORT 2          1
727-2Q8                      1
WMF                          1
M-8 EAGLE                    1
Name: MODEL, Length: 12312, dtype: int64
```

In [35]:

```
#check missing values values in the model column
```

```
print(aviation['MODEL'].isna().sum())
aviation['MODEL'].isna().sum()/len(aviation)
```

```
49
```

Out[35]:

```
0.0005516402855019926
```

In [36]:

```
'''missing values are very small we can drop them'''
aviation.dropna(subset='MODEL',inplace=True)
```

In [37]:

```
# 7.Check AMATEUR_BUILT
aviation['AMATEUR_BUILT'].value_counts(dropna=False)
```

Out[37]:

```
No     80240
Yes     8438
NaN       99
Name: AMATEUR_BUILT, dtype: int64
```

In [38]:

```
#fillna with placeholder for this misisng values dont want to miss data that could be imp
ortant to our analyis
aviation['AMATEUR_BUILT'].fillna('unknown',inplace=True)
```

In [39]:

```
# 8 Check schedule column
aviation['SCHEDULE'].value_counts(dropna=False,normalize=True)
```

Out[39]:

```
NaN     0.858837
NSCH    0.050238
UNK     0.046138
SCHD    0.044786
Name: SCHEDULE, dtype: float64
```

In [40]:

```
'''
we have 85% missing values, this column will not help much in our analysis so we drop it
'''
aviation.drop('SCHEDULE',axis=1,inplace=True)
```

In [41]:

```
#check if column has been dropped
aviation.columns
```

Out[41]:

```
Index(['EVENT_ID', 'INVESTIGATION_TYPE', 'ACCIDENT_NUMBER', 'EVENT_DATE',
       'COUNTRY', 'INJURY_SEVERITY', 'AIRCRAFT_DAMAGE', 'AIRCRAFT_CATEGORY',
       'REGISTRATION_NUMBER', 'MAKE', 'MODEL', 'AMATEUR_BUILT',
       'PURPOSE_OF_FLIGHT', 'TOTAL_FATAL_INJURIES', 'TOTAL_SERIOUS_INJURIES',
       'TOTAL_MINOR_INJURIES', 'TOTAL_UNINJURED', 'WEATHER_CONDITION',
       'BROAD_PHASE_OF_FLIGHT', 'REPORT_STATUS'],
      dtype='object')
```

In [42]:

```
#9 Check purpose of flight column
aviation['PURPOSE_OF_FLIGHT'].value_counts(dropna=False)
```

Out[42]:

```
Personal                       49413
Instructional                  10599
Unknown                         6787
NaN                             6138
Aerial Application              4710
Business                        4016
Positioning                     1645
Other Work Use                  1264
Ferry                            812
Aerial Observation               794
Public Aircraft                  720
Executive/corporate              553
Flight Test                      404
Skydiving                        182
External Load                    123
Public Aircraft - Federal        105
Banner Tow                       101
Air Race show                     99
Public Aircraft - Local           74
Public Aircraft - State           64
Air Race/show                     59
Glider Tow                        53
Firefighting                      40
Air Drop                          11
ASHO                               6
PUBS                               4
PUBL                               1
Name: PURPOSE_OF_FLIGHT, dtype: int64
```

In [43]:

```python
#fill NaN with the already existing placeholder
aviation['PURPOSE_OF_FLIGHT'].fillna('Unknown',inplace=True)
#check if missing values removed
aviation['PURPOSE_OF_FLIGHT'].isna().sum()
```

Out[43]:

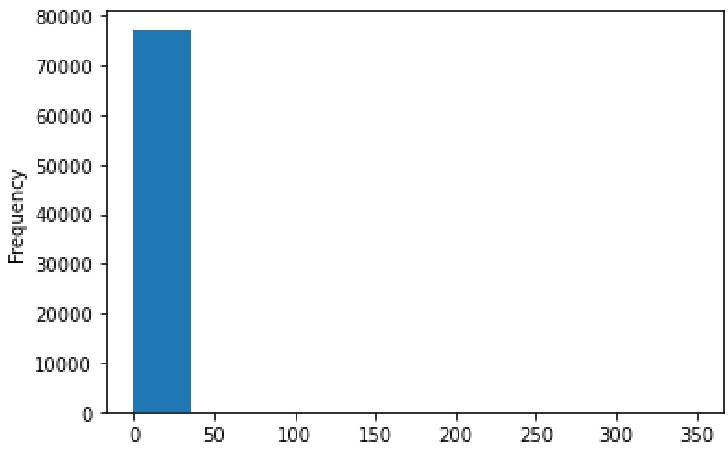
```
0
```

In [44]:

```python
''' There is a data entry issue where  Air Race show  and Air Race/show are different thi
ngs
'''
aviation['PURPOSE_OF_FLIGHT'] = aviation['PURPOSE_OF_FLIGHT'].replace('/',' ',regex=True)
```

In [45]:

```python
#10. check 'TOTAL_FATAL_INJURIES' columns
aviation['TOTAL_FATAL_INJURIES'].plot(kind='hist');
```



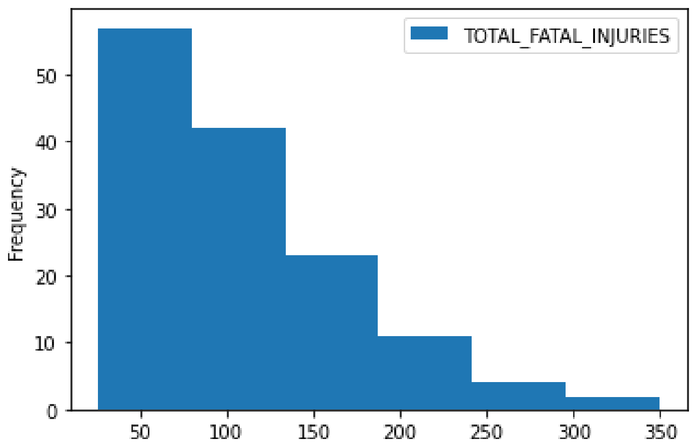
In [46]:

```python
# check for values above 25
```

```
aviation.query('TOTAL_FATAL_INJURIES >25')[['TOTAL_FATAL_INJURIES']].plot(kind='hist',bins=6)
```

Out[46]:

```
<AxesSubplot:ylabel='Frequency'>
```



In [47]:

```
aviation['TOTAL_FATAL_INJURIES'].isna().sum()
```

Out[47]:

```
11386
```

In [48]:

```
aviation.isna().sum()
```

Out[48]:

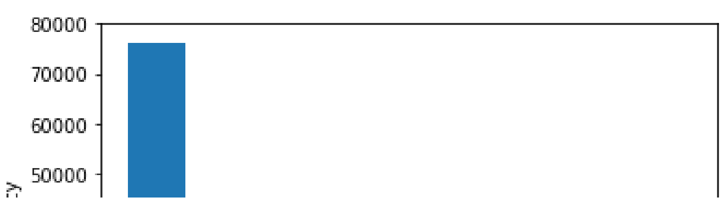
```
EVENT_ID                    0
INVESTIGATION_TYPE          0
ACCIDENT_NUMBER             0
EVENT_DATE                  0
COUNTRY                   225
INJURY_SEVERITY             0
AIRCRAFT_DAMAGE             0
AIRCRAFT_CATEGORY           0
REGISTRATION_NUMBER         0
MAKE                        0
MODEL                       0
AMATEUR_BUILT               0
PURPOSE_OF_FLIGHT           0
TOTAL_FATAL_INJURIES    11386
TOTAL_SERIOUS_INJURIES  12490
TOTAL_MINOR_INJURIES    11914
TOTAL_UNINJURED          5897
WEATHER_CONDITION        4439
BROAD_PHASE_OF_FLIGHT   27094
REPORT_STATUS            6335
dtype: int64
```

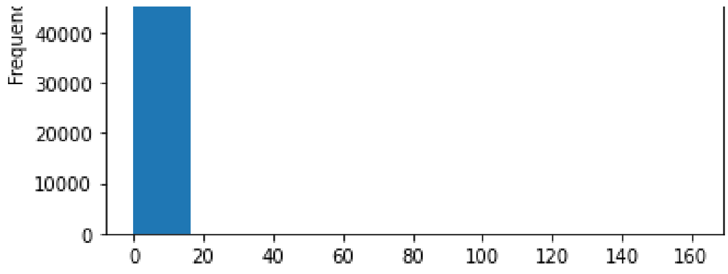
In [49]:

```
# 11. 'TOTAL_SERIOUS_INJURIES' columns
aviation['TOTAL_SERIOUS_INJURIES'].plot(kind='hist')
```

Out[49]:

```
<AxesSubplot:ylabel='Frequency'>
```



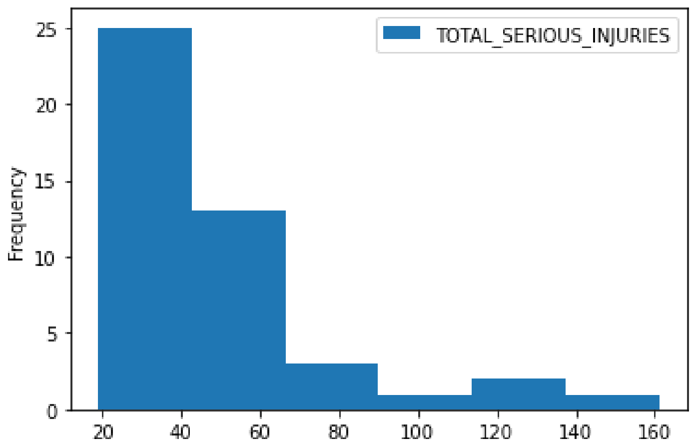


In [50]:

```
#hist for values above 18
# check for values above 25
aviation.query('TOTAL_SERIOUS_INJURIES >18')[['TOTAL_SERIOUS_INJURIES']].plot(kind='hist'
,bins=6)
```

Out[50]:

<AxesSubplot:ylabel='Frequency'>

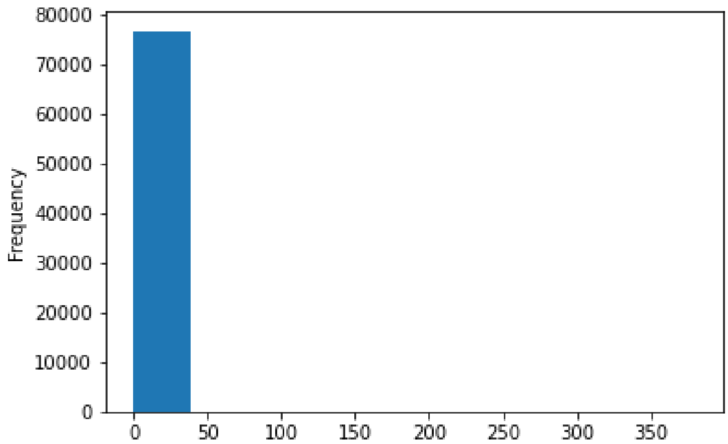


In [51]:

```
#12 'TOTAL_MINOR_INJURIES'
aviation['TOTAL_MINOR_INJURIES'].plot(kind='hist')
```

Out[51]:

<AxesSubplot:ylabel='Frequency'>

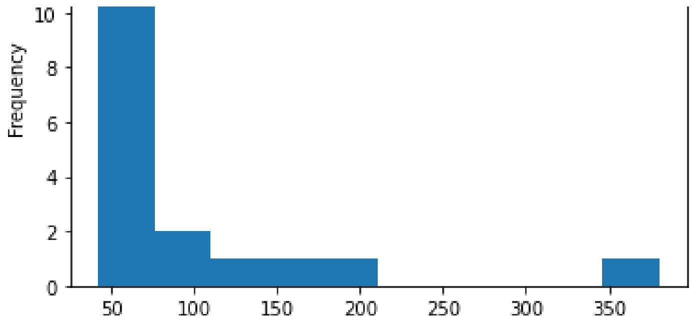


In [52]:

```
aviation.query('TOTAL_MINOR_INJURIES>40')['TOTAL_MINOR_INJURIES'].plot(kind='hist')
```

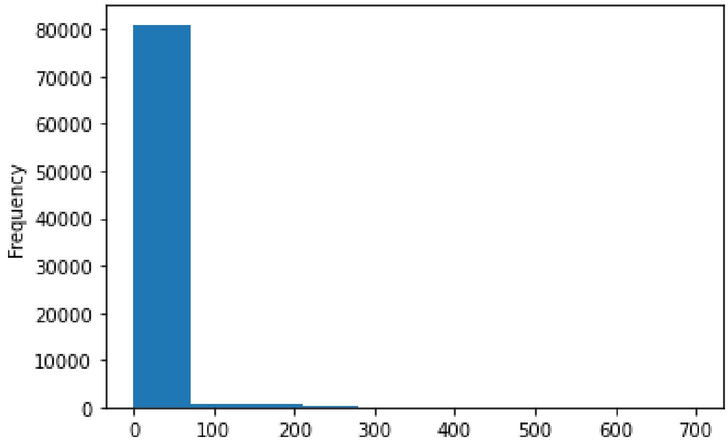
Out[52]:

<AxesSubplot:ylabel='Frequency'>



In [53]:

```
#13 , 'TOTAL_UNINJURED'
aviation['TOTAL_UNINJURED'].plot(kind='hist');
```

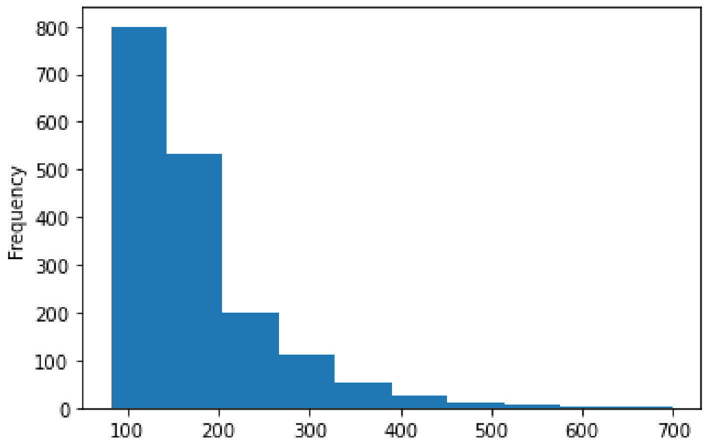


In [54]:

```
aviation.query('TOTAL_UNINJURED>80')['TOTAL_UNINJURED'].plot(kind='hist')
```

Out[54]:

```
<AxesSubplot:ylabel='Frequency'>
```



In [55]:

```
'''
All the 4 columns TOTAL_FATAL_INJURIES, TOTAL_SERIOUS_INJURIES, TOTAL_MINOR_INJURIES, TOT
AL_UNINJURED are skewed postitively
we will replace NA values with the median
'''
aviation['TOTAL_FATAL_INJURIES'].fillna(aviation['TOTAL_FATAL_INJURIES'].median(),inplace
=True)
aviation['TOTAL_SERIOUS_INJURIES'].fillna(aviation['TOTAL_SERIOUS_INJURIES'].median(),inp
lace=True)
aviation['TOTAL_MINOR_INJURIES'].fillna(aviation['TOTAL_MINOR_INJURIES'].median(),inplace
=True)
aviation['TOTAL_UNINJURED'].fillna(aviation['TOTAL_UNINJURED'].median(),inplace=True)
```

In [56]:

```python
#14 WEATHER_CONDITION columns
aviation['WEATHER_CONDITION'].value_counts(dropna=False)
```

Out[56]:

```
VMC    77251
IMC     5971
NaN     4439
UNK      854
Unk      262
Name: WEATHER_CONDITION, dtype: int64
```

In [57]:

```python
#convert to Upper case and also fillna with place holder unl
#title
aviation['WEATHER_CONDITION'] = aviation['WEATHER_CONDITION'].str.upper()
#fillna with  place holder 'UNK'
aviation['WEATHER_CONDITION'].fillna('UNK',inplace=True)
#check if okay
aviation['WEATHER_CONDITION'].value_counts(dropna=False)
```

Out[57]:

```
VMC    77251
IMC     5971
UNK     5555
Name: WEATHER_CONDITION, dtype: int64
```

In [58]:

```python
#15'BROAD_PHASE_OF_FLIGHT'
aviation['BROAD_PHASE_OF_FLIGHT'].value_counts(dropna=False)
```

Out[58]:

```
NaN            27094
Landing        15423
Takeoff        12481
Cruise         10263
Maneuvering     8138
Approach        6538
Climb           2031
Taxi            1958
Descent         1886
Go-around       1353
Standing         945
Unknown          548
Other            119
Name: BROAD_PHASE_OF_FLIGHT, dtype: int64
```

In [59]:

```python
#fill both other and Nan with place holder unknown
#replace other to placeholder unknown
aviation['BROAD_PHASE_OF_FLIGHT'] = aviation['BROAD_PHASE_OF_FLIGHT'].replace('Other','Un
known',regex=True)
#fillna with unknown
aviation['BROAD_PHASE_OF_FLIGHT'] .fillna('Unknown',inplace=True)
#check if okay
aviation['BROAD_PHASE_OF_FLIGHT'].value_counts(dropna=False)
```

Out[59]:

```
Unknown        27761
Landing        15423
Takeoff        12481
Cruise         10263
Maneuvering     8138
Approach        6538
Climb           2031
Taxi            1958
Descent         1886
```

```
Go-around        1353
Standing          945
Name: BROAD_PHASE_OF_FLIGHT, dtype: int64
```

In [60]:

```
#16. REPORT_STATUS
aviation['REPORT_STATUS'].value_counts(dropna=False).head(30)
```

Out[60]:

```
Probable Cause
61713
NaN
6335
Foreign
1986
<br /><br />
167
Factual
145
The pilot's failure to maintain directional control during the landing roll.
56
A loss of engine power for undetermined reasons.
52
The pilot's failure to maintain directional control during landing.
44
A total loss of engine power for undetermined reasons.
39
The loss of engine power for undetermined reasons.
29
The pilots failure to maintain directional control during the landing roll.\r\n\r
21
The pilot's improper recovery from a bounced landing.
19
The pilots failure to maintain directional control during the landing roll.
19
The pilot's failure to maintain directional control during takeoff.
17
The pilot's failure to maintain directional control of the airplane during landing.
17
None.
17
The pilots failure to maintain directional control during landing.
16
The pilot's improper landing flare, which resulted in a hard landing.
16
The student pilot's improper recovery from a bounced landing.
16
The pilot's failure to maintain directional control during the takeoff roll.
15
Preliminary
15
.
15
The pilots failure to maintain directional control during landing.\r\n\r
12
A partial loss of engine power for undetermined reasons.
12
The pilot's improper fuel management, which resulted in a loss of engine power due to fue
l exhaustion.              11
The student pilot's improper flare, which resulted in a hard landing.
10
The pilots improper landing flare, which resulted in a hard landing.\r\n\r
10
The pilot's failure to maintain directional control during the landing roll, which result
ed in a ground loop.        9
The pilots improper fuel management, which resulted in a loss of engine power due to fuel
exhaustion.                 9
The student pilot's failure to maintain directional control during takeoff.
9
Name: REPORT_STATUS, dtype: int64
```

In [61]:

```
aviation.columns
```

Out[61]:

```
Index(['EVENT_ID', 'INVESTIGATION_TYPE', 'ACCIDENT_NUMBER', 'EVENT_DATE',
       'COUNTRY', 'INJURY_SEVERITY', 'AIRCRAFT_DAMAGE', 'AIRCRAFT_CATEGORY',
       'REGISTRATION_NUMBER', 'MAKE', 'MODEL', 'AMATEUR_BUILT',
       'PURPOSE_OF_FLIGHT', 'TOTAL_FATAL_INJURIES', 'TOTAL_SERIOUS_INJURIES',
       'TOTAL_MINOR_INJURIES', 'TOTAL_UNINJURED', 'WEATHER_CONDITION',
       'BROAD_PHASE_OF_FLIGHT', 'REPORT_STATUS'],
      dtype='object')
```

In [62]:

```
aviation.isna().sum()*100/len(aviation)
```

Out[62]:

```
EVENT_ID                  0.000000
INVESTIGATION_TYPE        0.000000
ACCIDENT_NUMBER           0.000000
EVENT_DATE                0.000000
COUNTRY                   0.253444
INJURY_SEVERITY           0.000000
AIRCRAFT_DAMAGE           0.000000
AIRCRAFT_CATEGORY         0.000000
REGISTRATION_NUMBER       0.000000
MAKE                      0.000000
MODEL                     0.000000
AMATEUR_BUILT             0.000000
PURPOSE_OF_FLIGHT         0.000000
TOTAL_FATAL_INJURIES      0.000000
TOTAL_SERIOUS_INJURIES    0.000000
TOTAL_MINOR_INJURIES      0.000000
TOTAL_UNINJURED           0.000000
WEATHER_CONDITION         0.000000
BROAD_PHASE_OF_FLIGHT     0.000000
REPORT_STATUS             7.135857
dtype: float64
```

In [63]:

```
#update missing to None
aviation['REPORT_STATUS'].fillna('None',inplace=True)
```

In [64]:

```
#17. Country column
aviation['COUNTRY'].value_counts(dropna=False,normalize=True).head(30)
```

Out[64]:

```
United States         0.925882
Brazil                0.004168
Mexico                0.004021
Canada                0.003999
United Kingdom        0.003875
Australia             0.003345
France                0.002647
NaN                   0.002534
Spain                 0.002523
Bahamas               0.002433
Germany               0.002354
Colombia              0.002151
South Africa          0.001442
Japan                 0.001408
Venezuela             0.001352
Argentina             0.001250
Italy                 0.001250
Indonesia             0.001228
```

```
India                   0.001081
Peru                    0.001048
Russia                  0.001025
ATLANTIC OCEAN          0.000912
Ireland                 0.000867
Puerto Rico             0.000800
Dominican Republic      0.000766
Guatemala               0.000755
China                   0.000755
Eswatini                0.000687
New Zealand             0.000631
Sweden                  0.000631
Name: COUNTRY, dtype: float64
```

In [65]:

```python
#fill with a placeholder unknown
aviation['COUNTRY'].fillna('unknown',inplace=True)
```

In [66]:

```python
aviation.isna().sum()
```

Out[66]:

```
EVENT_ID                    0
INVESTIGATION_TYPE          0
ACCIDENT_NUMBER             0
EVENT_DATE                  0
COUNTRY                     0
INJURY_SEVERITY             0
AIRCRAFT_DAMAGE             0
AIRCRAFT_CATEGORY           0
REGISTRATION_NUMBER         0
MAKE                        0
MODEL                       0
AMATEUR_BUILT               0
PURPOSE_OF_FLIGHT           0
TOTAL_FATAL_INJURIES        0
TOTAL_SERIOUS_INJURIES      0
TOTAL_MINOR_INJURIES        0
TOTAL_UNINJURED             0
WEATHER_CONDITION           0
BROAD_PHASE_OF_FLIGHT       0
REPORT_STATUS               0
dtype: int64
```

d)All missing values cleaned.Lets check for duplicates

In [67]:

```python
aviation.duplicated().sum()  # They are no missing values in our dataset
```

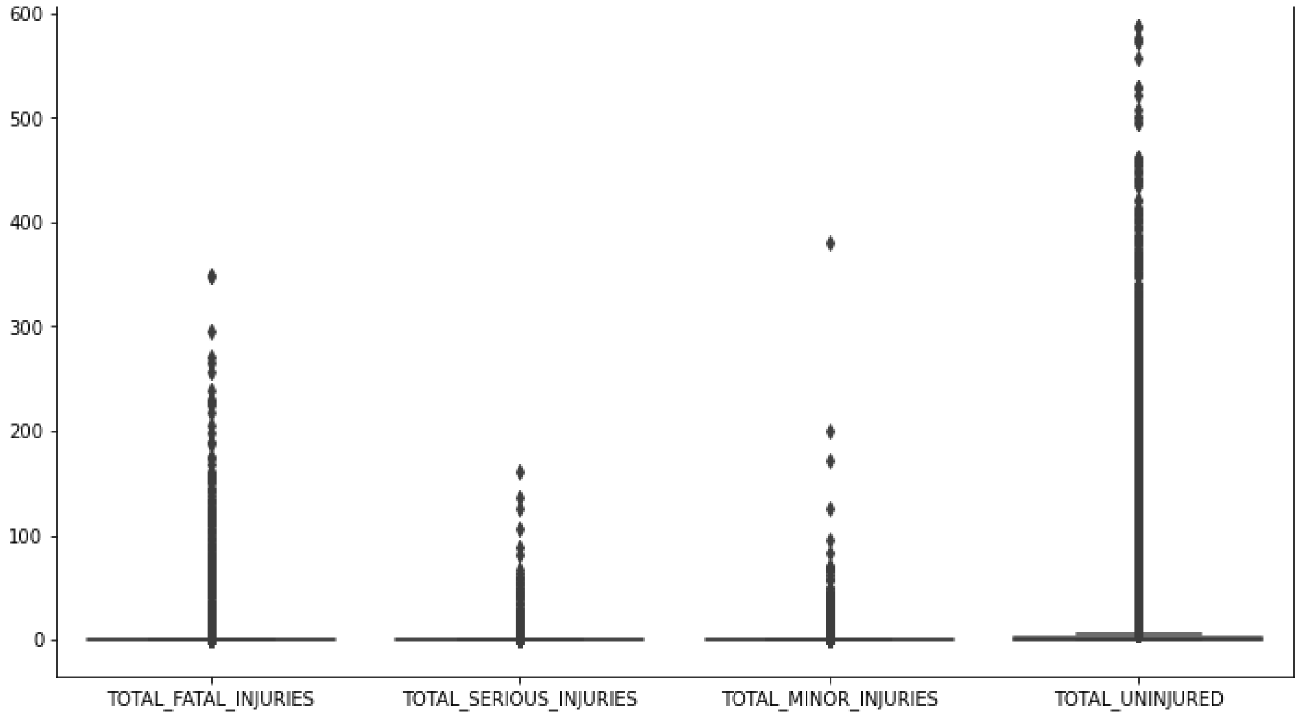
Out[67]:

```
0
```

e)Check for ouliers

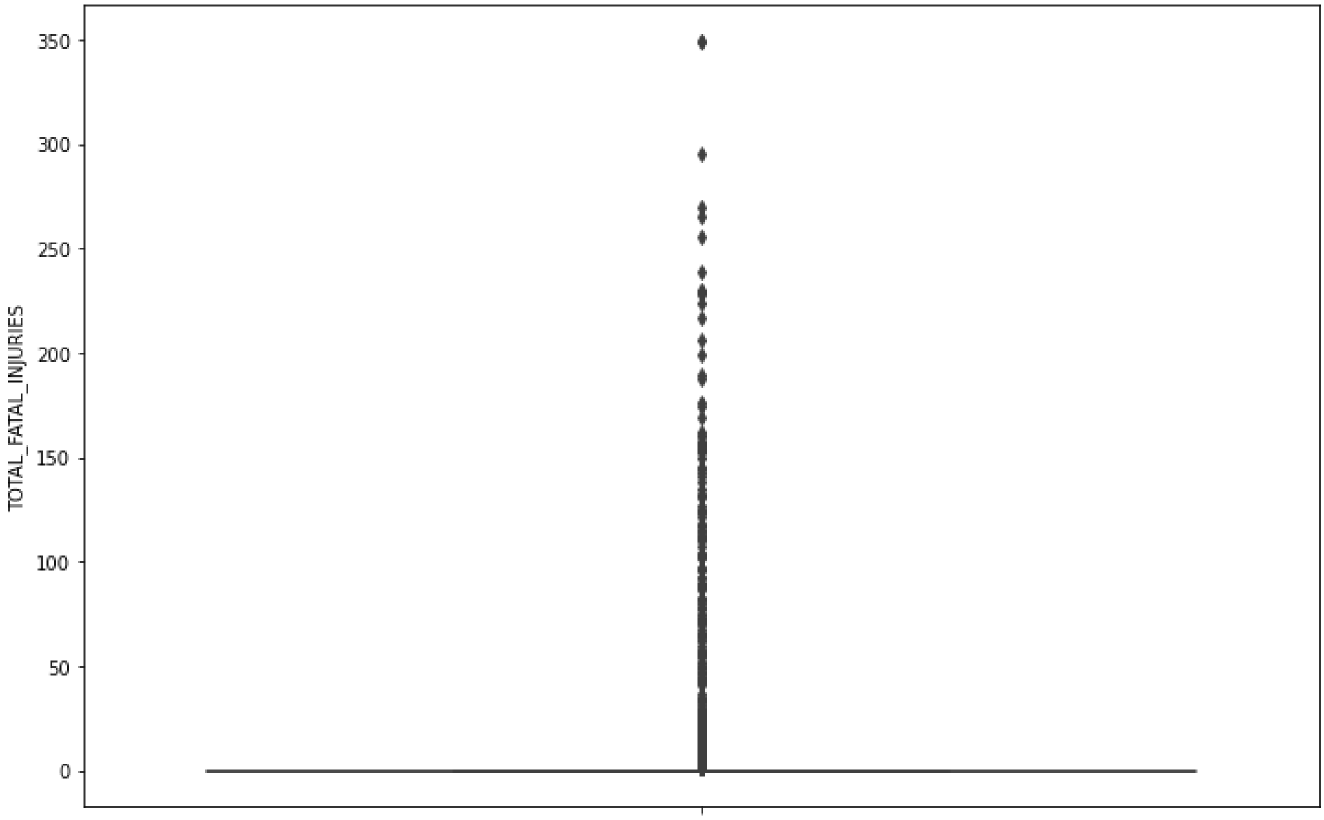
In [68]:

```python
plt.figure(figsize=(12,8))
sns.boxplot(data=aviation); #seem to be outliers for all these columns lets check by colu
mn
```



In [69]:

```
#1 total fatal injuries
plt.figure(figsize=(12,8))
sns.boxplot(data=aviation,y='TOTAL_FATAL_INJURIES');
```



In [70]:

```
#check outliers using IQR
#calculate Q1 and Q3
Q1, Q3 = aviation['TOTAL_UNINJURED'].quantile([0.25,0.75])
#IQR
IQR= Q3 - Q1
print(IQR)
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
print(lower_bound,upper_bound)
```

```
2.0
-3.0 5.0
```

In [71]:

```
'''
No need to remove outliers since these are real numbers for real accidents that took plac
e
'''
#aviation.query('~(TOTAL_UNINJURED <@lower_bound | TOTAL_UNINJURED >@upper_bound)')  # wo
rks as well #aviation.query('TOTAL_UNINJURED >=@lower_bound & TOTAL_UNINJURED <=@upper_bo
und')
```

Out[71]:

```
' \nNo need to remove outliers since these are real numbers for real accidents that took
place\n'
```

f)Convert Data types

In [72]:

```
#check date column'Event Date'
aviation['EVENT_DATE'].head(10)
```

Out[72]:

```
0    1948-10-24
1    1962-07-19
2    1974-08-30
3    1977-06-19
4    1979-08-02
5    1979-09-17
6    1981-08-01
7    1982-01-01
8    1982-01-01
9    1982-01-01
Name: EVENT_DATE, dtype: object
```

In [73]:

```
#convert to datetime format
aviation['EVENT_DATE'] = pd.to_datetime(aviation['EVENT_DATE'],format='%Y-%m-%d')
aviation['EVENT_DATE'].head(10)
```

Out[73]:

```
0   1948-10-24
1   1962-07-19
2   1974-08-30
3   1977-06-19
4   1979-08-02
5   1979-09-17
6   1981-08-01
7   1982-01-01
8   1982-01-01
9   1982-01-01
Name: EVENT_DATE, dtype: datetime64[ns]
```

g)Export the Cleaned dataset

In [74]:

```
aviation.to_csv('clean_aviationData.csv',index=0)
```

3. Importing cleaned Datasheet for EDA and Analysis

In [75]:

```
clean_df = pd.read_csv('clean_aviationData.csv')
```

```
clean_df.head()
```

Out[75]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	COUNTRY	INJURY_SEVERITY	AIRCRAFT_DA
0	20001218X45444	Accident	SEA87LA080	1948-10-24	United States	Fatal	Dest
1	20001218X45447	Accident	LAX94LA336	1962-07-19	United States	Fatal	Dest
2	20061025X01555	Accident	NYC07LA005	1974-08-30	United States	Fatal	Dest
3	20001218X45448	Accident	LAX96LA321	1977-06-19	United States	Fatal	Dest
4	20041105X01764	Accident	CHI79FA064	1979-08-02	United States	Fatal	Dest

In [76]:

```
#check for missing values
clean_df.isna().sum()
```

Out[76]:

```
EVENT_ID                  0
INVESTIGATION_TYPE        0
ACCIDENT_NUMBER           0
EVENT_DATE                0
COUNTRY                   0
INJURY_SEVERITY           0
AIRCRAFT_DAMAGE           0
AIRCRAFT_CATEGORY         0
REGISTRATION_NUMBER       0
MAKE                      0
MODEL                     0
AMATEUR_BUILT             0
PURPOSE_OF_FLIGHT         0
TOTAL_FATAL_INJURIES      0
TOTAL_SERIOUS_INJURIES    0
TOTAL_MINOR_INJURIES      0
TOTAL_UNINJURED           0
WEATHER_CONDITION         0
BROAD_PHASE_OF_FLIGHT     0
REPORT_STATUS             0
dtype: int64
```

In [77]:

```
#check for duplicates
clean_df.duplicated().sum()
```

Out[77]:

```
0
```

In [78]:

```
#do a copy of the dataframe
aviation_data = clean_df.copy()
aviation_data.head()
```

Out[78]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	COUNTRY	INJURY_SEVERITY	AIRCRAFT_DA
0	20001218X45444	Accident	SEA87LA080	1948-10-24	United States	Fatal	Dest
1	20001218X45447	Accident	LAX94LA336	1962-07-19	United States	Fatal	Dest

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	COUNTRY	INJURY_SEVERITY	AIRCRAFT_DAI
2	20061025X01555	Accident	NYC07LA005	1974-08-30	United States	Fatal	Dest
3	20001218X45448	Accident	LAX96LA321	1977-06-19	United States	Fatal	Dest
4	20041105X01764	Accident	CHI79FA064	1979-08-02	United States	Fatal	Dest

a)univariate Analysis

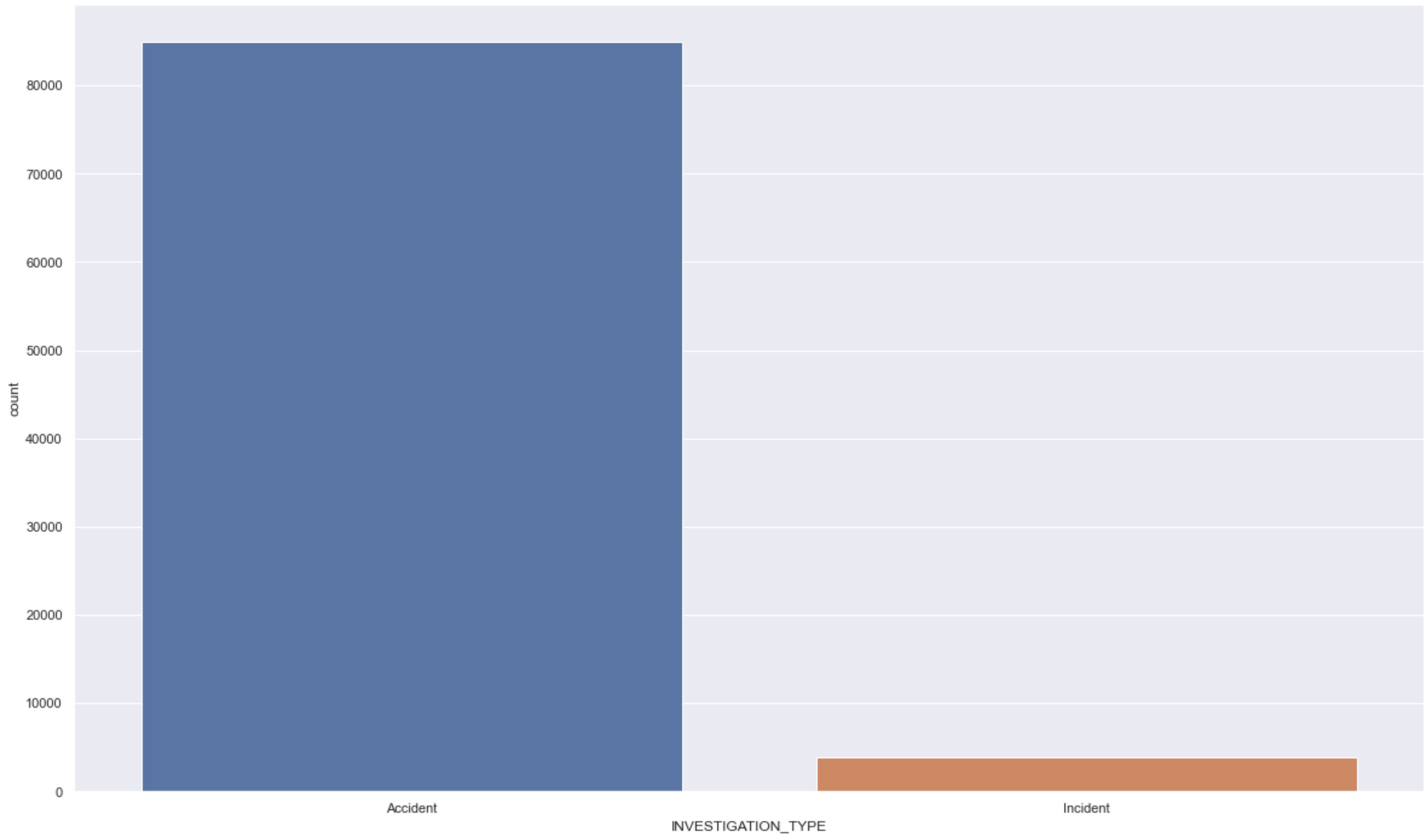
i)Check investigation_type column

In [79]:

```
#set figsize for all objects going forward
sns.set_theme(rc={'figure.figsize':(20,12)})
```

In [80]:

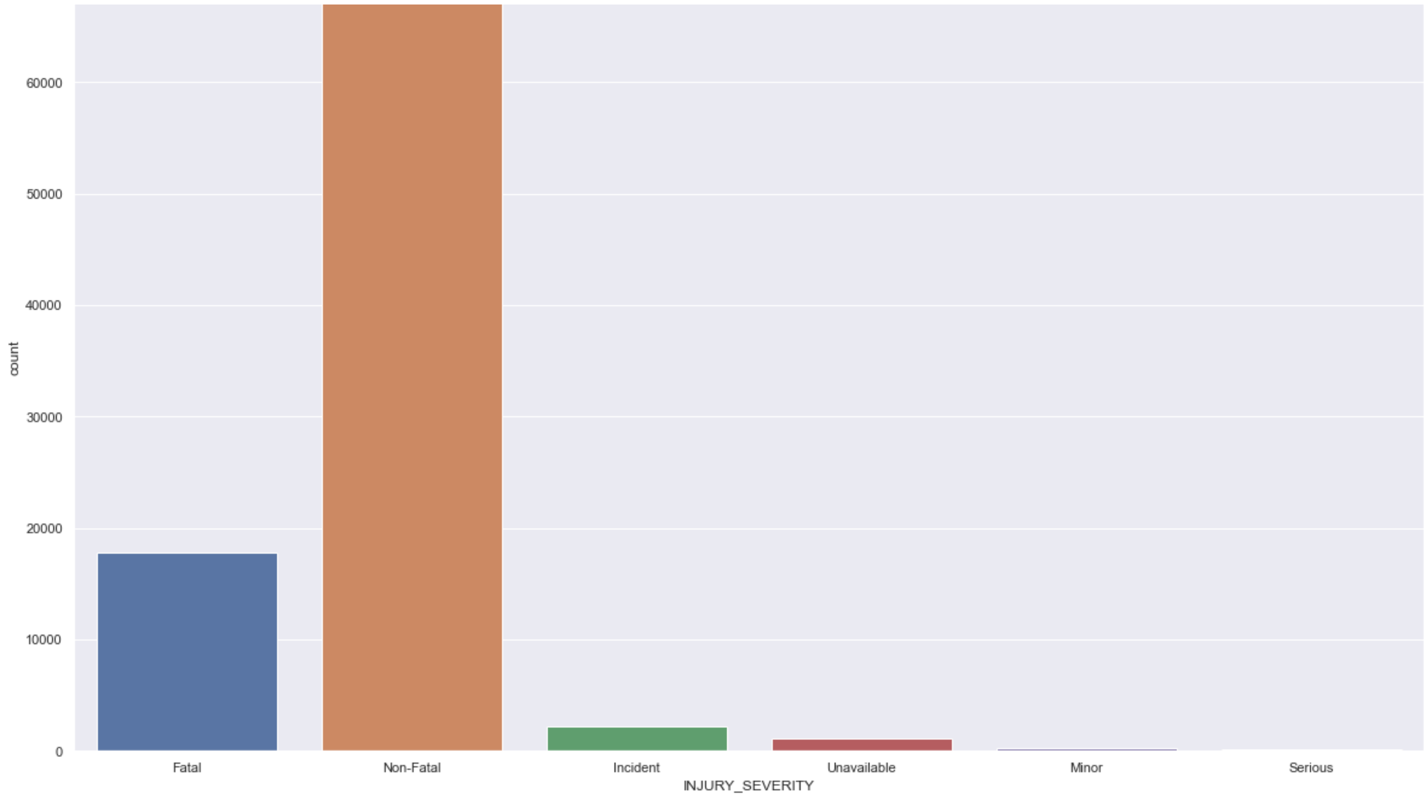
```
sns.countplot(data=aviation_data, x='INVESTIGATION_TYPE');
#We observe that accidents are more than incidents and by a very large percentage(96%)
```



ii)Check injury Severity column

In [81]:

```
sns.countplot(data=aviation_data,x='INJURY_SEVERITY');#Non fatal was the most common inju
ry severity followed by fatal
# Non-Fatal Injury: Any injury from an aviation event where the person survives.
# Fatal Injury: Death resulting from an aviation event, within 30 days.
# Serious Injury: Severe injury requiring extensive treatment or hospitalization, without
resulting in death.
# Minor Injury: Non-life-threatening injury with minimal medical impact.
# Incident: An event or operational issue affecting flight safety that does not result in
an accident, injury, or major damage.
```



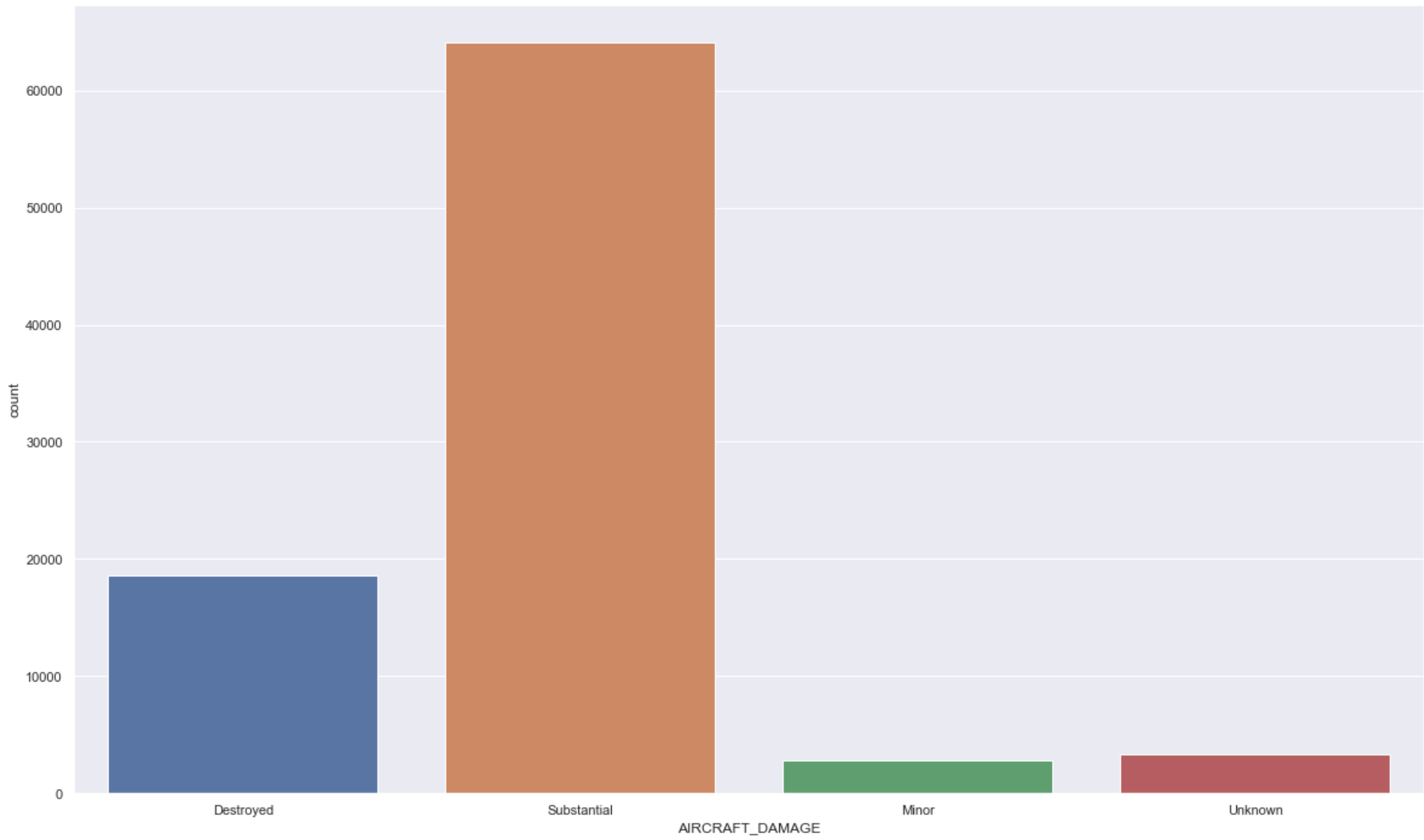
iii)Check aircraft damage

In [82]:

```
sns.countplot(data=aviation_data,x='AIRCRAFT_DAMAGE')# substantial damage is most common
# Destroyed: The aircraft is beyond repair and considered a total loss.
# Substantial Damage: Major structural damage that requires significant repair, but the a
ircraft can eventually be restored to service.
# Minor Damage: Superficial or minor repairs needed; the damage does not affect flight pe
rformance or safety.
```

Out[82]:

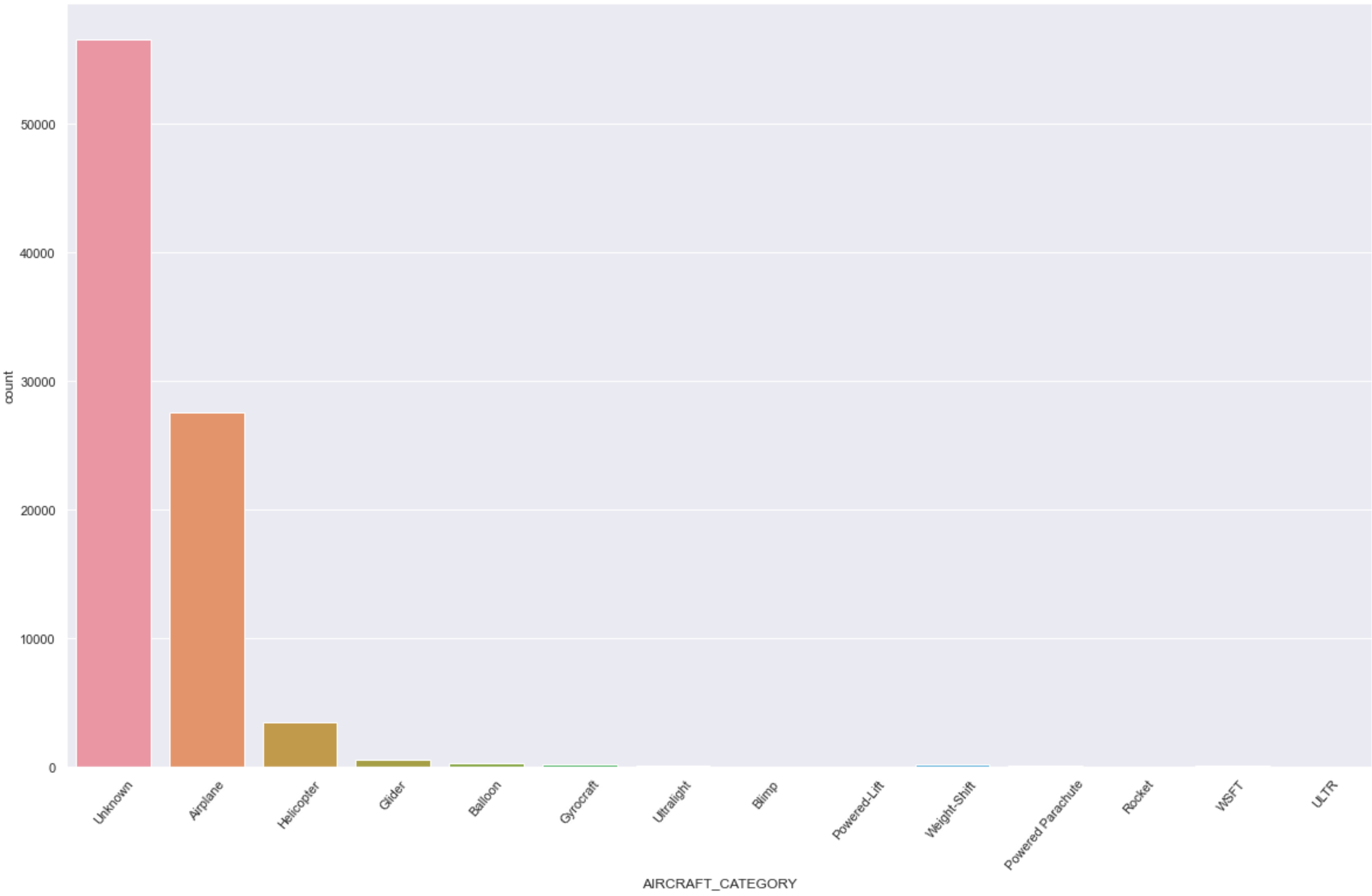
```
<AxesSubplot:xlabel='AIRCRAFT_DAMAGE', ylabel='count'>
```



iv)Check aircraft category

In [83]:

```
sns.countplot(data=aviation_data,x='AIRCRAFT_CATEGORY')
plt.xticks(rotation=50);#Airplane Category highest followed by helicopter and lastly ULTR
```



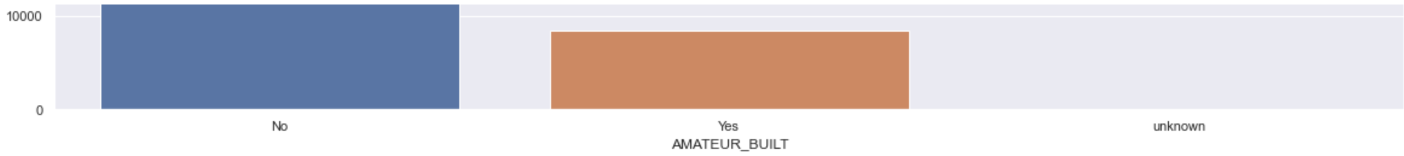
v)Check amateur built column

In [84]:

```
sns.countplot(data=aviation_data,x='AMATEUR_BUILT') #most aicrafts are not amateur built(
90%)
```

Out[84]:

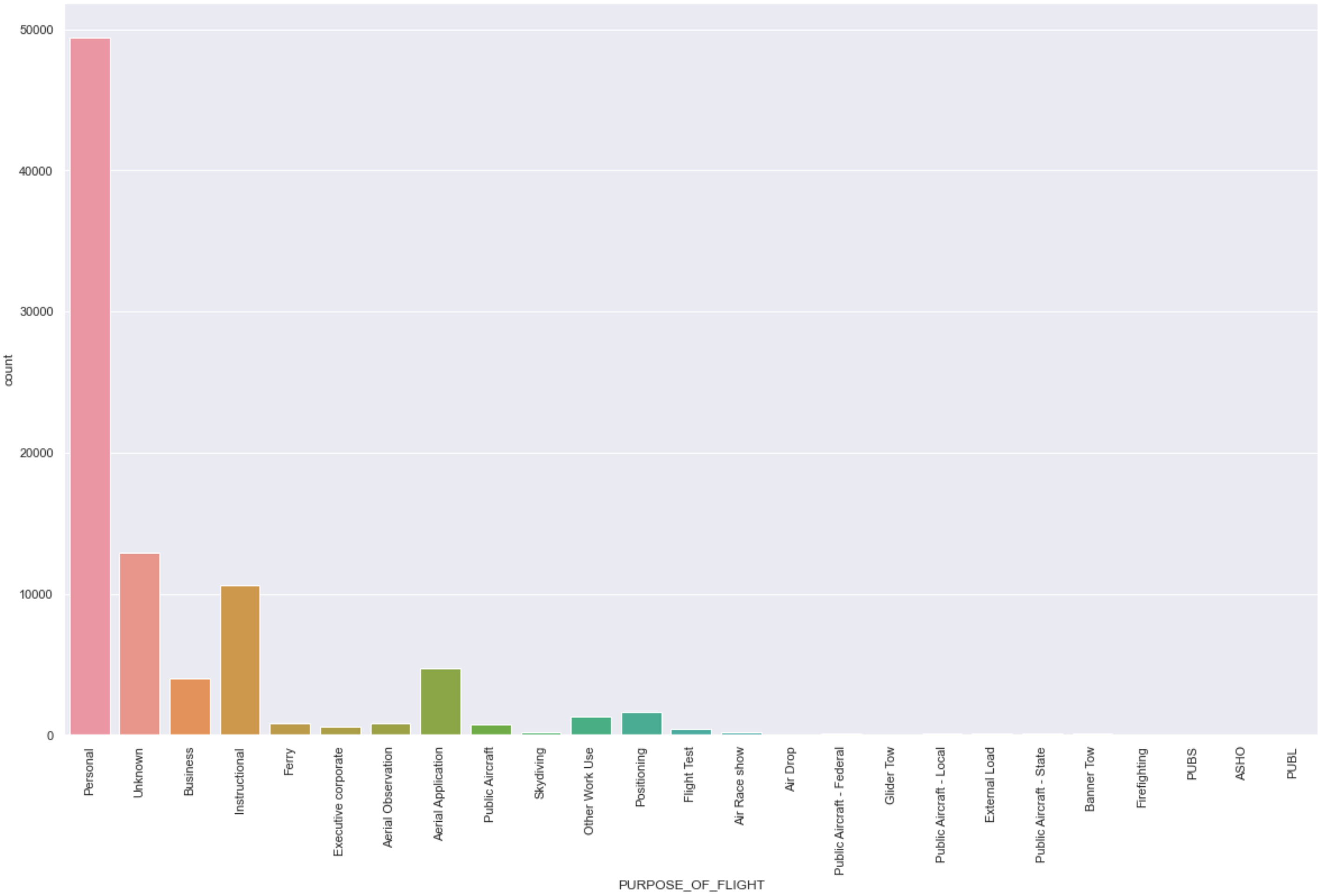
```
<AxesSubplot:xlabel='AMATEUR_BUILT', ylabel='count'>
```



vi)Purpose of flight column

In [85]:

```
sns.countplot(data=aviation_data,x='PURPOSE_OF_FLIGHT') #most flights were personal(more than half) followed by business then instructional
plt.xticks(rotation=90);
```



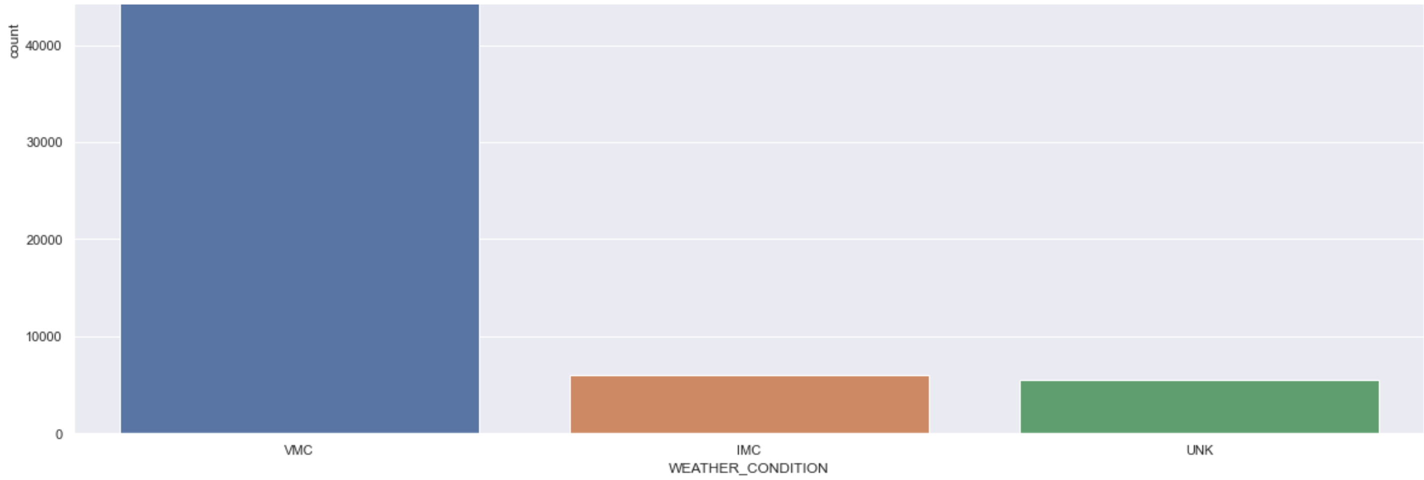
vii)Check weather column

In [86]:

```
weather_value = aviation_data['WEATHER_CONDITION'].value_counts().index
sns.countplot(data=aviation_data,x='WEATHER_CONDITION',order=weather_value) #VMC is the most common weather condition
```

Out[86]:

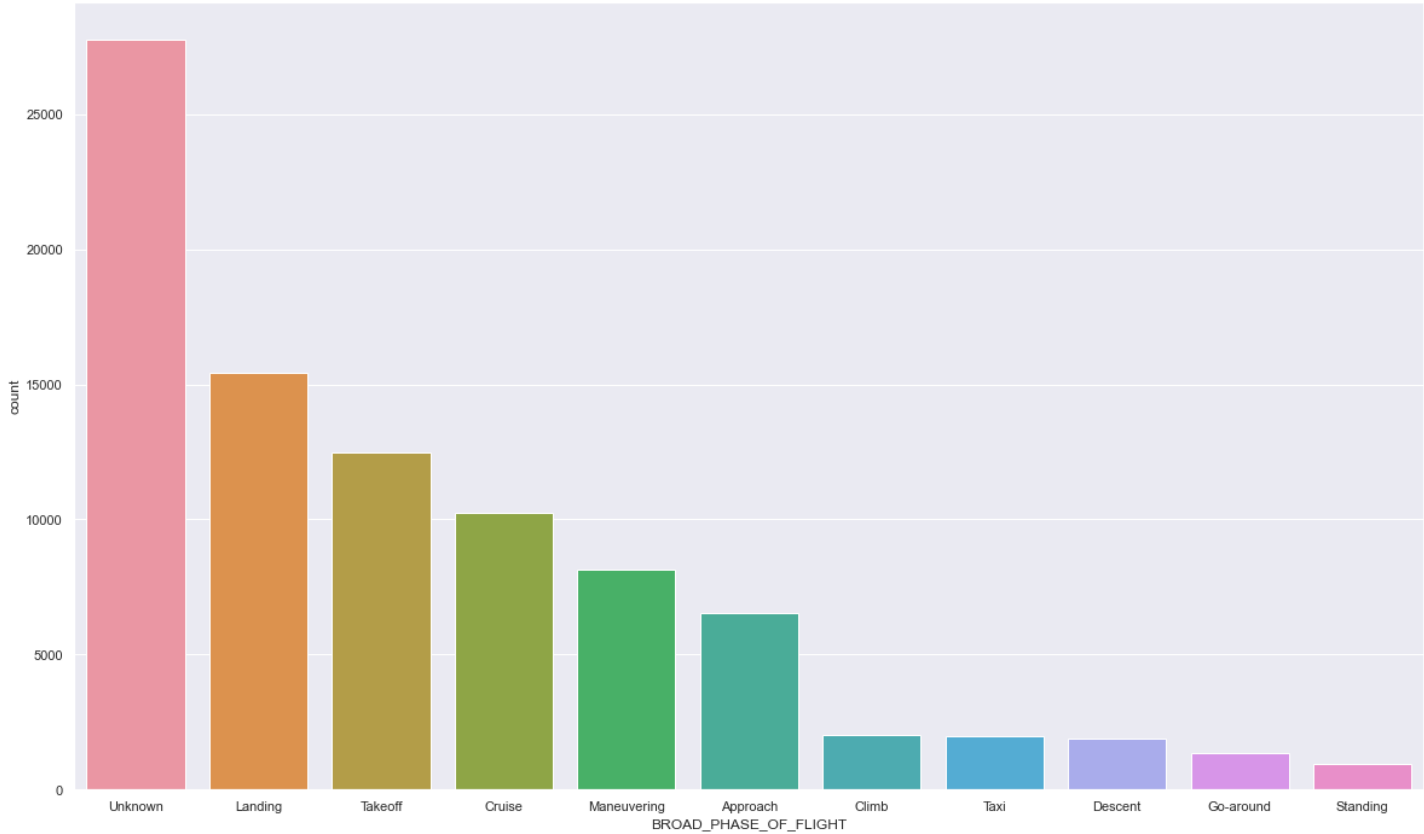
```
<AxesSubplot:xlabel='WEATHER_CONDITION', ylabel='count'>
```



viii)Check Broad Phase Of Flight column

In [87]:

```
broad_value= aviation_data['BROAD_PHASE_OF_FLIGHT'].value_counts().index
sns.countplot(data=aviation_data,x='BROAD_PHASE_OF_FLIGHT',order=broad_value); # landing
has the most cases followed by takeoff
```



ix)Check Event Date column

In [88]:

```
'''lets create column year, month and day that we could use for our analysis for this par
ticular analysis we need year but we can just add month and day probably for later '''
#1.Convert date to datetime format
aviation_data['EVENT_DATE'] = pd.to_datetime(aviation_data['EVENT_DATE'])
```

In [89]:

```
#2 get Year,Month and date column from the date
Year = aviation_data['EVENT_DATE'].dt.year
Month =aviation_data['EVENT_DATE'].dt.month
Day = aviation_data['EVENT_DATE'].dt.day
```

In [90]:

```
#3 insert this next to the Event date
aviation_data.insert(aviation_data.columns.get_loc('EVENT_DATE')+1,'YEAR',value=Year)
aviation_data.insert(aviation_data.columns.get_loc('EVENT_DATE')+2,'MONTH',value=Month)
aviation_data.insert(aviation_data.columns.get_loc('EVENT_DATE')+3,'DAY',value=Day)
```

In [91]:

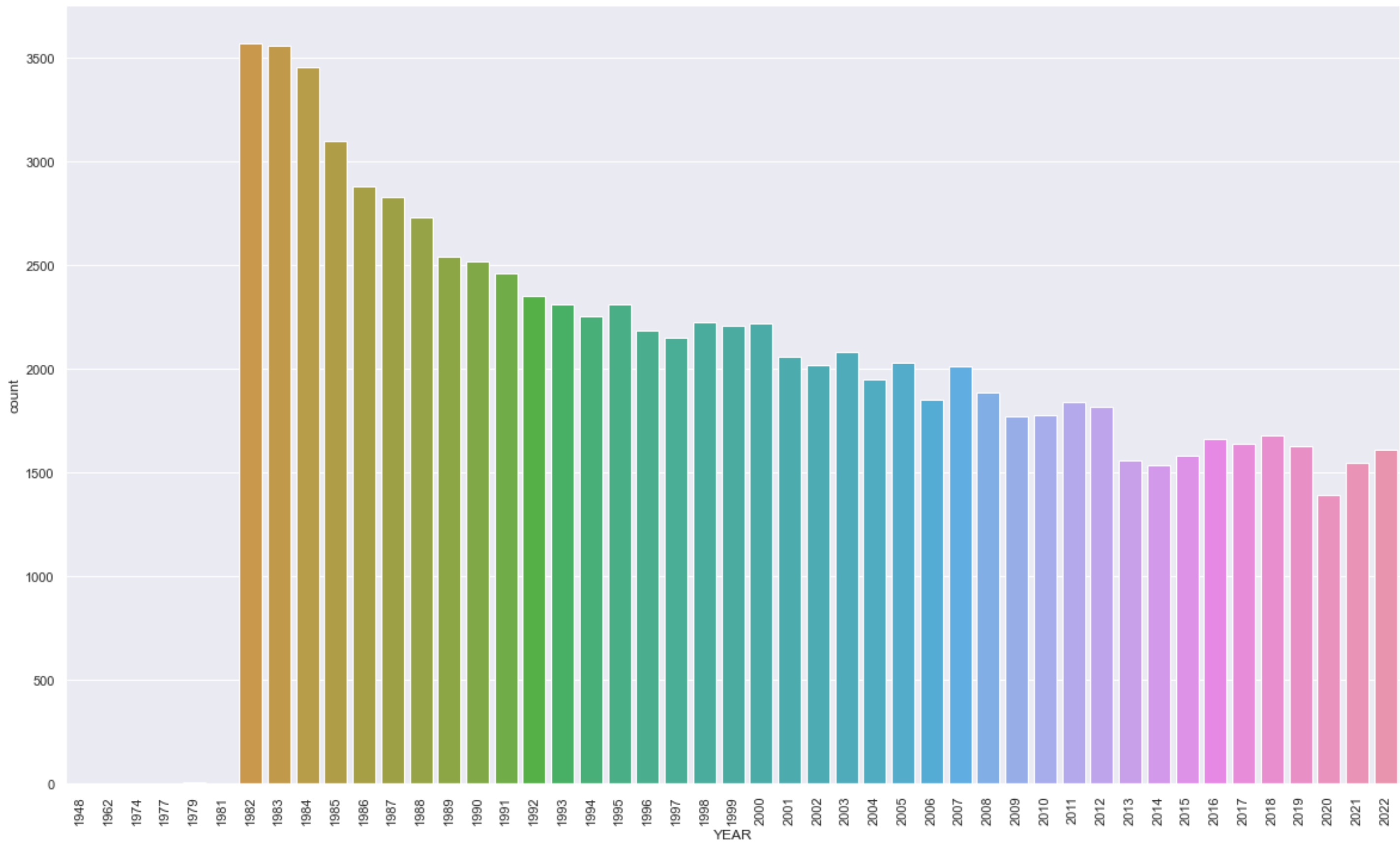
```
#4.Check if okay
aviation_data.head()
```

Out[91]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	YEAR	MONTH	DAY	COUNTRY	INJURY_SEV
0	20001218X45444	Accident	SEA87LA080	1948-10-24	1948	10	24	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	1962	7	19	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	1974	8	30	United States	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	1977	6	19	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	1979	8	2	United States	

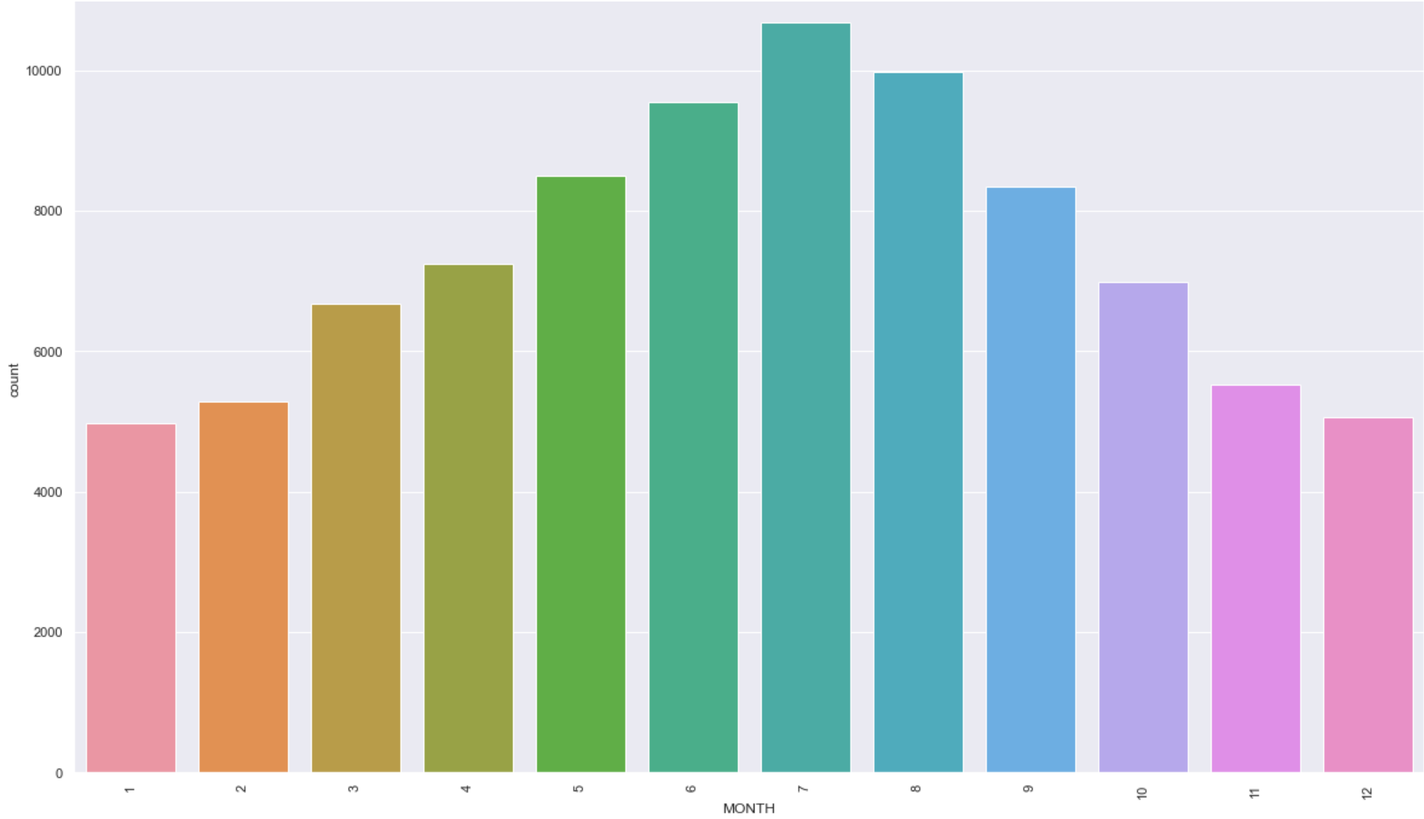
In [92]:

```
#5 plot years
sns.countplot(data=aviation_data,x='YEAR');
plt.xticks(rotation=90);#General decline in accidents
```



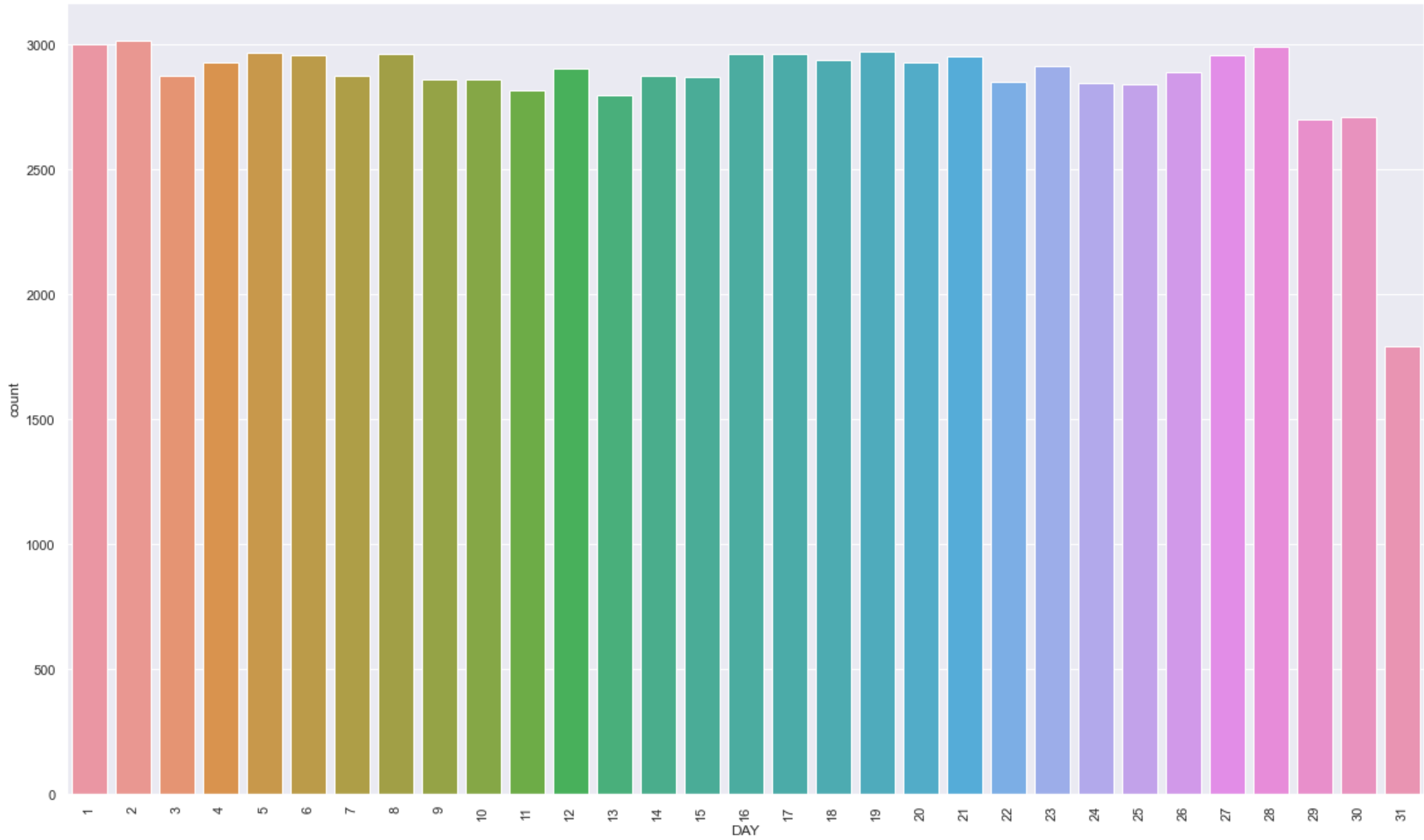
In [93]:

```
#6 plot months
sns.countplot(data=aviation_data,x='MONTH');
plt.xticks(rotation=90);#More accidents on the 6th,7th and 8th month#during summer
```



In [94]:

```
#5 plot days
sns.countplot(data=aviation_data,x='DAY');
plt.xticks(rotation=90);#no noticeable difference in days
```



x)Summary statistics for numeric columns

In [95]:

```
#get mean,median and skew on Fatal,serious,minor and uninjured columns
aviation_data[['TOTAL_FATAL_INJURIES','TOTAL_SERIOUS_INJURIES', 'TOTAL_MINOR_INJURIES', '
TOTAL_UNINJURED']].agg(['mean','median','max','min','skew']).T #data skewed positively w
ith minor ijuries being most skewed
```

Out[95]:

	mean	median	max	min	skew
TOTAL_FATAL_INJURIES	0.564493	0.0	349.0	0.0	35.308374
TOTAL_SERIOUS_INJURIES	0.240445	0.0	161.0	0.0	53.032755
TOTAL_MINOR_INJURIES	0.309258	0.0	380.0	0.0	93.348227
TOTAL_UNINJURED	5.034570	1.0	699.0	0.0	9.419666

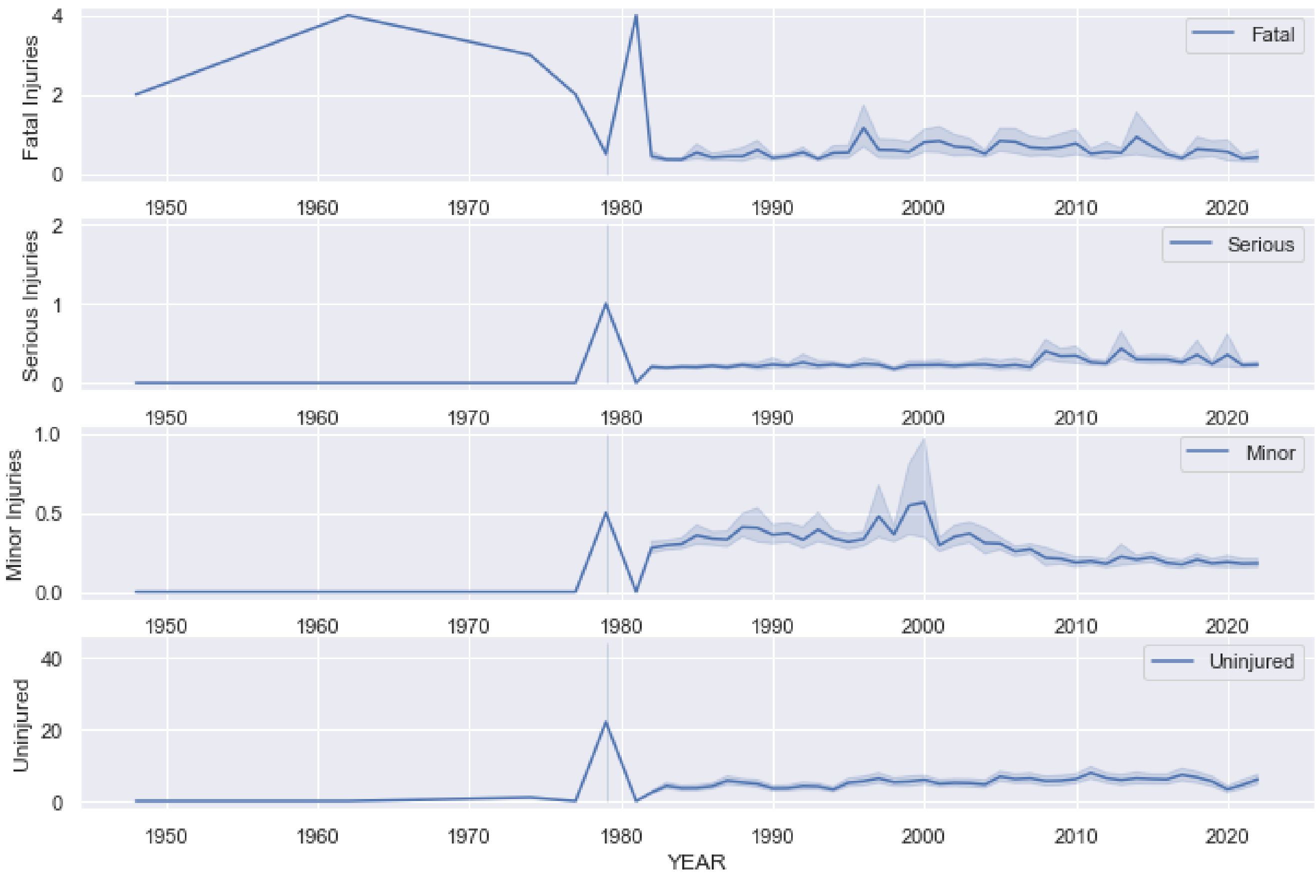
b)Bivariate Analysis

i)Check date and injury severity

In [96]:

```
fig, ax = plt.subplots(nrows=4,figsize=(12,8))
sns.lineplot(data=aviation_data,x='YEAR',y='TOTAL_FATAL_INJURIES',ax=ax[0],label='Fatal'
)
sns.lineplot(data=aviation_data,x='YEAR',y='TOTAL_SERIOUS_INJURIES',ax=ax[1],label='Seri
ous')
sns.lineplot(data=aviation_data,x='YEAR',y='TOTAL_MINOR_INJURIES',ax=ax[2],label='Minor'
)
sns.lineplot(data=aviation_data,x='YEAR',y='TOTAL_UNINJURED',ax=ax[3],label='Uninjured')

ax[0].set_ylabel('Fatal Injuries')
ax[1].set_ylabel('Serious Injuries')
ax[2].set_ylabel('Minor Injuries')
ax[3].set_ylabel('Uninjured');
#A general decline in totals of all as years went bar with uninjured topping the list
```



ii)Injury severity by make the aircraft

a)Private

In [97]:

```python
#check for private aicraft
private_aircraft = aviation_data.query('PURPOSE_OF_FLIGHT == "Personal"')
private_aircraft
```

Out[97]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	YEAR	MONTH	DAY	COUNTRY	INJURY
0	20001218X45444	Accident	SEA87LA080	1948-10-24	1948	10	24	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	1962	7	19	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	1974	8	30	United States	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	1977	6	19	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	1979	8	2	United States	
...	...	...	...	...	...	...	...	...	
88769	20221221106483	Accident	CEN23LA067	2022-12-21	2022	12	21	United States	
88772	20221227106491	Accident	ERA23LA093	2022-12-26	2022	12	26	United States	
88774	20221227106497	Accident	WPR23LA075	2022-12-26	2022	12	26	United States	
88775	20221227106498	Accident	WPR23LA076	2022-12-26	2022	12	26	United States	
88776	20221230106513	Accident	ERA23LA097	2022-12-29	2022	12	29	United States	

49413 rows × 23 columns

In [98]:

```python
make_counts = private_aircraft['MAKE'].value_counts()
#since makes are too many i will take top 20
top_makes = make_counts.nlargest(20).index
#filter data to only include this makes
private_aircraft_filtered = private_aircraft[private_aircraft['MAKE'].isin(top_makes)]
#cross tab to help in plotting injury severtity vs make
injury_make = pd.crosstab(private_aircraft_filtered['MAKE'],private_aircraft_filtered['IN
JURY_SEVERITY'])
injury_make
```

Out[98]:

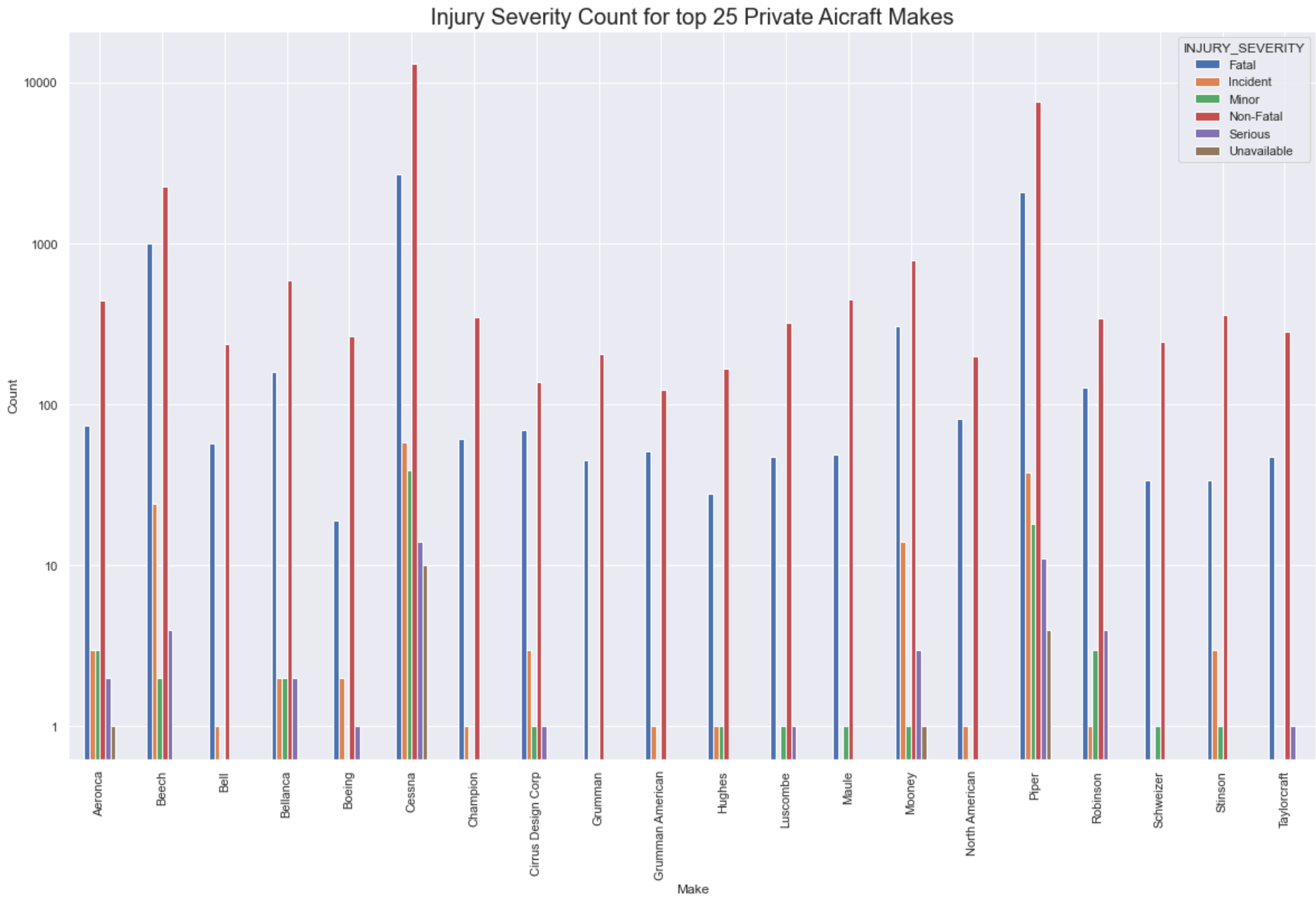
INJURY_SEVERITY	Fatal	Incident	Minor	Non-Fatal	Serious	Unavailable
MAKE						
Aeronca	74	3	3	445	2	1
Beech	999	24	2	2267	4	0
Bell	57	1	0	237	0	0
Bellanca	158	2	2	590	2	0
Boeing	19	2	0	267	1	0
Cessna	2704	58	39	13040	14	10
Champion	61	1	0	347	0	0
Cirrus Design Corp	69	3	1	138	1	0
Grumman	45	0	0	207	0	0

INJURY_SEVERITY MAKE	Fatal	Incident	Minor	Non-Fatal	Serious	Unavailable
Grumman American	51	1	0	124	0	0
Hughes	28	1	1	168	0	0
Luscombe	47	0	1	320	1	0
Maule	49	0	1	447	0	0
Mooney	306	14	1	784	3	1
North American	81	1	0	198	0	0
Piper	2088	38	18	7597	11	4
Robinson	127	1	3	340	4	0
Schweizer	34	0	1	244	0	0
Stinson	34	3	1	357	0	0
Taylorcraft	47	0	0	284	1	0

In [99]:

```
from matplotlib.ticker import ScalarFormatter

#fig,ax = plt.subplots(figsize=(20, 12))
ax= injury_make.plot(kind='bar')
ax.set_yscale('log')
ax.yaxis.set_major_formatter(ScalarFormatter())#prevents powers of 10 showing up
ax.set(xlabel='Make',ylabel='Count')
plt.title('Injury Severity Count for top 25 Private Aicraft Makes',fontsize=20);
#Boeing followed by sweinzwer and stinso have less fatal injuries
#Cessna followed by Piper has the highest no of minor injuries
```



b)Business

In [100]:

```
public_aircraft = aviation_data.query('PURPOSE_OF_FLIGHT == "Business"')
make_counts = public_aircraft['MAKE'].value_counts()
```

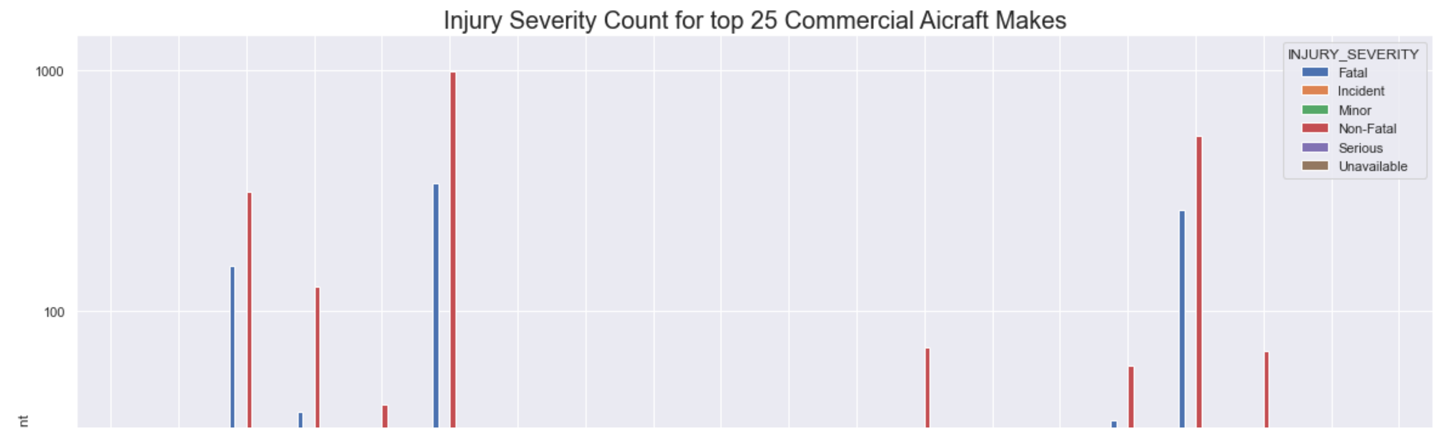
```python
#since makes are too many i will take top 20
top_makes = make_counts.nlargest(20).index
#filter data to only include this makes
public_aircraft_filtered = public_aircraft[public_aircraft['MAKE'].isin(top_makes)]
#cross tab to help in plotting injury severtity vs make
injury_make = pd.crosstab(public_aircraft_filtered['MAKE'],public_aircraft_filtered['INJURY_SEVERITY'])
injury_make
```

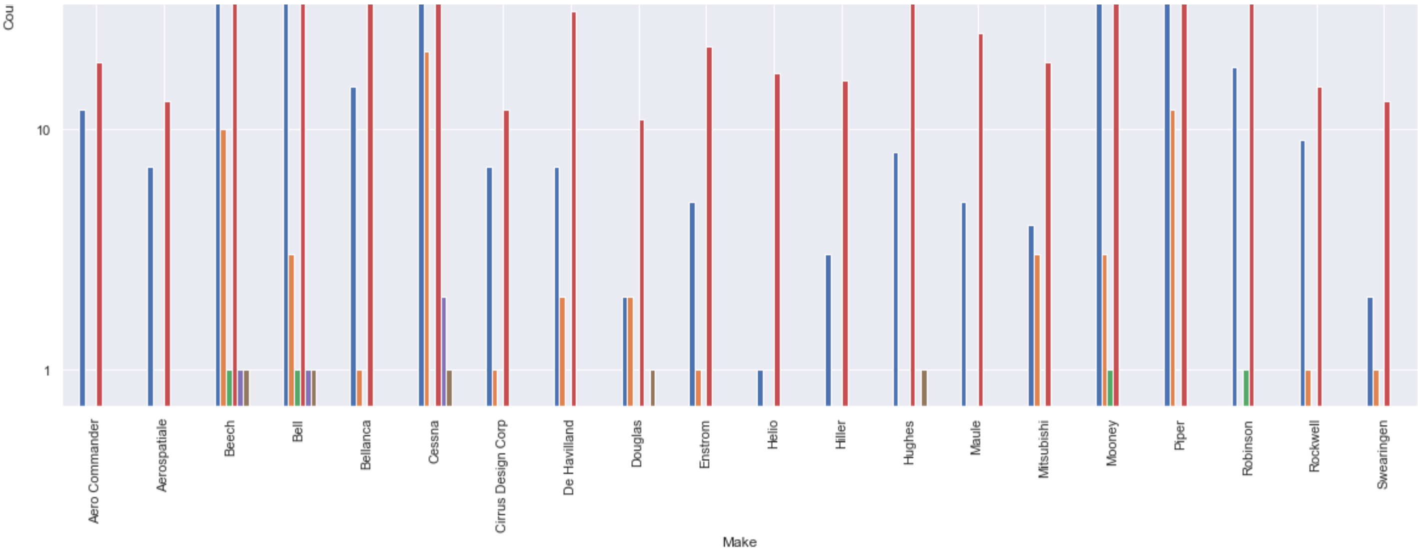
Out[100]:

INJURY_SEVERITY MAKE	Fatal	Incident	Minor	Non-Fatal	Serious	Unavailable
Aero Commander	12	0	0	19	0	0
Aerospatiale	7	0	0	13	0	0
Beech	154	10	1	314	1	1
Bell	38	3	1	127	1	1
Bellanca	15	1	0	41	0	0
Cessna	339	21	0	992	2	1
Cirrus Design Corp	7	1	0	12	0	0
De Havilland	7	2	0	31	0	0
Douglas	2	2	0	11	0	1
Enstrom	5	1	0	22	0	0
Helio	1	0	0	17	0	0
Hiller	3	0	0	16	0	0
Hughes	8	0	0	71	0	1
Maule	5	0	0	25	0	0
Mitsubishi	4	3	0	19	0	0
Mooney	35	3	1	59	0	0
Piper	264	12	0	535	0	0
Robinson	18	0	1	68	0	0
Rockwell	9	1	0	15	0	0
Swearingen	2	1	0	13	0	0

In [101]:

```python
from matplotlib.ticker import ScalarFormatter
ax= injury_make.plot(kind='bar')
ax.set_yscale('log')
ax.yaxis.set_major_formatter(ScalarFormatter())
ax.set(xlabel='Make',ylabel='Count')
plt.title('Injury Severity Count for top 25 Commercial Aicraft Makes',fontsize=20);
#Boeing has less fatal injuries followed by Helio and Swearingen
#Boeng has highest minor injuries followed by Bell
```





4. Answering Questions

i)Low risk Aircraft Based on Injury_severity

a)private Aircraft by Accidents

In [102]:

```
#filter data to accomodate only   private flights
private = aviation_data.query('PURPOSE_OF_FLIGHT == "Personal"')
private
```

Out[102]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	YEAR	MONTH	DAY	COUNTRY	INJURY
0	20001218X45444	Accident	SEA87LA080	1948-10-24	1948	10	24	United States	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	1962	7	19	United States	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	1974	8	30	United States	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	1977	6	19	United States	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	1979	8	2	United States	
...	...	...	...	...	...	...	...	...	
88769	20221221106483	Accident	CEN23LA067	2022-12-21	2022	12	21	United States	
88772	20221227106491	Accident	ERA23LA093	2022-12-26	2022	12	26	United States	
88774	20221227106497	Accident	WPR23LA075	2022-12-26	2022	12	26	United States	
88775	20221227106498	Accident	WPR23LA076	2022-12-26	2022	12	26	United States	
88776	20221230106513	Accident	ERA23LA097	2022-12-29	2022	12	29	United States	

49413 rows × 23 columns

In [103]:

```
#Filter severity to minor
severity_minor = private.query('INJURY_SEVERITY =="Minor"')
severity_minor
```

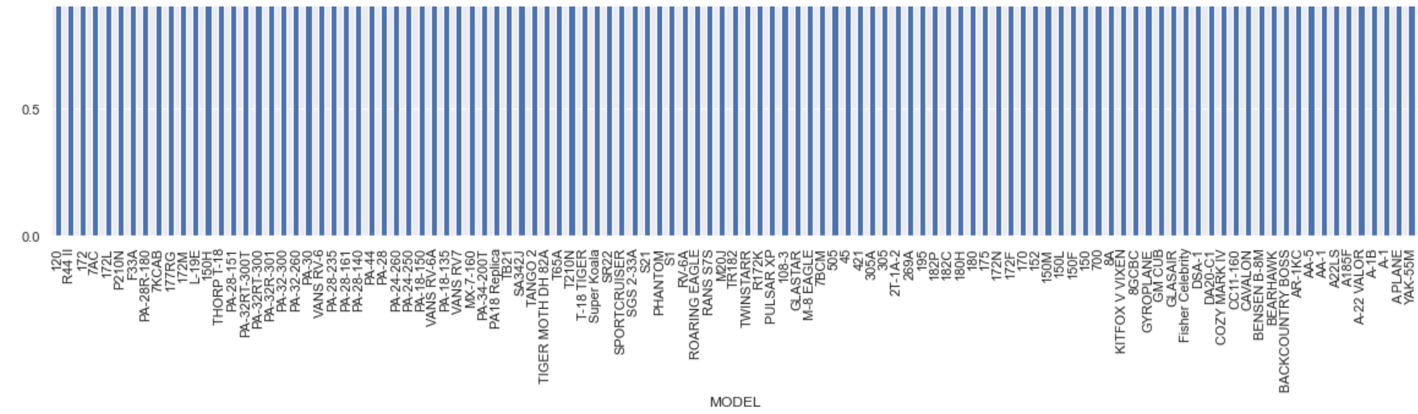
Out[103]:

	EVENT_ID	INVESTIGATION_TYPE	ACCIDENT_NUMBER	EVENT_DATE	YEAR	MONTH	DAY	COUNTRY	INJURY
87177	20220103104480	Accident	WPR22LA072	2022-01-03	2022	1	3	United States	
87221	20220119104542	Accident	ANC22LA013	2022-01-16	2022	1	16	United States	
87223	20220118104534	Accident	WPR22LA081	2022-01-18	2022	1	18	United States	
87237	20220124104548	Accident	CEN22LA106	2022-01-23	2022	1	23	United States	
87242	20220126104557	Accident	WPR22LA083	2022-01-26	2022	1	26	United States	
...	...	...	...	...	...	...	...	...	
88765	20221219106470	Accident	ERA23LA091	2022-12-16	2022	12	16	United States	
88766	20221227106496	Accident	WPR23LA074	2022-12-17	2022	12	17	United States	
88769	20221221106483	Accident	CEN23LA067	2022-12-21	2022	12	21	United States	
88772	20221227106491	Accident	ERA23LA093	2022-12-26	2022	12	26	United States	
88776	20221230106513	Accident	ERA23LA097	2022-12-29	2022	12	29	United States	

125 rows × 23 columns

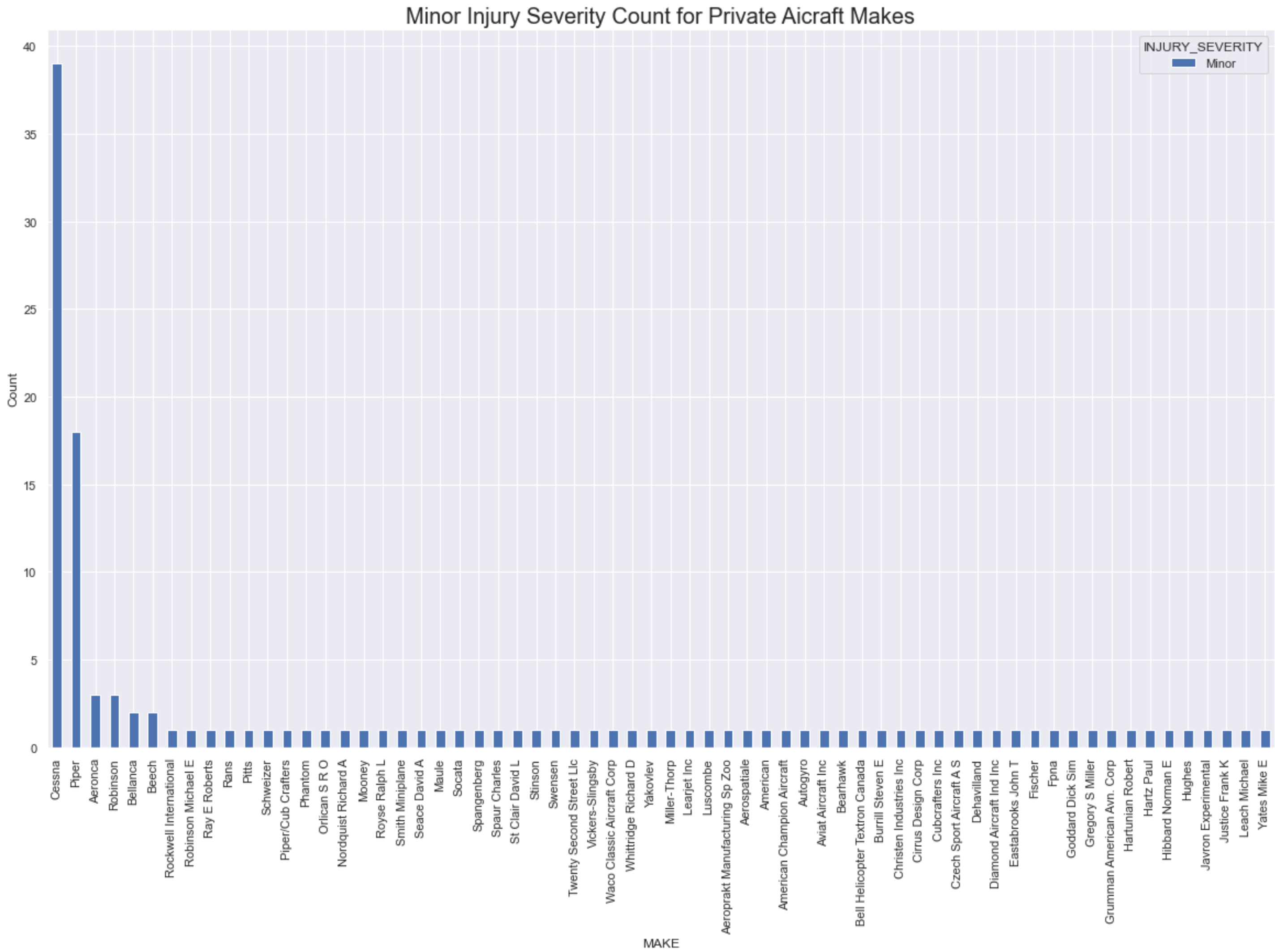
In [104]:

```
## check the Model of these aircrafts with minor injury
make_minor = pd.crosstab(severity_minor['MODEL'],severity_minor['INJURY_SEVERITY']).sort
_values('Minor',ascending=False)
ax=make_minor.plot(kind='bar')
plt.title('Minor Injury Severity Count for Private Aicraft Models',fontsize=20);
ax.set(xlabel='MODEL',ylabel='Count');
 ##use Model since its more descriptive than Make so we see that CESSNA 120 followed by R
obinson R44 11 then Cessna 172 have the highest minor injuries
```



In [105]:

```
# using make
## check the make of these aircrafts with minor injury
make_minor = pd.crosstab(severity_minor['MAKE'],severity_minor['INJURY_SEVERITY']).sort_
values('Minor',ascending=False)
ax=make_minor.plot(kind='bar')
plt.title('Minor Injury Severity Count for Private Aicraft Makes',fontsize=20);
ax.set(xlabel='MAKE',ylabel='Count'); ##Cessna followed by Piper then Aeronca
```

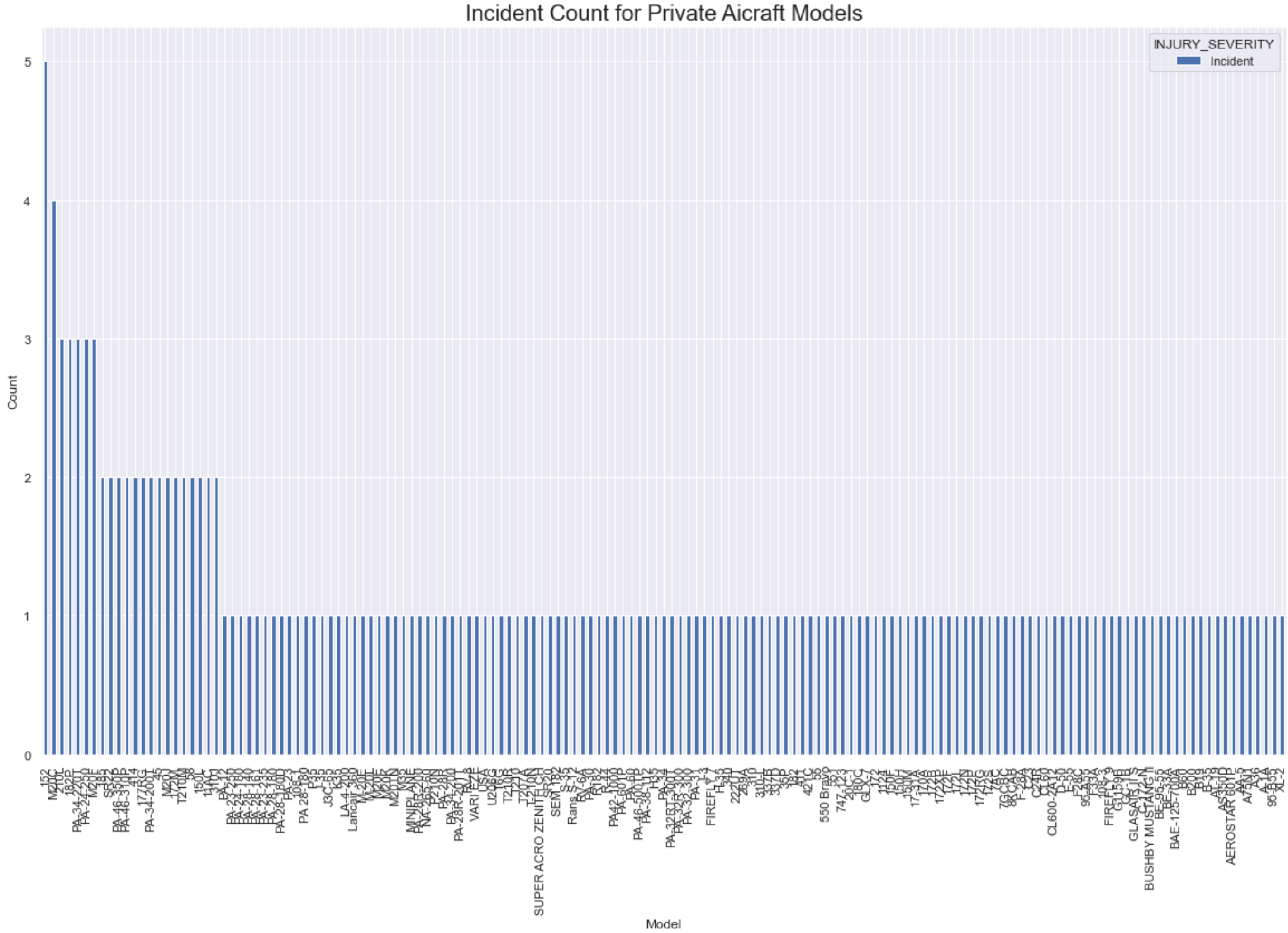


b)private Aircraft by incidents

In [106]:

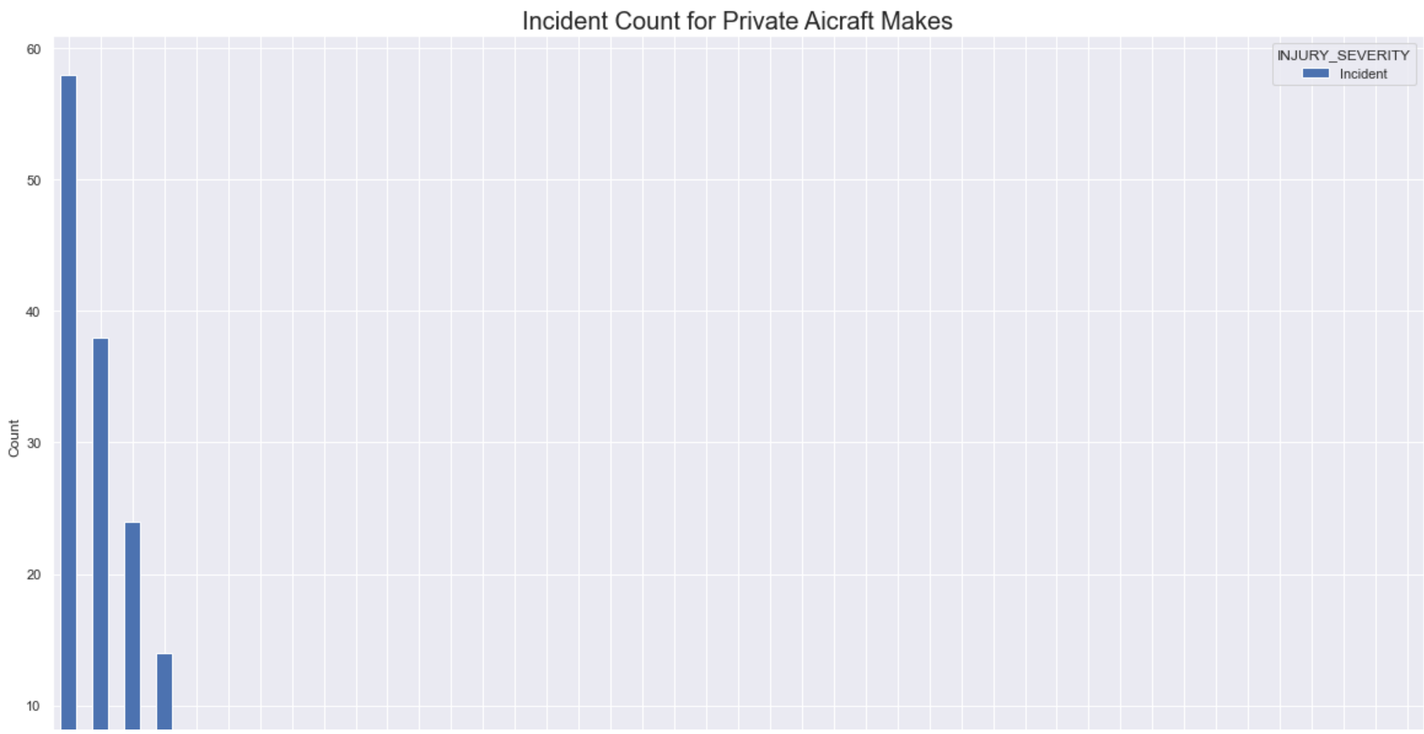
```
severity_Incident= private.query('INJURY_SEVERITY =="Incident"')
## check the Model of these aircrafts with incidents
make_incident = pd.crosstab(severity_Incident['MODEL'],severity_Incident['INJURY_SEVERIT
Y']).sort_values('Incident',ascending=False)
ax=make_incident.plot(kind='bar');
plt.title('Incident Count for Private Aicraft Models',fontsize=20);
```

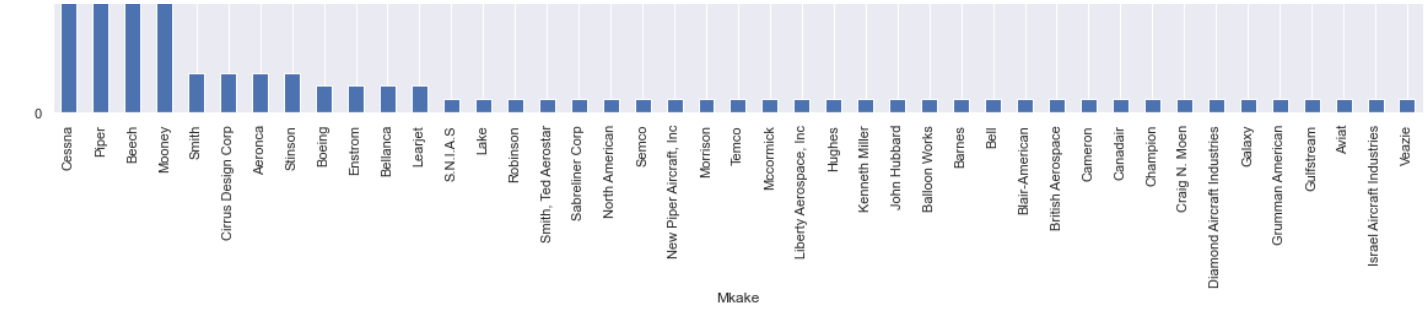
```
ax.set(xlabel='Model',ylabel='Count');
 ##use Model since its more descriptive than Make Cessna 152 followed by Moooney M20C the
n Cessna 210l
```



In [107]:

```
## check the Make of these aircrafts with incidents
make_incident = pd.crosstab(severity_Incident['MAKE'],severity_Incident['INJURY_SEVERITY
']).sort_values('Incident',ascending=False)
ax=make_incident.plot(kind='bar');
plt.title('Incident Count for Private Aicraft Makes',fontsize=20);
ax.set(xlabel='Mkake',ylabel='Count');
 #Cessna followed by Piper then Beech
```

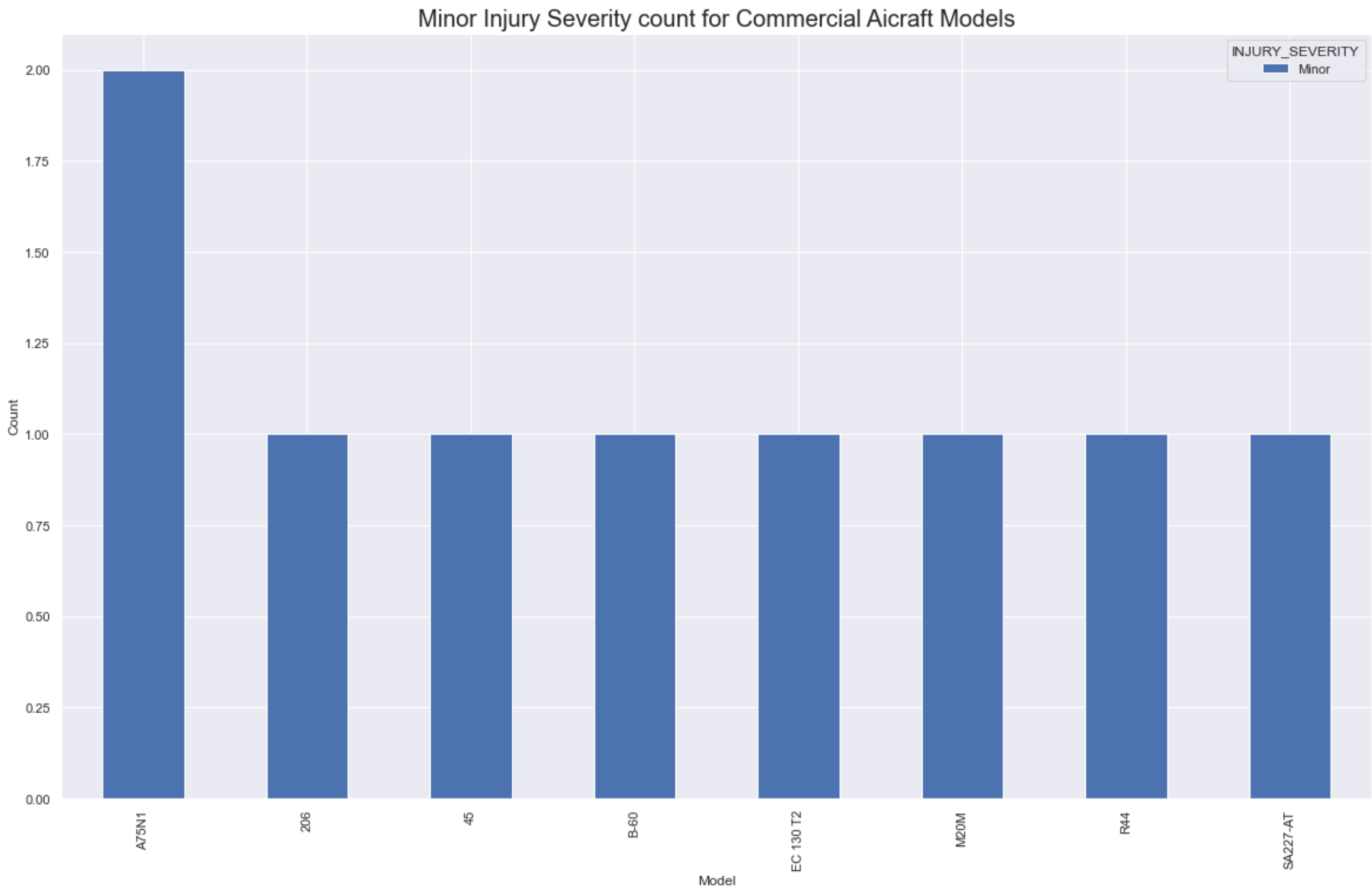




c)Commercial aircraft by accidents

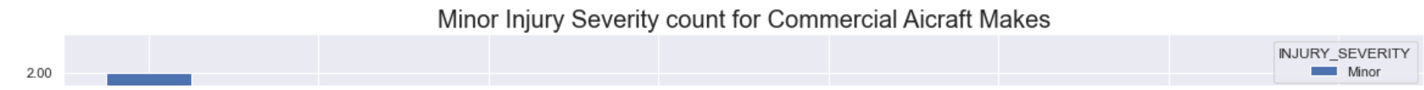
In [108]:

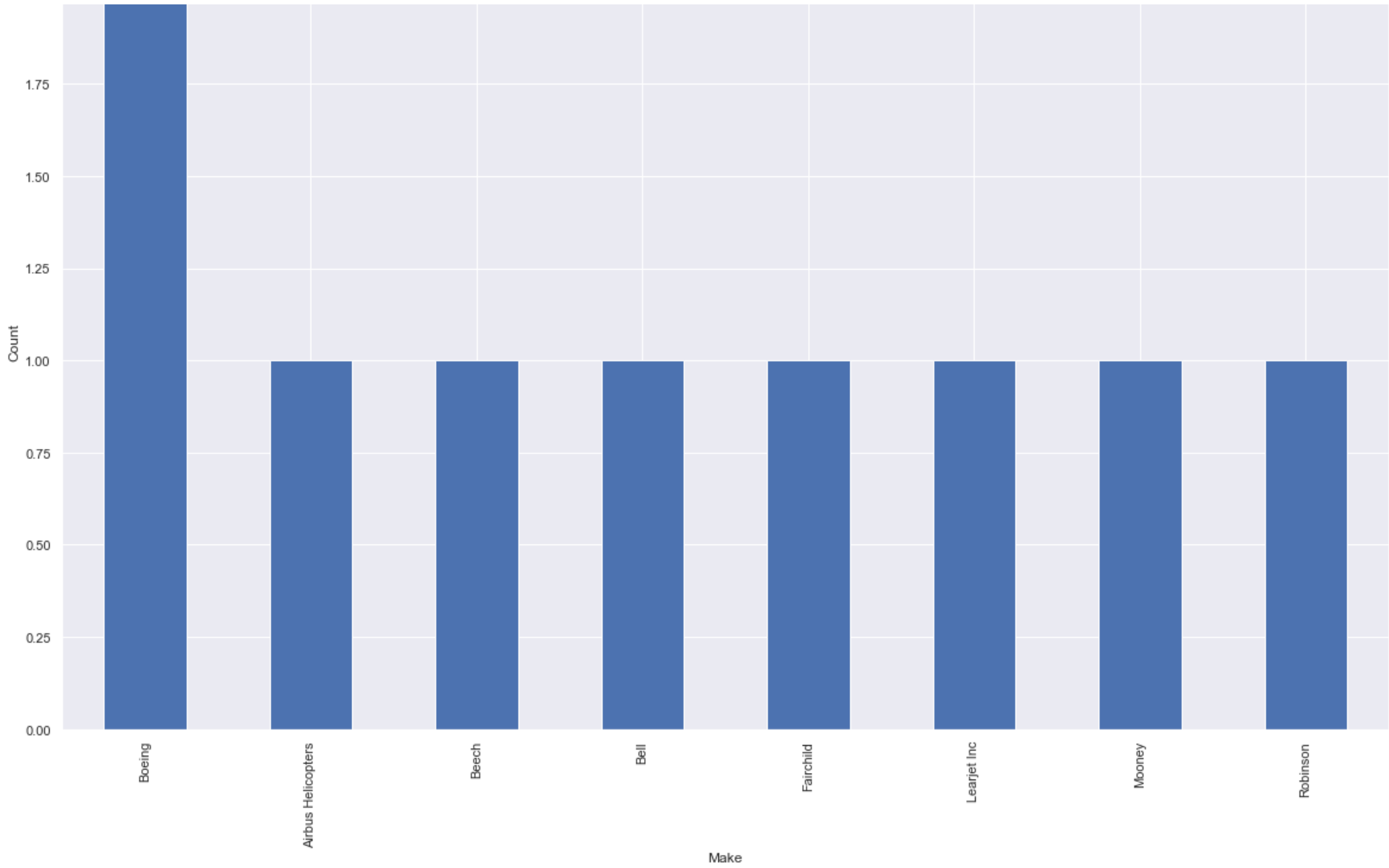
```
#filter data to accomodate only corporate and private flights and hw
commercial = aviation_data.query('PURPOSE_OF_FLIGHT == "Business"')
#Filter to minor incidents
severity_minor2 = commercial.query('INJURY_SEVERITY =="Minor"')
## check the Model of these aircrafts with minor injury
make_minor2 = pd.crosstab(severity_minor2['MODEL'],severity_minor2['INJURY_SEVERITY']).s
ort_values('Minor',ascending=False)
ax=make_minor2.plot(kind='bar')
plt.title('Minor Injury Severity count for Commercial Aicraft Models',fontsize=20);
ax.set(xlabel='Model',ylabel='Count');
#Boeing A75N1 has highest minor injuries others same
```



In [109]:

```
## check the Make of these aircrafts with minor injury
make_minor2 = pd.crosstab(severity_minor2['MAKE'],severity_minor2['INJURY_SEVERITY']).so
rt_values('Minor',ascending=False)
ax=make_minor2.plot(kind='bar');
plt.title('Minor Injury Severity count for Commercial Aicraft Makes',fontsize=20);
ax.set(xlabel='Make',ylabel='Count');
#Boeing others are same
```

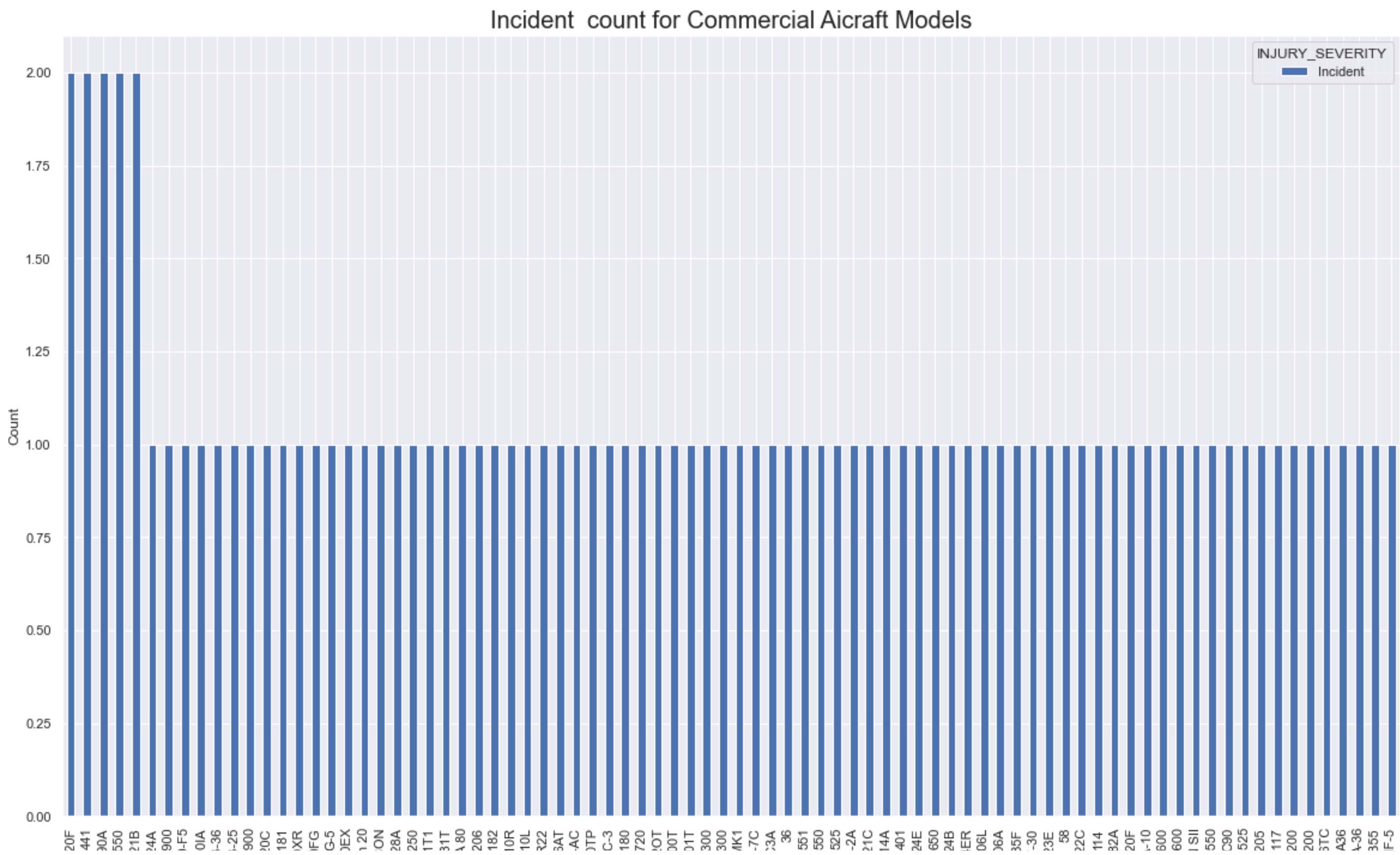


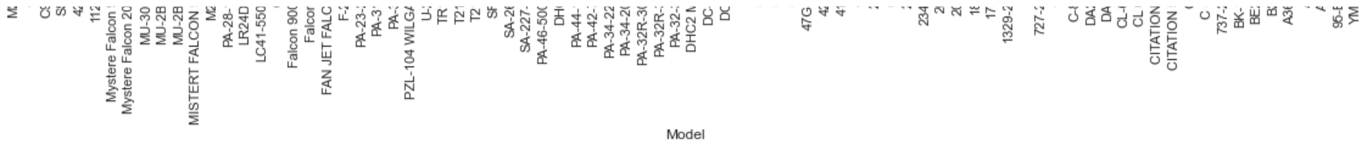


d)Commercial aircrafts by incidents

In [110]:

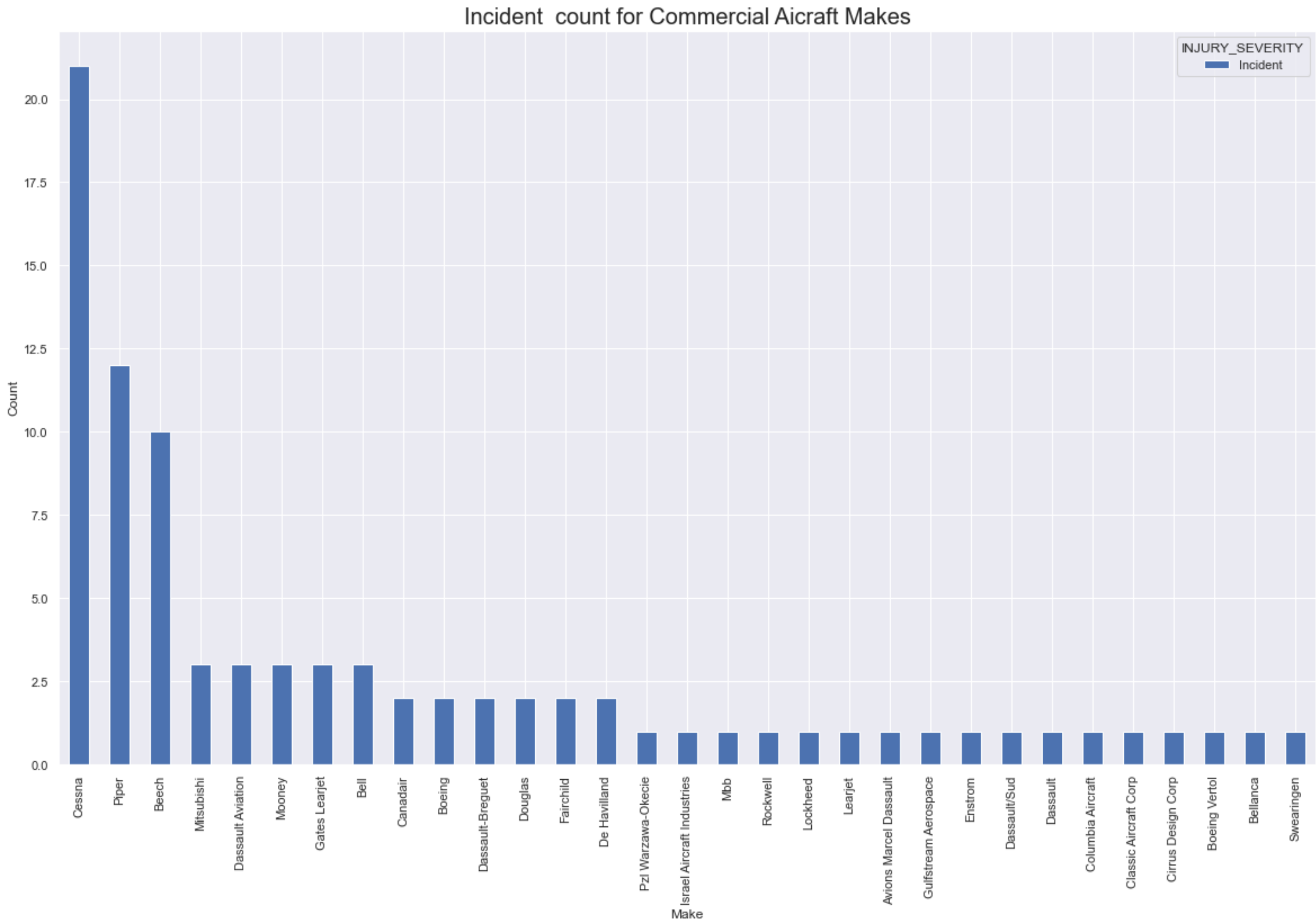
```
severity_Incident2 = commercial.query('INJURY_SEVERITY =="Incident"')
## check the Model of these aircrafts with minor incidenta
make_incident2 = pd.crosstab(severity_Incident2['MODEL'],severity_Incident2['INJURY_SEVER
ITY']).sort_values('Incident',ascending=False)
ax=make_incident2.plot(kind='bar')
plt.title('Incident  count for Commercial Aicraft Models',fontsize=20);
ax.set(xlabel='Model',ylabel='Count');#Mooney  M20F,cessna 441,Beech C90A,cessna S550 wi
th same probablity
```





In [111]:

```
## check the Make of these aircrafts with minor incidents
make_incident2 = pd.crosstab(severity_Incident2['MAKE'],severity_Incident2['INJURY_SEVERI
TY']).sort_values('Incident',ascending=False)
ax=make_incident2.plot(kind='bar')
plt.title('Incident  count for Commercial Aicraft Makes',fontsize=20);
ax.set(xlabel='Make',ylabel='Count'); #Cessna ,Piper, Beech
```

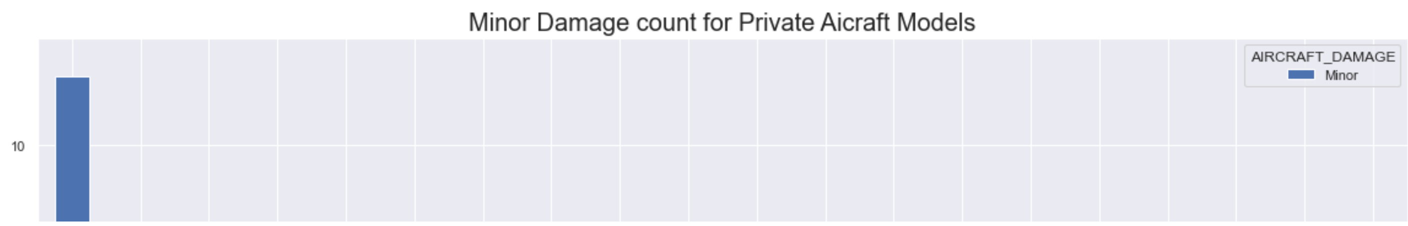


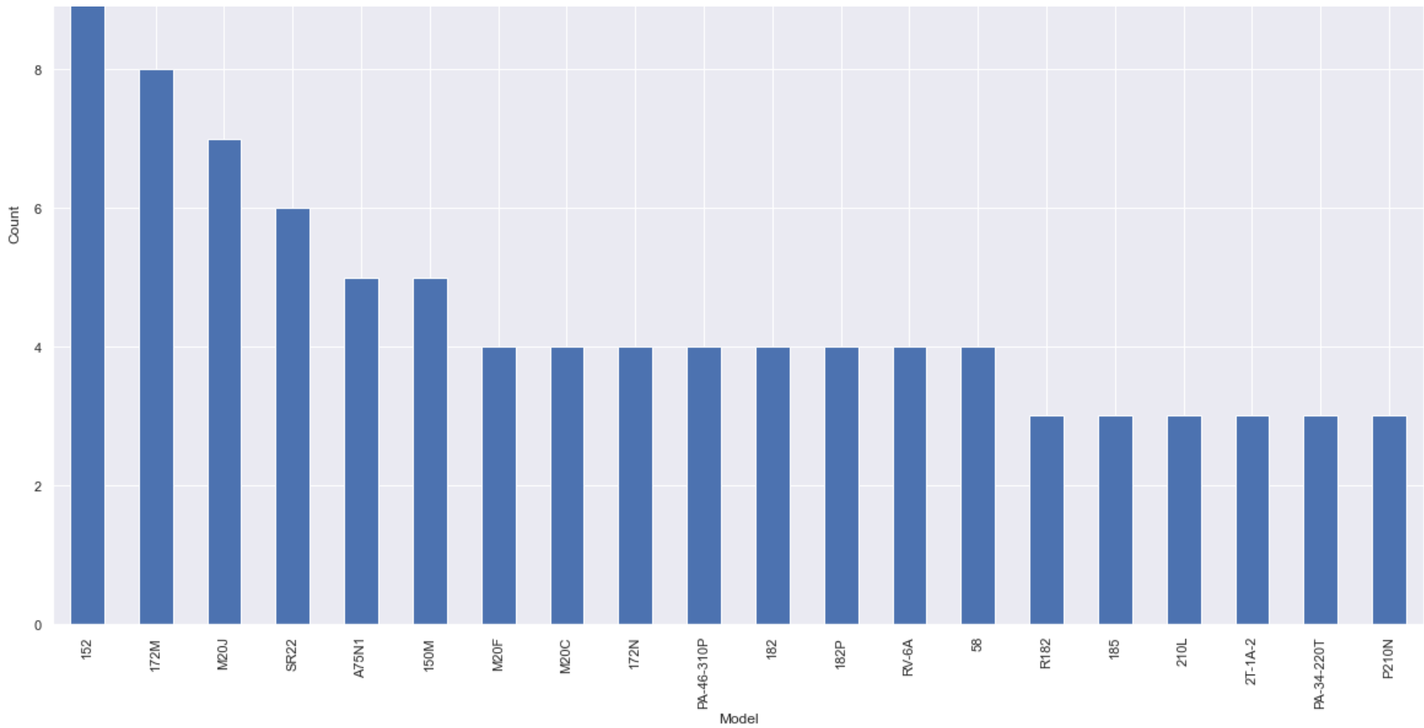
ii)Low risk Aircraft Based on Aicraft Damage

a)private Aircraft

In [116]:

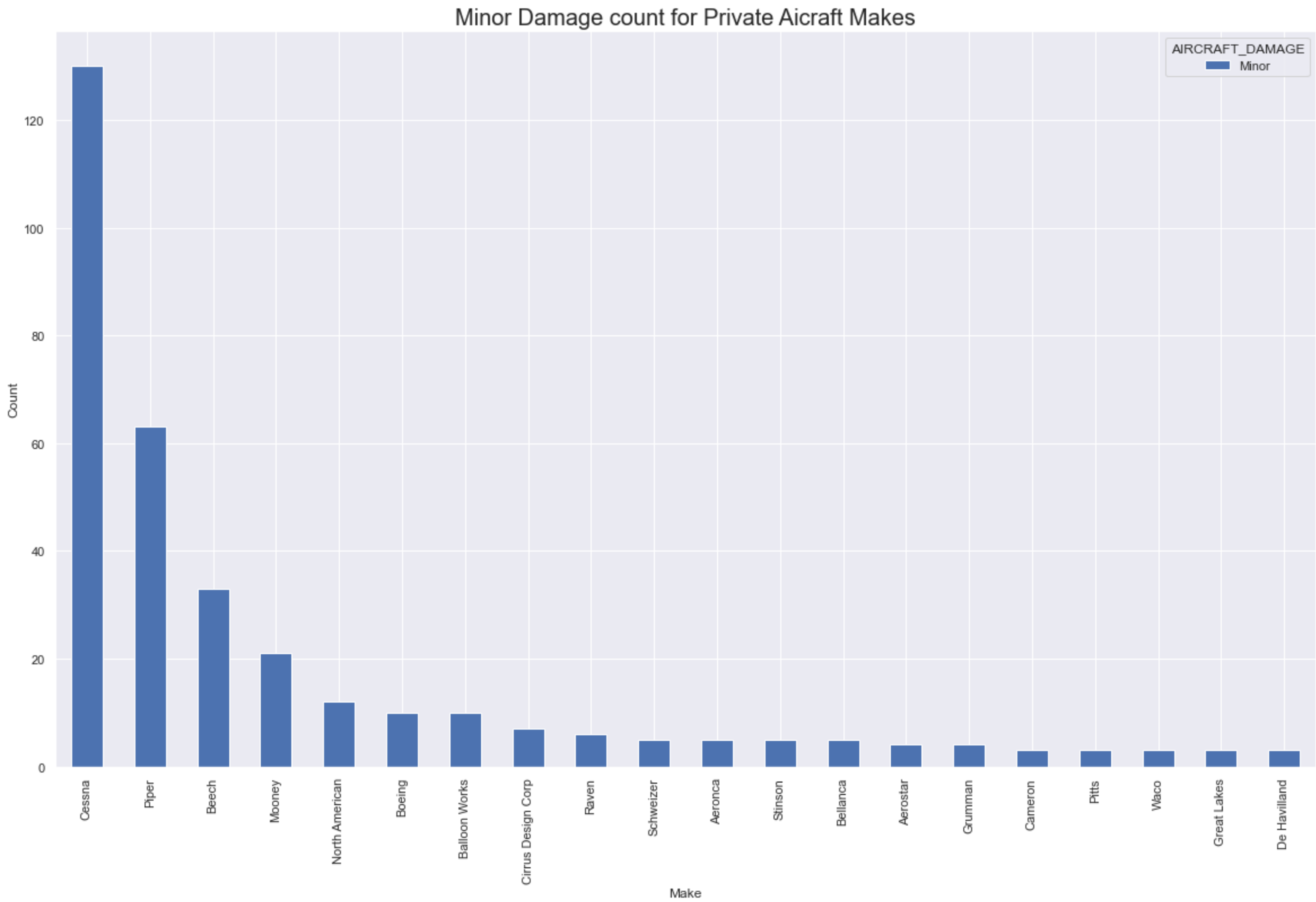
```
damage = private.query('AIRCRAFT_DAMAGE =="Minor"')
## check the Model of these aircrafts with minor injury  Pick top 20
damage_minor = pd.crosstab(damage['MODEL'],damage['AIRCRAFT_DAMAGE']).sort_values('Minor
',ascending=False)[:20]
ax=damage_minor.plot(kind='bar')
plt.title('Minor Damage count for Private Aicraft Models',fontsize=20);
ax.set(xlabel='Model',ylabel='Count'); #Cessna 152 ,folowed by Cessna 172M then Mooney M
20J has highest minor damages to aircrafts
```





In [117]:

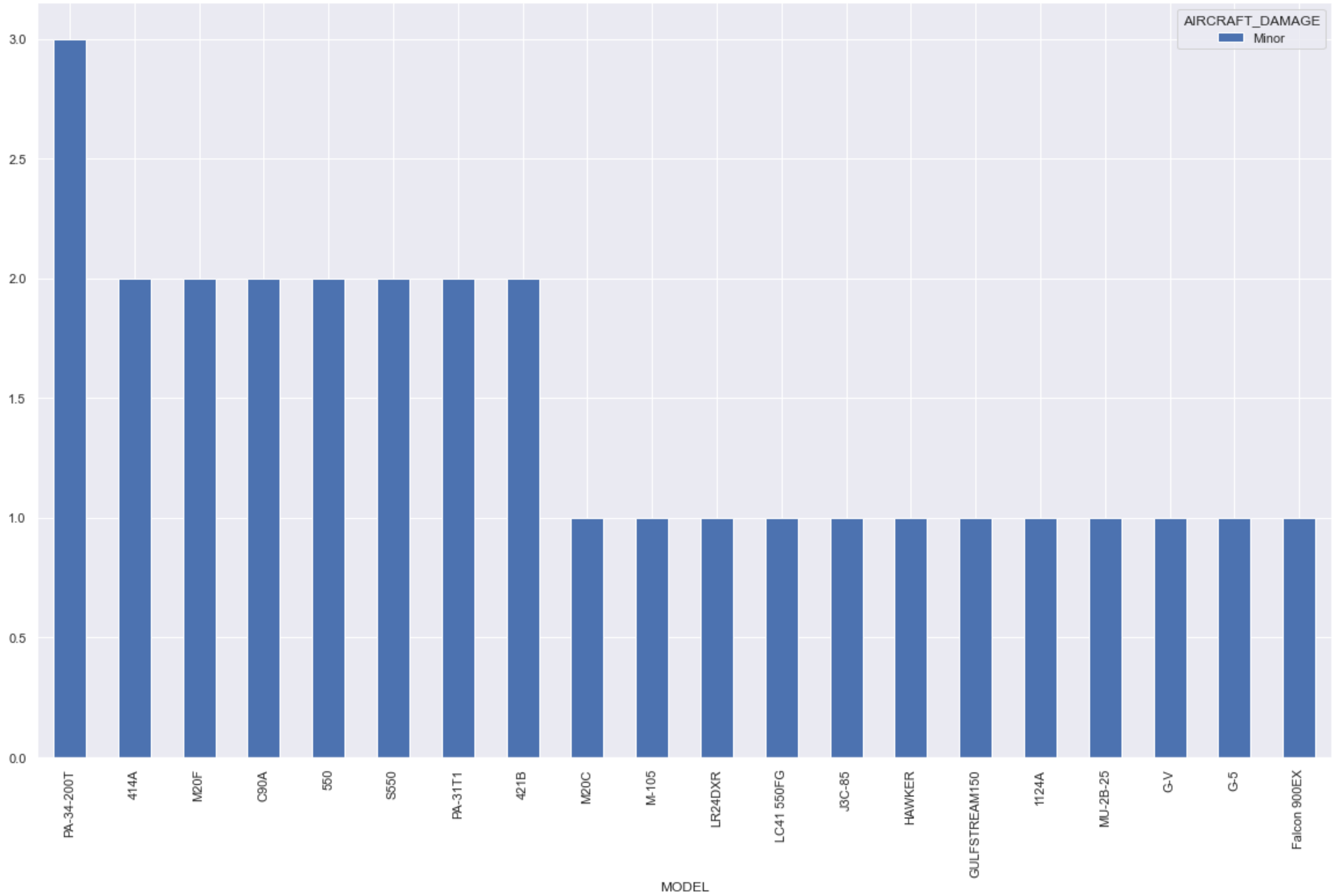
```
#check make
damage_minor = pd.crosstab(damage['MAKE'],damage['AIRCRAFT_DAMAGE']).sort_values('Minor'
,ascending=False)[:20]
ax=damage_minor.plot(kind='bar')
plt.title('Minor Damage count for Private Aicraft Makes',fontsize=20);
ax.set(xlabel='Make',ylabel='Count');
#Cessna,Piper beach
```



b)Commercial aircrafts

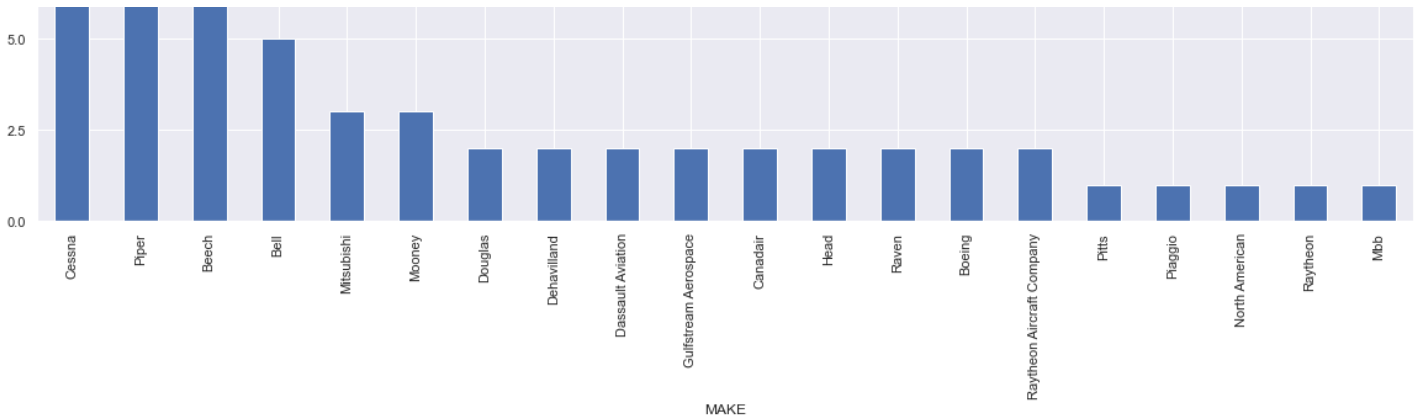
In [114]:

```
damage = commercial.query('AIRCRAFT_DAMAGE =="Minor"')
## check the Model of these aircrafts with minor injury  Pick top 20
damage_minor = pd.crosstab(damage['MODEL'],damage['AIRCRAFT_DAMAGE']).sort_values('Minor
',ascending=False)[:20]
ax=damage_minor.plot(kind='bar')
plt.title('Minor Damage count for Commercial Aicraft Models',fontsize=20);
ax.set(xlabel='Models',ylabel='Count'); #piper PA-34-200T has highest minor damages to ai
rcrafts
```



In [118]:

```
damage = commercial.query('AIRCRAFT_DAMAGE =="Minor"')
## check the Make of these aircrafts with minor injury
damage_minor = pd.crosstab(damage['MAKE'],damage['AIRCRAFT_DAMAGE']).sort_values('Minor'
,ascending=False)[:20]
damage_minor.plot(kind='bar')
plt.title('Minor Damage count for Commercial Aicraft Makes',fontsize=20);
ax.set(xlabel='Make',ylabel='Count'); #Cessna, Piper Beech
```

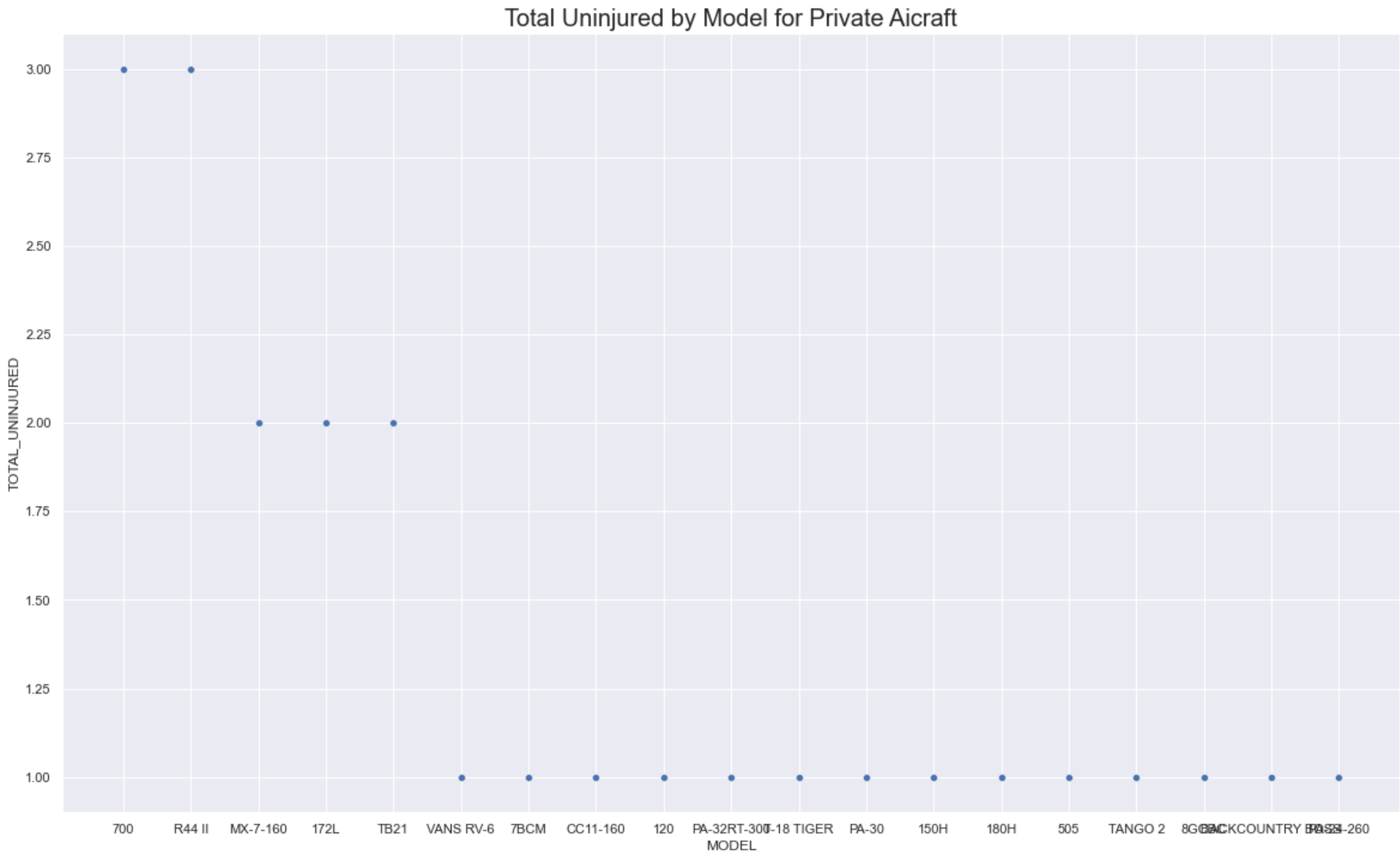


iii)Low risk by uninjured

a)Private Aircraft

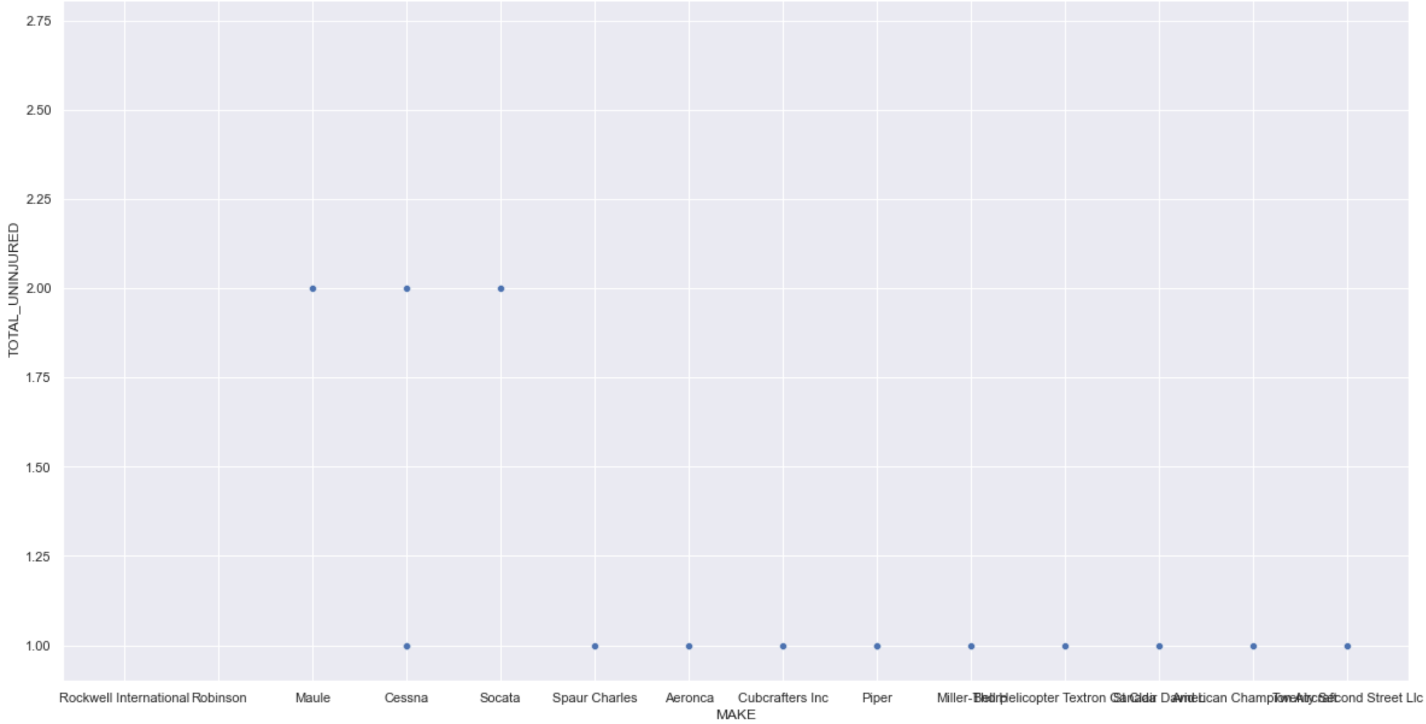
In [135]:

```
#Model by total unijured where injury is low take highest top 20
private_unijured= private .query('INJURY_SEVERITY == "Minor"').sort_values('TOTAL_UNINJUR
ED',ascending=False)[:20]
sns.scatterplot(data=private_unijured,x='MODEL',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Model for Private Aicraft',fontsize=20); #Rockwell 700 and
Robinson 411
```



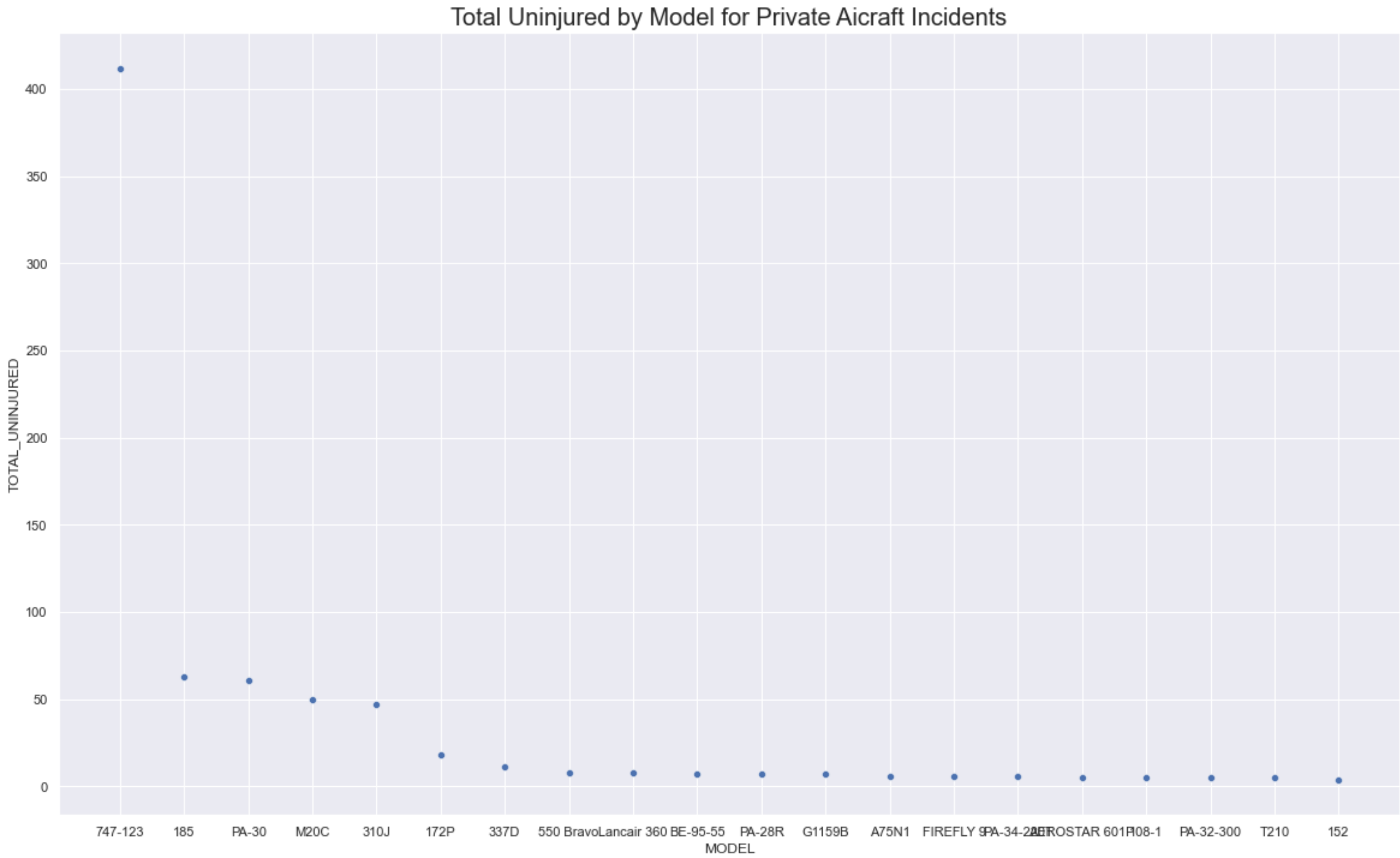
In [136]:

```
#Make by total unijured where injury is low take highest top 20
private_unijured= private .query('INJURY_SEVERITY == "Minor"').sort_values('TOTAL_UNINJUR
ED',ascending=False)[:20]
sns.scatterplot(data=private_unijured,x='MAKE',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Make for Private Aicraft',fontsize=20); #Rockwell followed
by Robinson
```



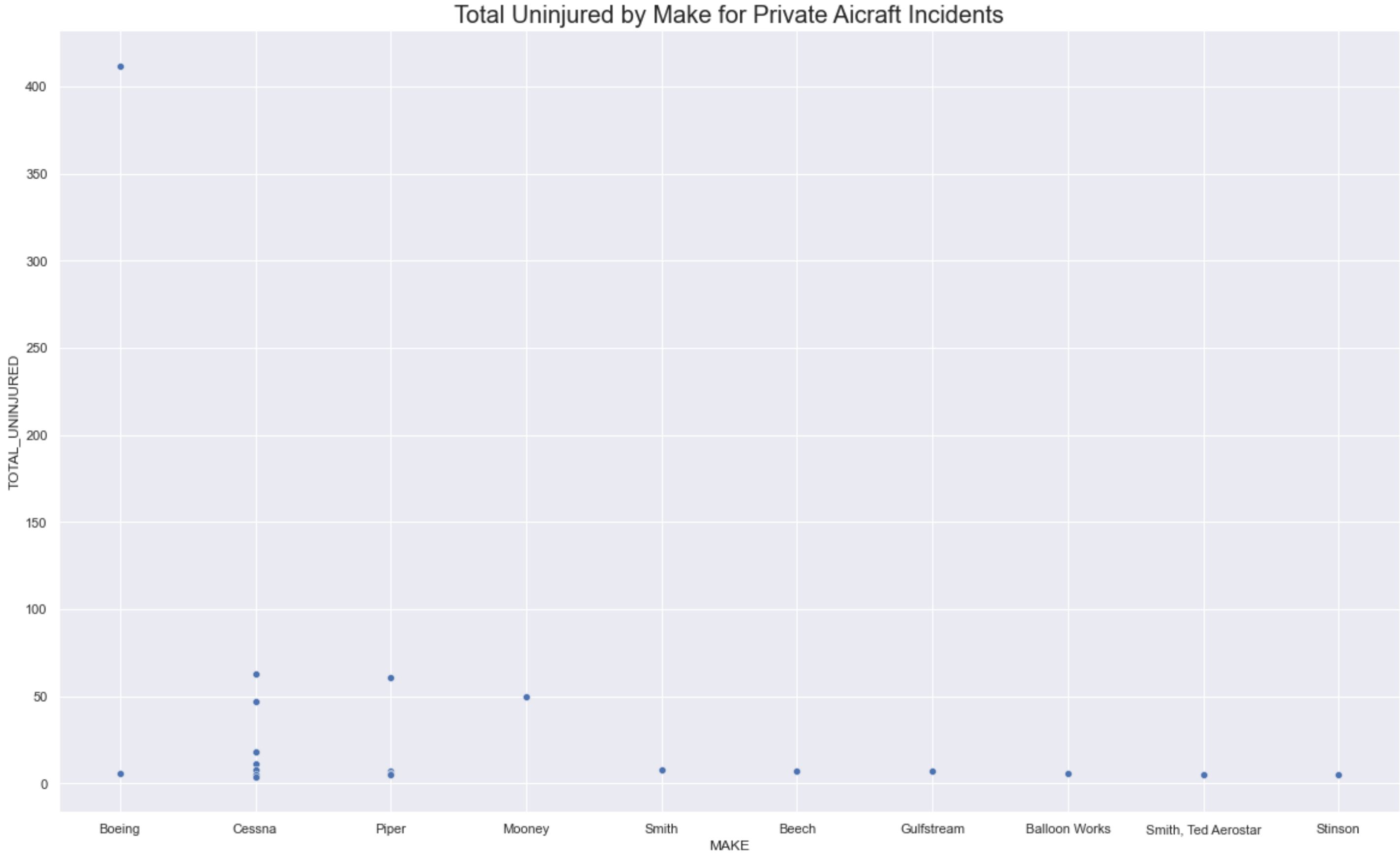
In [190]:

```
#check by incidents models
private_unijured= private .query('INJURY_SEVERITY == "Incident"').sort_values('TOTAL_UNI
NJURED',ascending=False)[:20]
sns.scatterplot(data=private_unijured,x='MODEL',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Model for Private Aicraft Incidents',fontsize=20);#Boeing 7
47-123 folowwed by Cessna 185 then piper PA-30
```



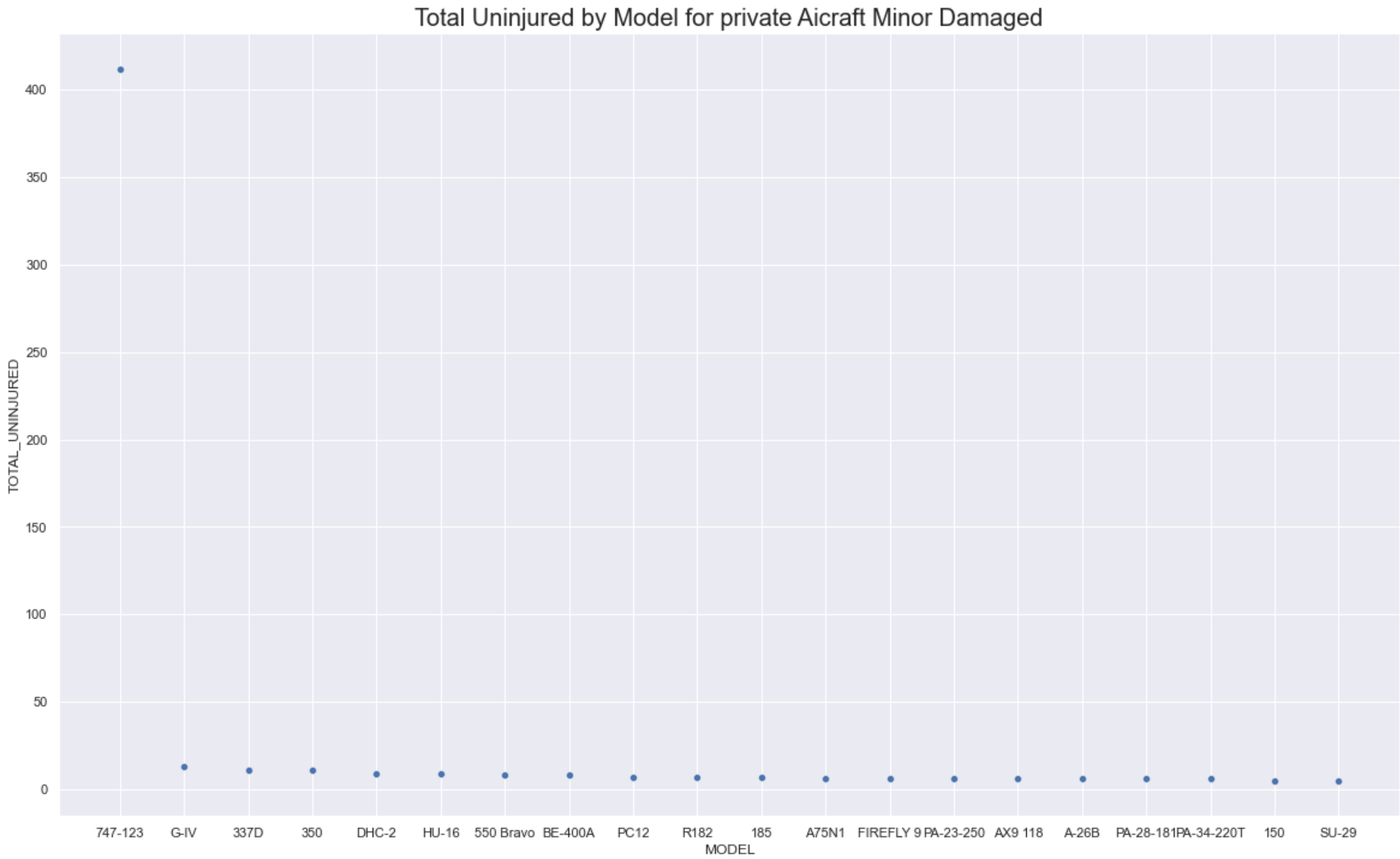
In [191]:

```
#check by incidents Makes
private_unijured= private .query('INJURY_SEVERITY == "Incident"').sort_values('TOTAL_UNI
NJURED',ascending=False)[:20]
sns.scatterplot(data=private_unijured,x='MAKE',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Make for Private Aicraft Incidents',fontsize=20);#Boeing fo
lowwed by Cessna  then piper
```



In [163]:

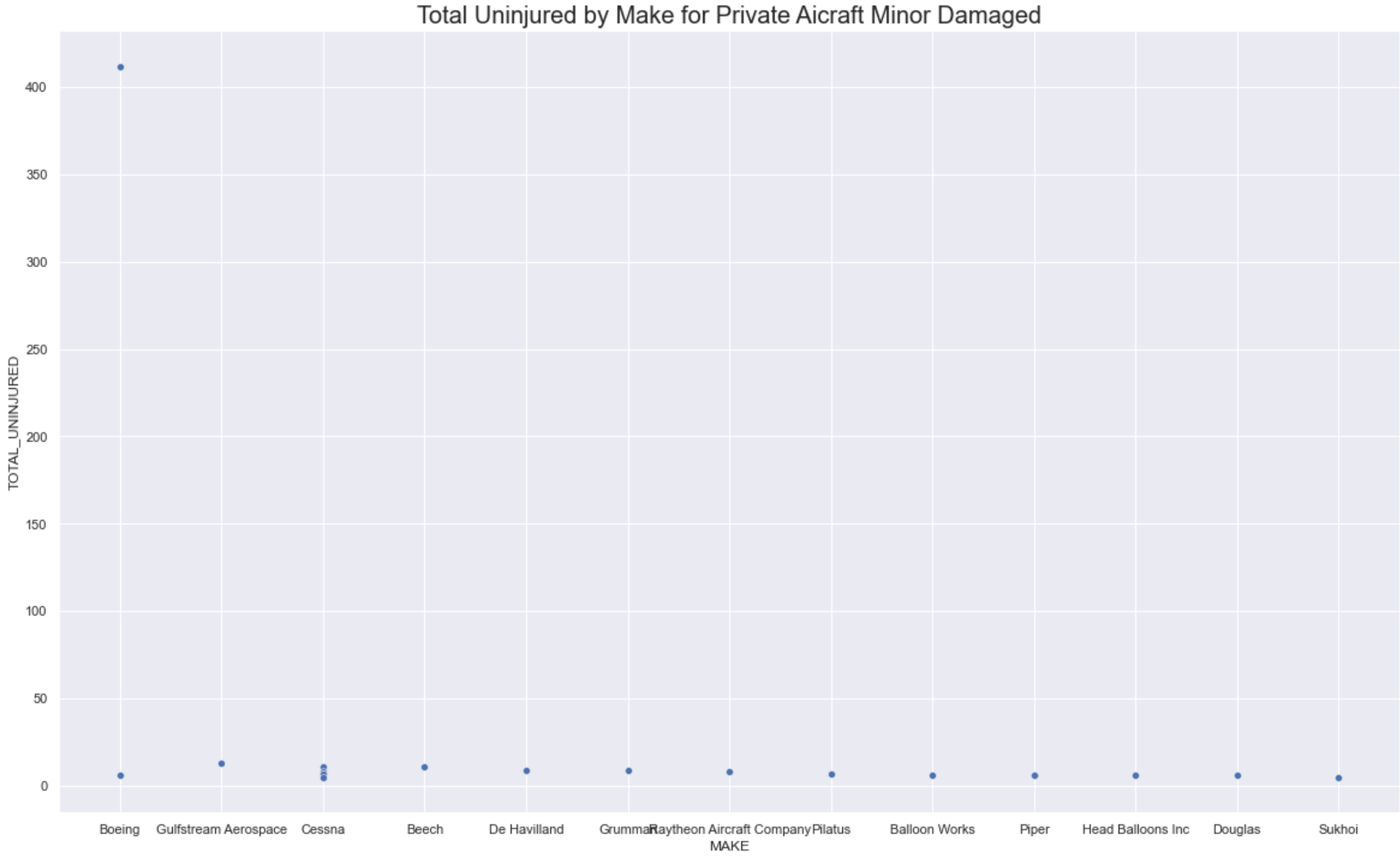
```
#Model by total unijured where Damage is low take highest top 20
private_damage=private .query('AIRCRAFT_DAMAGE == "Minor"').sort_values('TOTAL_UNINJURED'
,ascending=False)[:20]
sns.scatterplot(data=private_damage,x='MODEL',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Model for private Aicraft Minor Damaged',fontsize=20); #Boe
ing 747-123 has less unjuired
```



In [168]:

```
#Make by total unijured where Damage is low take highest top 20
```

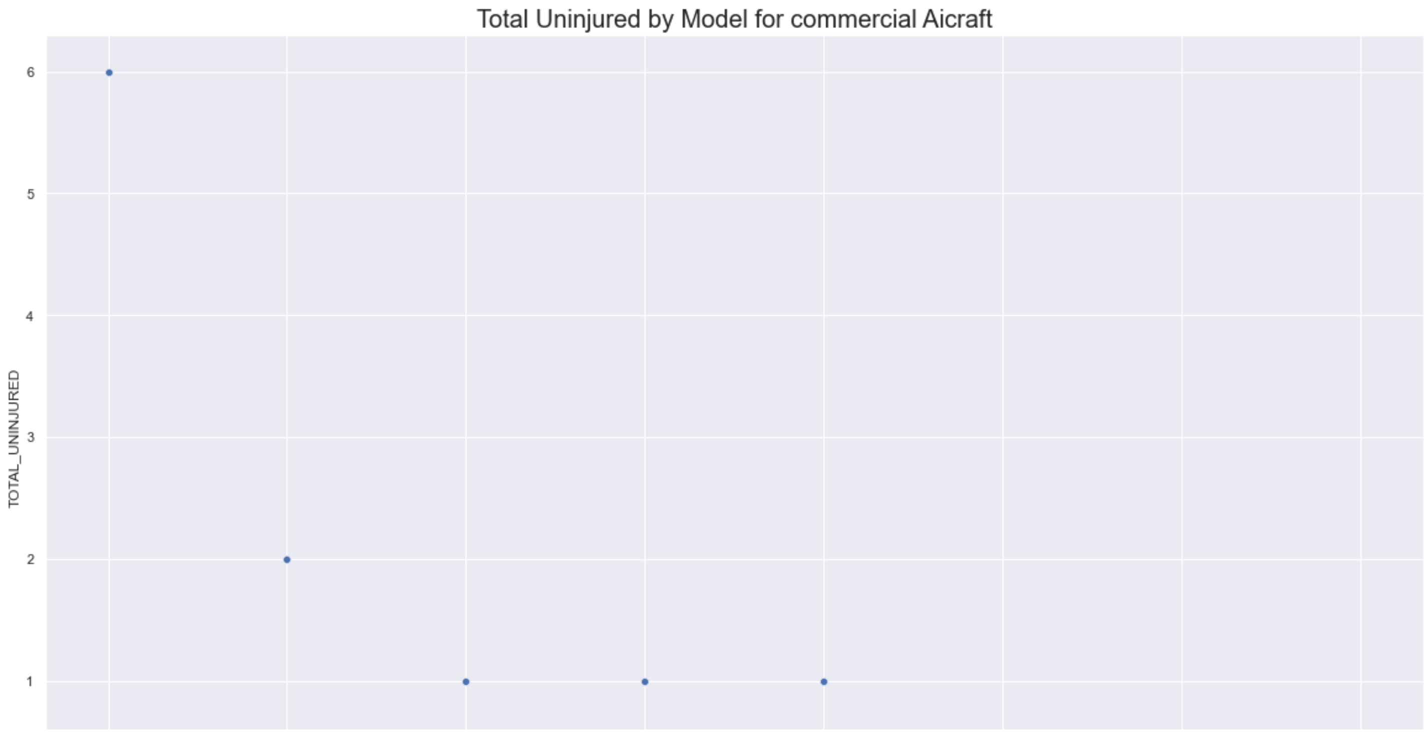
```
private_damage= private .query('AIRCRAFT_DAMAGE == "Minor"').sort_values('TOTAL_UNINJURED
',ascending=False)[:20]
sns.scatterplot(data=private_damage,x='MAKE',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Make for Private Aicraft Minor Damaged',fontsize=20); #Boei
ng  has less unjuired
```

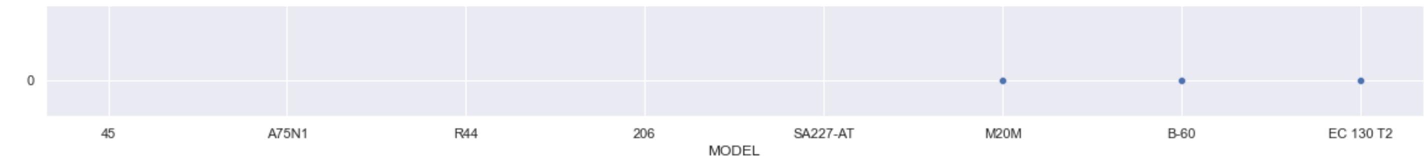


b)Commercial aircrafts

In [151]:

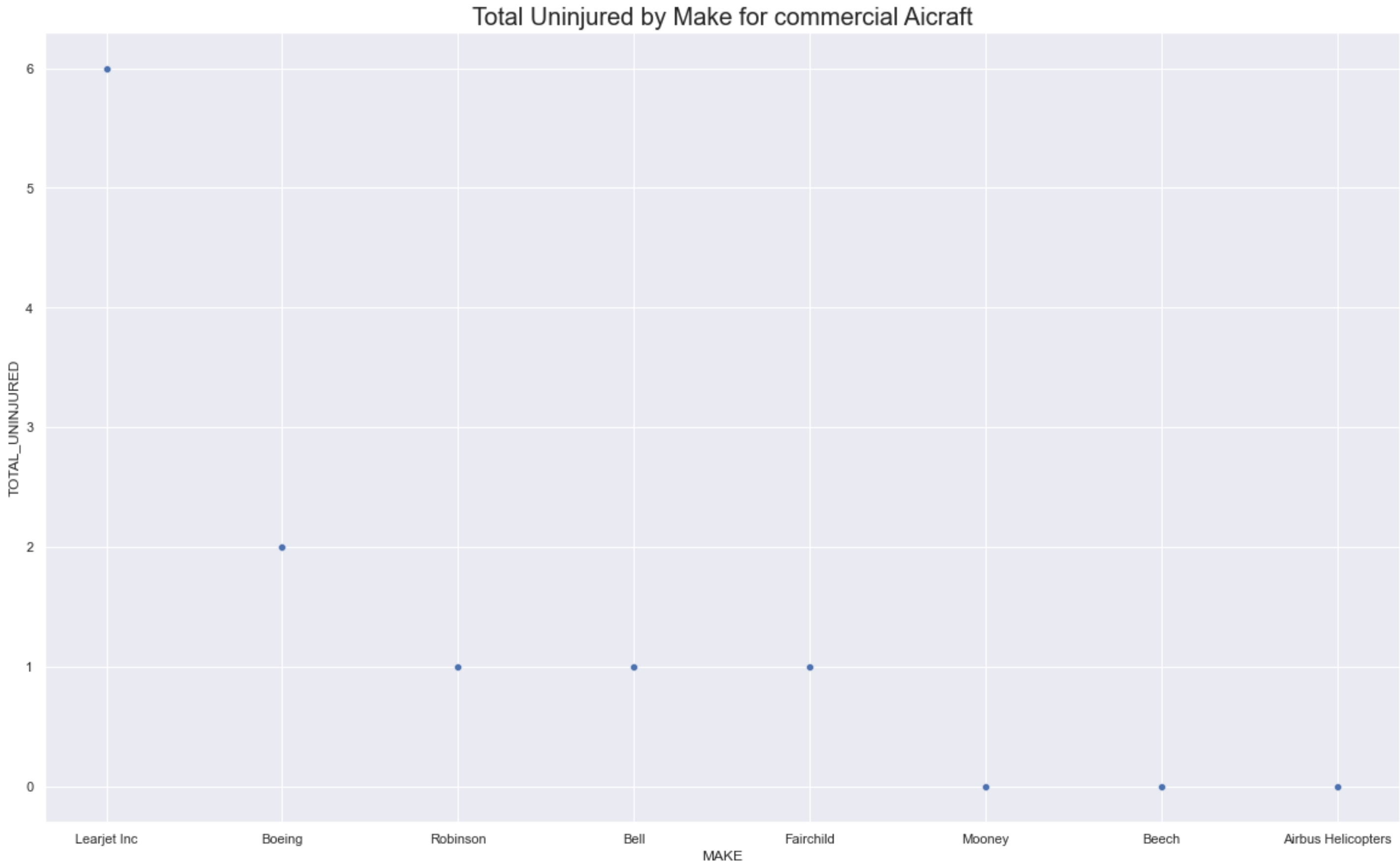
```
#minor injury severity models
#Model by total unijured where injury is low take highest top 20
commercial_unijured= commercial .query('INJURY_SEVERITY == "Minor"').sort_values('TOTAL_U
NINJURED',ascending=False)[:20]
sns.scatterplot(data=commercial_unijured,x='MODEL',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Model for commercial Aicraft',fontsize=20); #Learjet 45 fol
lowed by Boeing A75N1
```





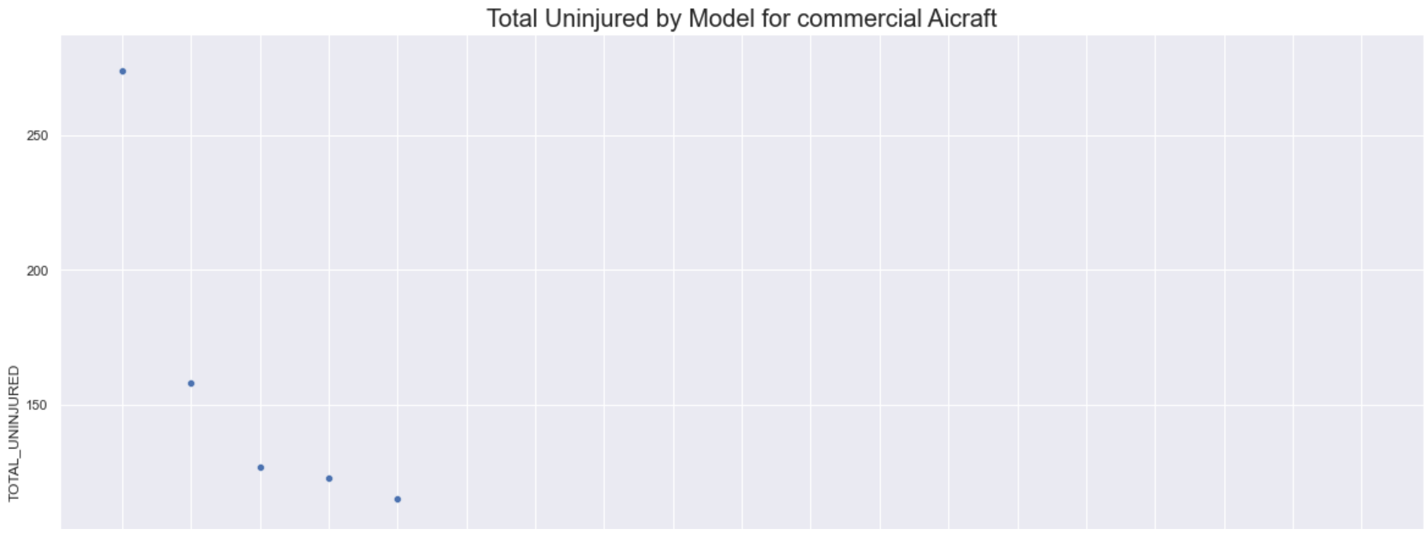
In [161]:

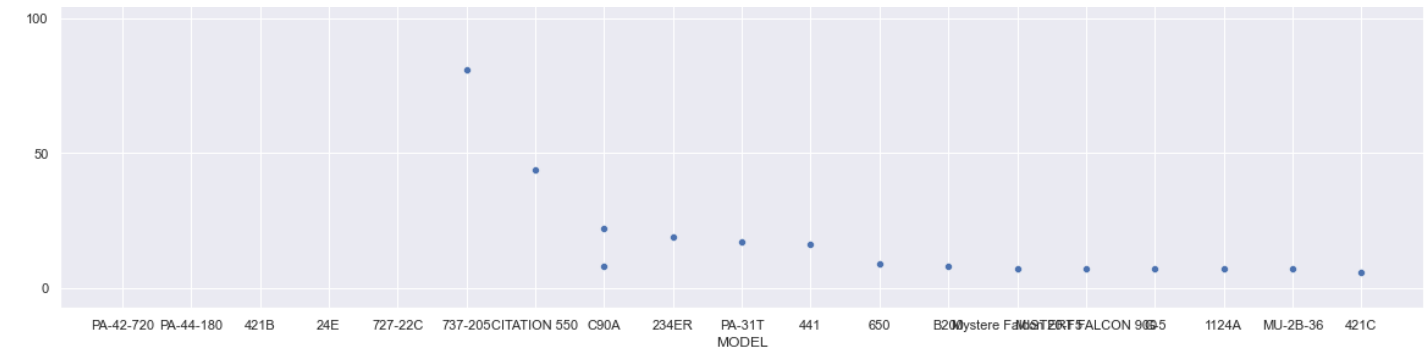
```
#minor injury severity makes
#Model by total unijured where injury is low take highest top 20
commercial_unijured= commercial .query('INJURY_SEVERITY == "Minor"').sort_values('TOTAL_U
NINJURED',ascending=False)[:20]
sns.scatterplot(data=commercial_unijured,x='MAKE',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Make for commercial Aicraft',fontsize=20); #Learjet  follow
ed by Boeing
```



In [192]:

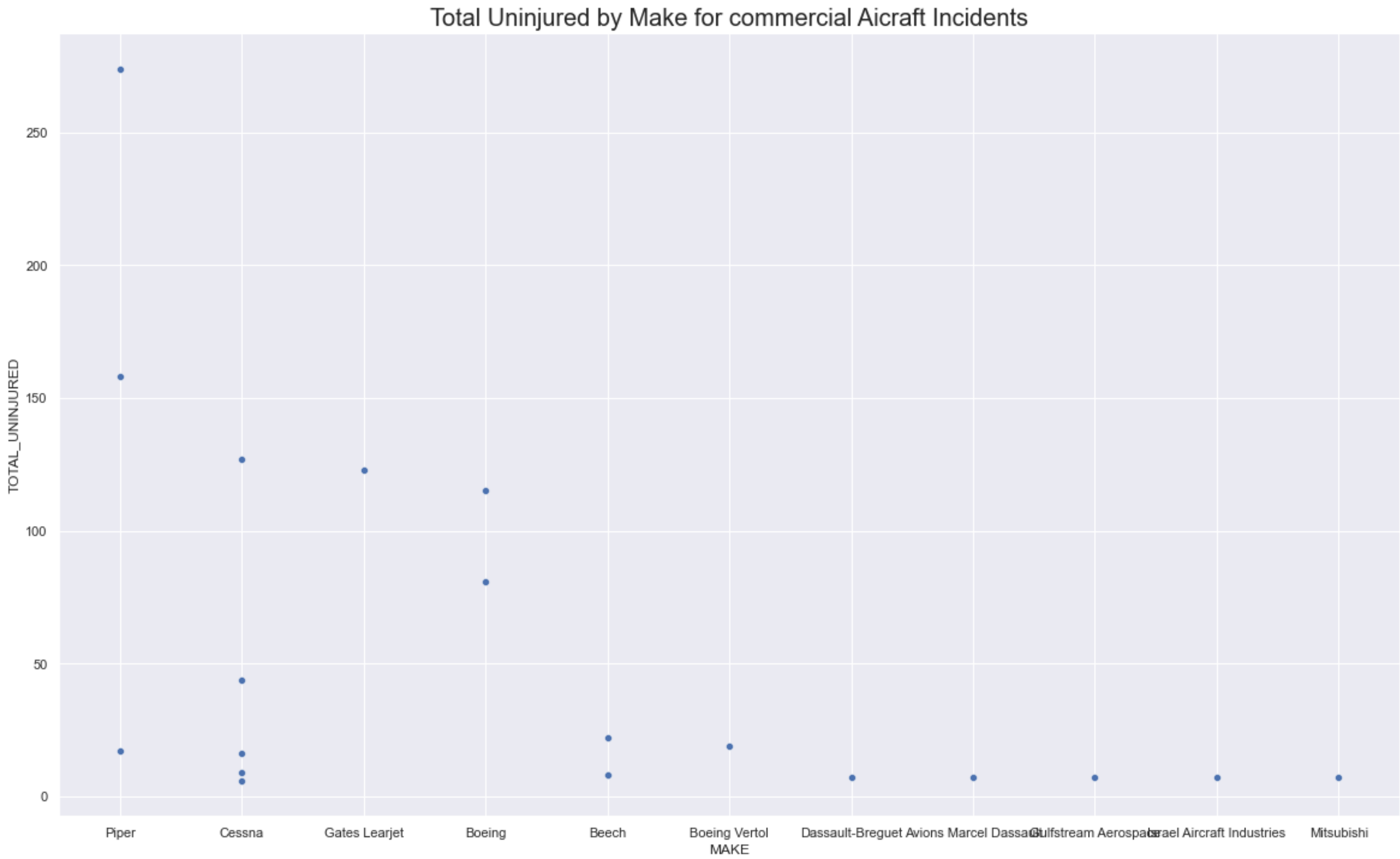
```
#Check by incidents Models
commercial_unijured= commercial .query('INJURY_SEVERITY == "Incident"').sort_values('TOTA
L_UNINJURED',ascending=False)[:20]
sns.scatterplot(data=commercial_unijured,x='MODEL',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Model for commercial Aicraft Incidents',fontsize=20); #Pipe
r PA-42-720 followed Piper PA-44-180 then Cessna 421B
```





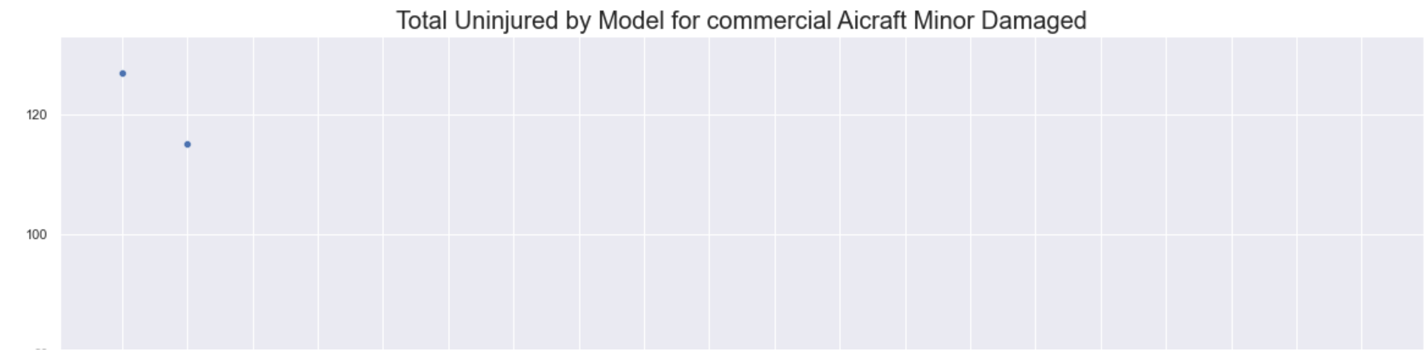
In [194]:

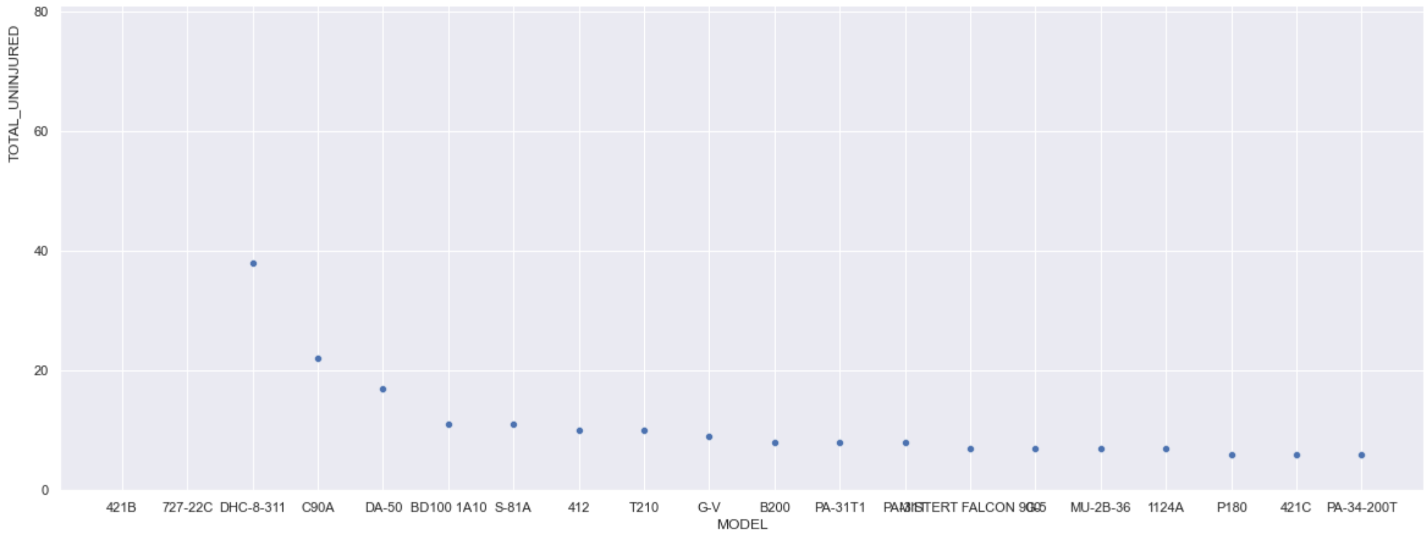
```
#Check by incidents Makes
commercial_unijured= commercial .query('INJURY_SEVERITY == "Incident"').sort_values('TOTA
L_UNINJURED',ascending=False)[:20]
sns.scatterplot(data=commercial_unijured,x='MAKE',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Make for commercial Aicraft Incidents',fontsize=20); #Piper
followedby then Cessna
```



In [171]:

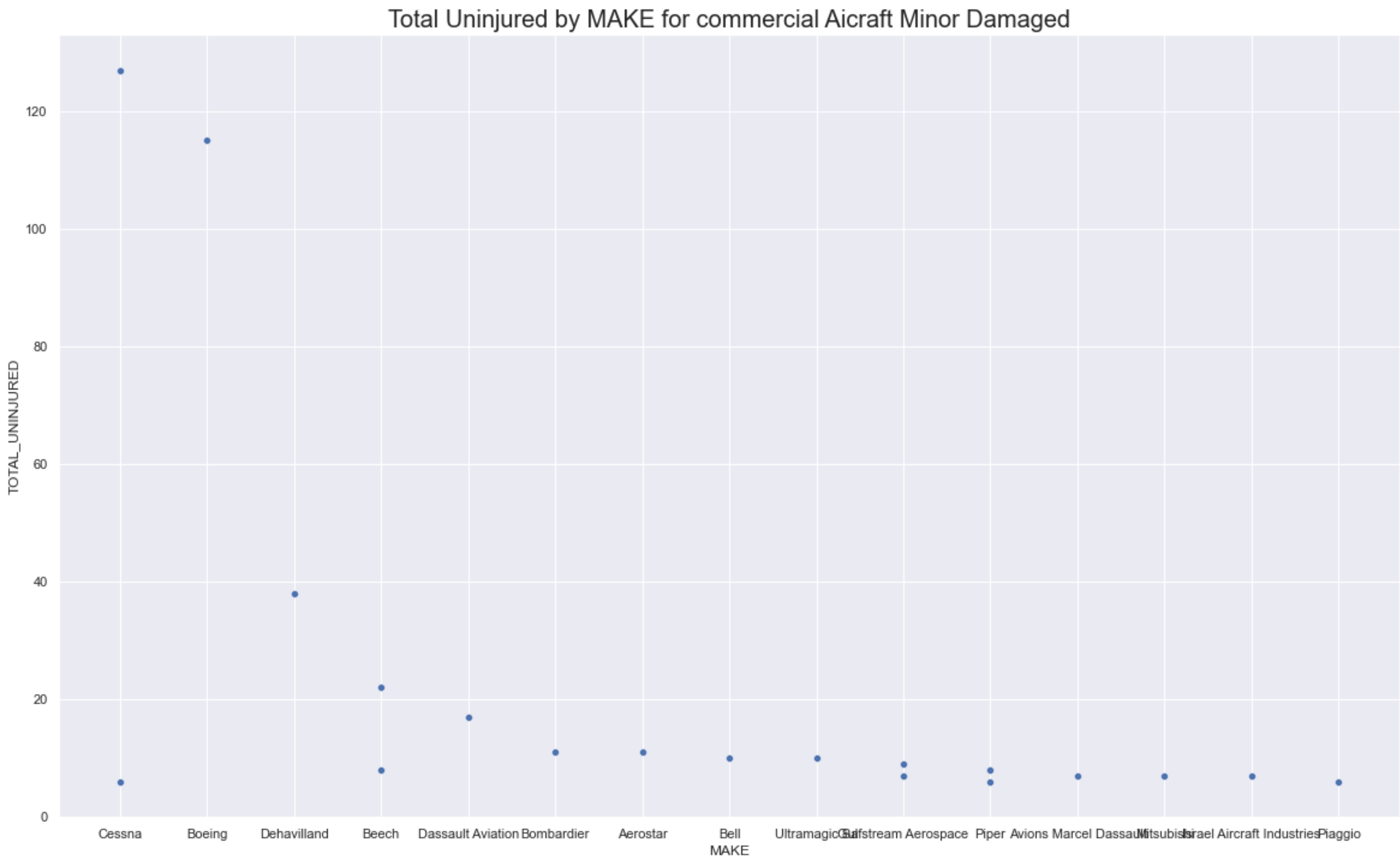
```
#minor aircraft damage models
#Model by total unijured where Damage is low take highest top 20
commercial_damage=commercial .query('AIRCRAFT_DAMAGE == "Minor"').sort_values('TOTAL_UNIN
JURED',ascending=False)[:20]
sns.scatterplot(data=commercial_damage,x='MODEL',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by Model for commercial Aicraft Minor Damaged',fontsize=20); #
Cessna 421B followed by Boeing 727-22C  has less unjuired
```





In [175]:

```
#minor aicraft damage Makes
#Make by total unijured where Damage is low take highest top 20
commercial_damage=commercial .query('AIRCRAFT_DAMAGE == "Minor"').sort_values('TOTAL_UNIN
JURED',ascending=False)[:20]
sns.scatterplot(data=commercial_damage,x='MAKE',y='TOTAL_UNINJURED')
plt.title('Total Uninjured by MAKE for commercial Aicraft Minor Damaged',fontsize=20); #C
essna followed by Boeing  has less unjured
```



Iv)Combing all risk factors i.e incidents,low damage and more uninjured

a)Private

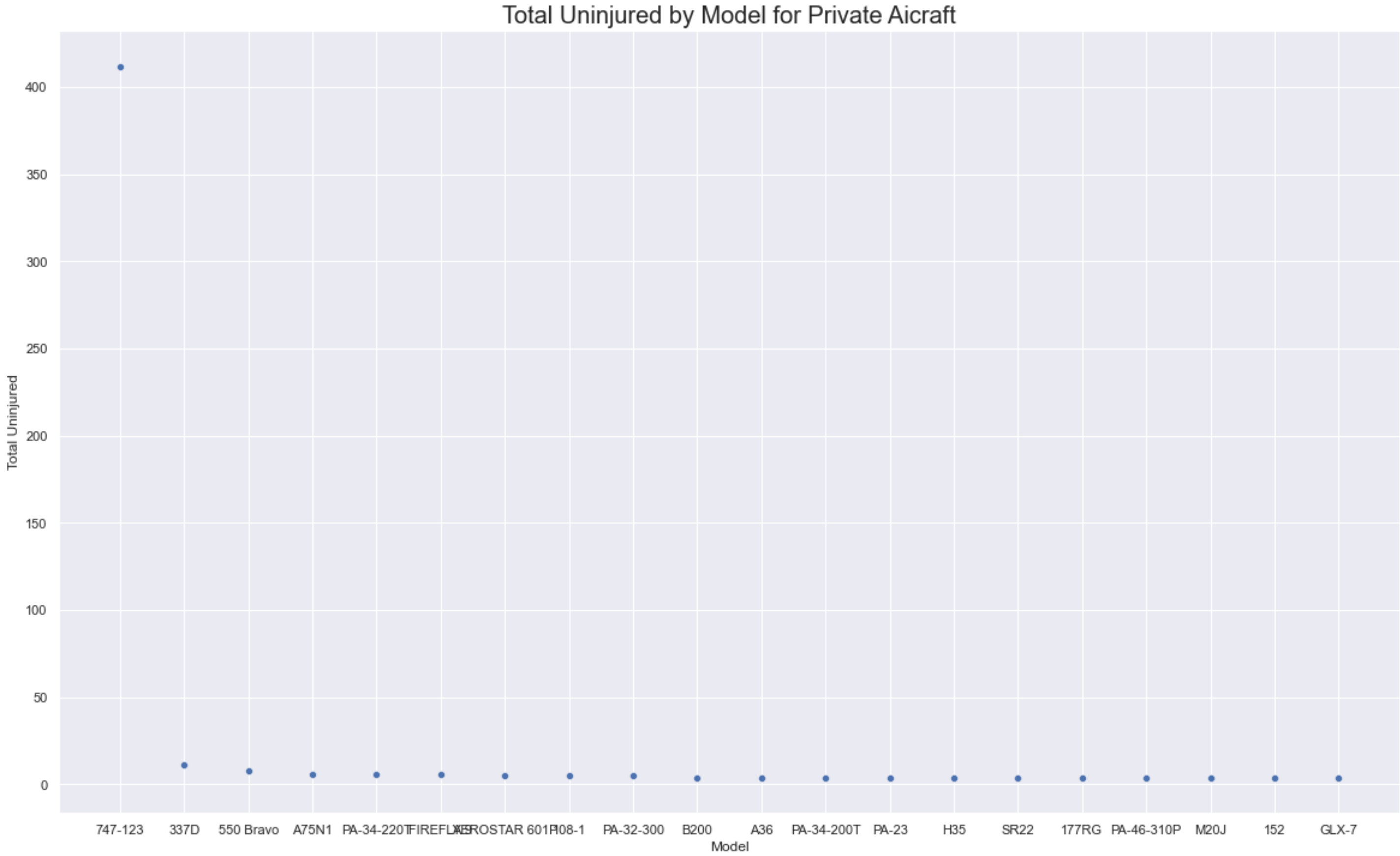
In [233]:

```
#Model by Incidents
private_risk_combined = private.query('INJURY_SEVERITY=="Incident" & AIRCRAFT_DAMAGE=="Mi
nor"').sort_values('TOTAL_UNINJURED',ascending=False)[:20]
sns.scatterplot(data=private_risk_combined,x='MODEL',y='TOTAL_UNINJURED')
plt.ylabel('Total Uninjured')
plt.xlabel('Model')
```

```
plt.title('Total Uninjured by Model for Private Aicraft',fontsize=20) #Boeing 747-123 fol
lowed by cessna 337D then Cessna 550 Bravo
```

Out[233]:

```
Text(0.5, 1.0, 'Total Uninjured by Model for Private Aicraft')
```

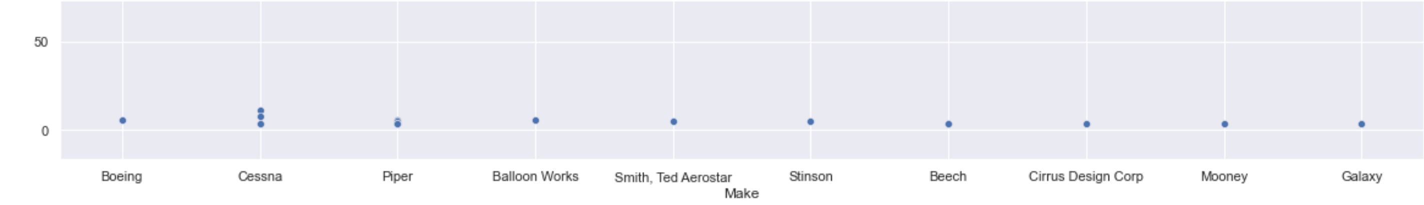


In [232]:

```
#Make by Incidents
private_risk_combined = private.query('INJURY_SEVERITY=="Incident" & AIRCRAFT_DAMAGE=="Mi
nor"').sort_values('TOTAL_UNINJURED',ascending=False)[:20]
sns.scatterplot(data=private_risk_combined,x='MAKE',y='TOTAL_UNINJURED')
plt.ylabel('Total Uninjured')
plt.xlabel('Make')
plt.title('Total Uninjured by Make for Private Aicraft',fontsize=20) #Boeing followed by
cessna then Piper
```

Out[232]:

```
Text(0.5, 1.0, 'Total Uninjured by Make for Private Aicraft')
```



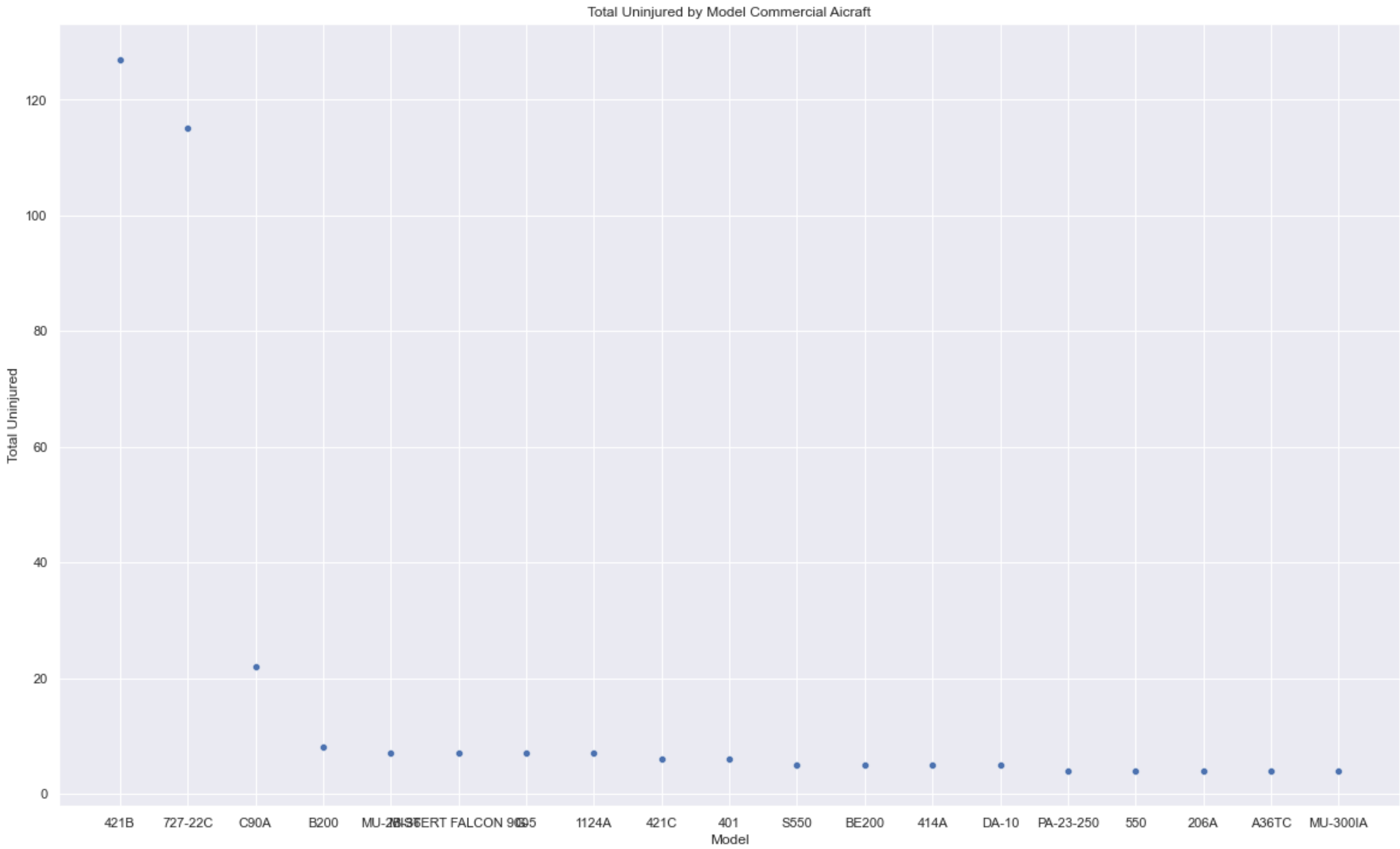
b)Commercial

In [234]:

```
#Model by Incidents
commercial_risk_combined = commercial.query('INJURY_SEVERITY=="Incident" & AIRCRAFT_DAMAG
E=="Minor"').sort_values('TOTAL_UNINJURED',ascending=False)[:20]
sns.scatterplot(data=commercial_risk_combined,x='MODEL',y='TOTAL_UNINJURED')
plt.ylabel('Total Uninjured')
plt.xlabel('Model')
plt.title('Total Uninjured by Model Commercial Aicraft'); #Cessna 421B followed by Boeng
727-22C
```

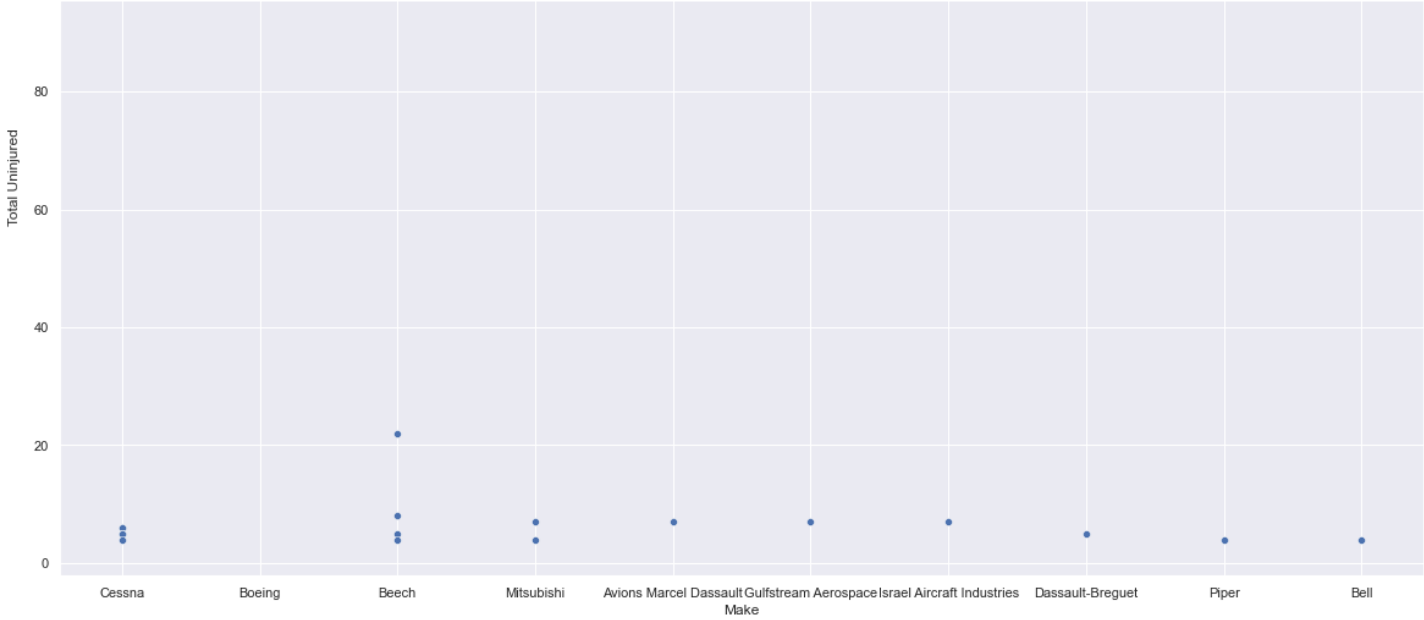
Out[234]:

```
Text(0.5, 1.0, 'Total Uninjured by Model Commercial Aicraft')
```



In [240]:

```
#Make by Incidents
commercial_risk_combined = commercial.query('INJURY_SEVERITY=="Incident" & AIRCRAFT_DAMAG
E=="Minor"').sort_values('TOTAL_UNINJURED',ascending=False)[:20]
sns.scatterplot(data=commercial_risk_combined,x='MAKE',y='TOTAL_UNINJURED')
plt.ylabel('Total Uninjured')
plt.xlabel('Make')
plt.title('Total Uninjured by Make Commercial Aicraft'); #Cessna followed by Boeng
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: