THIS IS MY PROJECT ABOUT THE FLIGHT DATA

```
BUSINESSPROBLEM
Your company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating air
```

```
  Cell In[92], line 2
    Your company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and
operating airplanes for commercial and private enterprises, but do not know anything about the potential risks of aircraft. You are
charged with determining which aircraft are the lowest risk for the company to start this new business endeavor. You must then
translate your findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to
purchase.
         ^
SyntaxError: invalid syntax
```

OBJECTIVE: To determine which aircraft has low risk. To determine which engine has low fatal rate. To determine between amateur and proffionally built aircaft has low risk. To determine on what wheather condition is the fatal rate high.

```
##Importing python libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
##Loading the aviation data with the extension CSV
df1=pd.read_csv("AviationData.csv",encoding="latin1")
df1
```

observation: The dataset has 88889 rows and 31 columns

```
##The first five rows
df1.head()
```

```
##The last five rows
df1.tail()
```

The dataset is uniform from top to bottom it is not corrupted

```
##Checking the column attribute
df1.columns
```

observation: This are the name of the columns

```
##checking the index attribute
df1.index
```

```
##Checking the summary of the dataset/info(verbose=(False)
df1.info()
```

Observation: This shows that in the data type we have float which are decimal numbers and object which is a mixture of both int,string and float

```
##checking the descriptive statistics of each columns
df1.describe(include="object").T
```

```
###Checking the shape attribute
df1.shape
```

observation: We have 88889 rows and 31 columns

```
##Create a copy to use in cleaning
df2=df1.copy(deep=True)
df2
```

```
##Changing all the columns into lowercase for uniformity
df2.columns=df2.columns.str.lower()
df2.columns
```

Observation: All have changed to lowercase

```
##Removing white space
df2.columns=df2.columns.str.replace(" ","")
df2.columns
```

```
## We can replace  the fullstop with the underscore
df2.columns=df2.columns.str.replace(".","_")
df2.columns
```

```
##Dropping unncessary columns eg event_id because they can cause noise in our data set
del df2["event_id"]
```

```
##Dropping publication_date column
del df2["publication_date"]
```

```
##Dropping latitude and longitude columns
del df2["latitude"]
```

```
##Dropping the longitude column
del df2["longitude"]
```

```
##Dropping the report status column
del df2["report_status"]
```

```
##Check the null values in the columns
df2.isnull().sum()
```

Observation There are number of null values in the data set

```
##Replace the null value in the total_fatal_injuries  column with the median because it is skewed to the right
median_fa=df2["total_fatal_injuries"].median()
df2["total_fatal_injuries"].fillna(median_fa,inplace=True)
```

```
##Replace the null value in the total_serious_injuries column with the median because it is skewed to the right
median_ser=df2["total_serious_injuries"].median()
df2["total_serious_injuries"].fillna(median_ser,inplace=True)
```

```
##Replace the null value in the total_minor_injuries column with the median because it is skewed to the right
median_min=df2["total_minor_injuries"].median()
df2["total_minor_injuries"].fillna(median_min,inplace=True)
```

```
##Replace the null value in the total_uninjuredcolumn with the median because it is skewed to the right
median_min=df2["total_uninjured"].median()
df2["total_uninjured"].fillna(median_min,inplace=True)
```

```
##Replace the null value in location using unkown
df2["location"].fillna("unknown",inplace=True)
```

```
##Replace the null value in country  using unknown
df2["country"].fillna("unknown",inplace=True)
```

```
##Replace the null value in airport_code using unknown
df2["airport_code"].fillna("unknown",inplace=True)
```

```
##Replace the null value in airport_name using unknown
df2["airport_name"].fillna("unknown",inplace=True)
```

```
##Replace the null value in injury_severity using unknown
df2["injury_severity"].fillna("unknown",inplace=True)
```

```
##Replace the null value in airport_damage using unknown
df2["aircraft_damage"].fillna("unknown",inplace=True)
```

```
##Replace the null value in aircraft_category using unknown
df2["aircraft_category"].fillna("unknown",inplace=True)
```

```
##Replace the null value in registration_number column using unknown
df2["registration_number"].fillna("unknown",inplace=True)
```

```
##Replace the null value in the make column using unknown
df2["make"].fillna("unknown",inplace=True)
```

```
##Replace the null value in model using unknown
df2["model"].fillna("unknown",inplace=True)
```

```
##Replace the null value in amateur_built using unknown
df2["amateur_built"].fillna("unknown",inplace=True)
```

```
##Replace the null value in number_of_engines using unknown
df2["number_of_engines"].fillna("unknown",inplace=True)
```

```
##Replace the null value in engine_type using unknown
df2["engine_type"].fillna("unknown",inplace=True)
```

```
##Replace the null value in far_description using unknown
df2["far_description"].fillna("unknown",inplace=True)
```

```
##Replace the null value in schedule using unknown
df2["schedule"].fillna("unknown",inplace=True)
```

```
##Replace the null value in purpose_of_flight using unknown
df2["purpose_of_flight"].fillna("unknown",inplace=True)
```

```
##Replace the null value in air_carrier using unknown
df2["air_carrier"].fillna("unknown",inplace=True)
```

```
##Replace the null value in airport_code using unknown
df2["broad_phase_of_flight"].fillna("unknown",inplace=True)
```

```
df2.columns
```

```
##In the weather column we have both unk and UNK which differ because of lower and upper case we must put them together
df2.groupby("weather_condition")["weather_condition"].count()
```

```
##Replacing the unk with UNK
df2["weather_condition"]=df2["weather_condition"].str.replace("Unk","UNK")
```

```
##To confirm that unk is in UNK
df2.groupby("weather_condition")["weather_condition"].count()
```

```
##Removing null values in weather using forward fill
df2["weather_condition"].fillna("UNK",inplace=True)
```

```
df2.groupby("engine_type")["engine_type"].count()
```

Observations:We have several unknown we must bundle them together

```
##Replacing the Unknown to the UNK
df2["engine_type"]=df2["engine_type"].str.replace("Unknown","UNK")
```

```
##Replacing the unknown to the UNK
df2["engine_type"]=df2["engine_type"].str.replace("unknown","UNK")
```

```
##Confirm if Unkown,unkown have been replaced by UKN
df2.groupby("engine_type")["engine_type"].count()
```

Observation:All null values have been removed

```
##Checking value counts
df2.groupby("broad_phase_of_flight")["broad_phase_of_flight"].count()
```

Observation:we have two unknowns

```
##Replacing the unknown to the Unknown
df2["broad_phase_of_flight"]=df2["broad_phase_of_flight"].str.replace("unknown","Unknown")
```

```
##Confirm the removal
df2.groupby("broad_phase_of_flight")["broad_phase_of_flight"].count()
```

Observation:The unknown has heen bundled with the Unknown

```
##confirm is all the null value have been removed
df2.isnull().sum()
```

```
##Checking for duplicates
df_duplicates=df2.duplicated().sum()
df_duplicates
```

Observation: There are zero duplicates

```
###Checking for outliers here we use boxplot to check for outliers
sns.boxplot(df2)
```

Observation: We can see that we have outliers in the 'total_minor_injuries', 'total_uninjured','total_fatal_injuries', 'total_serious_injuries' columns

```
##plotting the total_minor_injuries inorder to remove the outlier
sns.boxplot(df2["total_minor_injuries"])
```

```
##Use the maximum quantile method
max_total_minor=df2["total_minor_injuries"].quantile(0.995)
max_total_minor
```

```
df3=df2[df2["total_minor_injuries"]>max_total_minor]
df3=df2[df2["total_minor_injuries"]<max_total_minor]
sns.boxplot(df3["total_minor_injuries"])
```

Observation:Outliers have not been removed we use the second method that is the iqr

```
##Removing the outlier using the iqr
q1=df2["total_minor_injuries"].quantile(0.25)
q3=df2["total_minor_injuries"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df4=df2[(df2["total_minor_injuries"]>=lower_bound) & (df2["total_minor_injuries"]<=upper_bound)]
sns.boxplot(x="total_minor_injuries",data=df4);
```

Observation:After applying the IQR method to remove outliers, the resulting boxplot shows that the extreme values are no longer present. However, the data might now appear tightly concentrated around zero or within a narrow range

```
##plotting the 'total_fatal_injuries'inorder to remove the outlier
sns.boxplot(df2['total_fatal_injuries'])
```

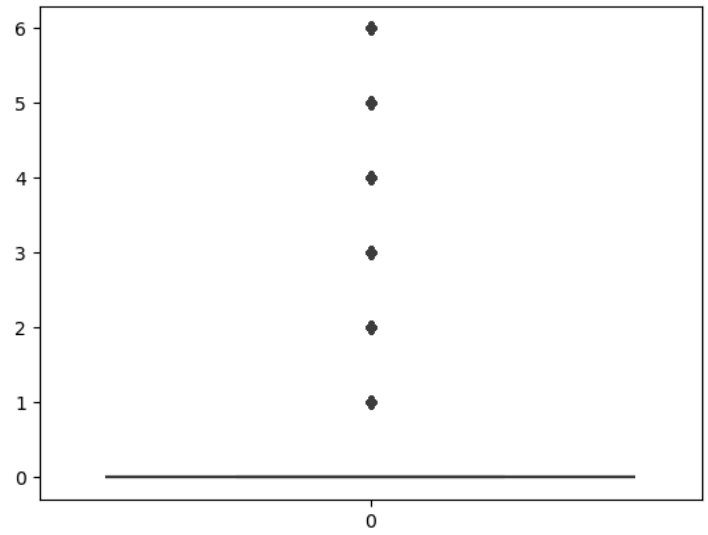
Observation:We can see an outlier

```
max_total_fatal=df2["total_fatal_injuries"].quantile(0.995)
max_total_fatal
```

```
7.0
```

```
df3=df2[df2["total_fatal_injuries"]>max_total_fatal]
df5=df2[df2["total_fatal_injuries"]<max_total_fatal]
sns.boxplot(df5["total_fatal_injuries"])
```

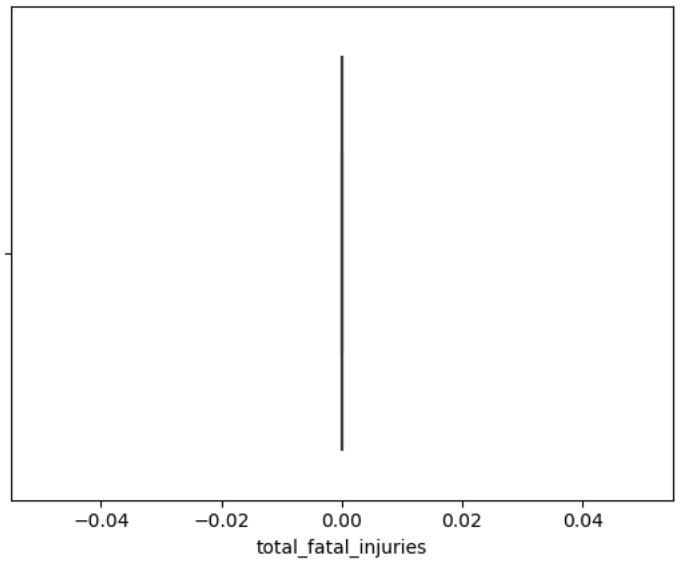
```
<Axes: >
```



Observation:This shows that the outliers have not been remove .We use the second method iqr

```
##Using the second method iqr
q1=df2["total_fatal_injuries"].quantile(0.25)
q3=df2["total_fatal_injuries"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df6=df2[(df2["total_fatal_injuries"]>=lower_bound) & (df2["total_fatal_injuries"]<=upper_bound)]
sns.boxplot(x="total_fatal_injuries",data=df6)
```

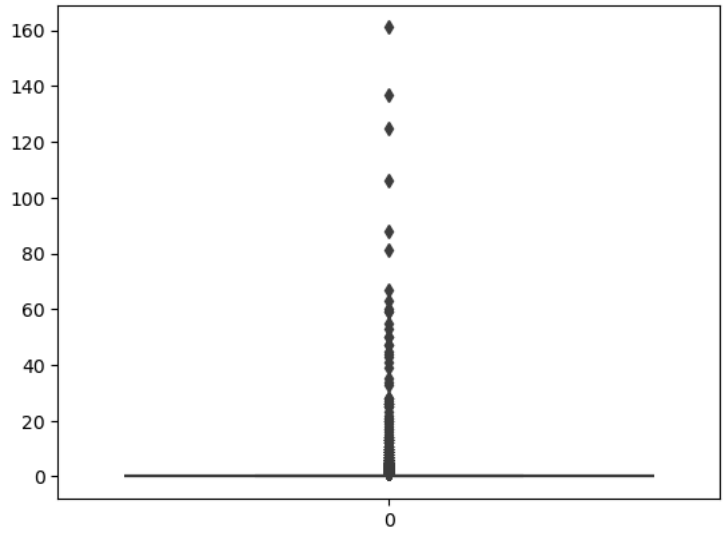
```
<Axes: xlabel='total_fatal_injuries'>
```



Observation:After applying the IQR method to remove outliers, the resulting boxplot shows that the extreme values are no longer present. However, the data might now appear tightly concentrated around zero or within a narrow range

```
#plotting the 'total_serious_injuries' inorder to remove the outlier
sns.boxplot(df2["total_serious_injuries"])
```

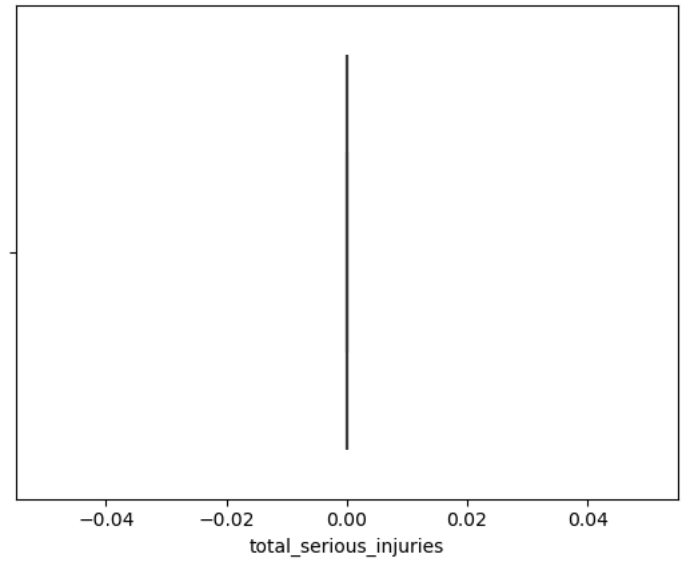
```
<Axes: >
```



Observation:We can see outliers

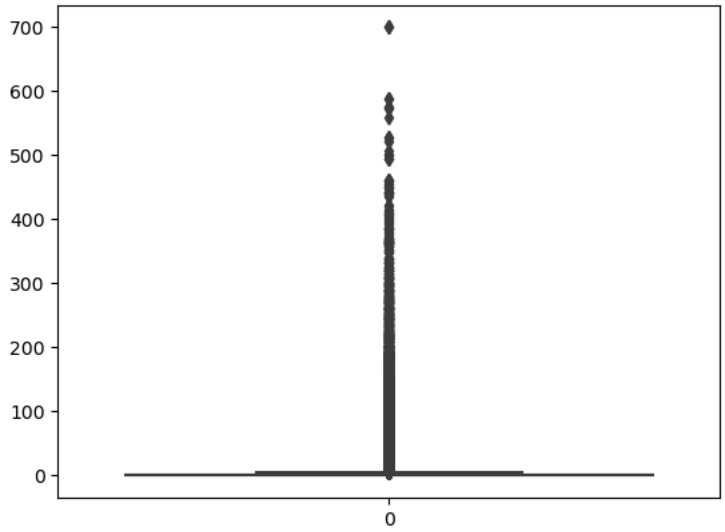
```
##We use the iqr to remove outliers
q1=df2["total_serious_injuries"].quantile(0.25)
q3=df2["total_serious_injuries"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df7=df2[(df2["total_serious_injuries"]>=lower_bound) & (df2["total_serious_injuries"]<=upper_bound)]
sns.boxplot(x="total_serious_injuries",data=df7)
```

```
<Axes: xlabel='total_serious_injuries'>
```



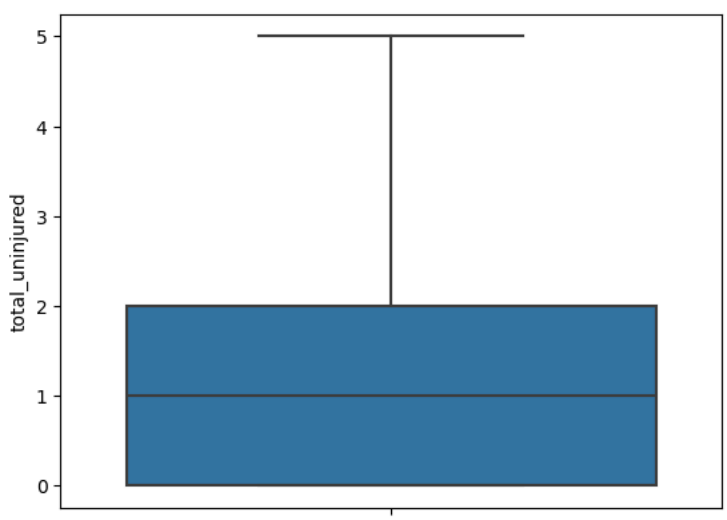

```
#plotting the 'total_uninjured' inorder to remove the outlier
sns.boxplot(df2["total_uninjured"])
```

```
<Axes: >
```



Observation:We can see one extreme value

```
##Removing the outlier using the iqr
q1=df2["total_uninjured"].quantile(0.25)
q3=df2["total_uninjured"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df8=df2[(df2["total_uninjured"]>=lower_bound) & (df2["total_uninjured"]<=upper_bound)]
sns.boxplot(y="total_uninjured",data=df8);
```




```
Observation:This shows all the otliers have been removed and the values are not congested at one point there is proper distribution
```

Saving the Clean Dataset

```
df8.to_csv("stainer_d.csv",index=False)
```

EDA

```
##Load the csv file you had saved as cleann_d inorder to do analysis
data=pd.read_csv("stainer_d.csv")
data
```

	investigation_type	accident_number	event_date	location	country	airport_code	airport_name	injury_severity	aircraft_dam
0	Accident	SEA87LA080	10/24/1948	MOOSE CREEK, ID	United States	unknown	unknown	Fatal(2)	Destro
1	Accident	LAX94LA336	7/19/1962	BRIDGEPORT, CA	United States	unknown	unknown	Fatal(4)	Destro
2	Accident	NYC07LA005	8/30/1974	Saltville, VA	United States	unknown	unknown	Fatal(3)	Destro
3	Accident	LAX96LA321	6/19/1977	EUREKA, CA	United States	unknown	unknown	Fatal(2)	Destro
4	Accident	CHI79FA064	8/2/1979	Canton, OH	United States	unknown	unknown	Fatal(1)	Destro
...	...	...	...	...	...	...	...	...	
84737	Accident	ERA23LA093	12/26/2022	Annapolis, MD	United States	unknown	unknown	Minor	unkn
84738	Accident	ERA23LA095	12/26/2022	Hampton, NH	United States	unknown	unknown	unknown	unkn
84739	Accident	WPR23LA075	12/26/2022	Payson, AZ	United States	PAN	PAYSON	Non-Fatal	Substa
84740	Accident	WPR23LA076	12/26/2022	Morgan, UT	United States	unknown	unknown	unknown	unkn
84741	Accident	ERA23LA097	12/29/2022	Athens, GA	United States	unknown	unknown	Minor	unkn

84742 rows × 26 columns

```
##To make acopy
data1=data.copy(deep=True)
data1
```

	investigation_type	accident_number	event_date	location	country	airport_code	airport_name	injury_severity	aircraft_dam
0	Accident	SEA87LA080	10/24/1948	MOOSE CREEK, ID	United States	unknown	unknown	Fatal(2)	Destro
1	Accident	LAX94LA336	7/19/1962	BRIDGEPORT, CA	United States	unknown	unknown	Fatal(4)	Destro
2	Accident	NYC07LA005	8/30/1974	Saltville, VA	United States	unknown	unknown	Fatal(3)	Destro
3	Accident	LAX96LA321	6/19/1977	EUREKA, CA	United States	unknown	unknown	Fatal(2)	Destro
4	Accident	CHI79FA064	8/2/1979	Canton, OH	United States	unknown	unknown	Fatal(1)	Destro
...	...	...	...	...	...	...	...	...	
84737	Accident	ERA23LA093	12/26/2022	Annapolis, MD	United States	unknown	unknown	Minor	unkn
84738	Accident	ERA23LA095	12/26/2022	Hampton, NH	United States	unknown	unknown	unknown	unkn
84739	Accident	WPR23LA075	12/26/2022	Payson, AZ	United States	PAN	PAYSON	Non-Fatal	Substa
84740	Accident	WPR23LA076	12/26/2022	Morgan, UT	United States	unknown	unknown	unknown	unkn
84741	Accident	ERA23LA097	12/29/2022	Athens, GA	United States	unknown	unknown	Minor	unkn

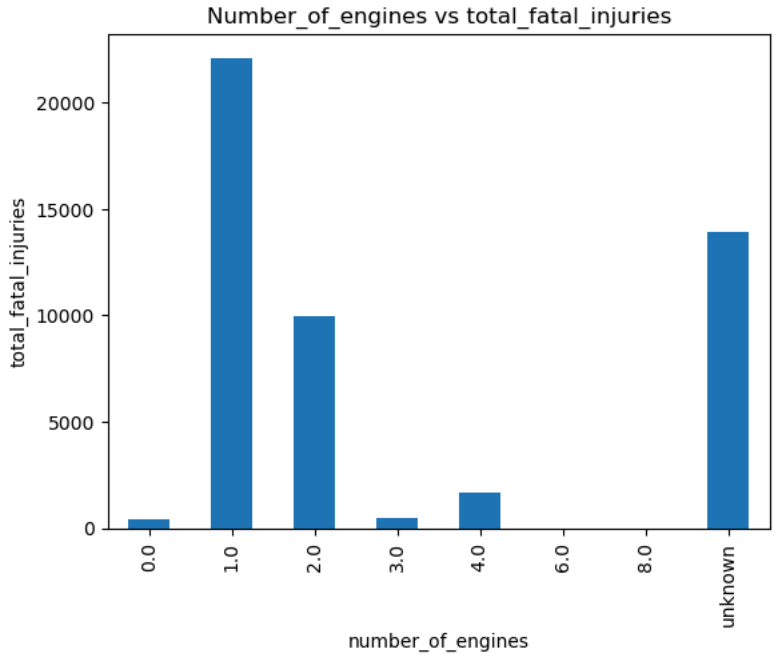
84742 rows × 26 columns

```
data.columns
```

```
Index(['investigation_type', 'accident_number', 'event_date', 'location',
       'country', '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',
```

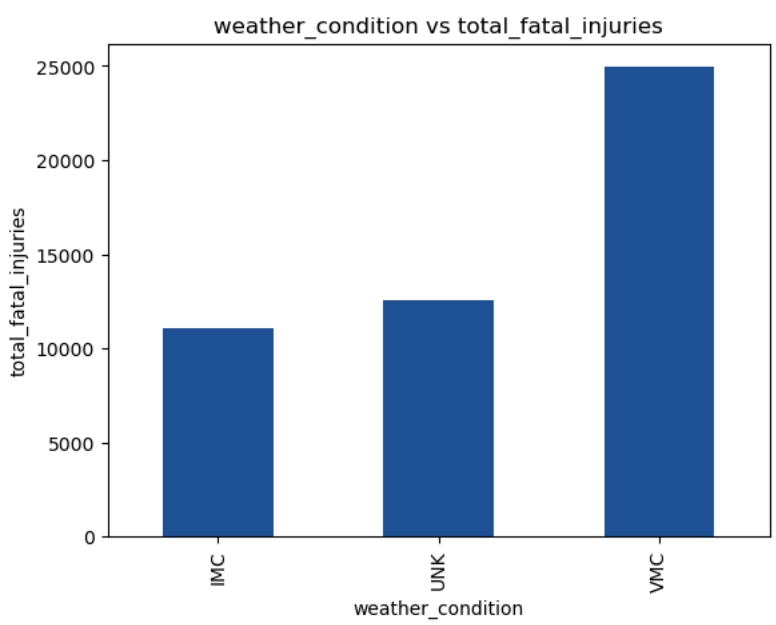
```
       'total_fatal_injuries', 'total_serious_injuries',
       'total_minor_injuries', 'total_uninjured', 'weather_condition',
       'broad_phase_of_flight'],
      dtype='object')
```

```
##plotting number_of_engine and total_fatal_injuries using matplot
total_gpr=data1.groupby("number_of_engines")["total_fatal_injuries"].sum()
total_gpr.plot(kind='bar');
plt.ylabel("total_fatal_injuries")
plt.title("Number_of_engines vs total_fatal_injuries")
plt.show()
```



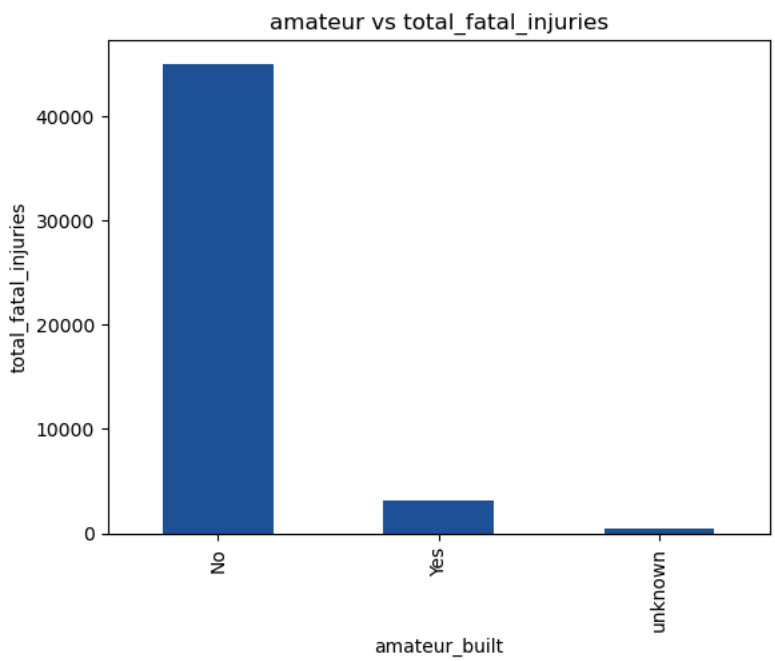
Observation:The more the number of engine the less the fatal engine

```
##Plotting wheather conditions and total fatal injuries variables using matplot
total_gpr=data1.groupby("weather_condition")["total_fatal_injuries"].sum()
total_gpr.plot(kind='bar',color='#1F509A')
plt.ylabel("total_fatal_injuries")
plt.title("weather_condition vs total_fatal_injuries")
plt.show()
```



Observation:When the weather condition is vmk the number of fatal injury is more

```
##plotting amateur_built and total_fatal_injuries using matplot
total_gpr=data1.groupby("amateur_built")["total_fatal_injuries"].sum()
total_gpr.plot(kind='bar',color='#1F509A')
plt.ylabel("total_fatal_injuries")
plt.title("amateur vs total_fatal_injuries")
plt.show()
```



Observation:Amateur built aircraft tend to have less fatal accident

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```