Start coding or generate with AI.

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

```
#load the aviation data with extension CSV
df=pd.read_csv("AviationData.csv",encoding="cp1252")
df
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\4271369542.py:2: DtypeWarning: Columns (6,7,28) have mixed types. Specify dtype option on
  df=pd.read_csv("AviationData.csv",encoding="cp1252")
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpo
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	

88889 rows × 31 columns

```
#head()
df.head()
```

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

5 rows × 31 columns

```
#checking bottom 5 rows
df.tail()
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airport.Na
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	N
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	N
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	PAYS(
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	N
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	N

5 rows × 31 columns

```
#saving the file to csv # use to_csv() and set index = False
#df.to_csv("clean.csv", index=False)
```

```
#setting the default data view # pd.set_option("display.max_columns", 500)
#pd.set_option("display.max_columns", 500)
```

```
#Displaying the top 3 rows # use head()
df.head()
```

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

5 rows × 31 columns

```
#checking top 5 and bottom 5 once
df
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpo
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	

88889 rows × 31 columns

```
#Displaying the last 3 rows # use tail()
```

```
#displaying top 5 and bottom 5 by default
```

```
#Displaying 3 random rows #use sample()
# we use random_state for Reproducibility
df.sample(3, random_state=42)
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpor
55758	20031231X02111	Accident	LAX04CA026	2003-10-25	PETALUMA, CA	United States	38.257778	-122.605556	O69	Pe Mu
75000	20140507X84024	Accident	ERA14CA225	2014-04-29	Chapin, SC	United States	034355N	0812138W	NaN	
81744	20180618X72911	Accident	WPR18LA173	2018-06-16	Hartford, CT	United States	414444N	0723740W	HFD	H E

3 rows × 31 columns

```
#checking the columns # use columns attribute
df.columns
```

```
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')
```

```
#checking the index # use df.index
df.index
```

```
RangeIndex(start=0, stop=88889, step=1)
```

```
#Checking the data Summary # use info(), we can also use info(verbose=False)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      88889 non-null  object
 2   Accident.Number         88889 non-null  object
 3   Event.Date              88889 non-null  object
 4   Location                88837 non-null  object
 5   Country                 88663 non-null  object
 6   Latitude                34382 non-null  object
 7   Longitude               34373 non-null  object
 8   Airport.Code            50132 non-null  object
 9   Airport.Name            52704 non-null  object
 10  Injury.Severity         87889 non-null  object
 11  Aircraft.damage         85695 non-null  object
 12  Aircraft.Category       32287 non-null  object
 13  Registration.Number     87507 non-null  object
 14  Make                    88826 non-null  object
 15  Model                   88797 non-null  object
 16  Amateur.Built           88787 non-null  object
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81793 non-null  object
 19  FAR.Description         32023 non-null  object
 20  Schedule                12582 non-null  object
 21  Purpose.of.flight       82697 non-null  object
 22  Air.carrier             16648 non-null  object
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object
 28  Broad.phase.of.flight   61724 non-null  object
 29  Report.Status           82505 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

```
#if we are intersted in concise info the we add an argument
df.info(verbose=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Columns: 31 entries, Event.Id to Publication.Date
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

```
#checking the datatypes of the columns
df.dtypes
```

```
Event.Id                   object
Investigation.Type         object
Accident.Number            object
Event.Date                 object
Location                   object
Country                    object
Latitude                   object
Longitude                  object
Airport.Code               object
Airport.Name               object
Injury.Severity            object
Aircraft.damage            object
Aircraft.Category          object
Registration.Number        object
Make                       object
Model                      object
Amateur.Built              object
Number.of.Engines         float64
Engine.Type                object
FAR.Description            object
Schedule                   object
Purpose.of.flight          object
```

```
Air.carrier                object
Total.Fatal.Injuries      float64
Total.Serious.Injuries    float64
Total.Minor.Injuries      float64
Total.Uninjured           float64
Weather.Condition          object
Broad.phase.of.flight      object
Report.Status              object
Publication.Date           object
dtype: object
```

```
#Checking concise Descriptive statistics for numerical variables# use df.describe()
df.describe()
```

	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

```
#Using transpose to change row into columns and vice_versa
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Number.of.Engines	82805.0	1.146585	0.446510	0.0	1.0	1.0	1.0	8.0
Total.Fatal.Injuries	77488.0	0.647855	5.485960	0.0	0.0	0.0	0.0	349.0
Total.Serious.Injuries	76379.0	0.279881	1.544084	0.0	0.0	0.0	0.0	161.0
Total.Minor.Injuries	76956.0	0.357061	2.235625	0.0	0.0	0.0	0.0	380.0
Total.Uninjured	82977.0	5.325440	27.913634	0.0	0.0	1.0	2.0	699.0

```
#Selecting the categorical columns and getting statistical summary
 # you use capital O or just mention "object"
df.describe(include='object')
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpo
count	88889	88889	88889	88889	88837	88663	34382	34373	50132	
unique	87951	2	88863	14782	27758	219	25592	27156	10374	
top	20001212X19172	Accident	CEN22LA149	1984-06-30	ANCHORAGE, AK	United States	332739N	0112457W	NONE	
freq	3	85015	2	25	434	82248	19	24	1488	

4 rows × 26 columns

```
df.describe(include= ["float","int"])
```

	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

```
#checking the number of variables and records, use df.shape,   or len(df)

print(f"This datast has is : {df.shape[0]} records and {df.shape[1]} columns")
```

```
This datast has is : 88889 records and 31 columns
```

```
#cheking the lenth
df.shape, len(df)
```

```
((88889, 31), 88889)
```

```
print(f"This dataset has : {df.shape[0]} records")
```

```
This dataset has : 88889 records
```

```
df.columns
```

```
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')
```

```
#subsetting a few columns
df= df[["Event.Id", "Investigation.Type", "Accident.Number", "Event.Date"]]
df
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date
0	20001218X45444	Accident	SEA87LA080	1948-10-24
1	20001218X45447	Accident	LAX94LA336	1962-07-19
2	20061025X01555	Accident	NYC07LA005	1974-08-30
3	20001218X45448	Accident	LAX96LA321	1977-06-19
4	20041105X01764	Accident	CHI79FA064	1979-08-02
...	...	...	...	...
88884	20221227106491	Accident	ERA23LA093	2022-12-26
88885	20221227106494	Accident	ERA23LA095	2022-12-26
88886	20221227106497	Accident	WPR23LA075	2022-12-26
88887	20221227106498	Accident	WPR23LA076	2022-12-26
88888	20221230106513	Accident	ERA23LA097	2022-12-29

88889 rows × 4 columns

```
#selecting the specific columns# use df[['col1', 'col2', 'col3']]
```

```
#checking the column names of the new sub_frame
df.columns
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number', 'Event.Date'], dtype='object')
```

```
# we can use slicing as well e.g. df[df.columns[:5]]
#this select the specific columns
df_specific = df[df.columns[:4]]
df_specific.head()
```

	Event.Id	Investigation.Type	Accident.Number	Event.Date
0	20001218X45444	Accident	SEA87LA080	1948-10-24
1	20001218X45447	Accident	LAX94LA336	1962-07-19
2	20061025X01555	Accident	NYC07LA005	1974-08-30
3	20001218X45448	Accident	LAX96LA321	1977-06-19
4	20041105X01764	Accident	CHI79FA064	1979-08-02

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

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Country	Latitude	Longitude	Airport.Code	Airpo
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	NaN	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	NaN	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	NaN	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	NaN	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
88884	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	NaN	
88885	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	NaN	
88886	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	
88887	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	NaN	
88888	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	NaN	

88889 rows × 31 columns

```
#Changing all the columns into lowercase
df1.columns=df1.columns.str.lower()
df1.columns
```

```
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')
```

```
#Remove white space
df1.columns=df1.columns.str.replace(" ","")
```

```
df1.columns
```

```
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')
```

```
#replace the fullstop with the underscore
df1.columns=df1.columns.str.replace(".","_")
df1.columns
```

```
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')
```

```
#Replace the null value in location using unkown
df1["location"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\1783529423.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["location"].fillna("unknown",inplace=True)
```

```
#Replace the null value in amateur_built using unknown
df1["amateur_built"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\1679168083.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["amateur_built"].fillna("unknown",inplace=True)
```

```
#Replace the null value in model using unknown
df1["model"].fillna("unknown",inplace=True)
```

```
#Replace the null value in the make column using unknown
df1["make"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3829749286.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["make"].fillna("unknown",inplace=True)
```

```
#Replace the null value in registration_number column using unknown
df1["registration_number"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2476904887.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["registration_number"].fillna("unknown",inplace=True)
```

```
#Replace the null value in aircraft_category using unknown
df1["aircraft_category"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3018217498.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me

  df1["aircraft_category"].fillna("unknown",inplace=True)
```

```
#Replace the null value in airport_damage using unknown
df1["aircraft_damage"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3957280873.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me

  df1["aircraft_damage"].fillna("unknown",inplace=True)
```

```
#Replace the null value in injury_severity using unknown
df1["injury_severity"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\116777537.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Ser
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me

  df1["injury_severity"].fillna("unknown",inplace=True)
```

```
#Replace the null value in airport_name using unknown
df1["airport_name"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2521067570.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me

  df1["airport_name"].fillna("unknown",inplace=True)
```

```
#Replace the null value in airport_code using unknown
df1["airport_code"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2051564295.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me

  df1["airport_code"].fillna("unknown",inplace=True)
```

```
#Replace the null value in country  using unknown
df1["country"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\932111147.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Ser
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me

  df1["country"].fillna("unknown",inplace=True)
```

```
#Replace the null value in the total_uninjuredcolumn with the median because it is skewed to the right
median_min=df1["total_uninjured"].median()
df1["total_uninjured"].fillna(median_min,inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3227217894.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["total_uninjured"].fillna(median_min,inplace=True)
```

```
#Replace the null value in the total_minor_injuries column with the median because it is skewed to the right
median_min=df1["total_minor_injuries"].median()
df1["total_minor_injuries"].fillna(median_min,inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3162184480.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["total_minor_injuries"].fillna(median_min,inplace=True)
```

```
#Replace the null value in airport_code using unknown
df1["broad_phase_of_flight"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\1538067145.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["broad_phase_of_flight"].fillna("unknown",inplace=True)
```

```
#Replace the null value in air_carrier using unknown
df1["air_carrier"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\388962569.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Ser
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["air_carrier"].fillna("unknown",inplace=True)
```

```
#Replace the null value in purpose_of_flight using unknown
df1["purpose_of_flight"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3184402981.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["purpose_of_flight"].fillna("unknown",inplace=True)
```

```
#Replace the null value in schedule using unknown
df1["schedule"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2268203621.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["schedule"].fillna("unknown",inplace=True)
```

```
#Replace the null value in far_description using unknown
df1["far_description"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\3026423423.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["far_description"].fillna("unknown",inplace=True)
```

```
#Replace the null value in engine_type using unknown
df1["engine_type"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\690722306.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Ser
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["engine_type"].fillna("unknown",inplace=True)
```

```
#Replace the null value in number_of_engines using unknown
df1["number_of_engines"].fillna("unknown",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2325848231.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df1["number_of_engines"].fillna("unknown",inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2325848231.py:2: FutureWarning: Setting an item of incompatible dtype is deprecated and wi
  df1["number_of_engines"].fillna("unknown",inplace=True)
```

```
df1.columns
```

```
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 the weather column we have both unk and UNK which differ because of lower and upper case we must put them together
df1.groupby("weather_condition")["weather_condition"].count()
```

```
weather_condition
IMC     5976
UNK      856
Unk      262
VMC    77303
Name: weather_condition, dtype: int64
```

```
#Replacing the unk with UNK
df1["weather_condition"]=df1["weather_condition"].str.replace("Unk","UNK")
```

```
#To confirm that unk is in UNK
df1.groupby("weather_condition")["weather_condition"].count()
```

```
weather_condition
IMC     5976
UNK     5610
VMC    77303
Name: weather_condition, dtype: int64
```

```
#Removing null values in weather using forward fill
df1["weather_condition"].fillna("UNK",inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\2038692622.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Se
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
```

```
df1["weather_condition"].fillna("UNK",inplace=True)
```

```
df1.groupby("engine_type")["engine_type"].count()
```

```
engine_type
Electric              10
Geared Turbofan       12
Hybrid Rocket          1
LR                     2
NONE                   2
Reciprocating      69530
Turbo Fan           2481
Turbo Jet            703
Turbo Prop          3391
Turbo Shaft         3609
UNK                    1
Unknown             2051
unknown             7096
Name: engine_type, dtype: int64
```

Observation: there are many unknowns that should be put together

```
#Replacing the Unknown to the UNK
df1["engine_type"]=df1["engine_type"].str.replace("Unknown","UNK")
```

```
#Replacing the unknown to the UNK
df1["engine_type"]=df1["engine_type"].str.replace("unknown","UNK")
```

```
#Confirm if Unkown,unkown have been replaced by UKN
df1.groupby("engine_type")["engine_type"].count()
```

```
engine_type
Electric              10
Geared Turbofan       12
Hybrid Rocket          1
LR                     2
NONE                   2
Reciprocating      69530
Turbo Fan           2481
Turbo Jet            703
Turbo Prop          3391
Turbo Shaft         3609
UNK                 9148
Name: engine_type, dtype: int64
```

Observation: The null values have been removed

```
#Checking value counts
df1.groupby("broad_phase_of_flight")["broad_phase_of_flight"].count()
```

```
broad_phase_of_flight
Approach        6546
Climb           2034
Cruise         10269
Descent         1887
Go-around       1353
Landing        15428
Maneuvering     8144
Other            119
Standing         945
Takeoff        12493
Taxi            1958
Unknown          548
unknown        27165
Name: broad_phase_of_flight, dtype: int64
```

Observation: Two unknowns

```
#Replacing the unknown to the Unknown
df1["broad_phase_of_flight"]=df1["broad_phase_of_flight"].str.replace("unknown","Unknown")
```

```
#Confirm the removal
df1.groupby("broad_phase_of_flight")["broad_phase_of_flight"].count()
```

```
broad_phase_of_flight
Approach        6546
Climb           2034
Cruise         10269
Descent         1887
Go-around       1353
Landing        15428
Maneuvering     8144
Other            119
Standing         945
Takeoff        12493
Taxi            1958
Unknown        27713
Name: broad_phase_of_flight, dtype: int64
```

```
#confirm is all the null value have been removed
df1.isnull().sum()
```

```
event_id                      0
investigation_type            0
accident_number               0
event_date                    0
location                      0
country                       0
latitude                  54507
longitude                 54516
airport_code                  0
airport_name                  0
injury_severity               0
aircraft_damage               0
aircraft_category             0
registration_number           0
make                          0
model                         0
amateur_built                 0
number_of_engines             0
engine_type                   0
far_description               0
schedule                      0
purpose_of_flight             0
air_carrier                   0
total_fatal_injuries      11401
total_serious_injuries        0
total_minor_injuries          0
total_uninjured               0
weather_condition             0
broad_phase_of_flight         0
report_status              6384
publication_date          13771
dtype: int64
```

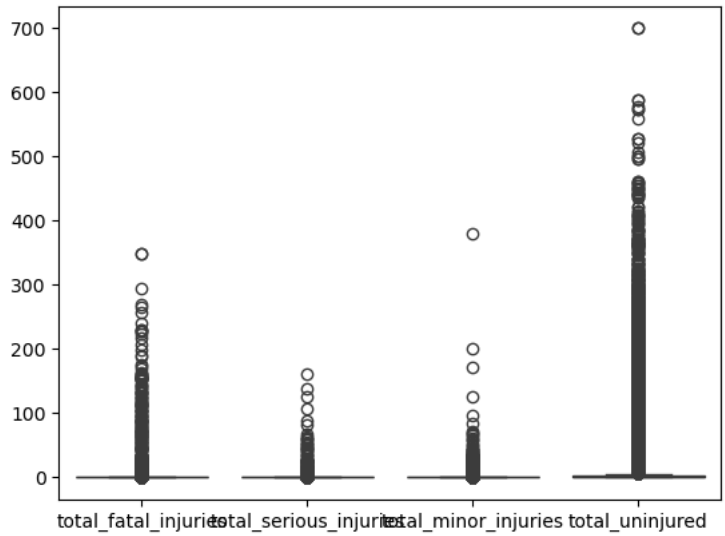
```
#Checking for duplicates
df_duplicates=df.duplicated().sum()
df_duplicates
```

```
0
```

Observation: there are no duplicates

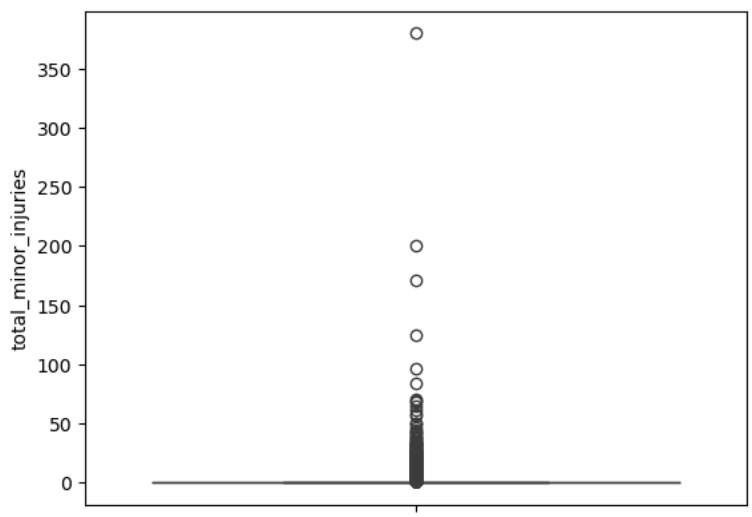
```
#Checking for outliers here we use boxplot to check for outliers
sns.boxplot(df1)
```

```
<Axes: >
```




```
#plotting the total_minor_injuries inorder to remove the outlier
sns.boxplot(df1["total_minor_injuries"])
```

```
<Axes: ylabel='total_minor_injuries'>
```

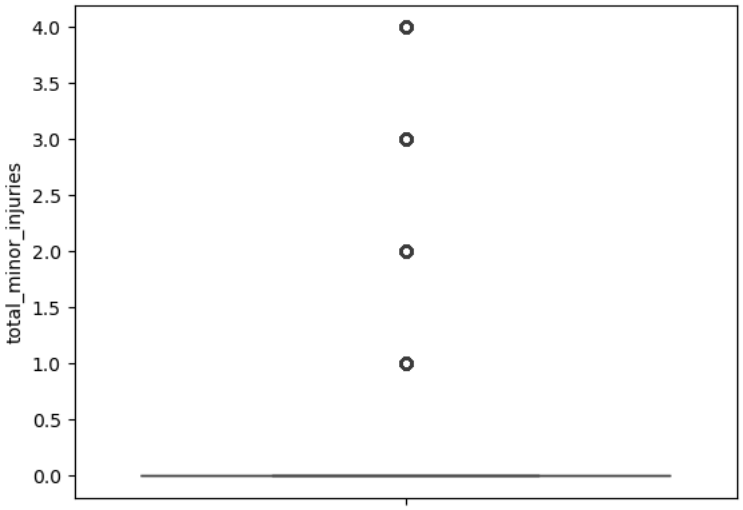



```
#Use the maximum quantile method
max_total_minor=df1["total_minor_injuries"].quantile(0.995)
max_total_minor
```

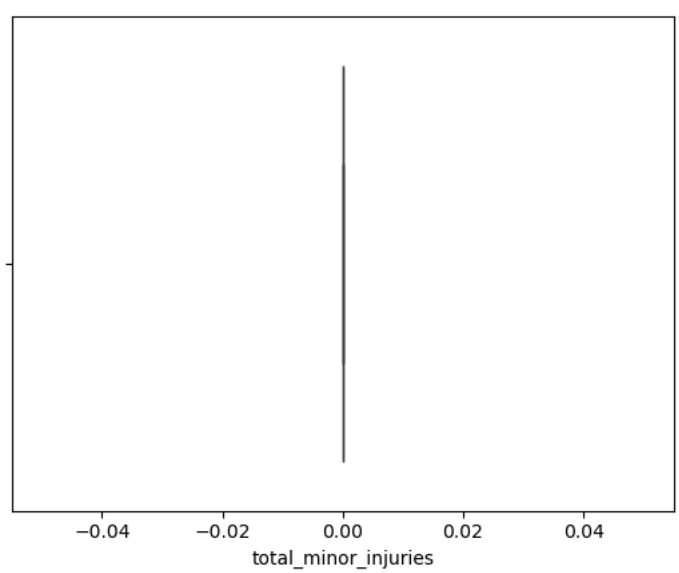
```
5.0
```

```
df2=df1[df1["total_minor_injuries"]>max_total_minor]
df2=df1[df1["total_minor_injuries"]<max_total_minor]
sns.boxplot(df2["total_minor_injuries"])
```

```
<Axes: ylabel='total_minor_injuries'>
```

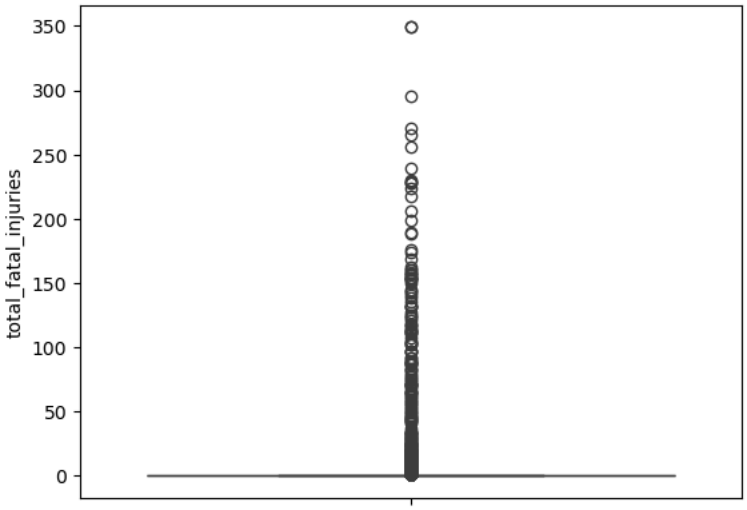



```
#Removing the outlier using the iqr
q1=df1["total_minor_injuries"].quantile(0.25)
q3=df1["total_minor_injuries"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df3=df1[(df1["total_minor_injuries"]>=lower_bound) & (df1["total_minor_injuries"]<=upper_bound)]
sns.boxplot(x="total_minor_injuries",data=df3);
```




```
#plotting the 'total_fatal_injuries'inorder to remove the outlier
sns.boxplot(df1['total_fatal_injuries'])
```

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

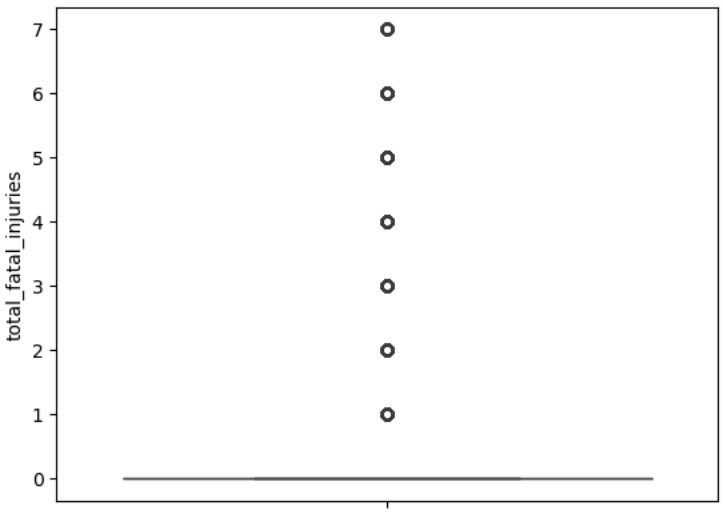



```
max_total_fatal=df1["total_fatal_injuries"].quantile(0.995)
max_total_fatal
```

```
8.0
```

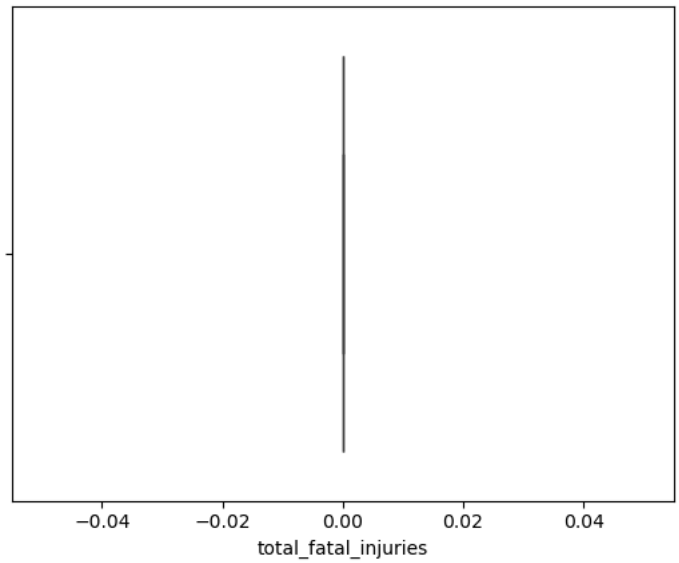
```
df2=df1[df1["total_fatal_injuries"]>max_total_fatal]
df4=df1[df1["total_fatal_injuries"]<max_total_fatal]
sns.boxplot(df4["total_fatal_injuries"])
```

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



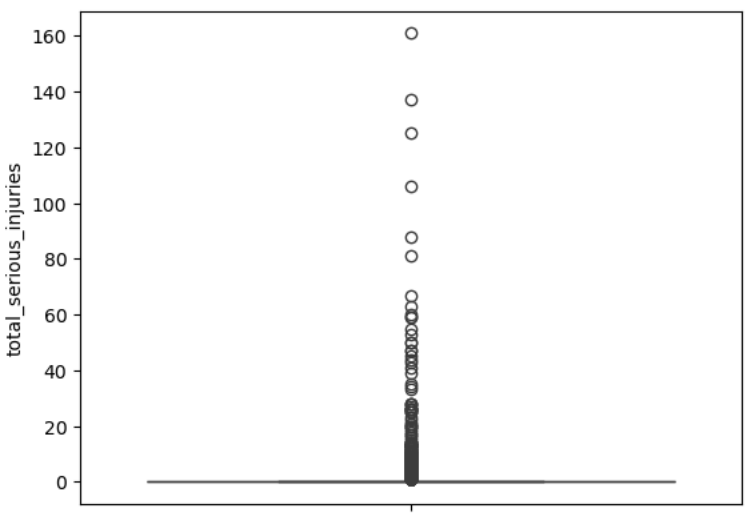

```
#Using the second method iqr
q1=df1["total_fatal_injuries"].quantile(0.25)
q3=df1["total_fatal_injuries"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df5=df1[(df1["total_fatal_injuries"]>=lower_bound) & (df1["total_fatal_injuries"]<=upper_bound)]
sns.boxplot(x="total_fatal_injuries",data=df5)
```

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



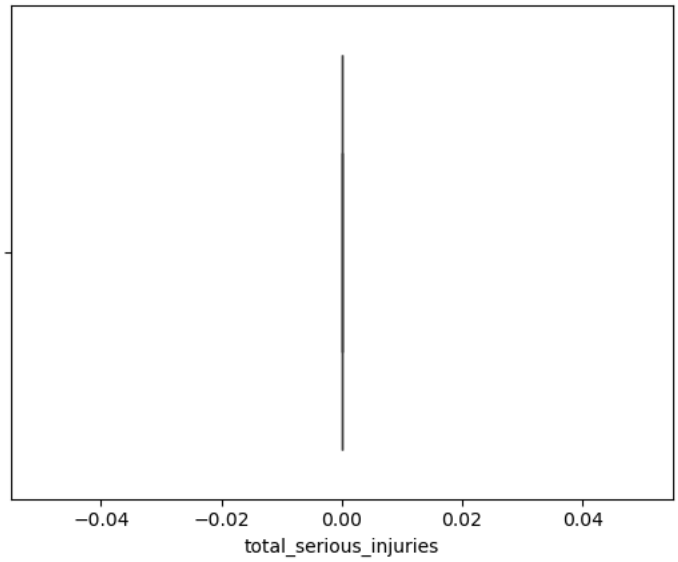

```
#plotting the 'total_serious_injuries' inorder to remove the outlier
sns.boxplot(df1["total_serious_injuries"])
```

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



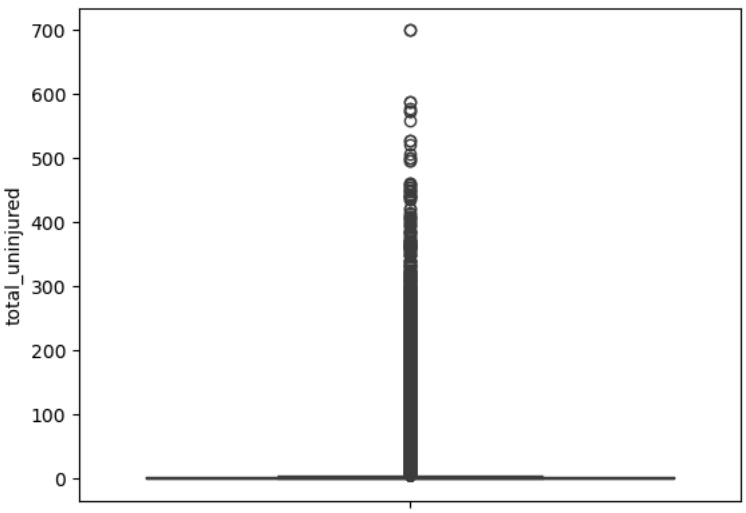

```
#We use the iqr to remove outliers
q1=df1["total_serious_injuries"].quantile(0.25)
q3=df1["total_serious_injuries"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df6=df1[(df1["total_serious_injuries"]>=lower_bound) & (df1["total_serious_injuries"]<=upper_bound)]
sns.boxplot(x="total_serious_injuries",data=df6)
```

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



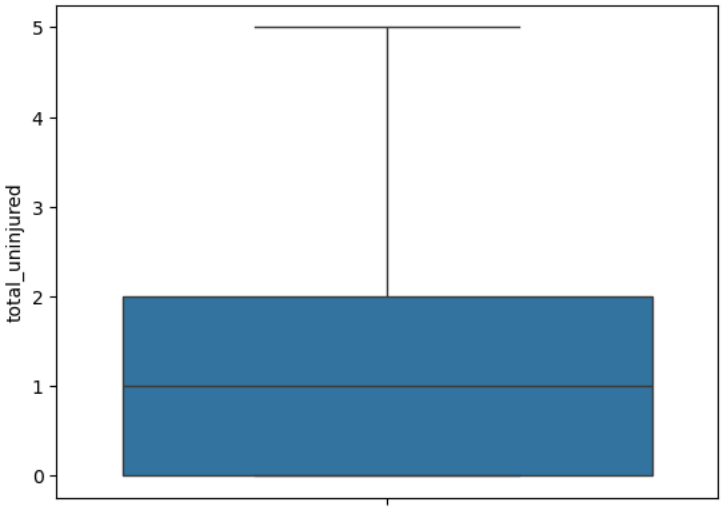

```
#plotting the 'total_uninjured' inorder to remove the outlier
sns.boxplot(df1["total_uninjured"])
```

```
<Axes: ylabel='total_uninjured'>
```




```
#Removing the outlier using the iqr
q1=df1["total_uninjured"].quantile(0.25)
q3=df1["total_uninjured"].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df7=df1[(df1["total_uninjured"]>=lower_bound) & (df1["total_uninjured"]<=upper_bound)]
sns.boxplot(y="total_uninjured",data=df7)
```

```
<Axes: ylabel='total_uninjured'>
```



SAVING THE NEW DATASET

```
df7.to_csv("cleannynt_d.csv",index=False)
```

EDA

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9640\757037666.py:2: DtypeWarning: Columns (6,7) have mixed types. Specify dtype option on impo
  data=pd.read_csv("cleannynt_d.csv")
```

	event_id	investigation_type	accident_number	event_date	location	country	latitude	longitude	airport_code	airpo
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	unknown	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	unknown	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	unknown	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	unknown	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	unknown	
...	...	...	...	...	...	...	...	...	...	
84737	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	unknown	
84738	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	unknown	
84739	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	
84740	20221227106498	Accident	WPR23LA076	2022-12-26	Morgan, UT	United States	NaN	NaN	unknown	
84741	20221230106513	Accident	ERA23LA097	2022-12-29	Athens, GA	United States	NaN	NaN	unknown	

84742 rows × 31 columns

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

	event_id	investigation_type	accident_number	event_date	location	country	latitude	longitude	airport_code	airpo
0	20001218X45444	Accident	SEA87LA080	1948-10-24	MOOSE CREEK, ID	United States	NaN	NaN	unknown	
1	20001218X45447	Accident	LAX94LA336	1962-07-19	BRIDGEPORT, CA	United States	NaN	NaN	unknown	
2	20061025X01555	Accident	NYC07LA005	1974-08-30	Saltville, VA	United States	36.922223	-81.878056	unknown	
3	20001218X45448	Accident	LAX96LA321	1977-06-19	EUREKA, CA	United States	NaN	NaN	unknown	
4	20041105X01764	Accident	CHI79FA064	1979-08-02	Canton, OH	United States	NaN	NaN	unknown	
...	...	...	...	...	...	...	...	...	...	
84737	20221227106491	Accident	ERA23LA093	2022-12-26	Annapolis, MD	United States	NaN	NaN	unknown	
84738	20221227106494	Accident	ERA23LA095	2022-12-26	Hampton, NH	United States	NaN	NaN	unknown	
84739	20221227106497	Accident	WPR23LA075	2022-12-26	Payson, AZ	United States	341525N	1112021W	PAN	
						United				