

# PROJECT 1 ASSIGNMENT

## Import the necessary libraries

```
In [63]: import pandas as pd
import numpy as np
from numbers import Number
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

## Load the dataset

The encoding parameter is used to handle any special characters that may be present in the file.

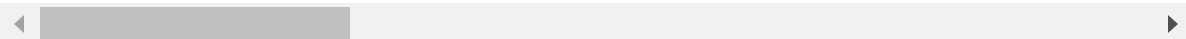
```
In [2]: df = pd.read_csv('AviationData.csv', encoding='latin-1')
```

```
In [3]: # Display the first few rows of the dataframe.
df.head()
```

```
Out[3]:
```

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

5 rows × 31 columns



```
In [4]: # Display the first few rows of the dataframe.
df.describe()
```

Out[4]:

	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	Total.Minor.Injuries	Tot
<b>count</b>	82805.000000	77488.000000	76379.000000	76956.000000	8
<b>mean</b>	1.146585	0.647855	0.279881	0.357061	
<b>std</b>	0.446510	5.485960	1.544084	2.235625	
<b>min</b>	0.000000	0.000000	0.000000	0.000000	
<b>25%</b>	1.000000	0.000000	0.000000	0.000000	
<b>50%</b>	1.000000	0.000000	0.000000	0.000000	
<b>75%</b>	1.000000	0.000000	0.000000	0.000000	
<b>max</b>	8.000000	349.000000	161.000000	380.000000	

In [5]: *# Display the last few rows of the dataframe*  
`df.tail()`

Out[5]:

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

5 rows × 31 columns

In [6]: `df.shape`

Out[6]: (88889, 31)

## CLEANING THE DATA

In [9]: `print(df.isnull().sum())`

Event.Id	0
Investigation.Type	0
Accident.Number	0
Event.Date	0
Location	52
Country	226
Latitude	54507
Longitude	54516
Airport.Code	38757
Airport.Name	36185
Injury.Severity	1000
Aircraft.damage	3194
Aircraft.Category	56602
Registration.Number	1382
Make	63
Model	92
Amateur.Built	102
Number.of.Engines	6084
Engine.Type	7096
FAR.Description	56866
Schedule	76307
Purpose.of.flight	6192
Air.carrier	72241
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	6384
Publication.Date	13771

dtype: int64

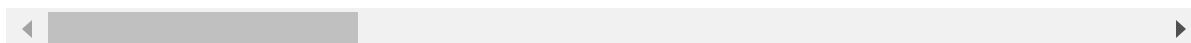
```
In [10]: # - Handle missing values (e.g., drop rows, fill with mean/median/mode, or use im
df.fillna(method='ffill', inplace=True) # Forward-fill missing values
```

```
In [11]: df.head()
```

Out[11]:

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

5 rows × 31 columns



Convert 'Publication.Date' and 'Event.Date' columns to datetime objects.

Convert specific columns to integer type.

```
In [12]: # - Convert columns to appropriate data types (e.g., date, numerical)
df['Publication.Date'] = pd.to_datetime(df['Publication.Date'])
df['Total.Fatal.Injuries'] = df['Total.Fatal.Injuries'].astype(int)
df['Total.Serious.Injuries'] = df['Total.Serious.Injuries'].astype(int)
df['Total.Minor.Injuries'] = df['Total.Minor.Injuries'].astype(int)
df['Total.Uninjured'] = df['Total.Uninjured'].astype(int)
df['Event.Date'] = pd.to_datetime(df['Event.Date'])
```

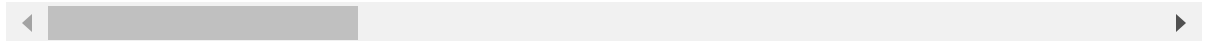
```
In [13]: # Mode imputation for 'Airport.Code'
df['Airport.Code'].fillna(df['Airport.Code'].mode()[0], inplace=True)
```

```
In [14]: df.head()
```

Out[14]:

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

5 rows × 31 columns

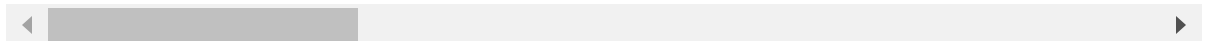


In [15]: df.head()

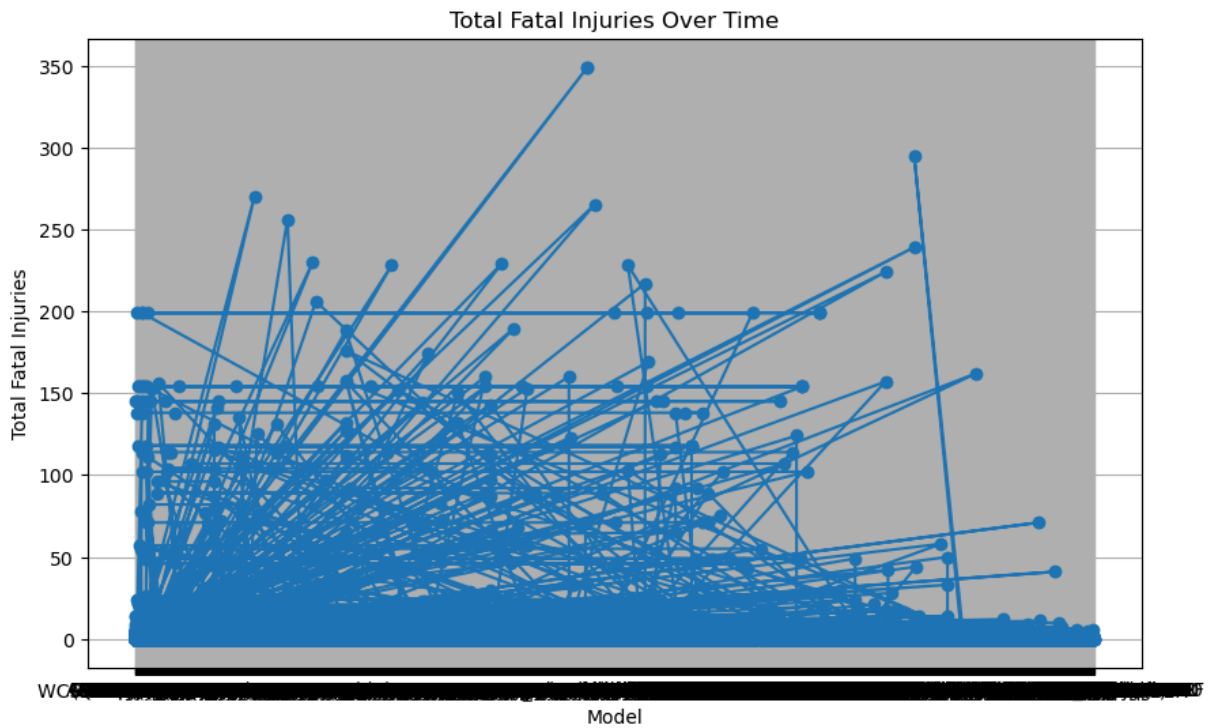
Out[15]:

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

5 rows × 31 columns



```
In [16]: # Analyze accident trends over time
plt.figure(figsize=(10, 6))
plt.plot(df['Model'], df['Total.Fatal.Injuries'], marker='o')
plt.title('Total Fatal Injuries Over Time')
plt.xlabel('Model')
plt.ylabel('Total Fatal Injuries')
plt.grid(True)
plt.show()
```



Check for and handle any remaining missing values after initial cleaning

Further data cleaning based on specific column needs

Example: If 'Latitude' or 'Longitude' have missing values, consider imputation or removal

... (add more specific cleaning steps as needed)

```
In [31]: print(df.isnull().sum())

# Remove rows where 'Total.Fatal.Injuries' are greater than the total number of people
total_injured = df['Total.Fatal.Injuries'] + df['Total.Serious.Injuries'] + df['Total.Minor.Injuries']
df = df[df['Total.Fatal.Injuries'] <= total_injured]

# Ensure consistent data types across the dataframe
for col in df.columns:
    if df[col].dtype == 'object':
        try:
            df[col] = pd.to_numeric(df[col])
        except ValueError:
            pass # Skip if conversion to numeric fails

# Example: Handling inconsistent values in a categorical column (replace with appropriate value)
# df['Make'] = df['Make'].replace({'CESSNA AIRCRAFT': 'CESSNA'}) # Replace inconsistent values

# Drop rows where 'Total.Fatal.Injuries' is negative (if any)
df = df[df['Total.Fatal.Injuries'] >= 0]

df.head()
```

```

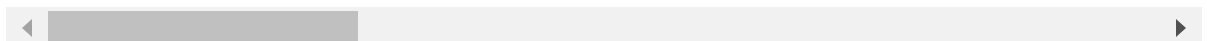
Event.Id          0
Investigation.Type 0
Accident.Number   0
Event.Date        0
Location          0
Country           0
Latitude          2
Longitude         2
Airport.Code      0
Airport.Name      7
Injury.Severity   0
Aircraft.damage   0
Aircraft.Category 5
Registration.Number 0
Make             0
Model            0
Amateur.Built     0
Number.of.Engines 0
Engine.Type       0
FAR.Description   5
Schedule          5
Purpose.of.flight 0
Air.carrier       5
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
Publication.Date  1
dtype: int64

```

Out[31]:

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

5 rows × 31 columns

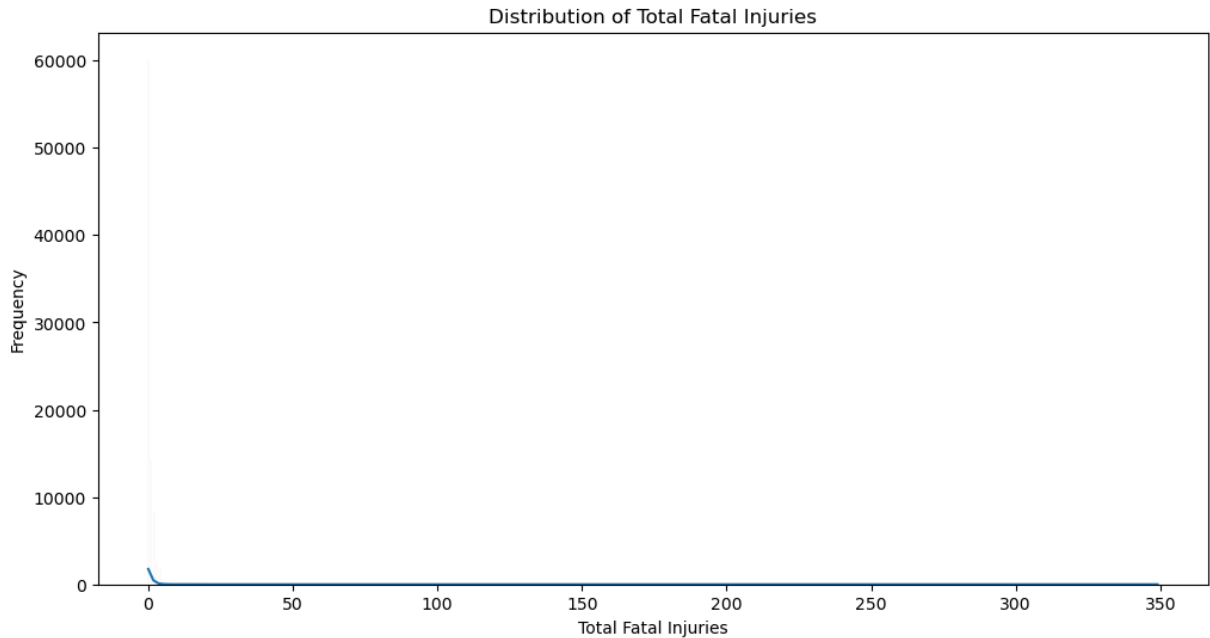


```

In [44]: # Analyze injury severity distribution
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.In
df[injury_columns].describe()

```

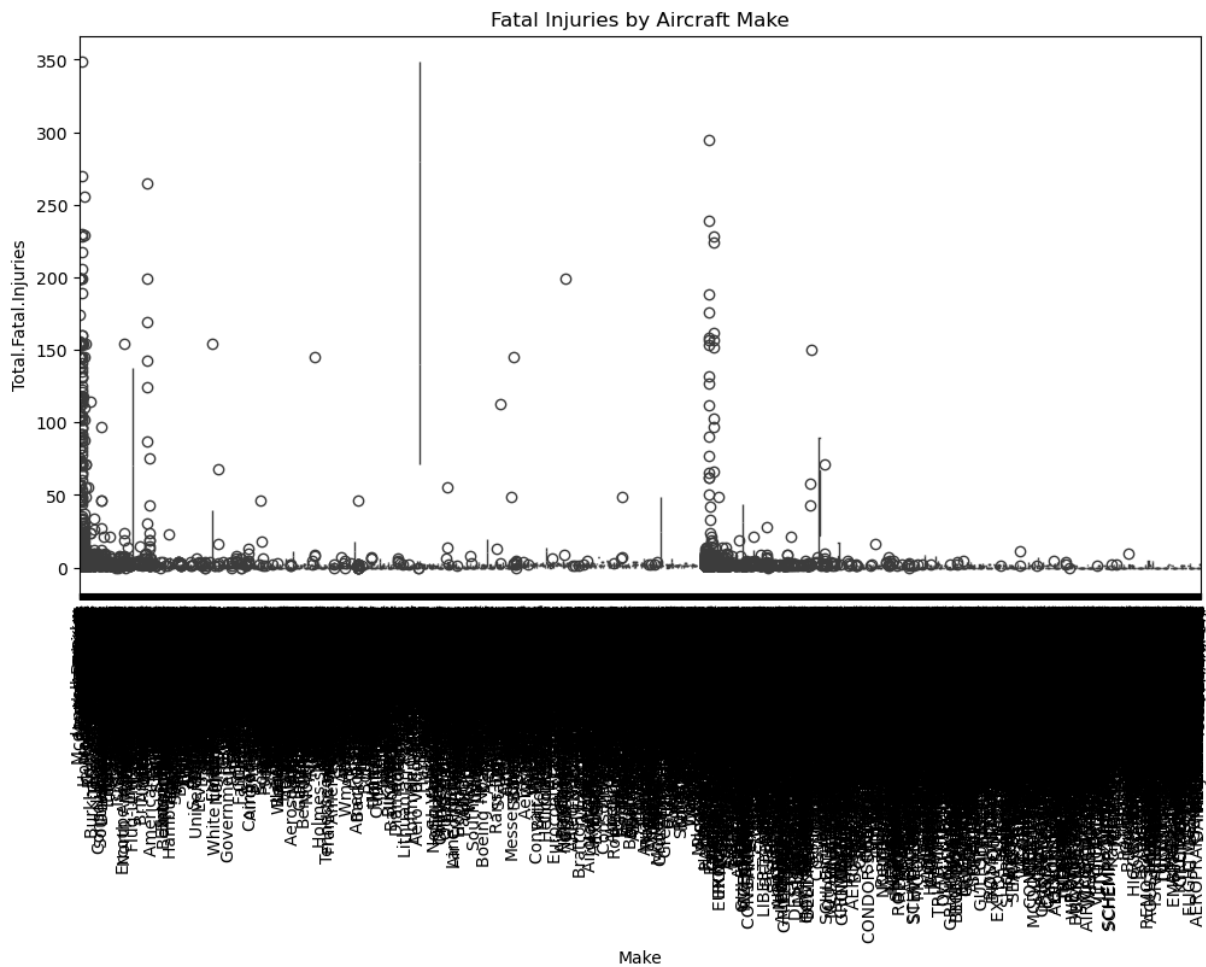
```
# Visualize the distribution of total fatal injuries using a histogram.  
plt.figure(figsize=(12, 6))  
sns.histplot(df['Total.Fatal.Injuries'], kde=True)  
plt.title('Distribution of Total Fatal Injuries')  
plt.xlabel('Total Fatal Injuries')  
plt.ylabel('Frequency')  
plt.show()
```



**Investigate the relationship between aircraft make and fatal injuries using a boxplot.**

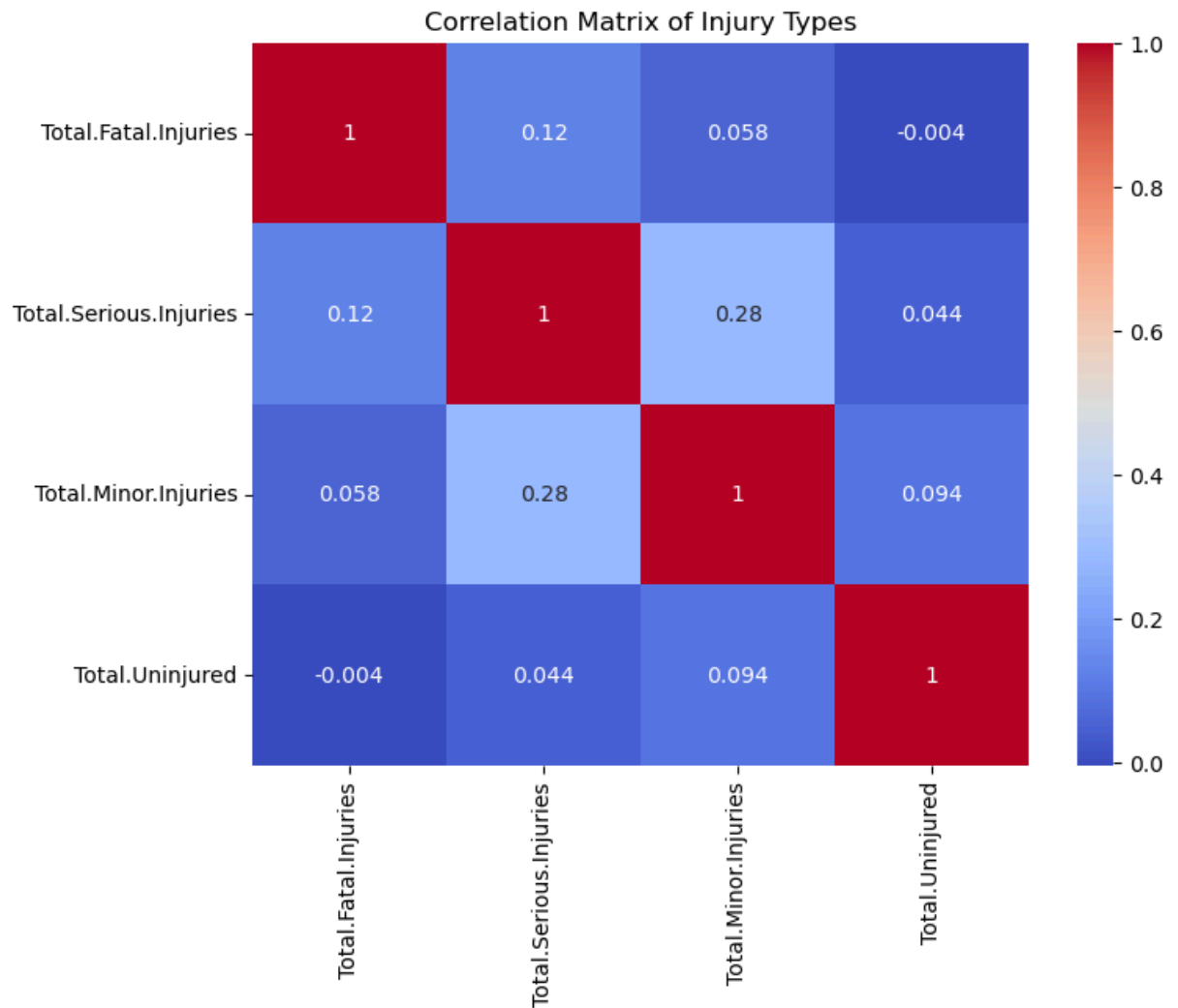
```
In [47]: plt.figure(figsize=(12, 6))  
sns.boxplot(x='Make', y='Total.Fatal.Injuries', data=df)  
plt.xticks(rotation=90)  
plt.title('Fatal Injuries by Aircraft Make')  
plt.show()
```





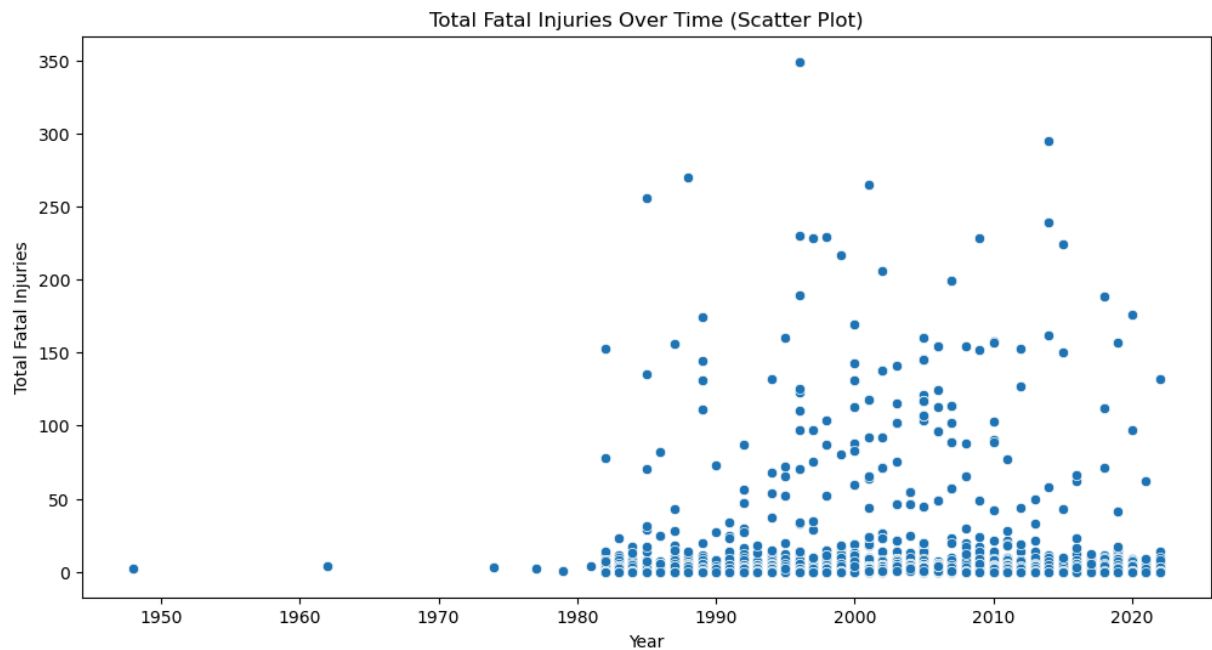
Analyze the correlation between features using a heatmap.

```
In [48]: correlation_matrix = df[['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.M
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Injury Types')
plt.show()
```



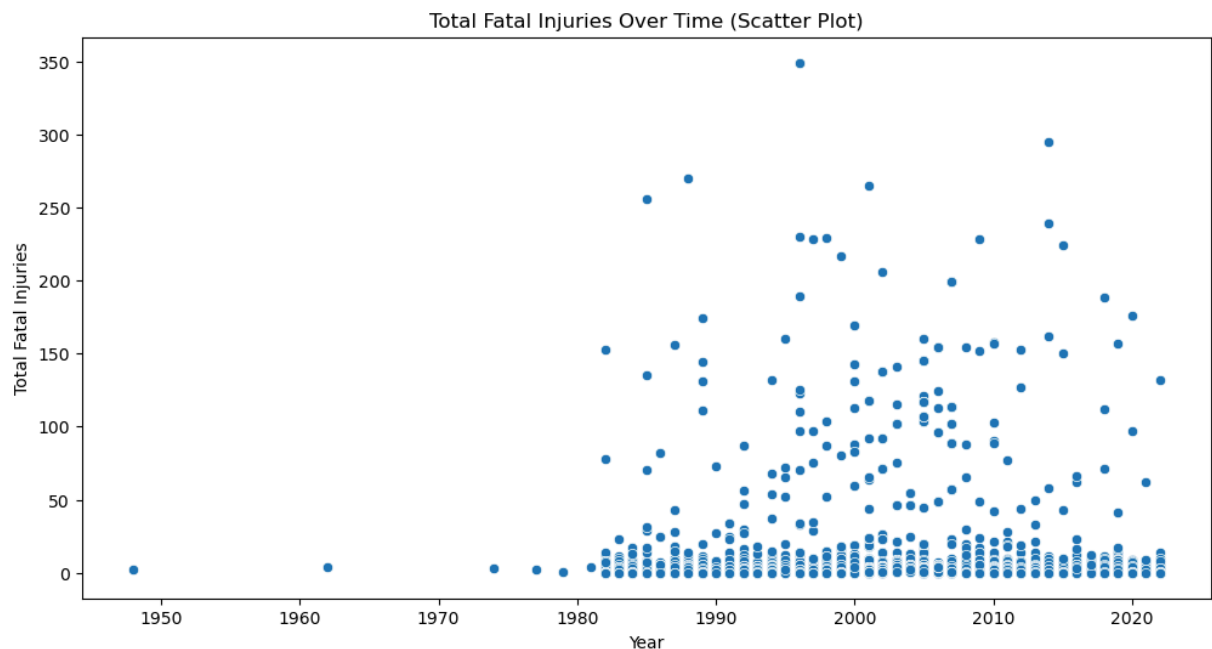
**Analyze the relationship between the year of the event and total fatal injuries**

```
In [35]: df['Event.Year'] = df['Event.Date'].dt.year
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Event.Year', y='Total.Fatal.Injuries', data=df)
plt.title('Total Fatal Injuries Over Time (Scatter Plot)')
plt.xlabel('Year')
plt.ylabel('Total Fatal Injuries')
plt.show()
```



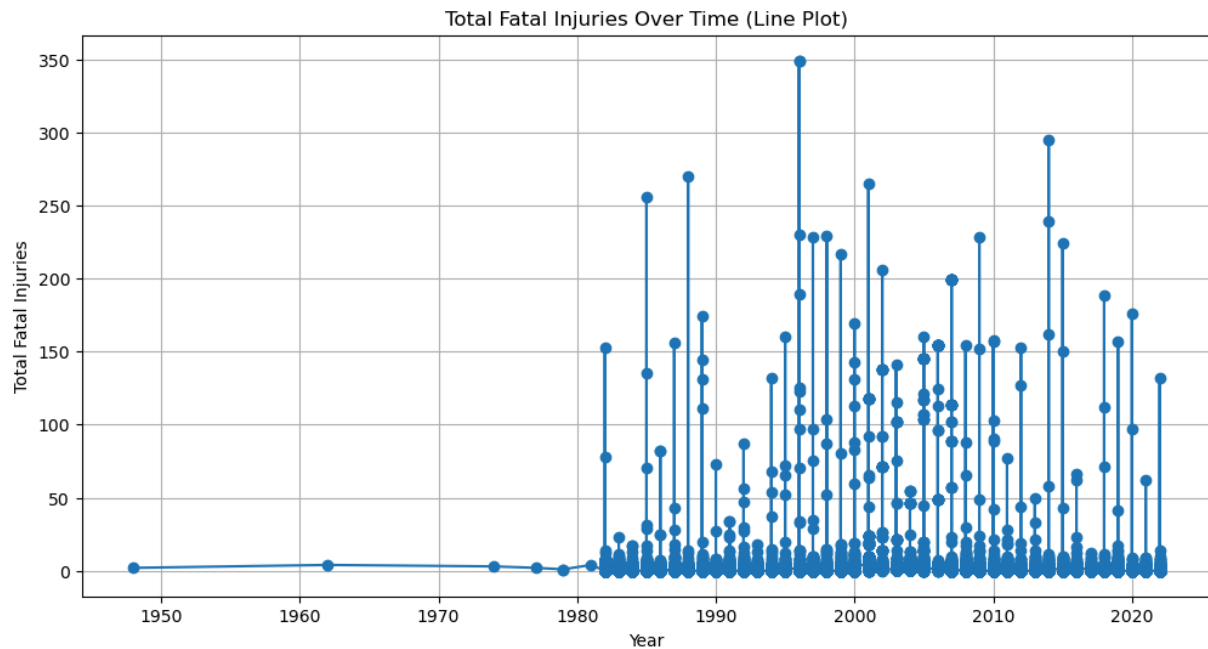
Analyze the relationship between the year and total fatal injuries using a scatterplot.

```
In [42]: plt.figure(figsize=(12, 6))
sns.scatterplot(x='Event.Year', y='Total.Fatal.Injuries', data=df)
plt.title('Total Fatal Injuries Over Time (Scatter Plot)')
plt.xlabel('Year')
plt.ylabel('Total Fatal Injuries')
plt.show()
```



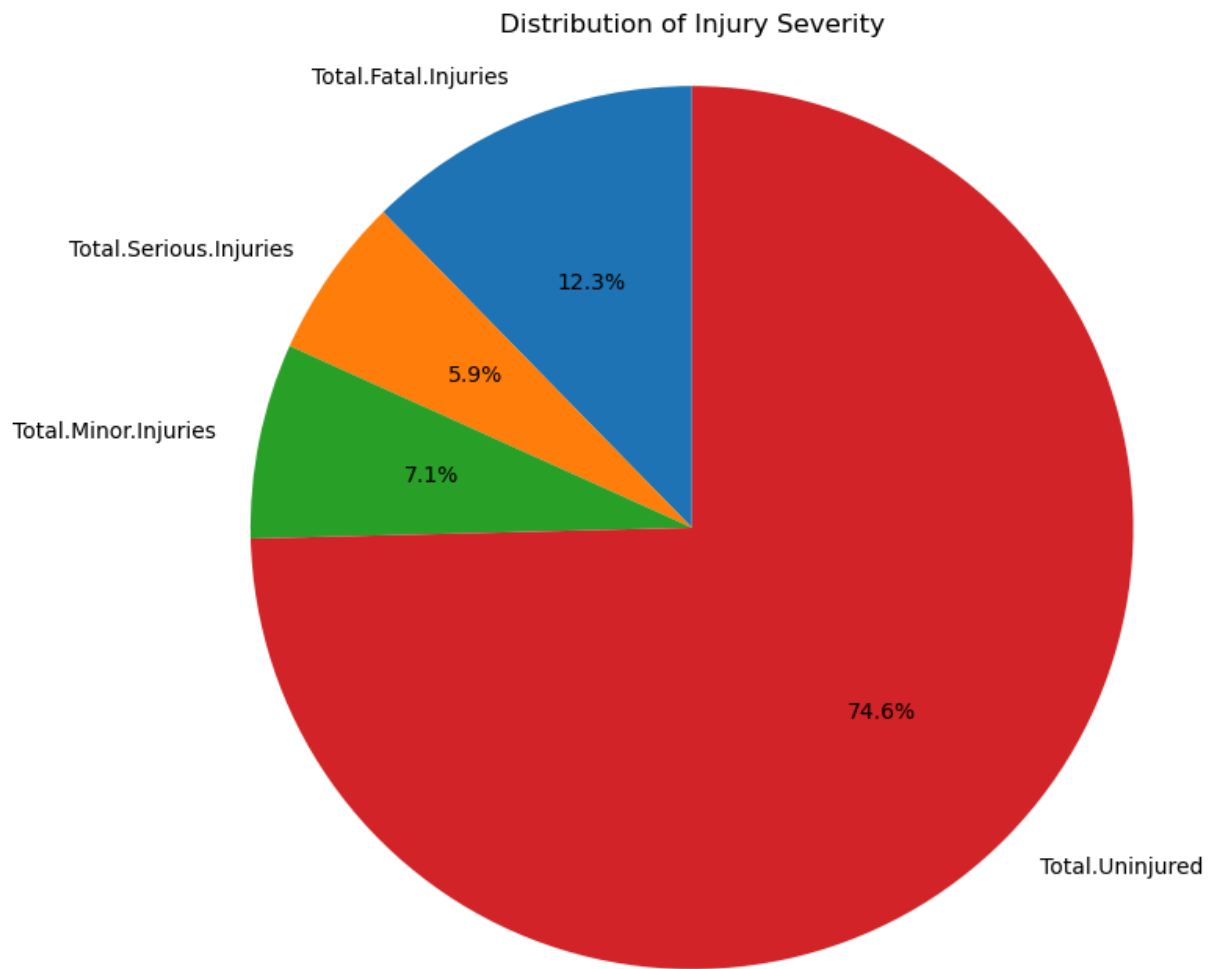
```
In [59]: # Analyze the relationship between the year of the event and total fatal injuries u
plt.figure(figsize=(12, 6))
plt.plot(df['Event.Year'], df['Total.Fatal.Injuries'], marker='o', linestyle='--')
plt.title('Total Fatal Injuries Over Time (Line Plot)')
```

```
plt.xlabel('Year')
plt.ylabel('Total Fatal Injuries')
plt.grid(True) # Add grid for better readability
plt.show()
```



```
In [61]: # Assuming 'injury_columns' from your existing code holds the relevant data
injury_columns = ['Total.Fatal.Injuries', 'Total.Serious.Injuries', 'Total.Minor.In
injury_data = df[injury_columns].sum()

plt.figure(figsize=(8, 8))
plt.pie(injury_data, labels=injury_data.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Injury Severity')
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



In [ ]: