Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
```

In [2]:
```
df = pd.read_csv("/Users/apple/Desktop/Data 557/Project/US_Accidents_King_County.csv")
```

In [3]:
```
df.columns
```

Out[3]:
```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
       'Astronomical_Twilight', 'year', 'weekday', 'hours'],
      dtype='object')
```

Exploratory Data Analysis

In [4]:
```
df['Severity'].describe()
```

Out[4]:
```
count    15903.000000
mean         2.276237
std          0.620546
min          1.000000
25%          2.000000
50%          2.000000
75%          2.000000
max          4.000000
Name: Severity, dtype: float64
```

In [5]:
```
df['Distance(mi)'].describe()
```

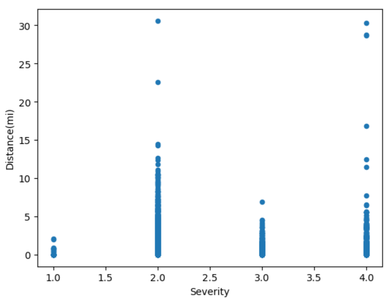
Out[5]:
```
count    15903.000000
mean         0.551042
std          1.076373
min          0.000000
25%          0.058000
50%          0.249000
75%          0.645000
max         30.579000
Name: Distance(mi), dtype: float64
```

In [6]:
```
df.corr()['Severity']['Distance(mi)']
```

Out[6]:
```
-0.00025702207729784914
```

In [7]:
```
df.plot.scatter(x='Severity', y='Distance(mi)')
```

Out[7]:
```
<AxesSubplot:xlabel='Severity', ylabel='Distance(mi)'>
```



MAIN ANALYSIS

Extract Severity and Distance(mi) into a new dataframe

In [8]:
```
df_new = df[['Severity','Distance(mi)']]
```

Create new dataframe of severe accidents where we have values of severity 3 and 4

In [9]:
```
df_severe = df_new.loc[df_new['Severity']>2]
```

Calculate variance of the severe accidents dataframe

In [10]:
```
print(np.var(df_severe['Distance(mi)']))
```

```
1.353799066026002
```

Create new dataframe of non-severe accidents where we have values of severity 1 and 2

In [11]:
```
df_not_severe = df_new.loc[df_new['Severity'] <= 2]
```

Calculate variance of the non-severe accidents dataframe

In [12]:
```
print(np.var(df_not_severe['Distance(mi)']))
```

```
1.100356638635528
```

Two sample t-test with unequal variance

In [13]:
```
stats.ttest_ind(a=df_severe['Distance(mi)'], b=df_not_severe['Distance(mi)'], equal_var=False)
```

Out[13]:
```
Ttest_indResult(statistic=-5.296842257720126, pvalue=1.2265537883671376e-07)
```

Calculating 95% confidence interval for mean distance(mi) in the not severe group and point value of mean distance(mi)

In [14]:
```
import statsmodels.stats.api as sms

# Calculate the confidence interval
ci = sms.DescrStatsW(df_not_severe['Distance(mi)']).tconfint_mean()

# Print the confidence interval
print(f"The 95% confidence interval is {ci}")
```

```
The 95% confidence interval is (0.5580211949019754, 0.5949554142285295)
```

In [15]:
```
avg_non_severe = np.mean(df_not_severe['Distance(mi)'])
print(avg_non_severe)
```

```
0.5764883045652548
```

Calculating 95% confidence interval for mean distance(mi) in the severe group and point value of mean distance(mi)

In [16]:
```
# Calculate the confidence interval
ci = sms.DescrStatsW(df_severe['Distance(mi)']).tconfint_mean()

# Print the confidence interval
print(f"The 95% confidence interval is {ci}")
```

```
The 95% confidence interval is (0.4224935980176451, 0.49957031068420954)
```

In [17]:
```
avg_severe = np.mean(df_severe['Distance(mi)'])
print(avg_severe)
```

```
0.46103195435092786
```

Preparing 4 groups from our initial data for one-way ANOVA test

In [18]:
```
x1 = df_new['Distance(mi)'].loc[df_new['Severity'] == 1]
x2 = df_new['Distance(mi)'].loc[df_new['Severity'] == 2]
x3 = df_new['Distance(mi)'].loc[df_new['Severity'] == 3]
x4 = df_new['Distance(mi)'].loc[df_new['Severity'] == 4]
```

In [19]:
```
stats.f_oneway(x1,x2,x3,x4)
```

Out[19]:
```
F_onewayResult(statistic=62.82573535823872, pvalue=2.2526865859276457e-40)
```

In [20]:
```
import matplotlib.pyplot as plt
```

In [21]:
```
# Create a figure and axis
fig, ax = plt.subplots(figsize=(8,6))

# Set the title and axis labels
ax.set_title('Boxplots of Multiple Categories')
ax.set_xlabel('Groups')
ax.set_ylabel('Values')

# Create the boxplot for each group
ax.boxplot([df_new[df_new['Severity'] == g]['Distance(mi)'] for g in df_new['Severity'].unique()])

# Set the x-axis tick labels to the group names
ax.set_xticklabels(df_new['Severity'].unique())

# Show the plot
plt.show()
```