 # S AVINASH GUPTA 21BCE7754 SLOT-> MORNING SLOT 10-12 AL-ML

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

data=sns.load_dataset('car_crashes')

data.head()
```

```
   total  speeding  alcohol  not_distracted  no_previous  ins_premium
\
0   18.8     7.332    5.640          18.048       15.040       784.55

1   18.1     7.421    4.525          16.290       17.014      1053.48

2   18.6     6.510    5.208          15.624       17.856       899.47

3   22.4     4.032    5.824          21.056       21.280       827.34

4   12.0     4.200    3.360          10.920       10.680       878.41


   ins_losses abbrev
0      145.08     AL
1      133.93     AK
2      110.35     AZ
3      142.39     AR
4      165.63     CA
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   total           51 non-null     float64
 1   speeding        51 non-null     float64
 2   alcohol         51 non-null     float64
 3   not_distracted  51 non-null     float64
 4   no_previous     51 non-null     float64
 5   ins_premium     51 non-null     float64
 6   ins_losses      51 non-null     float64
 7   abbrev          51 non-null     object
dtypes: float64(7), object(1)
memory usage: 3.3+ KB
```

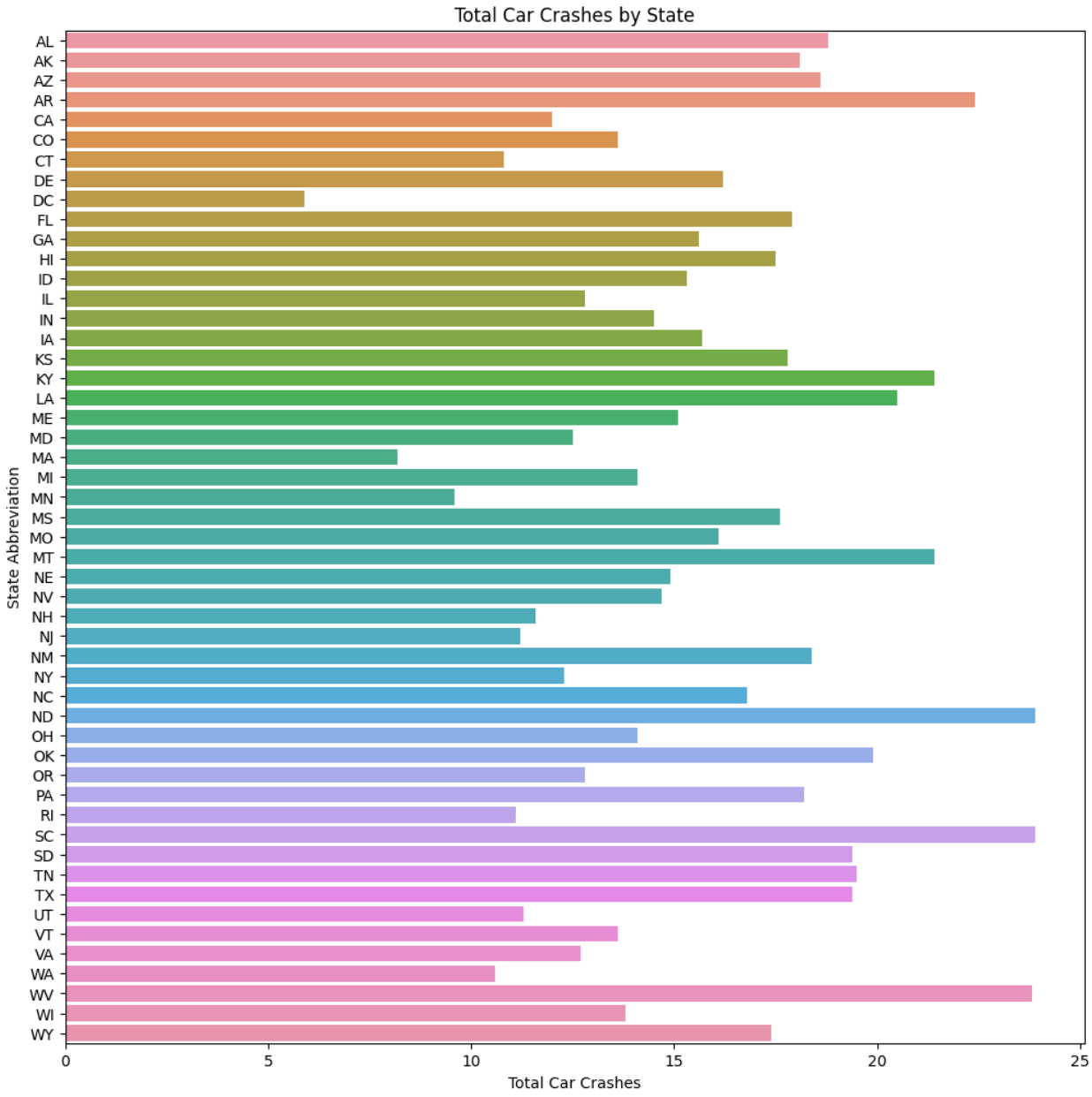
```
data.shape
```

```
(51, 8)
```

 # Inference: This bar plot provides a visual representation of the total number of car crashes for each U.S. state. States are listed on the y-axis (represented by their abbreviations), and the length of each bar corresponds to the total number of car crashes in that state in X-axis

 # This visualization allows you to quickly identify which states have the highest and lowest total car crash counts. It's evident that some states have significantly higher car crash rates compared to others.

```
# Create a bar plot showing the total car crashes by state
plt.figure(figsize=(12, 12))
sns.barplot(x="total", y="abbrev", data=data, orient="h")
plt.xlabel("Total Car Crashes")
plt.ylabel("State Abbreviation")
plt.title("Total Car Crashes by State")
plt.show()
```

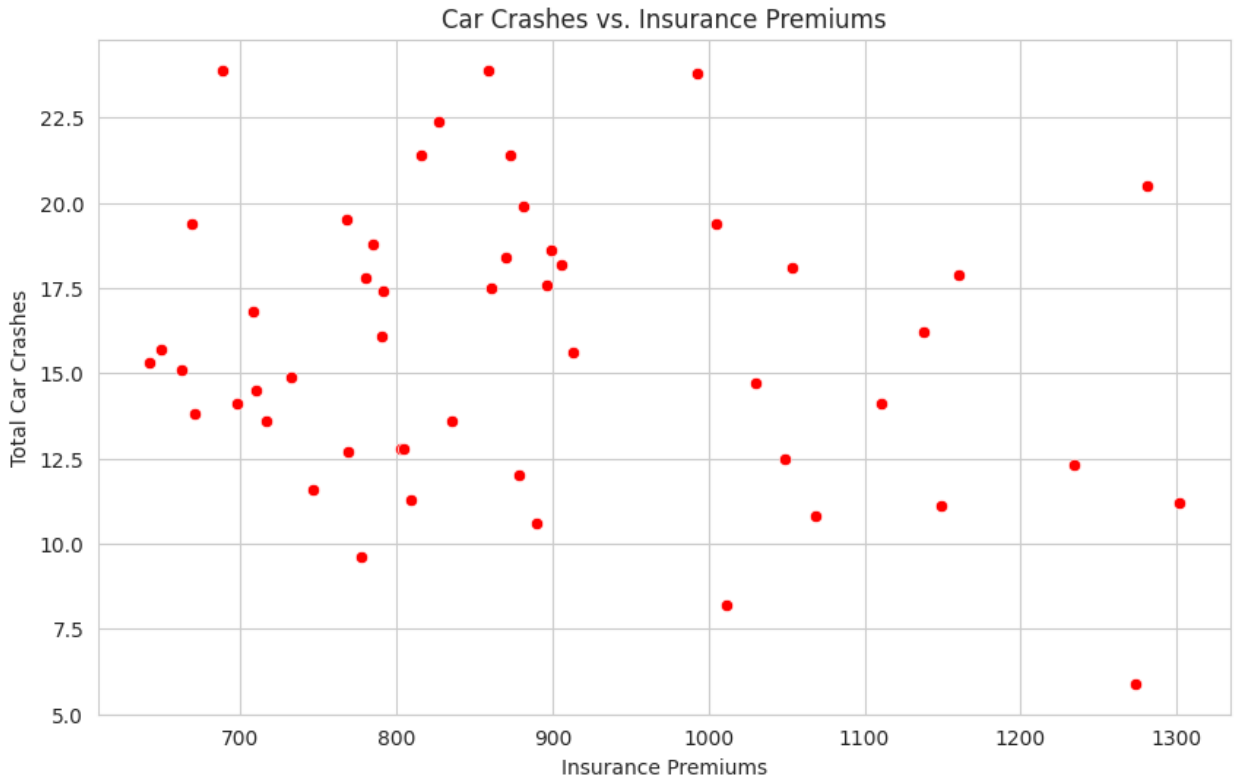


*** The scatter plot shows the relationship between the total number of car crashes and the average insurance premiums for each state. Each point on the plot represents a state, and its position indicates the relationship between car crashes and insurance premiums.***

*** By examining the scatter plot, you can assess whether there is any correlation between higher insurance premiums and lower car crash rates or vice versa. The plot can help identify states that deviate from the overall trend***

```
# Create a Scatter plot to visualize the distribution of total and
ins_premium.
plt.figure(figsize=(10, 6))
sns.scatterplot(x="ins_premium", y="total",color="red", data=data)
```

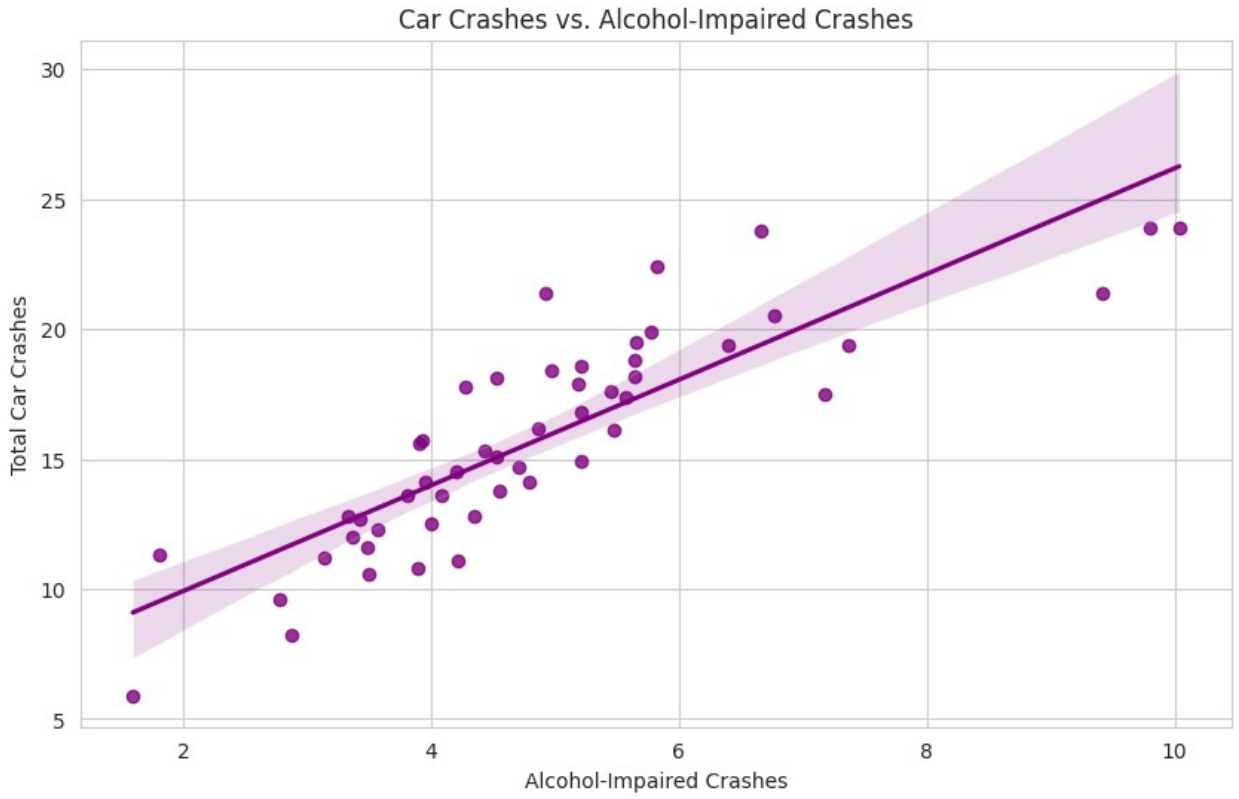
```
plt.xlabel("Insurance Premiums")
plt.ylabel("Total Car Crashes")
plt.title("Car Crashes vs. Insurance Premiums")
plt.show()
```



 # Inference: This regression plot illustrates the potential linear relationship between the total number of car crashes and the number of alcohol-impaired crashes in each state. The regression line represents the best-fit linear relationship between the two variables.

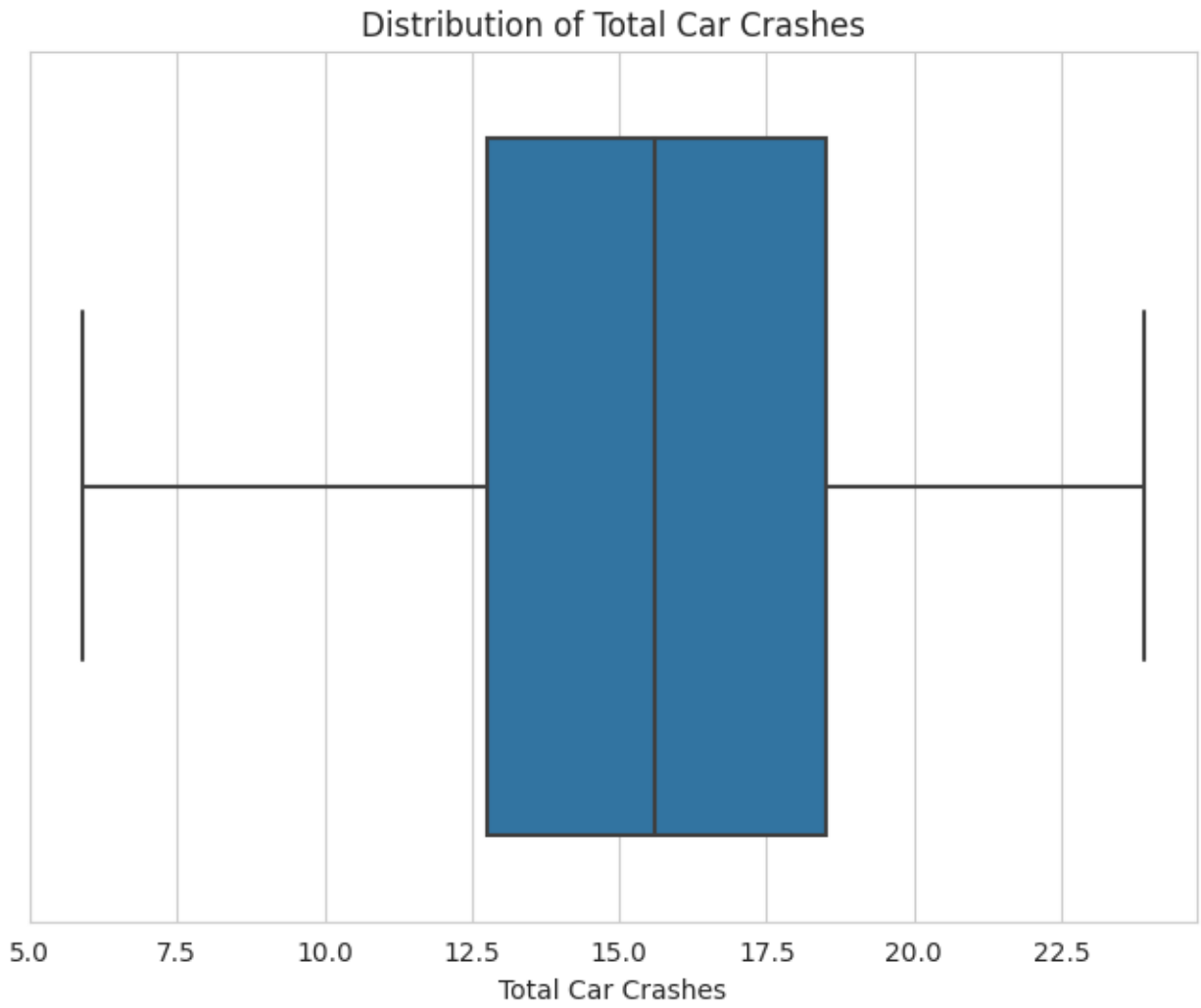
The slope and direction of the regression line can provide insights into whether an increase in alcohol-impaired crashes tends to correspond to an increase in total car crashes. If the line slopes upward, it suggests a positive correlation, while a downward slope indicates a negative correlation.

```
# Create a regression plot to visualize the distribution of total and alcohol.
plt.figure(figsize=(10, 6))
sns.regplot(x="alcohol", y="total",color="purple", data=data)
plt.xlabel("Alcohol-Impaired Crashes")
plt.ylabel("Total Car Crashes")
plt.title("Car Crashes vs. Alcohol-Impaired Crashes")
plt.show()
```



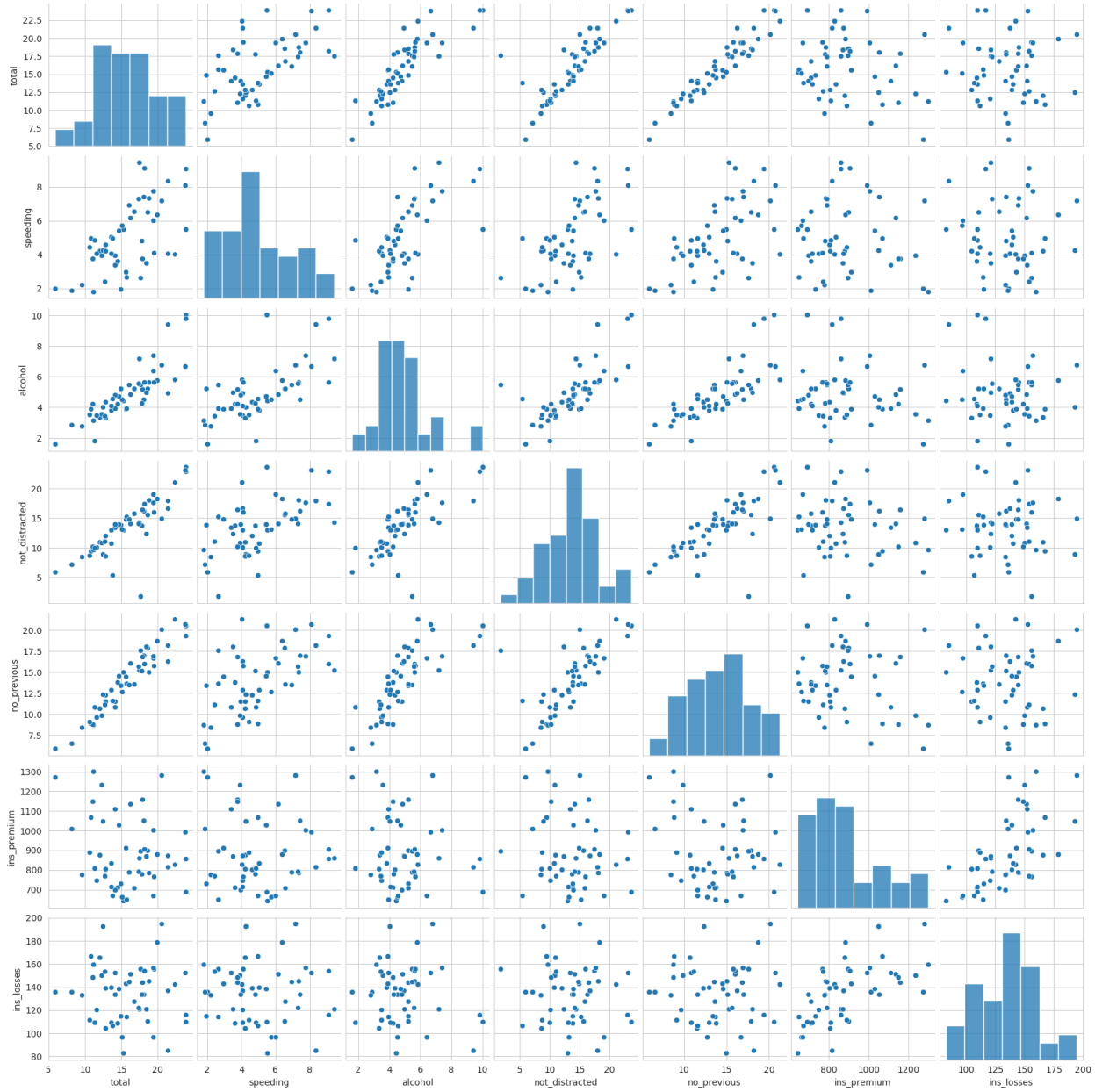
 # Inference: The box plot provides insights into the distribution of total car crashes. You can see the median, quartiles, and identify any potential outliers.

```
# Create a box plot to visualize the distribution of total car crashes
plt.figure(figsize=(8, 6))
sns.boxplot(x="total", data=data)
plt.xlabel("Total Car Crashes")
plt.title("Distribution of Total Car Crashes")
plt.show()
```



 # The pair plot allows you to visualize the pairwise relationships between numeric variables in the dataset. You can identify potential correlations and patterns among variables.

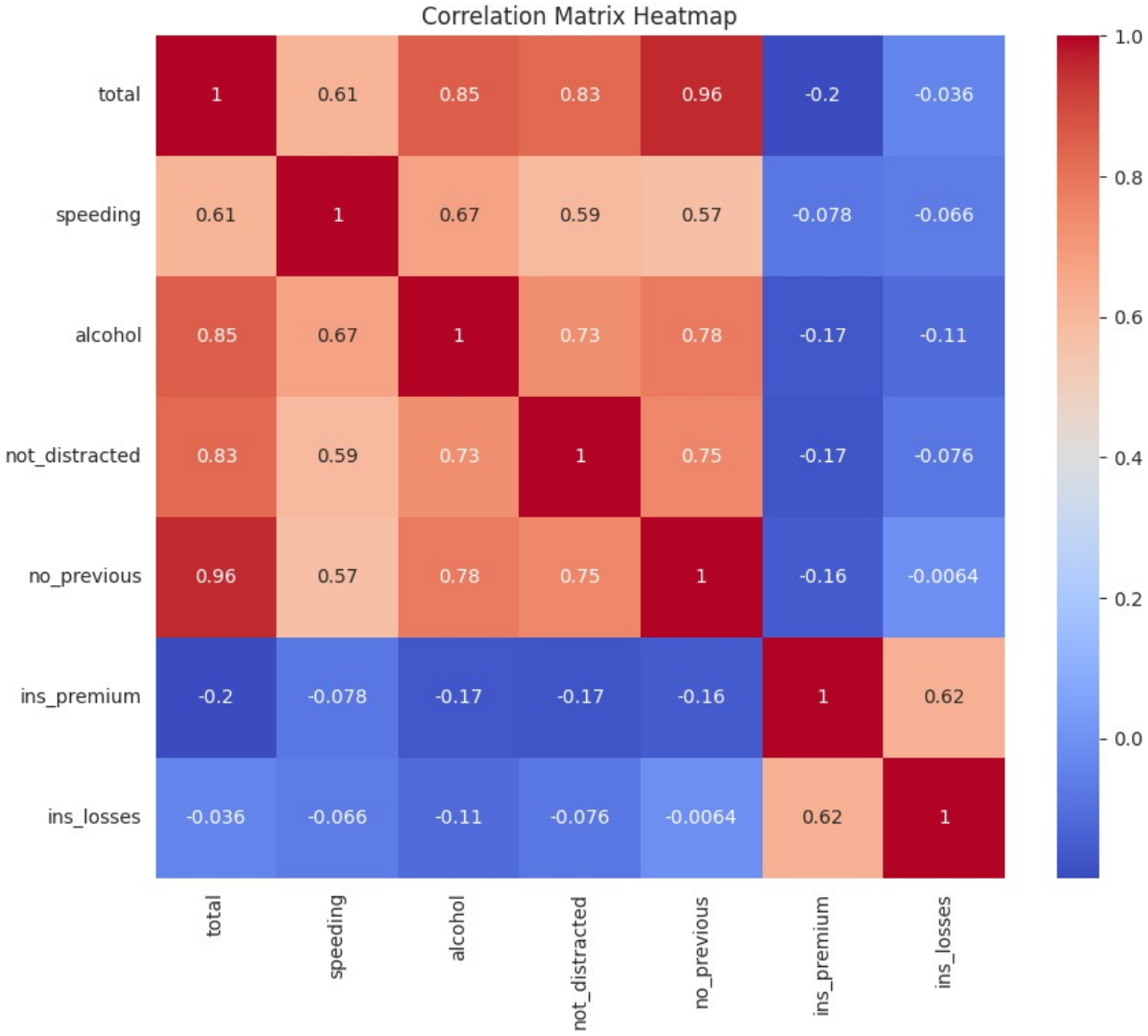
```
# Create a pair plot to visualize pairwise relationships between
numeric variables
sns.pairplot(data)
plt.show()
```



 # Inference: The heatmap of the correlation matrix helps you identify correlations between numeric variables in the dataset. Darker colors indicate stronger correlations, and you can infer which variables are positively or negatively correlated.

```
# Create a heatmap to visualize the correlation matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix Heatmap")
plt.show()
```

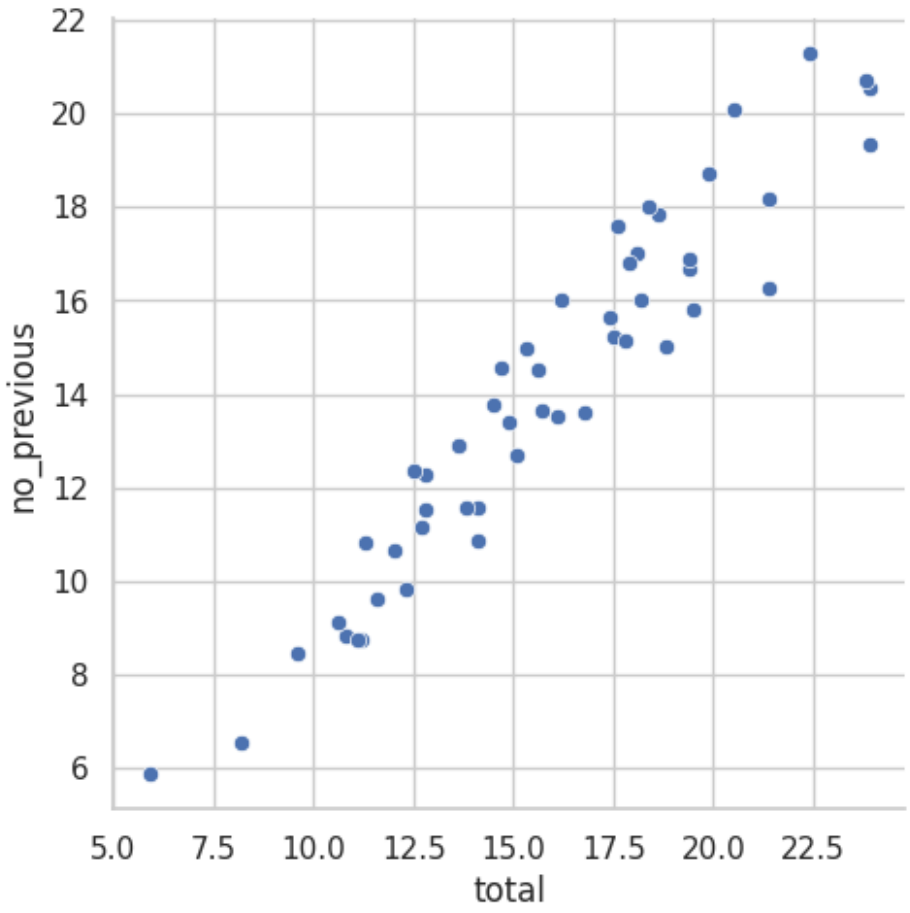
```
<ipython-input-31-1eff0baf0d77>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```




```
# here we can see that as even through have  no previous accident case
still total acccidents increasing whil no_orevios increasing
plt.figure(figsize=(15, 19))
sns.relplot(x="total",y="no_previous",data=data,palette="husl")
plt.show()
```

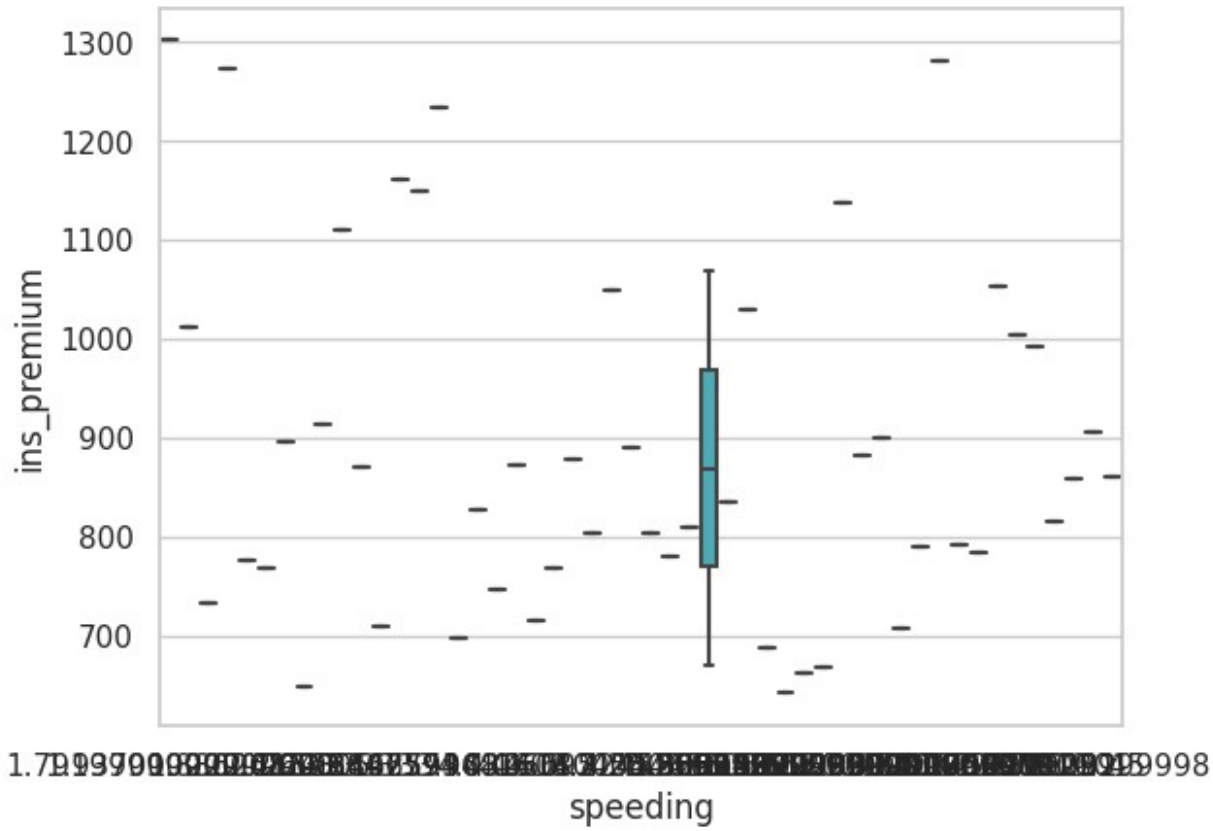
```
<ipython-input-63-9249dd364c6a>:2: UserWarning: Ignoring `palette`
because no `hue` variable has been assigned.
  sns.relplot(x="total",y="no_previous",data=data,palette="husl")
```

```
<Figure size 1500x1900 with 0 Axes>
```



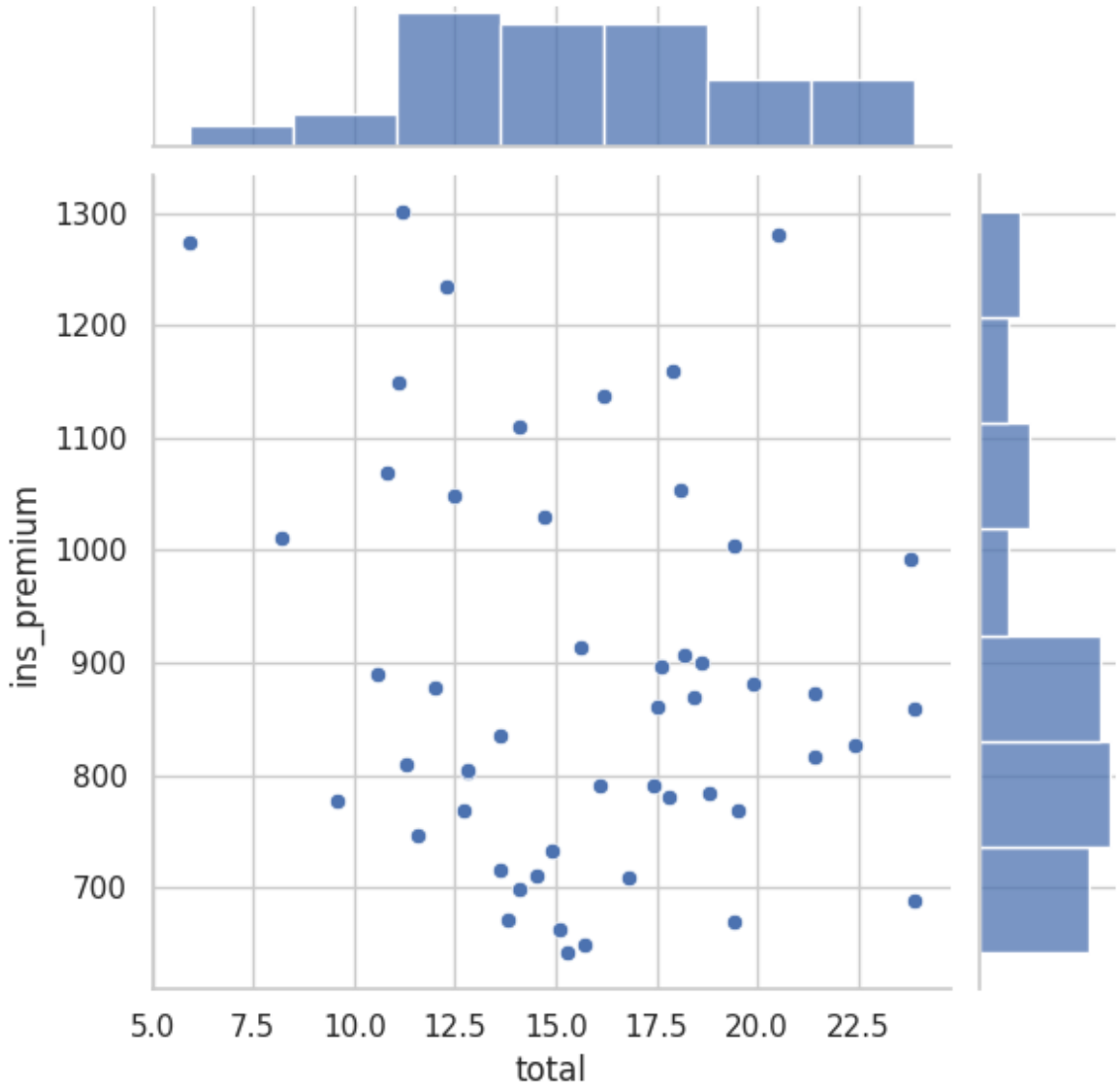

```
# here using this boxlpot we can anlyaze the data which is out of box
and to get high accurate by removing them
sns.boxplot(x="speeding",y="ins_premium",data=data) # here we consider
x as speeding and y as insurance policy
```

```
<Axes: xlabel='speeding', ylabel='ins_premium'>
```



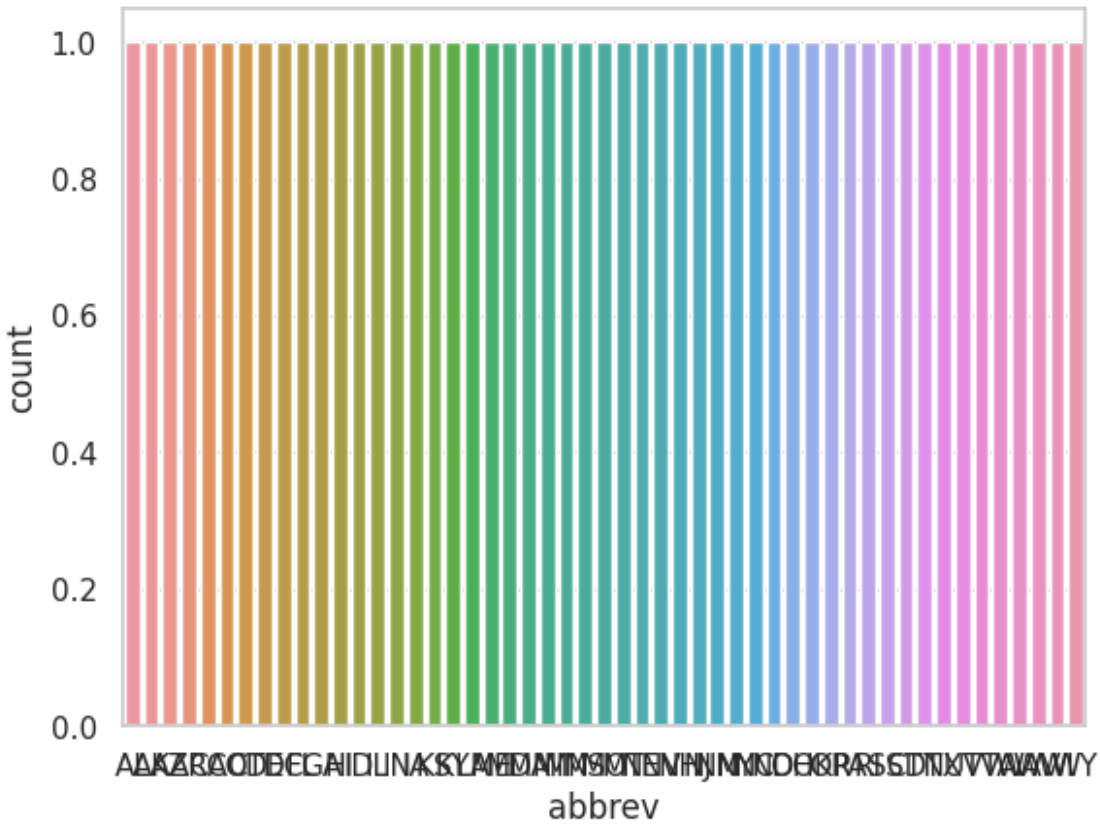

```
# where here from this graph we are able to see differnt graphs to
compare each graph  to get the resultant conclusion here we consider x
as total and y as insurance premium
sns.jointplot(x="total",y="ins_premium",data=data)
```

```
<seaborn.axisgrid.JointGrid at 0x7c6ba8fd5f60>
```




```
sns.countplot(x="abbrev",data=data)
```

```
<Axes: xlabel='abbrev', ylabel='count'>
```



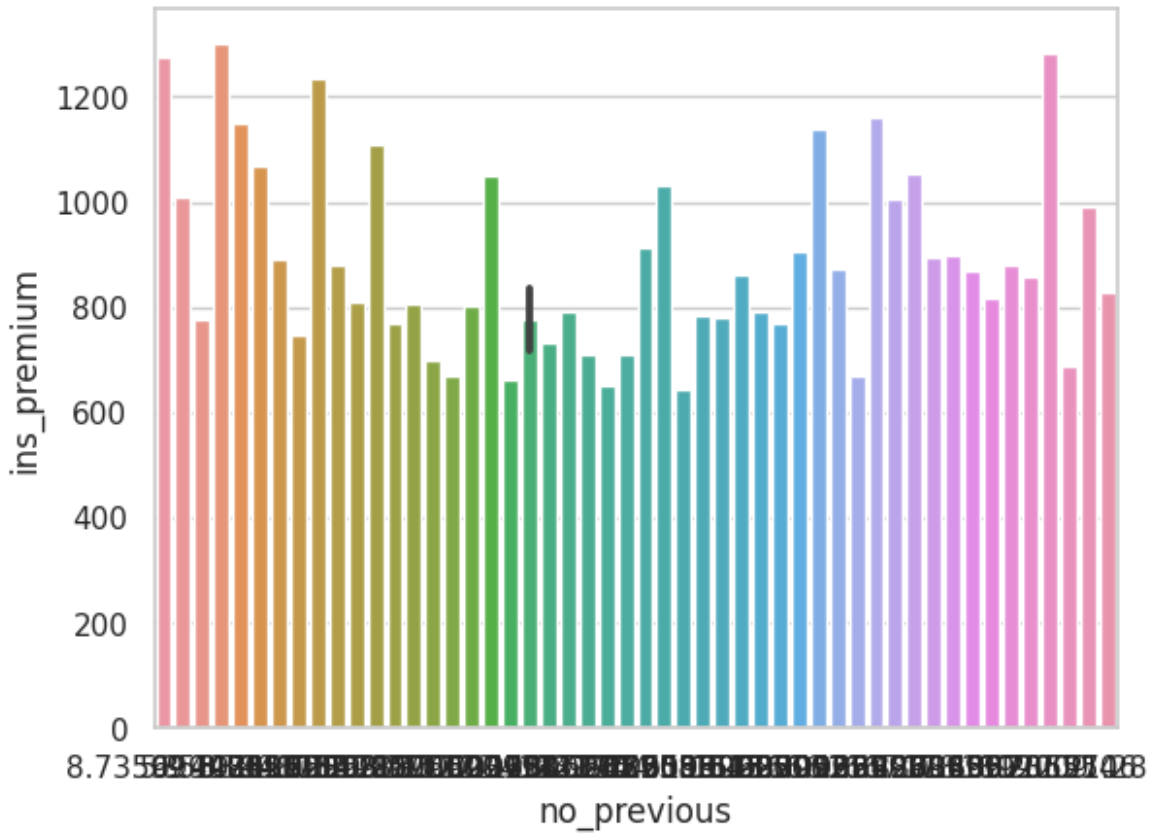

```
data["abbrev"].value_counts()
```

```
AL    1
PA    1
NV    1
NH    1
NJ    1
NM    1
NY    1
NC    1
ND    1
OH    1
OK    1
OR    1
RI    1
MT    1
SC    1
SD    1
TN    1
TX    1
UT    1
VT    1
VA    1
WA    1
```

```
WV    1
WI    1
NE    1
MO    1
AK    1
ID    1
AZ    1
AR    1
CA    1
CO    1
CT    1
DE    1
DC    1
FL    1
GA    1
HI    1
IL    1
MS    1
IN    1
IA    1
KS    1
KY    1
LA    1
ME    1
MD    1
MA    1
MI    1
MN    1
WY    1
Name: abbrev, dtype: int64
```

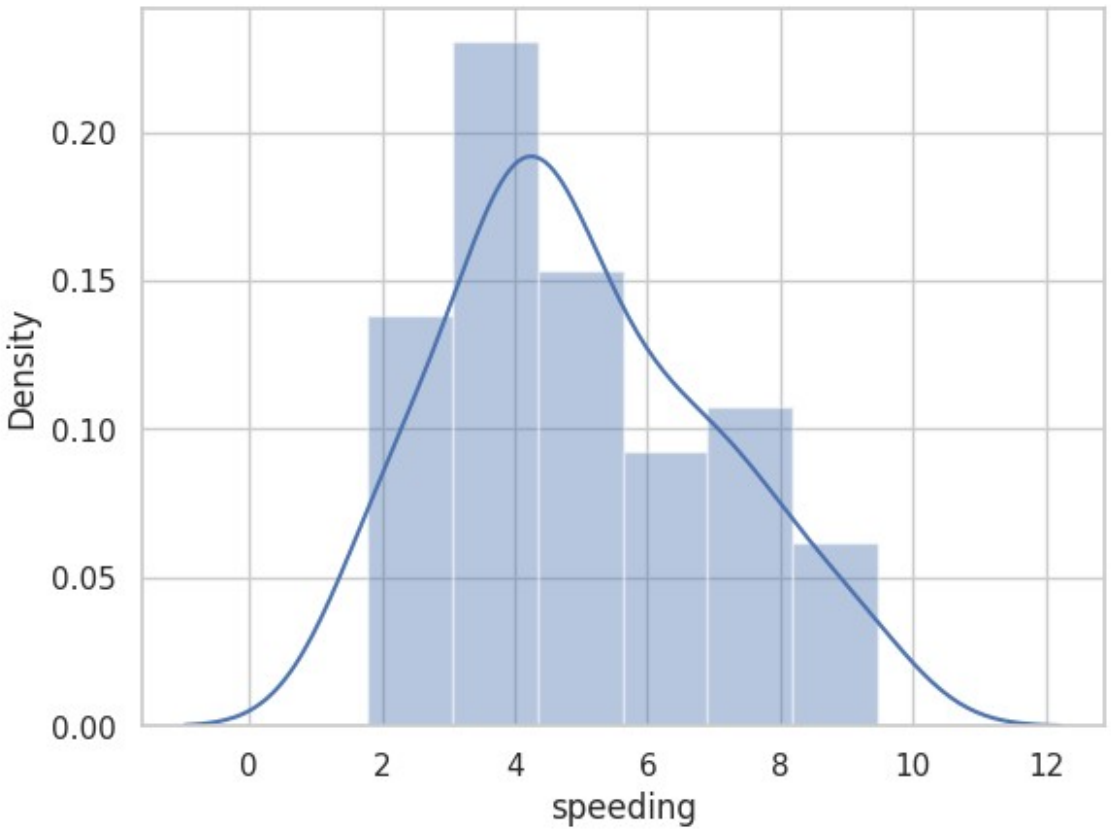
```
# here we are analysizing the people who have no_prevoius accidents
and who had simultaneoulsy have insurance policy at particular
locations
sns.barplot(x="no_previous",y="ins_premium",data=data)
```

```
<Axes: xlabel='no_previous', ylabel='ins_premium'>
```



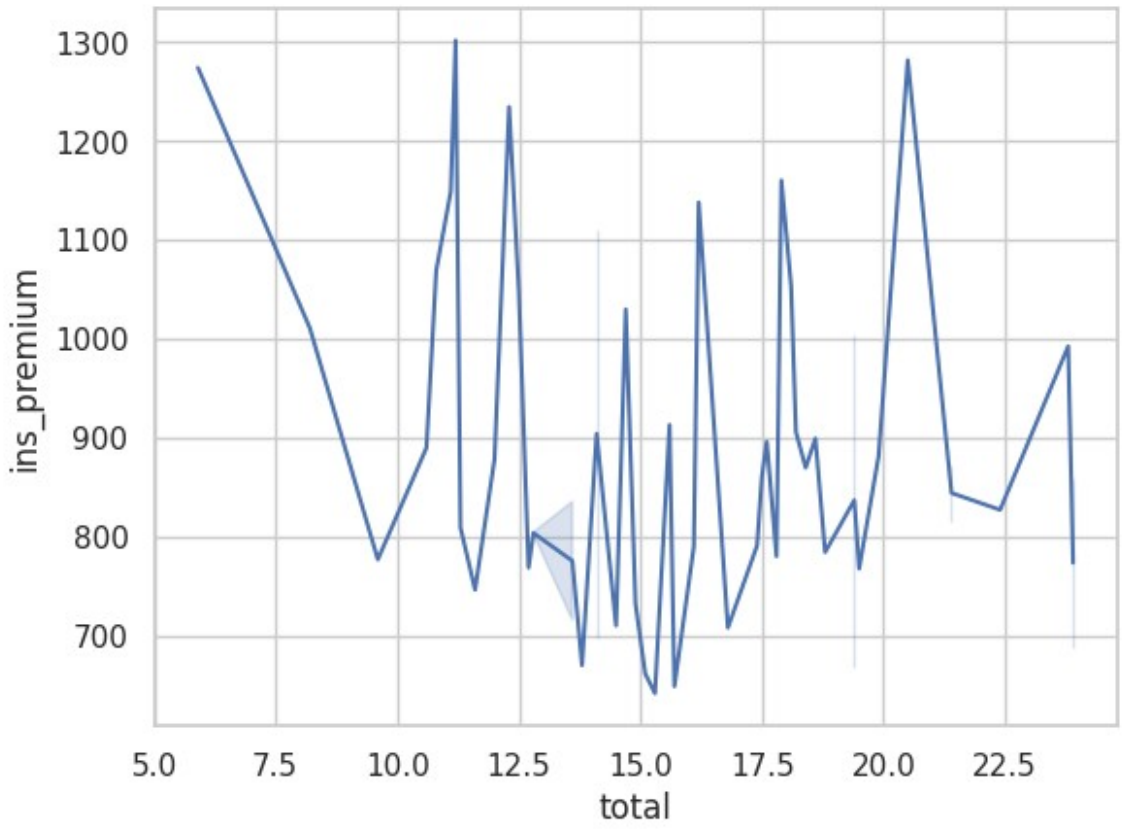

```
# here we are analyzing the graph at particulr where speed is high
through distplot so we can see where it has high and low
sns.distplot(data["speeding"])
```

```
<ipython-input-100-51334965cf24>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data["speeding"])

<Axes: xlabel='speeding', ylabel='Density'>
```




```
 # the  semi line which show between the line is the confidence
interval of that particular mean value.
 # here we consider that x as total y as ins_premium where between
12.5 and 15 we observe that confidence interval where most of the
people have insurance premium.
sns.lineplot(y="ins_premium",x="total",data=data)
```

```
<Axes: xlabel='total', ylabel='ins_premium'>
```


1300
1200
1100
1000
900
800
700
ins_premium
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
total