```
pip install seaborn   #library already installed
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: seaborn in c:\programdata\anaconda3\
lib\site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\programdata\
anaconda3\lib\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\programdata\
anaconda3\lib\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\
programdata\anaconda3\lib\site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1-
>seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
import matplotlib.pyplot as plt #Import the matplotlib.pyplot module
and alias it as plt
```

```
import seaborn as sns # Import the Seaborn library and alias it as
'sns'
```

```
print(sns.get_dataset_names())  #car_crashes is the inbuilt dataset 
which we will import
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 
'diamonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 
'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 
'seaice', 'taxis', 'tips', 'titanic']
```

```
df=sns.load_dataset('car_crashes') # Load the 'car_crashes' dataset 
from Seaborn and assign it to the variable 'df'

df   # 'df' contains the 'car_crashes' dataset, which provides 
information about car crash statistics.
```

```
    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

5    13.6     5.032    3.808          10.744       12.920       835.50

6    10.8     4.968    3.888           9.396        8.856      1068.73

7    16.2     6.156    4.860          14.094       16.038      1137.87

8     5.9     2.006    1.593           5.900        5.900      1273.89

9    17.9     3.759    5.191          16.468       16.826      1160.13

10   15.6     2.964    3.900          14.820       14.508       913.15

11   17.5     9.450    7.175          14.350       15.225       861.18

12   15.3     5.508    4.437          13.005       14.994       641.96

13   12.8     4.608    4.352          12.032       12.288       803.11

14   14.5     3.625    4.205          13.775       13.775       710.46

15   15.7     2.669    3.925          15.229       13.659       649.06

16   17.8     4.806    4.272          13.706       15.130       780.45

17   21.4     4.066    4.922          16.692       16.264       872.51
```

```
18   20.5     7.175    6.765          14.965       20.090      1281.55
19   15.1     5.738    4.530          13.137       12.684       661.88
20   12.5     4.250    4.000           8.875       12.375      1048.78
21    8.2     1.886    2.870           7.134        6.560      1011.14
22   14.1     3.384    3.948          13.395       10.857      1110.61
23    9.6     2.208    2.784           8.448        8.448       777.18
24   17.6     2.640    5.456           1.760       17.600       896.07
25   16.1     6.923    5.474          14.812       13.524       790.32
26   21.4     8.346    9.416          17.976       18.190       816.21
27   14.9     1.937    5.215          13.857       13.410       732.28
28   14.7     5.439    4.704          13.965       14.553      1029.87
29   11.6     4.060    3.480          10.092        9.628       746.54
30   11.2     1.792    3.136           9.632        8.736      1301.52
31   18.4     3.496    4.968          12.328       18.032       869.85
32   12.3     3.936    3.567          10.824        9.840      1234.31
33   16.8     6.552    5.208          15.792       13.608       708.24
34   23.9     5.497   10.038          23.661       20.554       688.75
35   14.1     3.948    4.794          13.959       11.562       697.73
36   19.9     6.368    5.771          18.308       18.706       881.51
37   12.8     4.224    3.328           8.576       11.520       804.71
38   18.2     9.100    5.642          17.472       16.016       905.99
39   11.1     3.774    4.218          10.212        8.769      1148.99
40   23.9     9.082    9.799          22.944       19.359       858.97
41   19.4     6.014    6.402          19.012       16.684       669.31
42   19.5     4.095    5.655          15.990       15.795       767.91
43   19.4     7.760    7.372          17.654       16.878      1004.75
```

```
44   11.3      4.859    1.808              9.944         10.848         809.38

45   13.6      4.080    4.080             13.056         12.920         716.20

46   12.7      2.413    3.429             11.049         11.176         768.95

47   10.6      4.452    3.498              8.692          9.116         890.03

48   23.8      8.092    6.664             23.086         20.706         992.61

49   13.8      4.968    4.554              5.382         11.592         670.31

50   17.4      7.308    5.568             14.094         15.660         791.14


    ins_losses abbrev
0       145.08     AL
1       133.93     AK
2       110.35     AZ
3       142.39     AR
4       165.63     CA
5       139.91     CO
6       167.02     CT
7       151.48     DE
8       136.05     DC
9       144.18     FL
10      142.80     GA
11      120.92     HI
12       82.75     ID
13      139.15     IL
14      108.92     IN
15      114.47     IA
16      133.80     KS
17      137.13     KY
18      194.78     LA
19       96.57     ME
20      192.70     MD
21      135.63     MA
22      152.26     MI
23      133.35     MN
24      155.77     MS
25      144.45     MO
26       85.15     MT
27      114.82     NE
28      138.71     NV
29      120.21     NH
30      159.85     NJ
31      120.75     NM
32      150.01     NY
33      127.82     NC
```

```
34      109.72     ND
35      133.52     OH
36      178.86     OK
37      104.61     OR
38      153.86     PA
39      148.58     RI
40      116.29     SC
41       96.87     SD
42      155.57     TN
43      156.83     TX
44      109.48     UT
45      109.61     VT
46      153.72     VA
47      111.62     WA
48      152.56     WV
49      106.62     WI
50      122.04     WY
```

```
df.info()  # Display information about the DataFrame 'df', including
data types, non-null counts, and memory usage.
```

```
<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
```

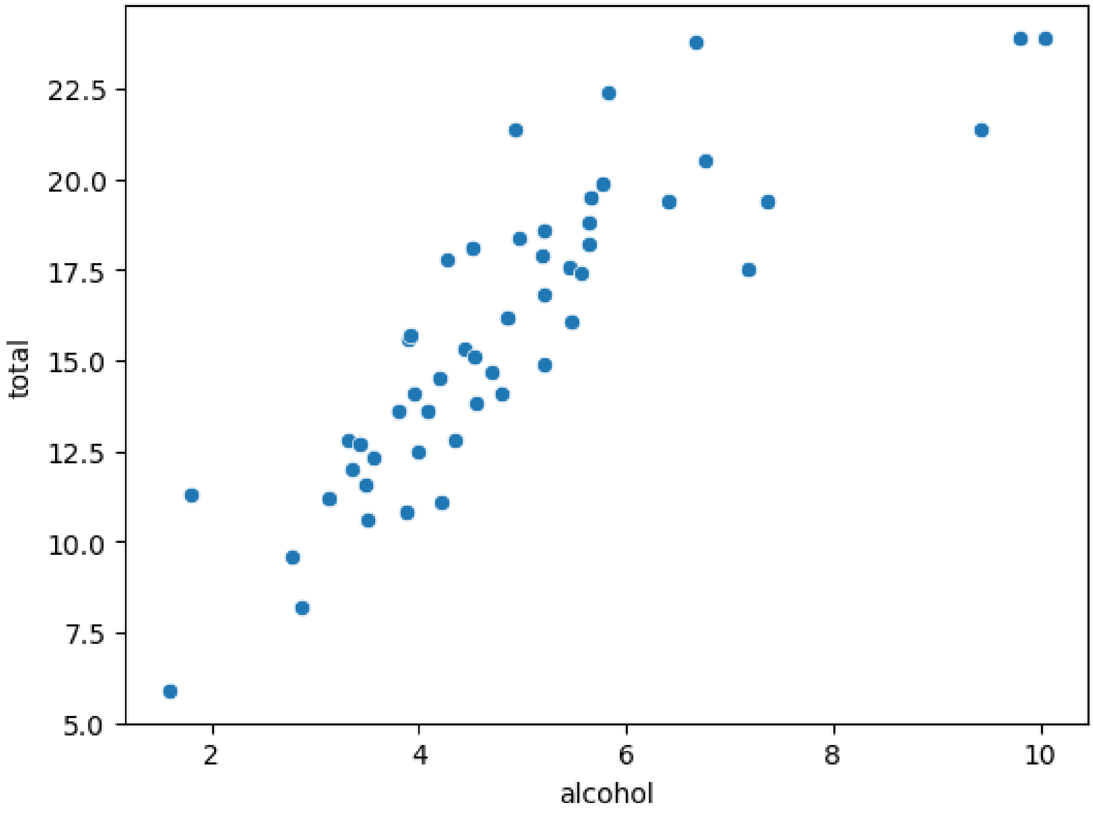
```
df.head(5)  # Display the first 5 rows of the DataFrame 'df' to
provide an overview of its data.
```

```
   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
```

```
sns.scatterplot(x="alcohol",y="total",data=df)  # Create a scatter plot to visualize the relationship between alcohol-related crashes and total crashes.
```

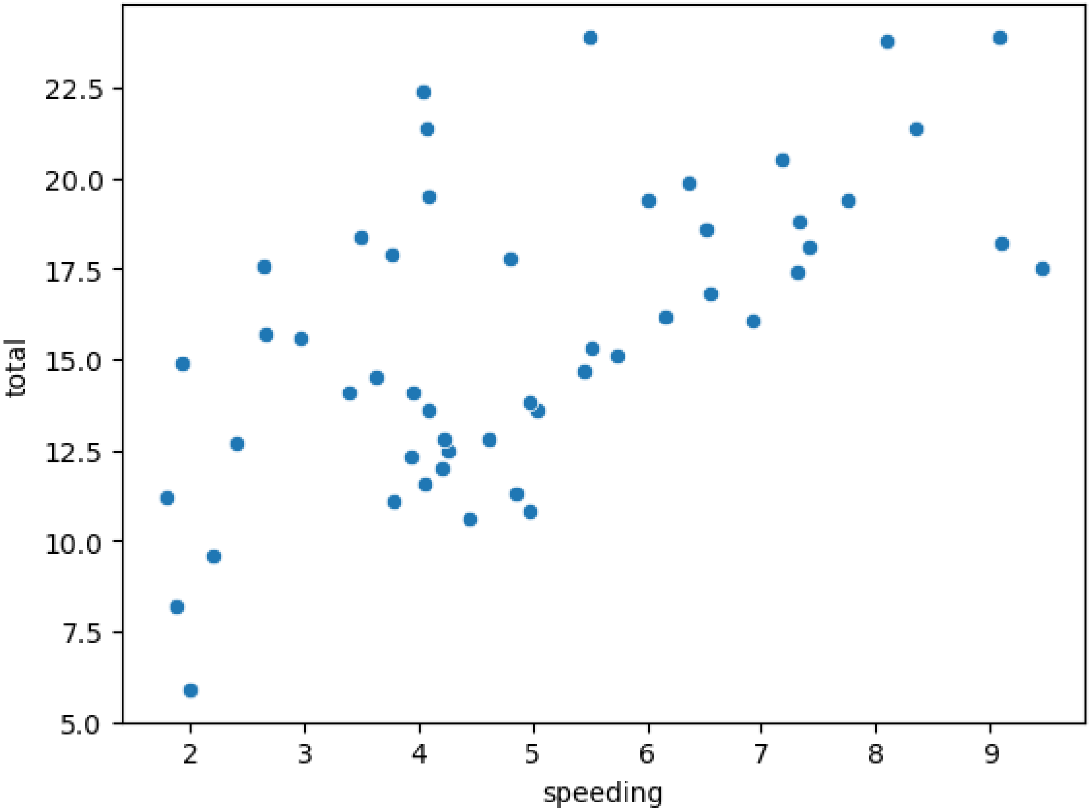
```
<Axes: xlabel='alcohol', ylabel='total'>
```



```
# As the number of alcohol-related crashes increases, the total number of crashes also tends to increase
```

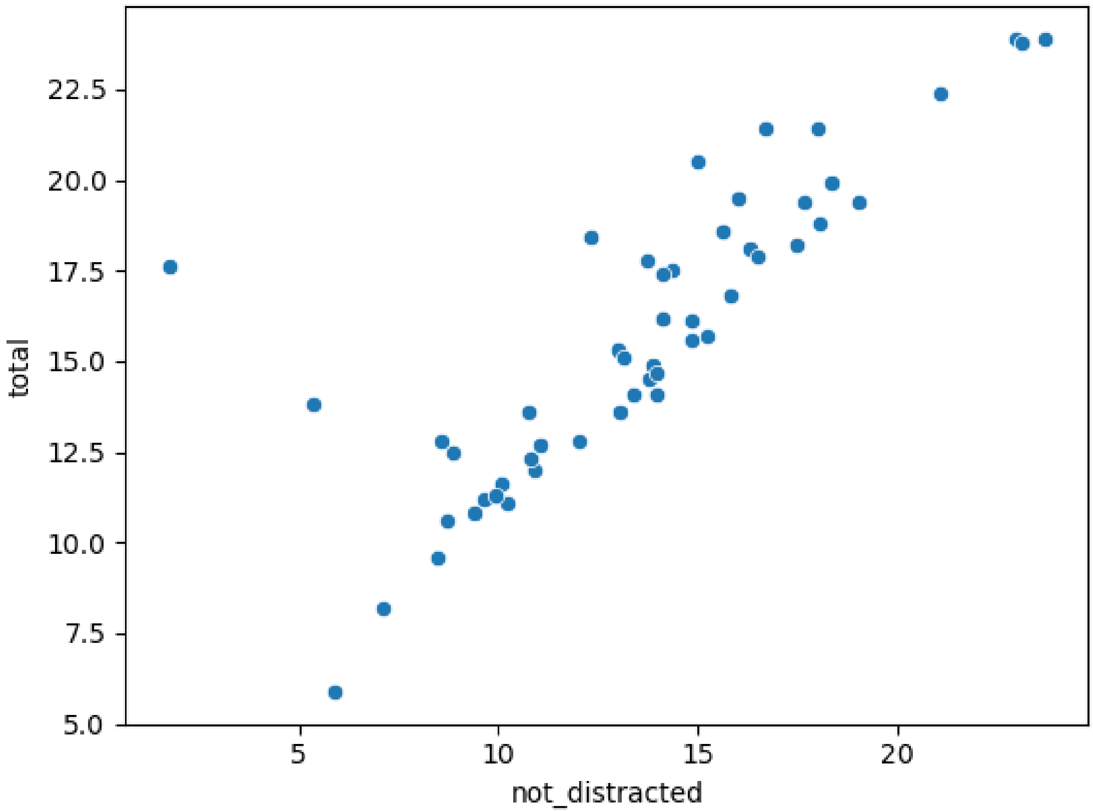
```
sns.scatterplot(x="speeding",y="total",data=df) #speeding vs total
```

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



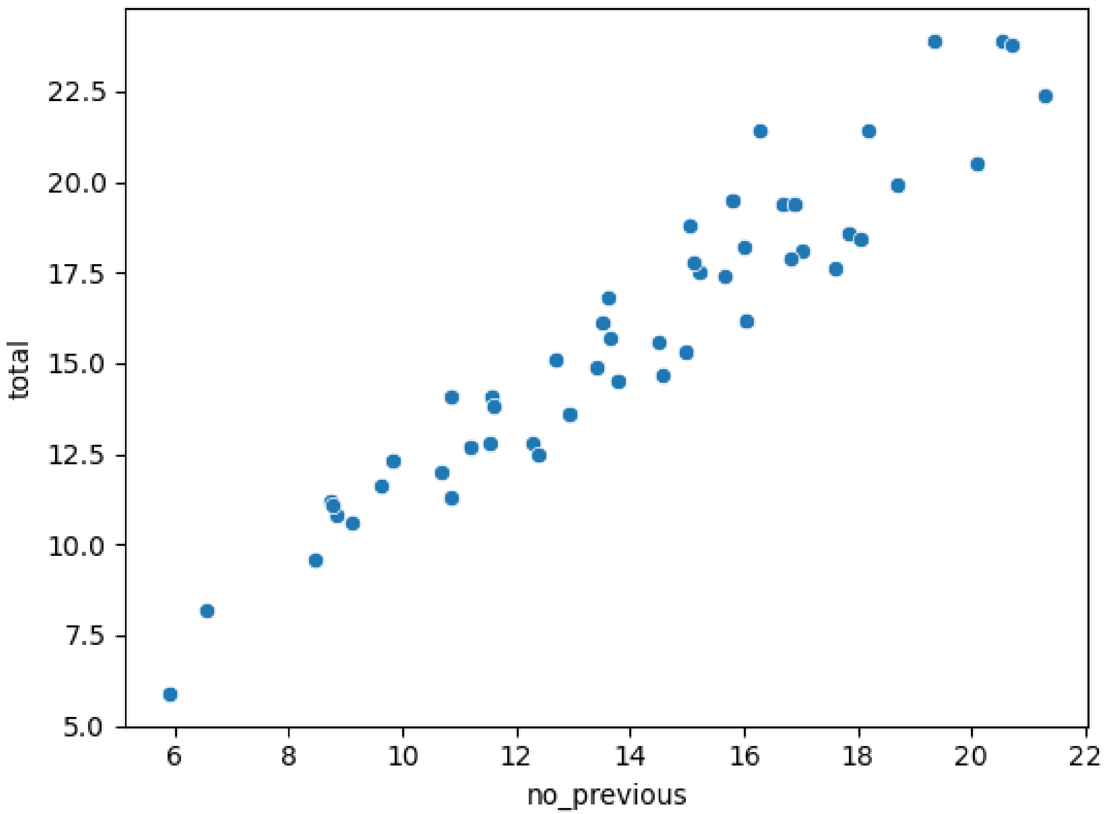
```
sns.scatterplot(x="not_distracted",y="total",data=df) #not_distracted
vs total
```

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

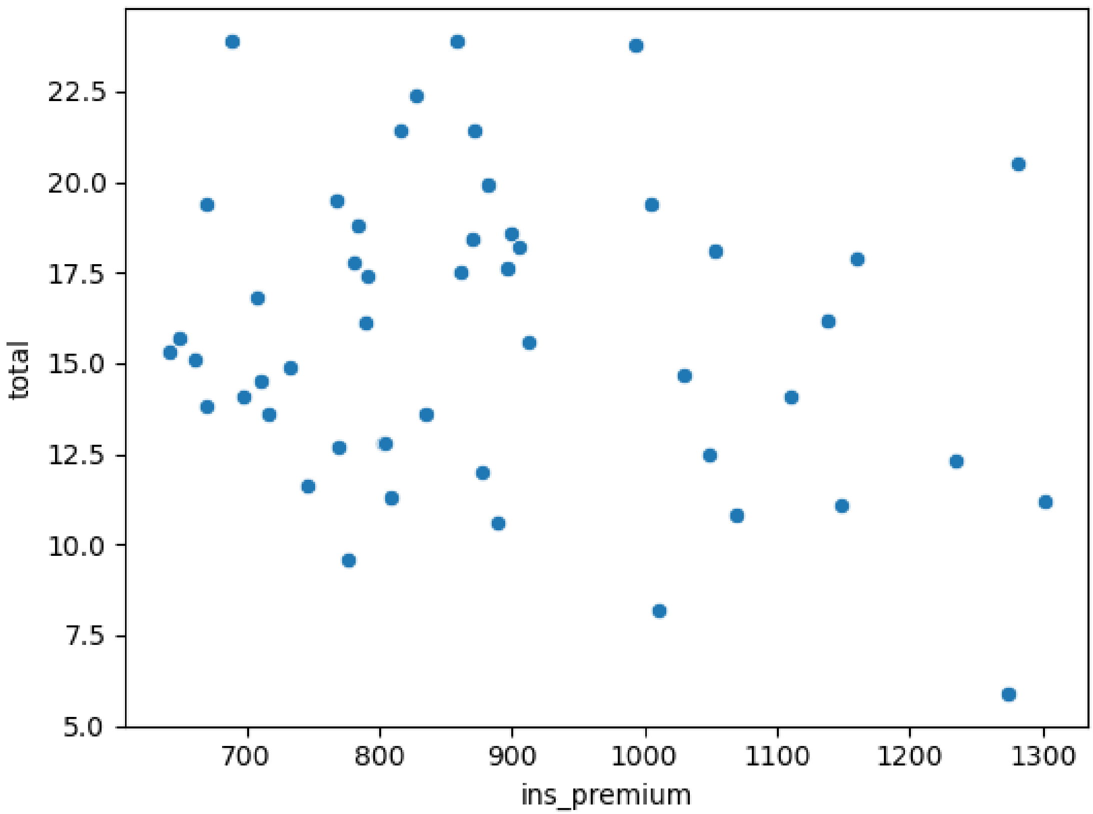


```
sns.scatterplot(x="no_previous",y="total",data=df) #no_previous vs total
```

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

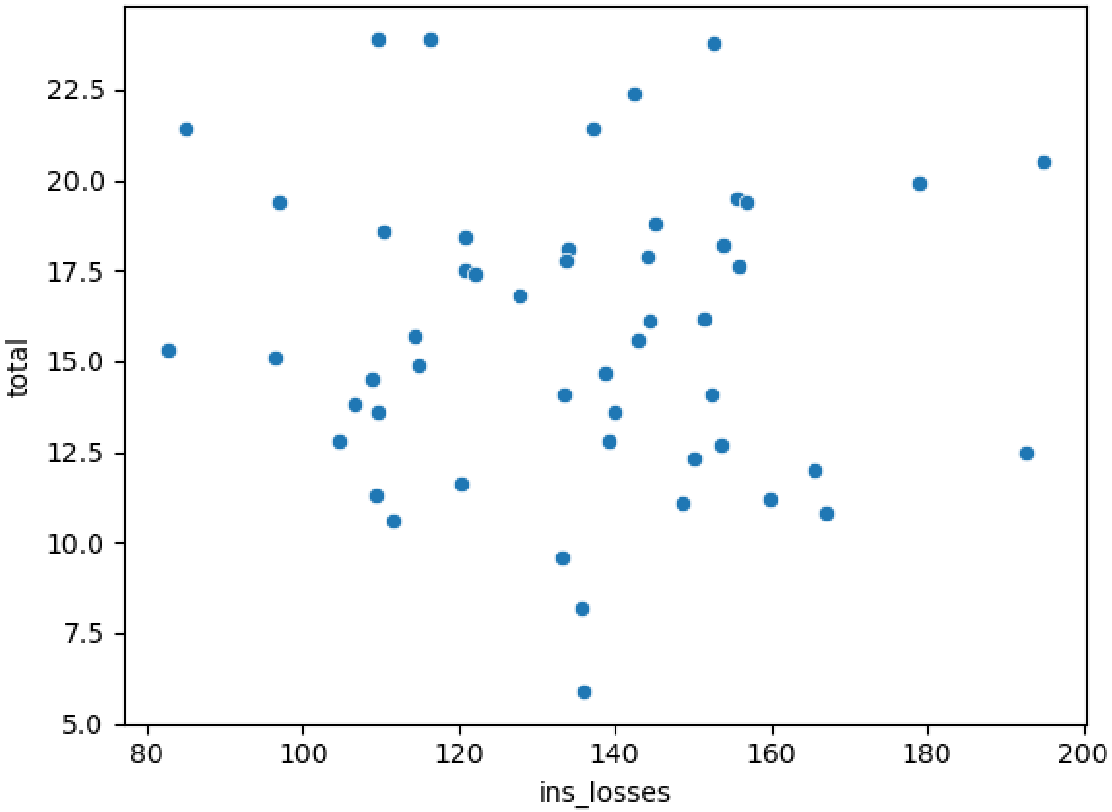


```
sns.scatterplot(x="ins_premium",y="total",data=df) #ins_premium vs
total

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



```
sns.scatterplot(x="ins_losses",y="total",data=df) #ins_losses vs total
```

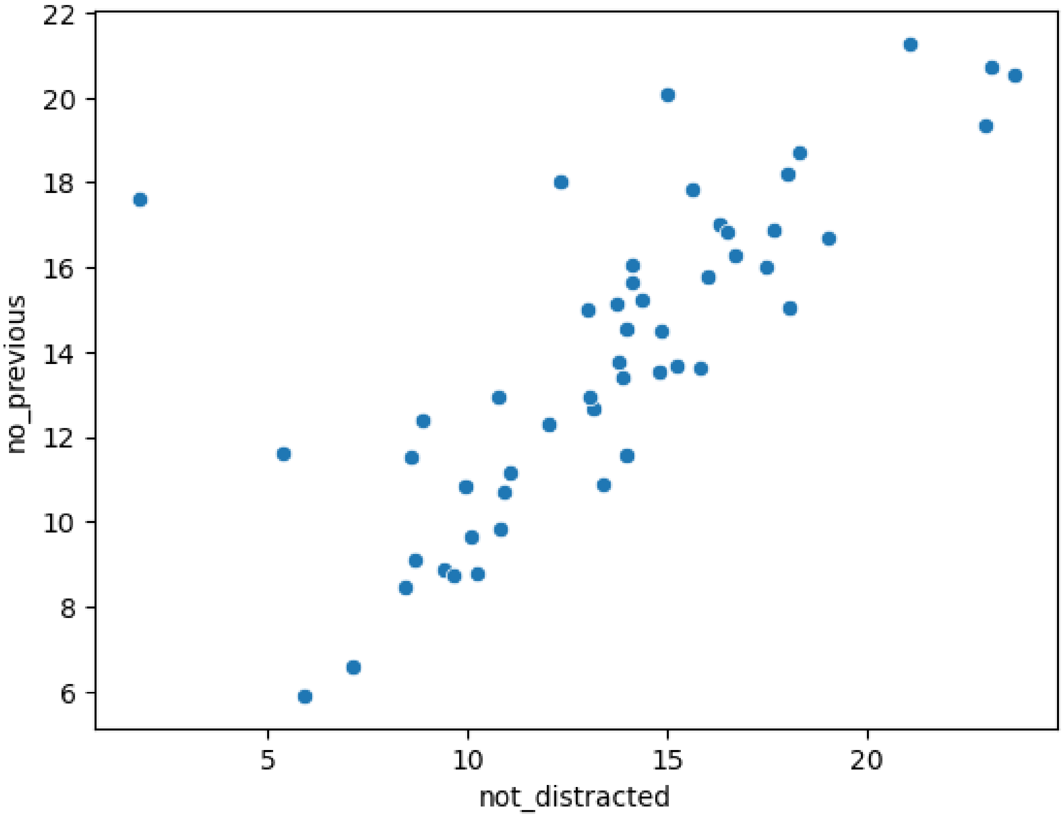
```
<Axes: xlabel='ins_losses', ylabel='total'>
```



#Lets take Alcohol vs Total

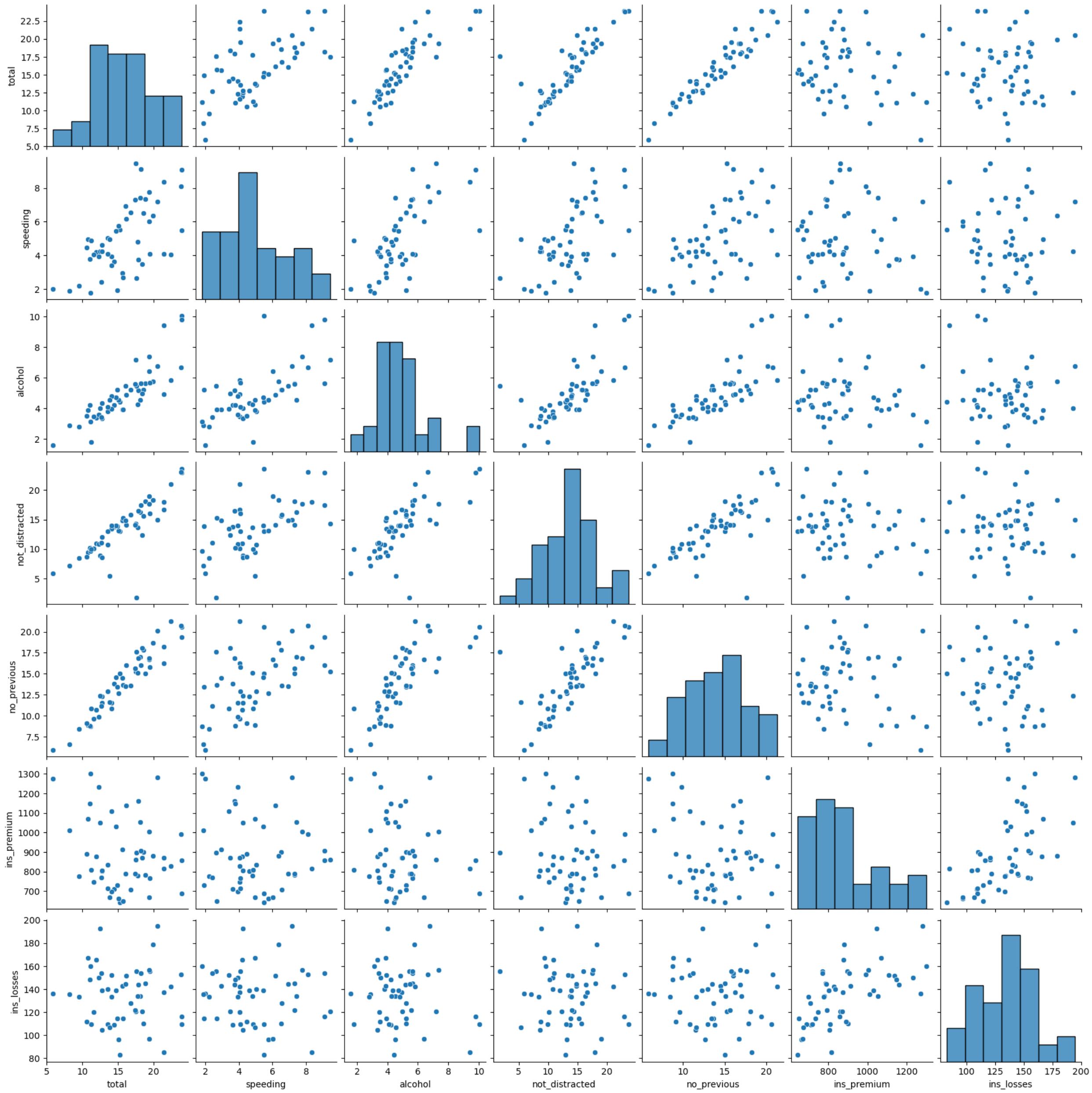
```
sns.scatterplot(x="not_distracted",y="no_previous",data=df)  # Create a scatter plot to visualize the relationship between alcohol-related crashes and total crashes.
```

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



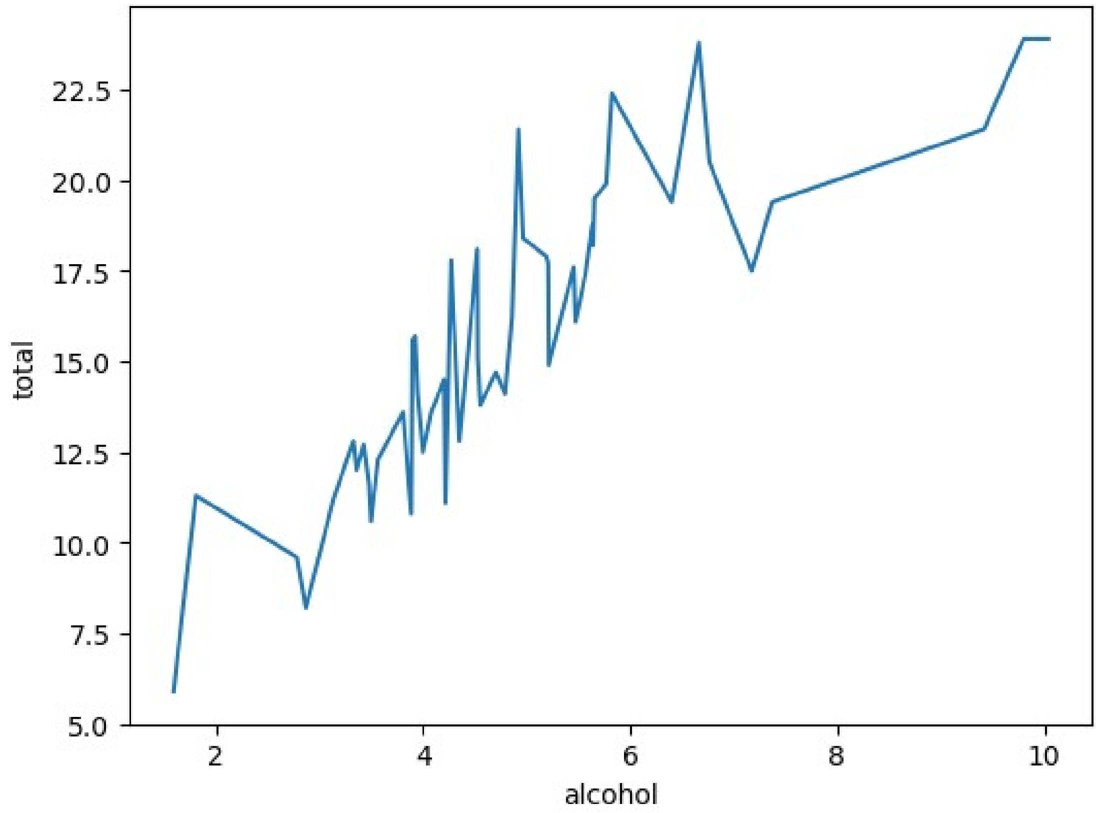
```
sns.pairplot(df)#all the scatter plots and barplots
```

```
<seaborn.axisgrid.PairGrid at 0x1662cb847d0>
```



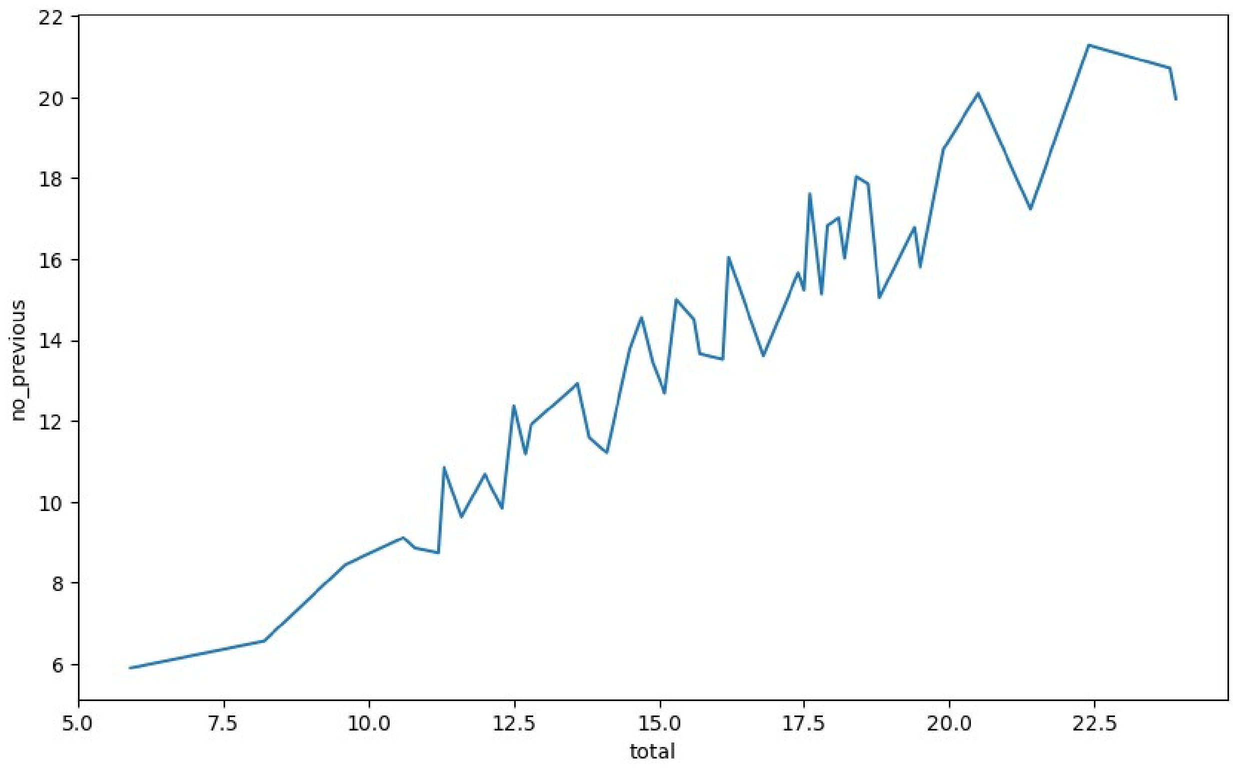
```
sns.lineplot(x="alcohol",y="total",data=df,errorbar=None)
```

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



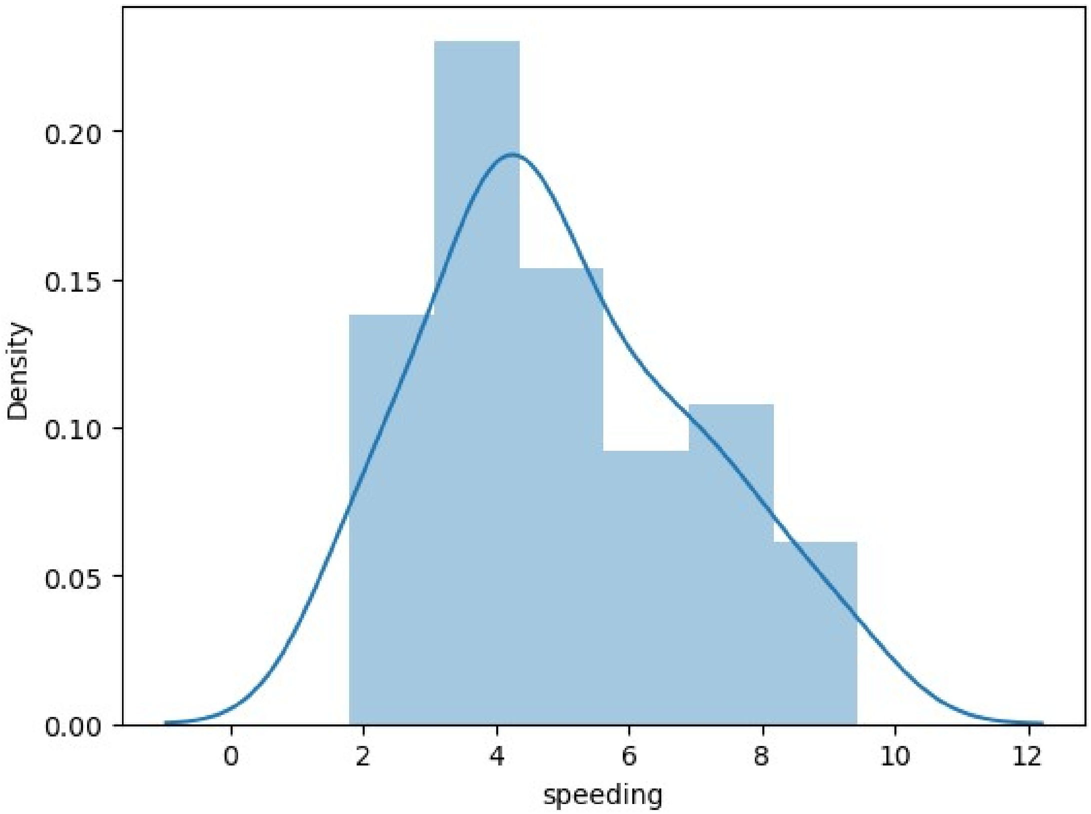
```
#This line plot provides an initial visual insight into the
relationship between alcohol consumption and car crashes
#alcohol increases the total car crashes we can say by the above plot

plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x='total', y='no_previous',errorbar=None)#error
bar is used since data contains a string type data
```

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



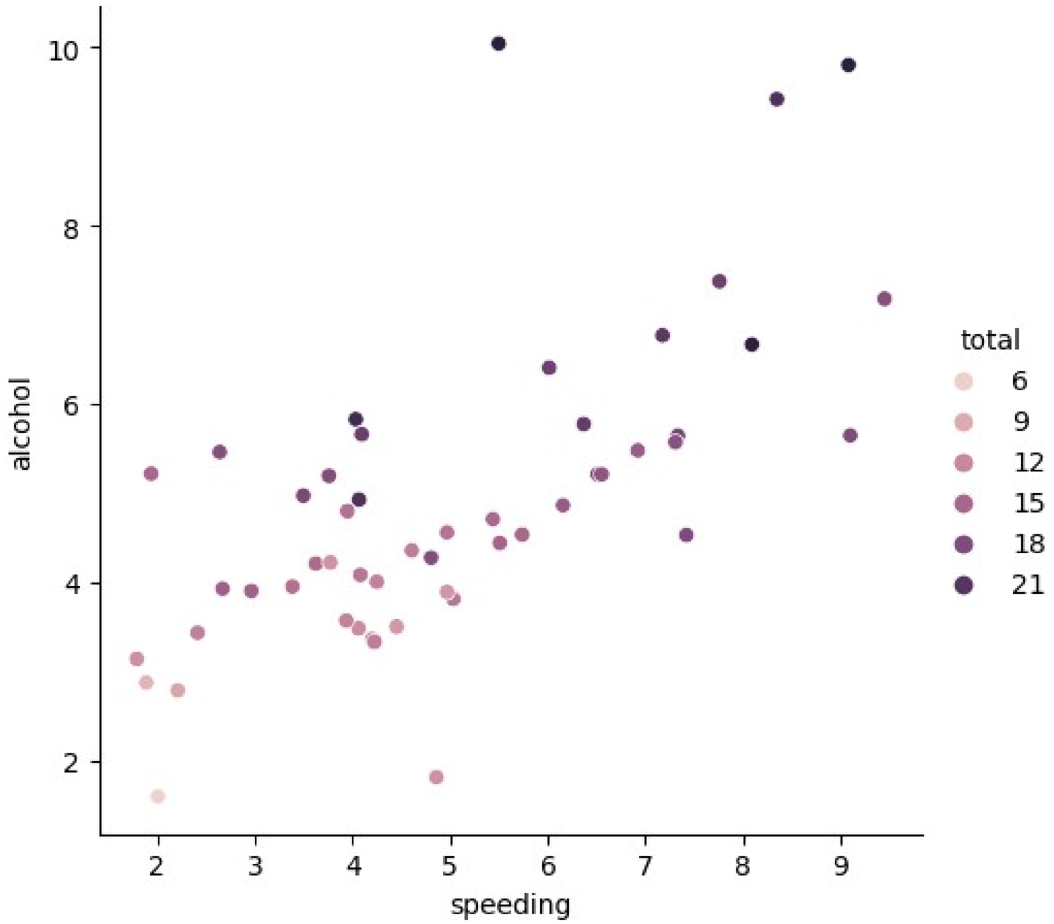
```
#The line plot suggests a relationship between the 'total' (possibly total car crashes or incidents) and 'no_previous' (possibly incidents where there were no previous accidents) variables
#as the increase in no_previous increases the total car crashes increases

sns.distplot(df["speeding"])
```

```
C:\Users\kaler\AppData\Local\Temp\ipykernel_24612\2127910581.py: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(df["speeding"])

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



```
#This plot helps you understand the distribution of the "speeding"
data, including the central tendency (e.g., where the data tends to
cluster) and any patterns, such as whether it follows a normal
distribution, is skewed, or has multiple modes
#The curve is bell shaped
# data represented in the histogram is roughly symmetrically
distributed around a central value, and the histogram takes on a shape
that resembles a bell curve

sns.relplot(x="speeding",y="alcohol",data=df,hue="total")
```

```
<seaborn.axisgrid.FacetGrid at 0x21ea479f950>
```



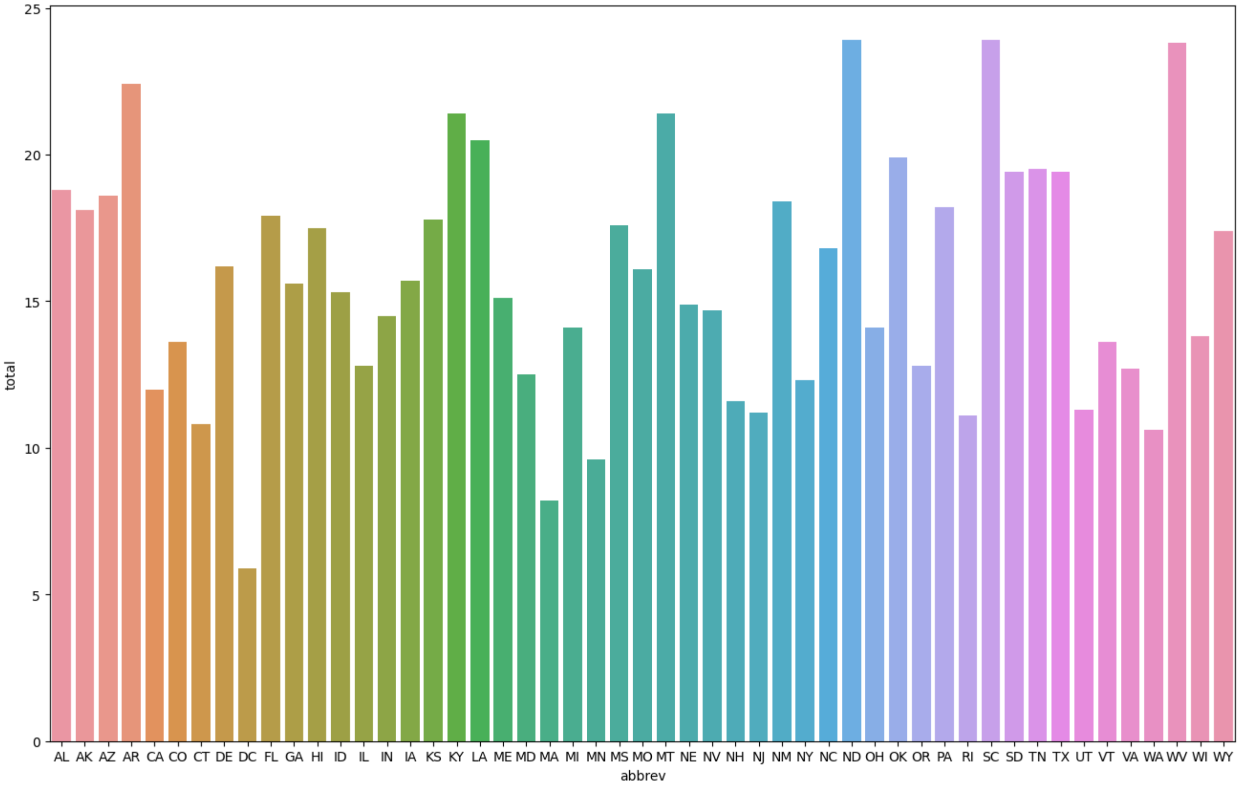
```
#The plot will display data points with "speeding" values on the x-
axis, "alcohol" values on the y-axis, and use different colors to
distinguish between different categories or values in the "total"
column.

df["no_previous"].value_counts()
```

```
12.920    2
15.040    1
16.016    1
14.553    1
9.628     1
8.736     1
18.032    1
9.840     1
13.608    1
20.554    1
11.562    1
18.706    1
11.520    1
8.769     1
```

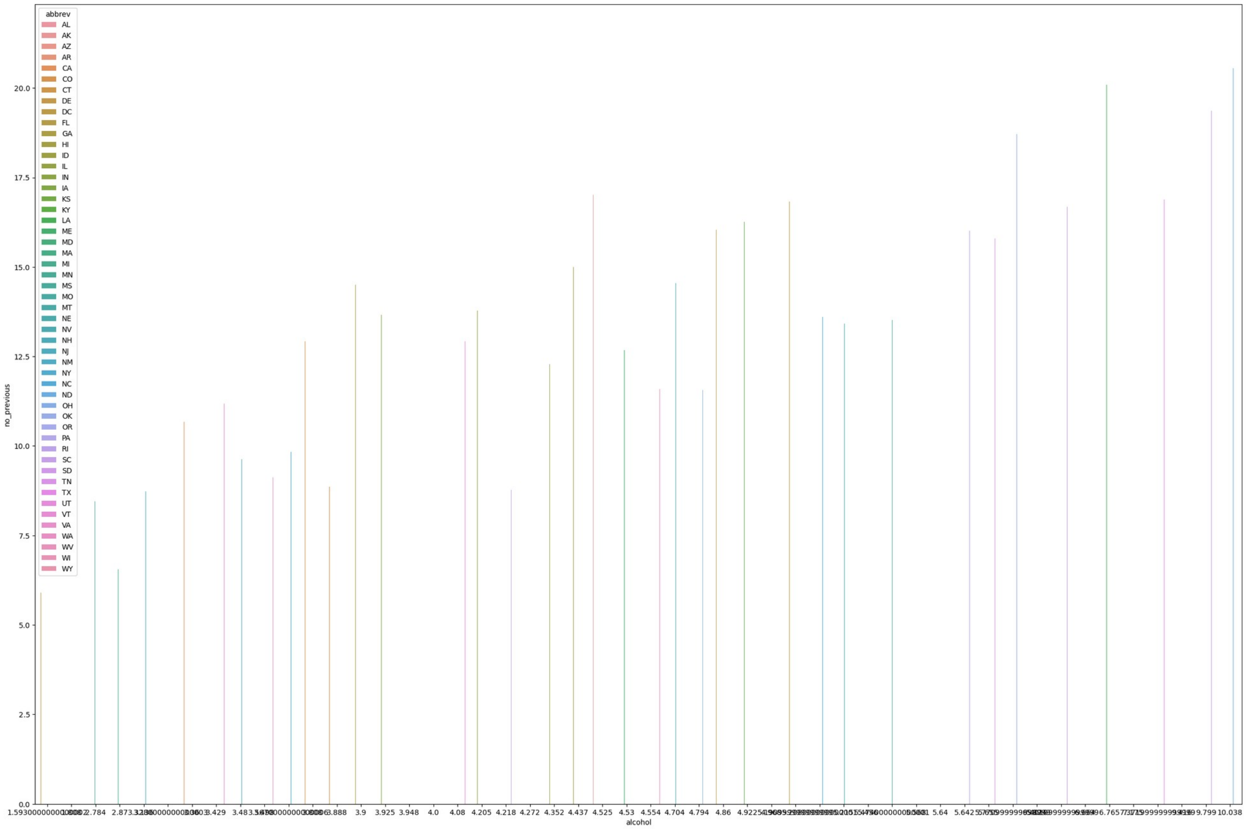
```
18.190    1
19.359    1
16.684    1
15.795    1
16.878    1
10.848    1
11.176    1
9.116     1
20.706    1
11.592    1
13.410    1
13.524    1
17.014    1
17.600    1
17.856    1
21.280    1
10.680    1
8.856     1
16.038    1
5.900     1
16.826    1
14.508    1
15.225    1
14.994    1
12.288    1
13.775    1
13.659    1
15.130    1
16.264    1
20.090    1
12.684    1
12.375    1
6.560     1
10.857    1
8.448     1
15.660    1
Name: no_previous, dtype: int64
```

```
#The value_counts() operation on the "no_previous" column provides a
count of each unique value in that column. The output will show how
many times each unique value appears in the column.
#it shows the count of total accidents counts done by
no_previous_accident done attribute

plt.subplots(figsize=(16,10))
sns.barplot(x='abbrev',y='total',data=df)
```

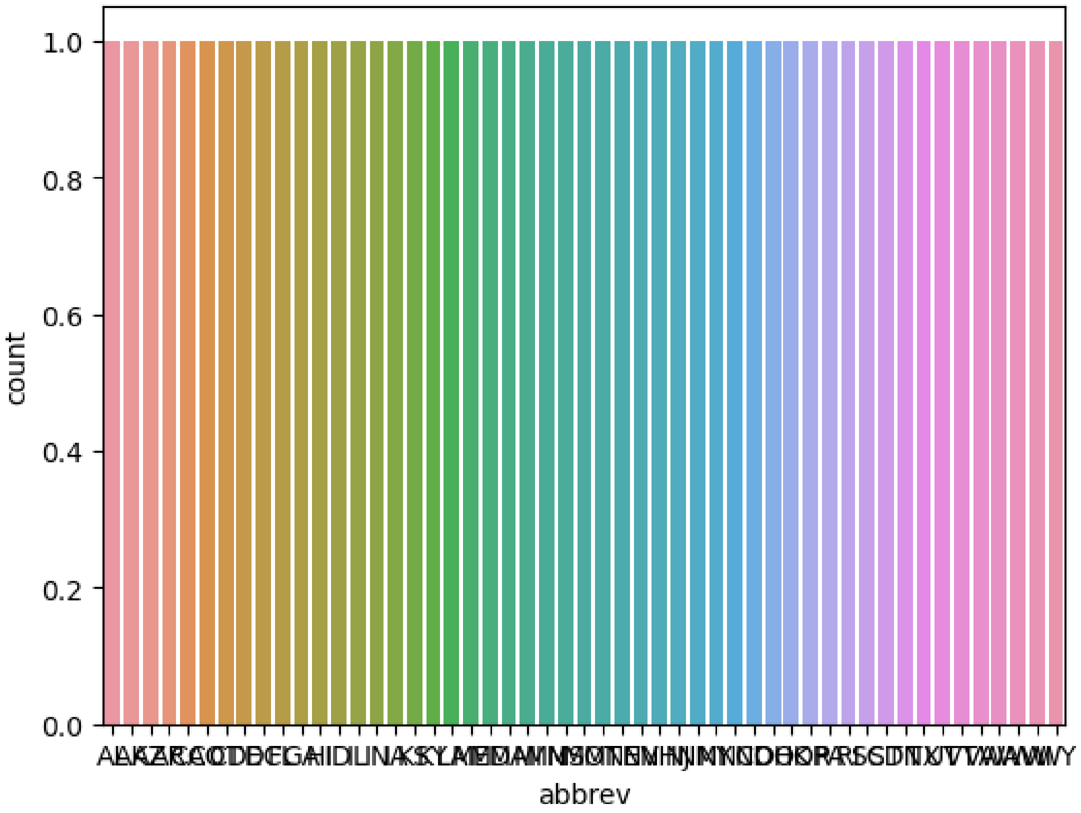
```
<Axes: xlabel='abbrev', ylabel='total'>
```



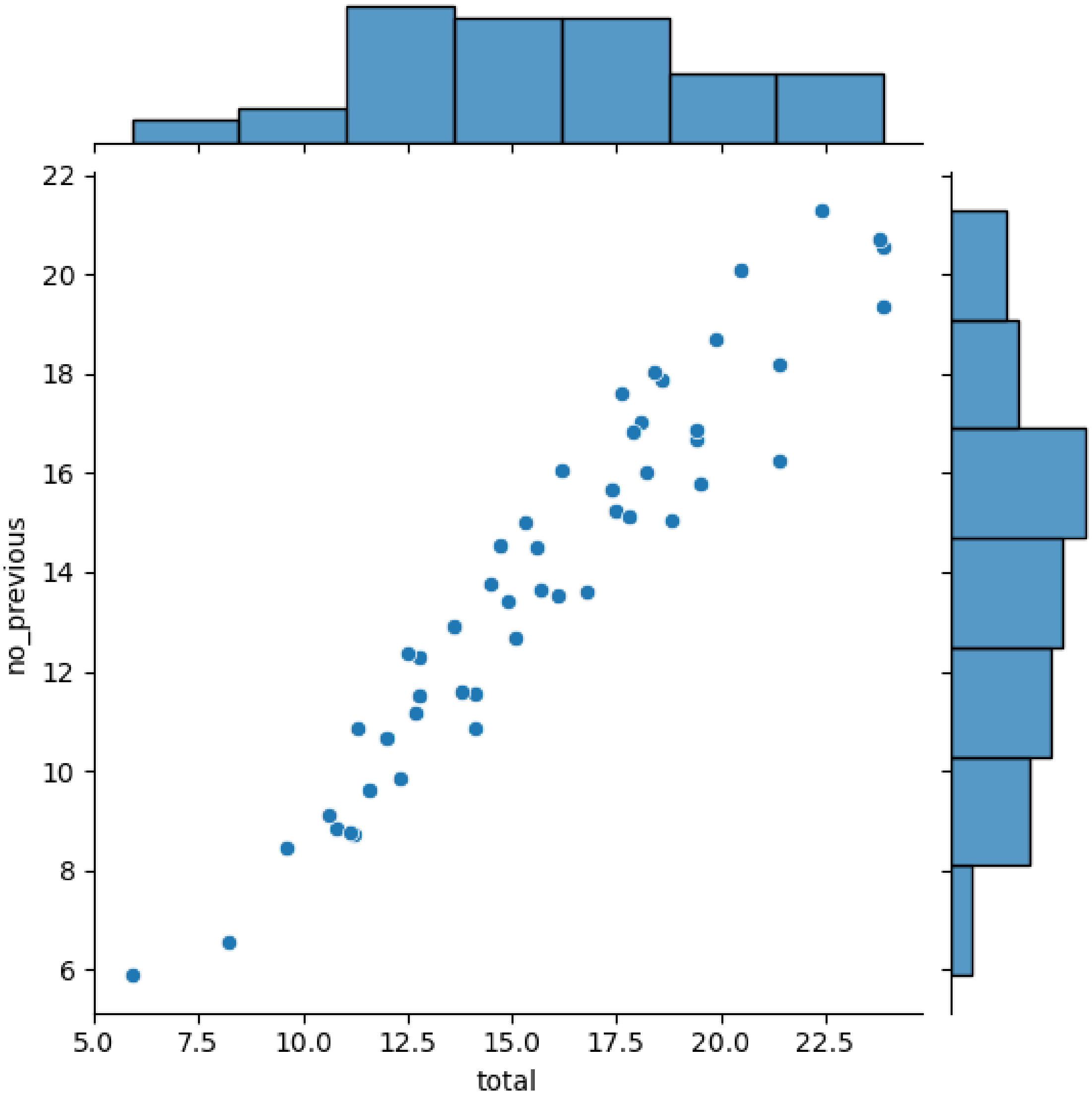
```
#The resulting bar plot will display bars for each unique value in the
'abbrev' column (assuming 'abbrev' represents abbreviations or
labels), with the height of each bar corresponding to the 'total'
value associated with that abbreviation

plt.figure(figsize=(30, 20))  # Adjust the figure size as needed
sns.barplot(data=df, x='alcohol', y='no_previous', hue='abbrev')

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



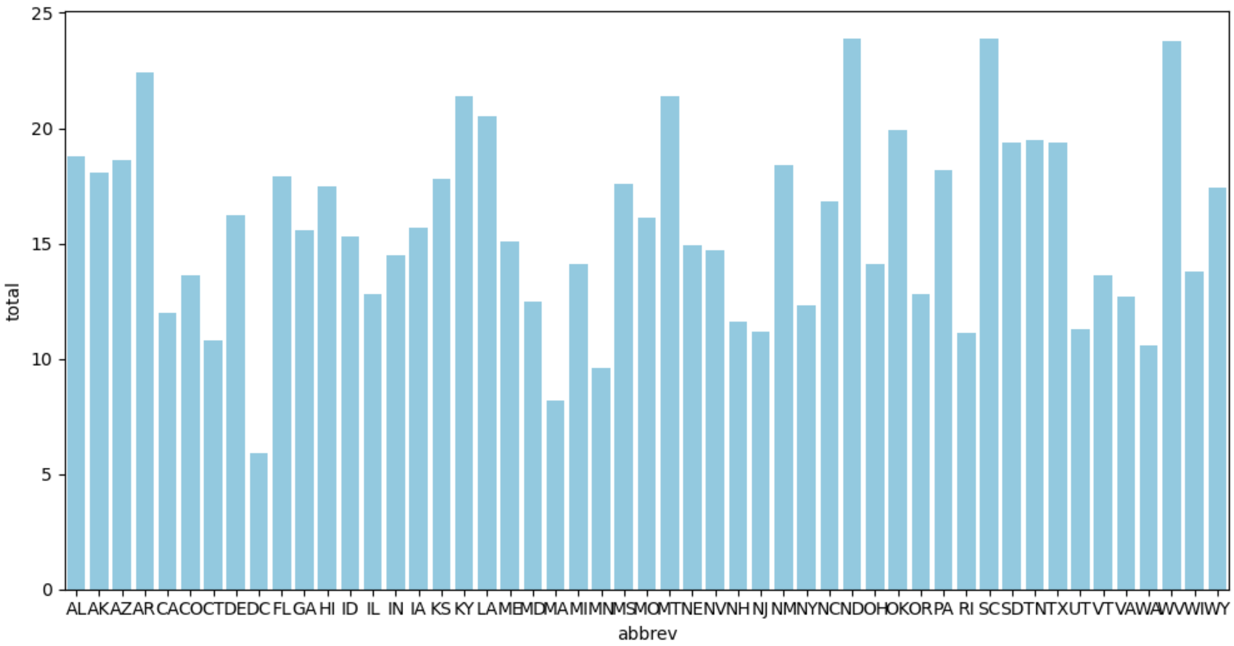
```
#Creating a Seaborn bar plot with a larger figure size (width: 30
units, height: 6 units) to visualize the relationship between the
'no_previous' column and the 'alcohol' column, with the 'abbrev'
column used for hue

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

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



```
#The resulting count plot displays bars for each unique value in the
"abbrev" column, showing the frequency or count of each category
#we can aslo see the same count in the value count

sns.jointplot(x="total",y="no_previous",data=df)

<seaborn.axisgrid.JointGrid at 0x1665d11b310>
```

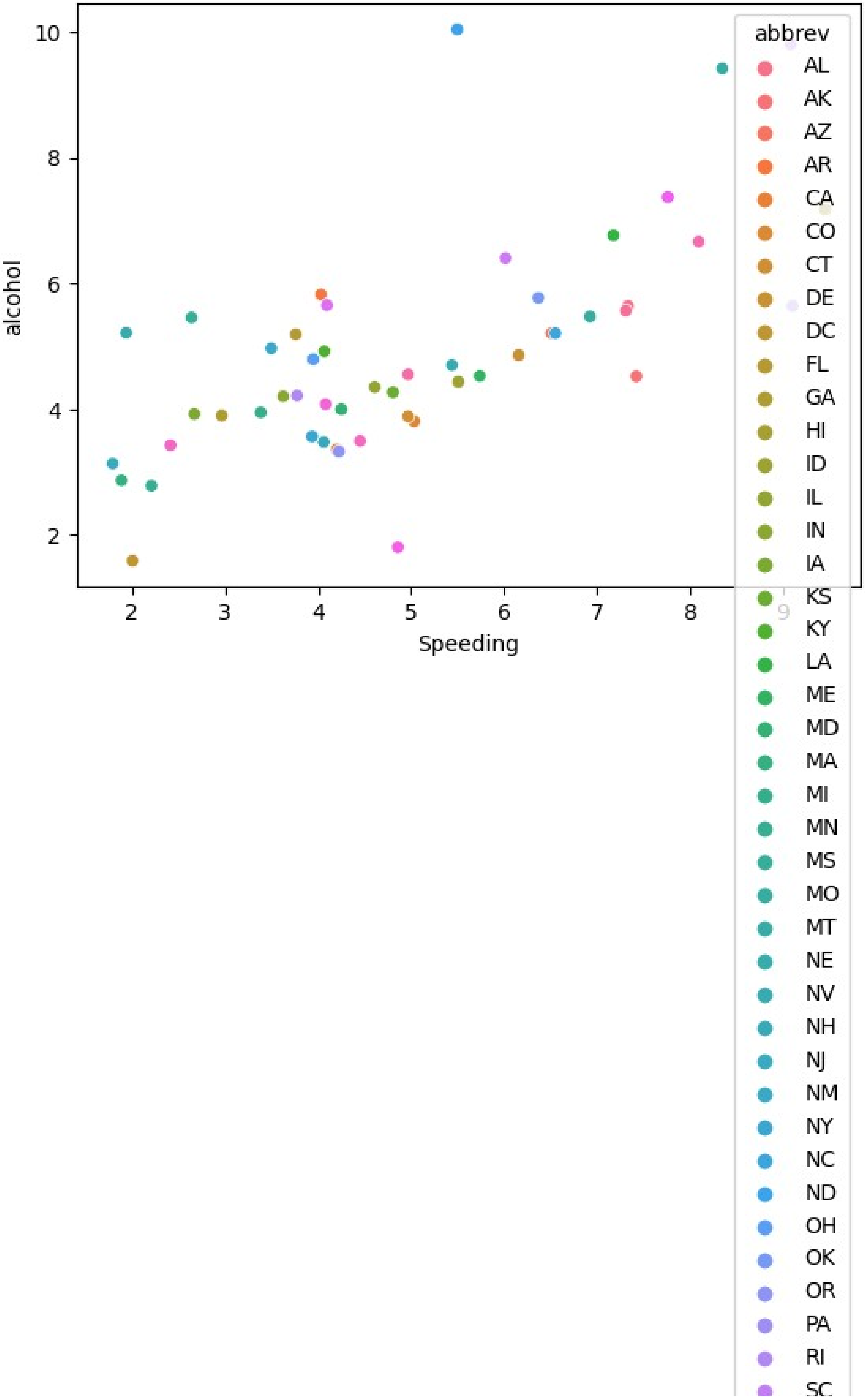


```
#The resulting joint plot provides several insights into the
relationship between the "total" and "no_previous" columns

plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='abbrev', y='total', color='skyblue',
label='Total Crashes')

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



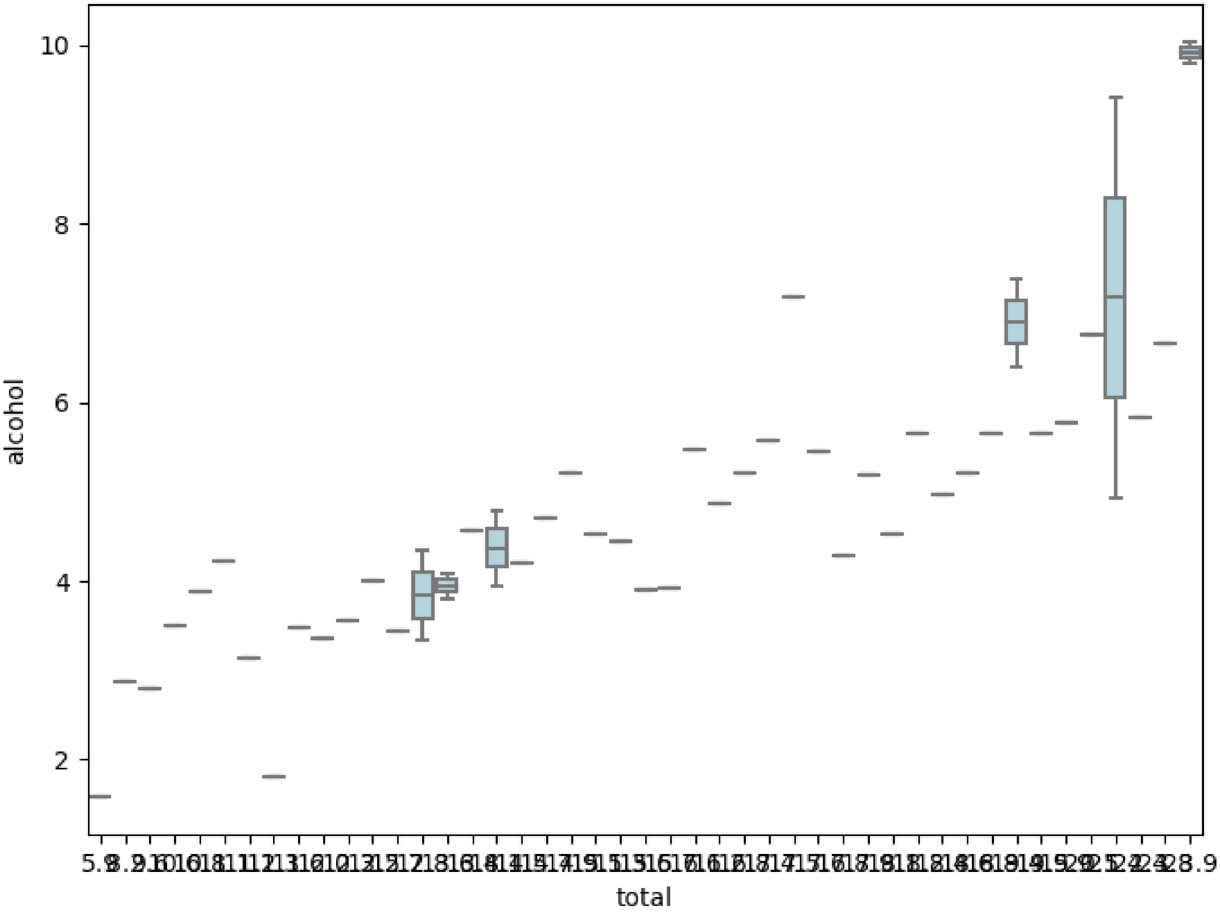
```
#The resulting bar plot will display bars for each unique value in the
'abbrev' column, with the height of each bar representing the 'total'
value associated with that abbreviation

sns.scatterplot(data=df, x='speeding', y='alcohol', hue='abbrev')
plt.xlabel('Speeding')
```

```
Text(0.5, 0, 'Speeding')
```


10
8
6
4
2
alcohol
2
3
4
5
6
7
8
9
Speeding
abbrev
AL
AK
AZ
AR
CA
CO
CT
DE
DC
FL
GA
HI
ID
IL
IN
IA
KS
KY
LA
ME
MD
MA
MI
MN
MS
MO
MT
NE
NV
NH
NJ
NM
NY
NC
ND
OH
OK
OR
PA
RI
SC

```
#The resulting scatter plot provides several insights into the
relationship between 'speeding' and 'alcohol,' with consideration of
the 'abbrev'

plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='total',y="alcohol", color='lightblue')
```

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



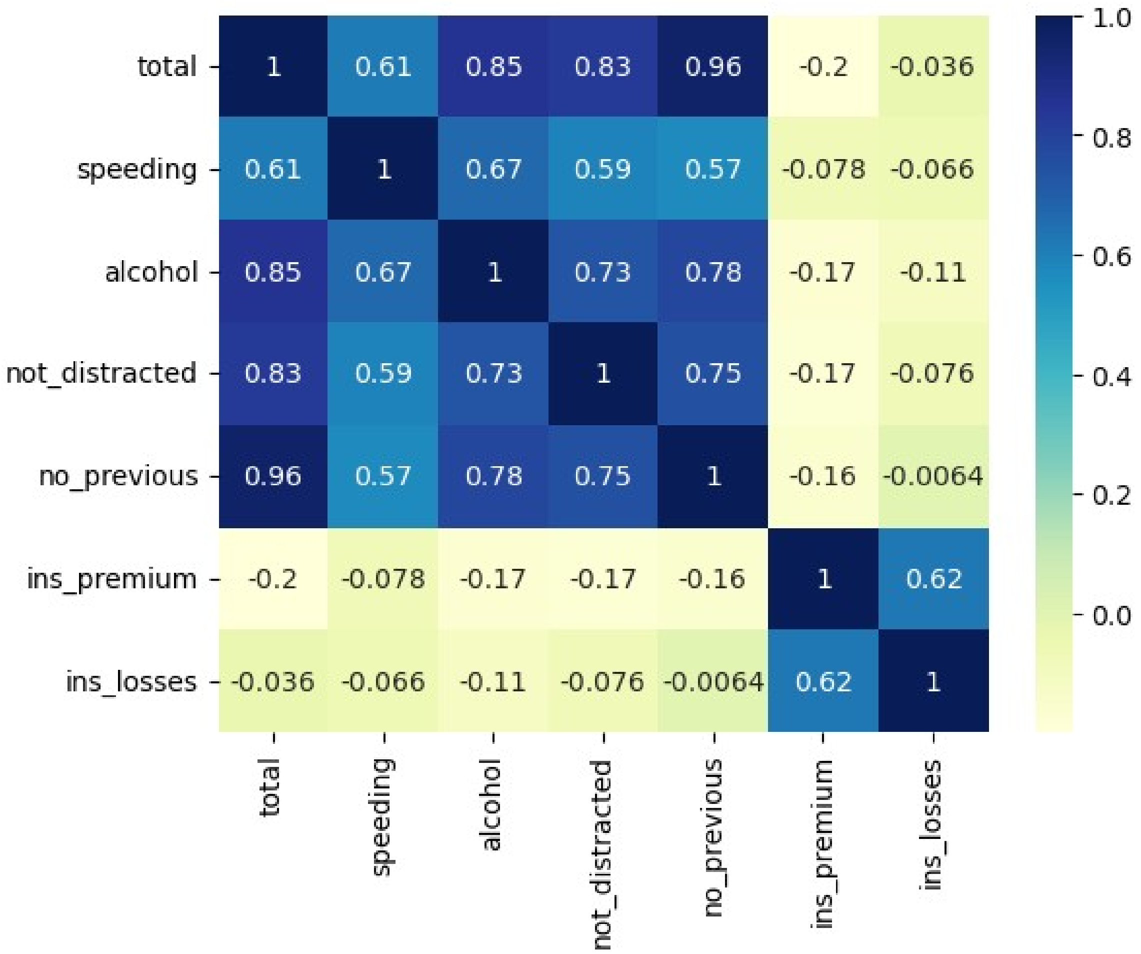
```
# To analyze the relationship between the total number of crashes
('total') and the number of alcohol-related crashes ('alcohol').

corr=df.corr()
corr
```

```
C:\Users\kaler\AppData\Local\Temp\ipykernel_15308\3182140910.py:1:
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.
  corr=df.corr()
```

```
                   total  speeding   alcohol  not_distracted
no_previous  \
total           1.000000  0.611548  0.852613        0.827560
0.956179
speeding        0.611548  1.000000  0.669719        0.588010
0.571976
alcohol         0.852613  0.669719  1.000000        0.732816
0.783520
not_distracted  0.827560  0.588010  0.732816        1.000000
0.747307
no_previous     0.956179  0.571976  0.783520        0.747307
1.000000
ins_premium    -0.199702 -0.077675 -0.170612       -0.174856    -
0.156895
ins_losses     -0.036011 -0.065928 -0.112547       -0.075970    -
0.006359

                ins_premium  ins_losses
total             -0.199702   -0.036011
speeding          -0.077675   -0.065928
alcohol           -0.170612   -0.112547
not_distracted    -0.174856   -0.075970
no_previous       -0.156895   -0.006359
ins_premium        1.000000    0.623116
ins_losses         0.623116    1.000000
```

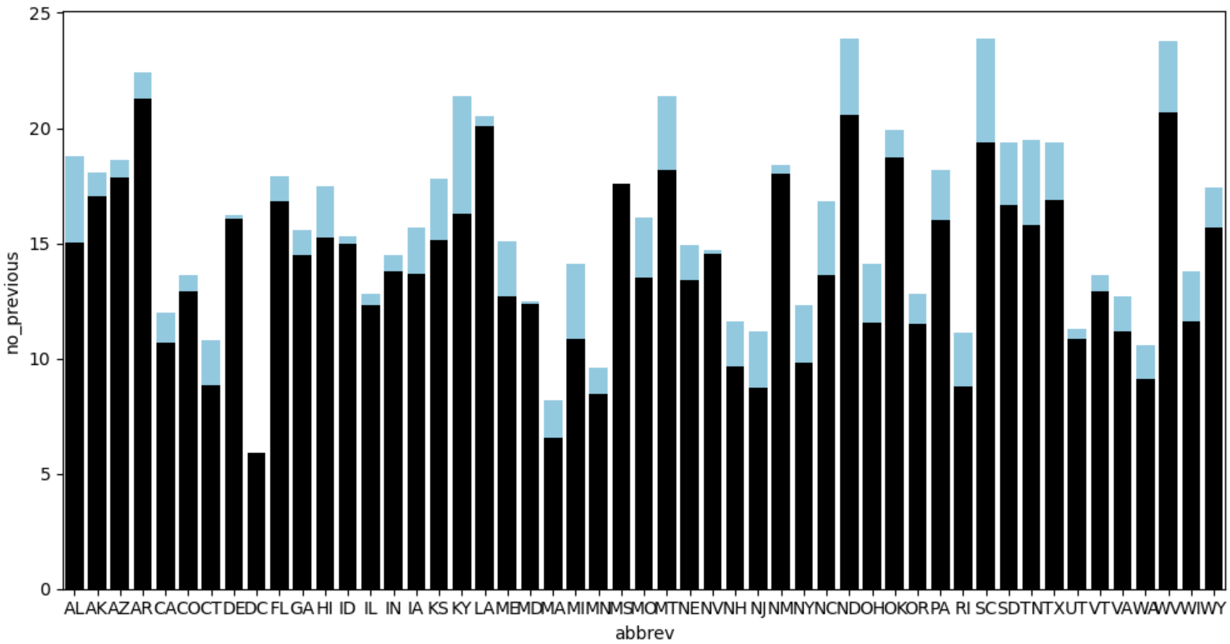
```
#we can see that "no_previous" is highly correlated to "total".we can
also see all the correlations

sns.heatmap(corr,annot=True,cmap="YlGnBu")
```

```
<Axes: >
```



```
#corellation between every attribute is given here
#>0.5 is highly correlated
#<0.5 is less correlated
#infernce:no_previous is highly collerated to total car crashes

plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='abbrev', y='total', color='skyblue',
label='Total Crashes')
sns.barplot(data=df,x='abbrev',y='no_previous',color='black',label="no
-previous-crashes")
sns.barplot
```

```
<function seaborn.categorical.barplot(data=None, *, x=None, y=None,
hue=None, order=None, hue_order=None, estimator='mean',
errorbar=('ci', 95), n_boot=1000, units=None, seed=None, orient=None,
color=None, palette=None, saturation=0.75, width=0.8, errcolor='.26',
errwidth=None, capsize=None, dodge=True, ci='deprecated', ax=None,
**kwargs)>
```



```
#by the bar plot we can say that no_previous and total are highly
corellated
```