In [ ]:
```
#Assignment-2 by Bhavesh-21BCE9264
```

In [3]:
```
#Take car_crashes dataset from seaborn library
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
import matplotlib as mpl
import matplotlib.pyplot as plt

sns.set(style="ticks")
np.random.seed(sum(map(ord, "axis_grids")))
```

In [5]:
```
#Load the dataset
df = sns.load_dataset("car_crashes")
df.head()
```

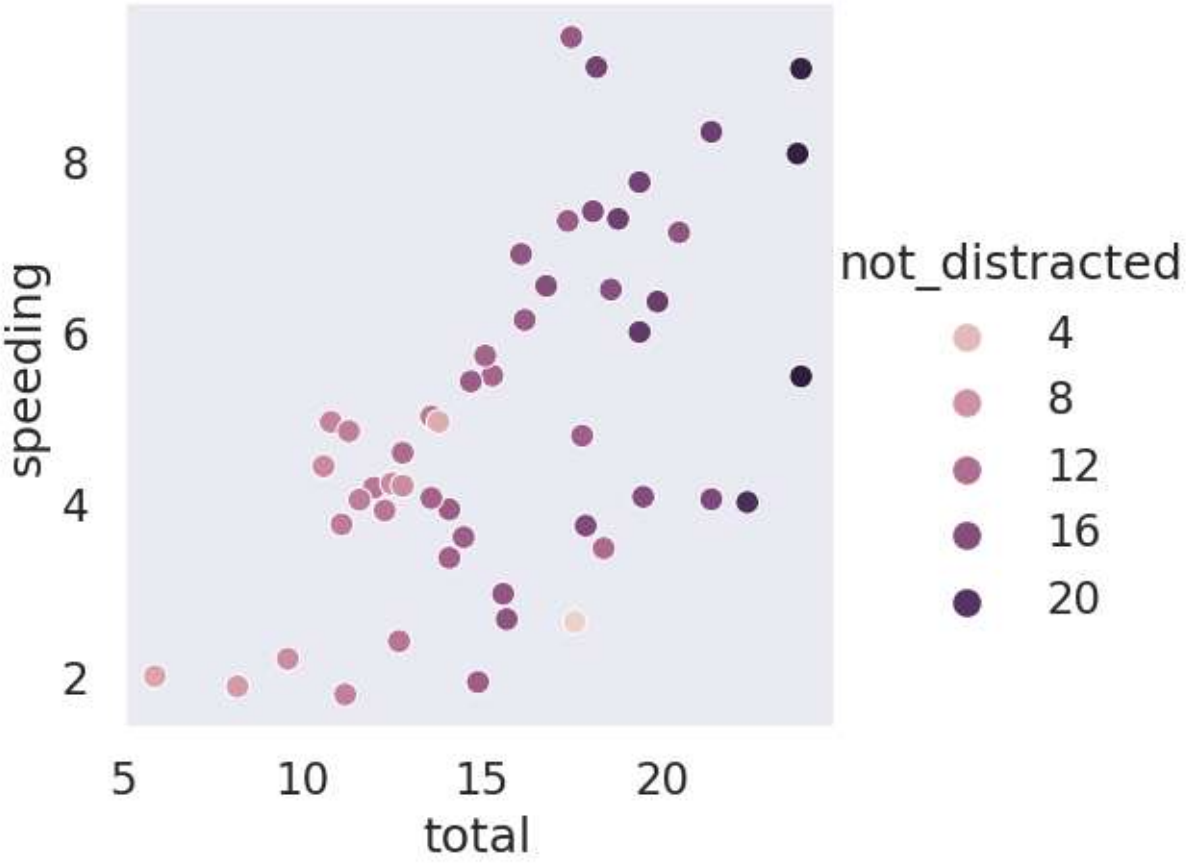
Out[5]:

	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses	abbrev
0	18.8	7.332	5.640	18.048	15.040	784.55	145.08	AL
1	18.1	7.421	4.525	16.290	17.014	1053.48	133.93	AK
2	18.6	6.510	5.208	15.624	17.856	899.47	110.35	AZ
3	22.4	4.032	5.824	21.056	21.280	827.34	142.39	AR
4	12.0	4.200	3.360	10.920	10.680	878.41	165.63	CA

Performing data Visualisation

In [24]:
```
#Relational Plot
sns.relplot(data=df, x="total", y="speeding", hue="not_distracted")
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x7a16e11e06a0>
```

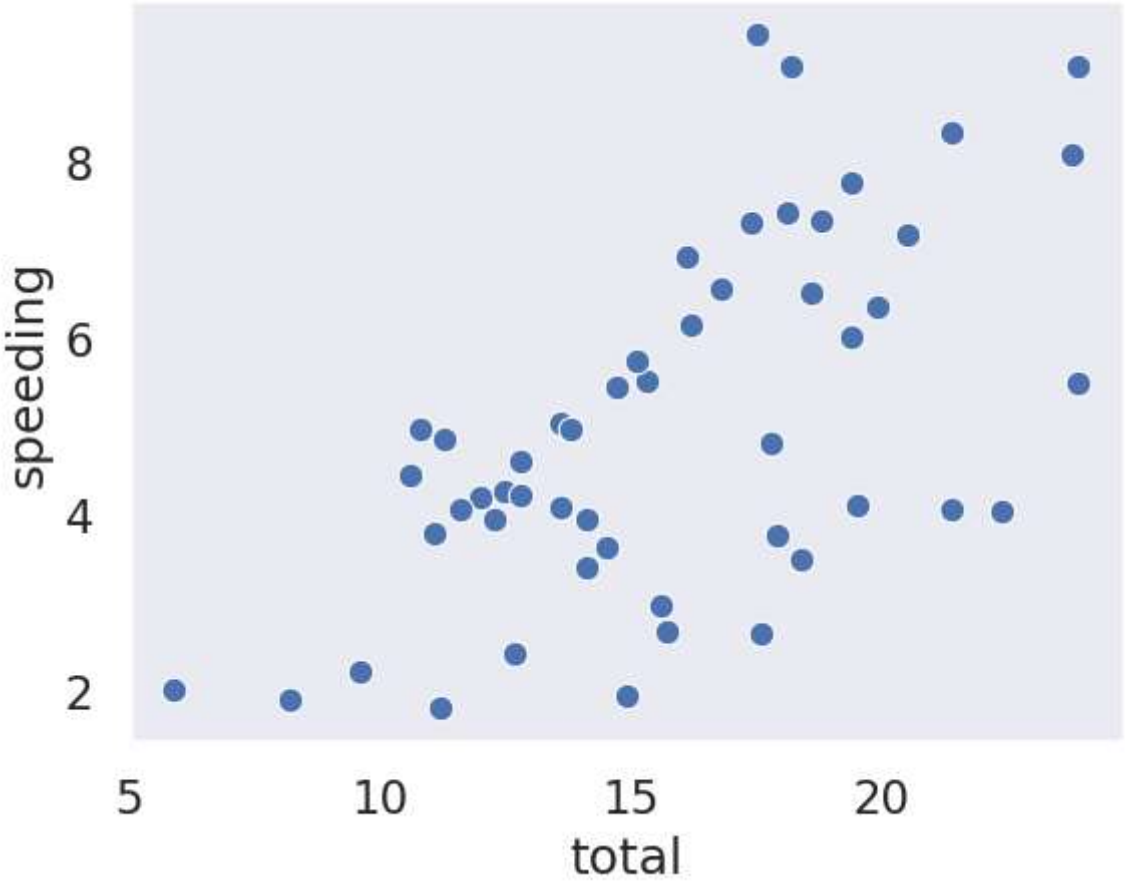


Inference : The Relational Plot (relplot) allows us to visualise how variables within a dataset relate to each other.

In [29]:
```
#Scatter Plot
sns.scatterplot(data=df, x="total", y="speeding", palette="deep")
```

```
<ipython-input-29-11e9ef104022>:2: UserWarning: Ignoring `palette` because no `hue` v
ariable has been assigned.
  sns.scatterplot(data=df, x="total", y="speeding",palette="deep")
```

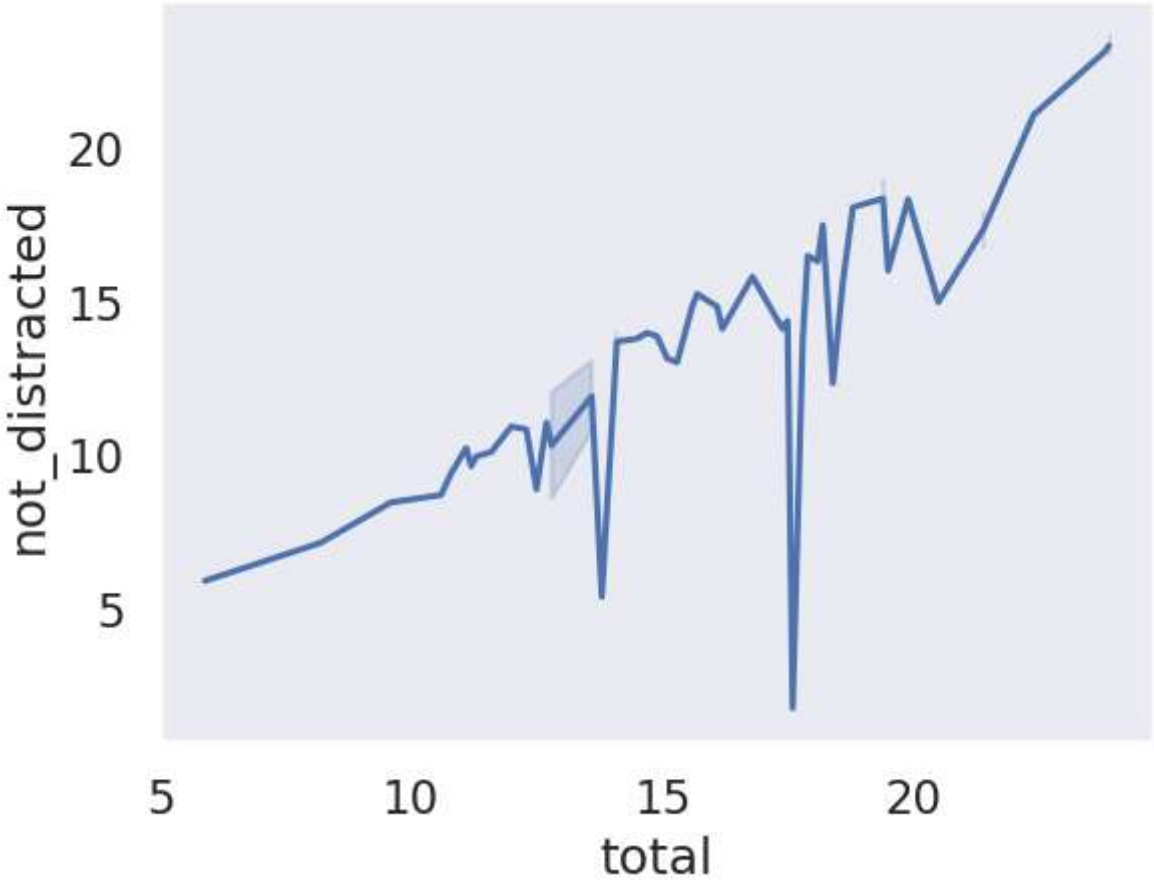
Out[29]:
```
<Axes: xlabel='total', ylabel='speeding'>
```



Inference :The scatterplot() method helps to draw a scatter plot with the possibility of several semantic groupings.

In [32]:
```
#Line Plot
sns.lineplot(data=df, x="total", y="not_distracted")
```

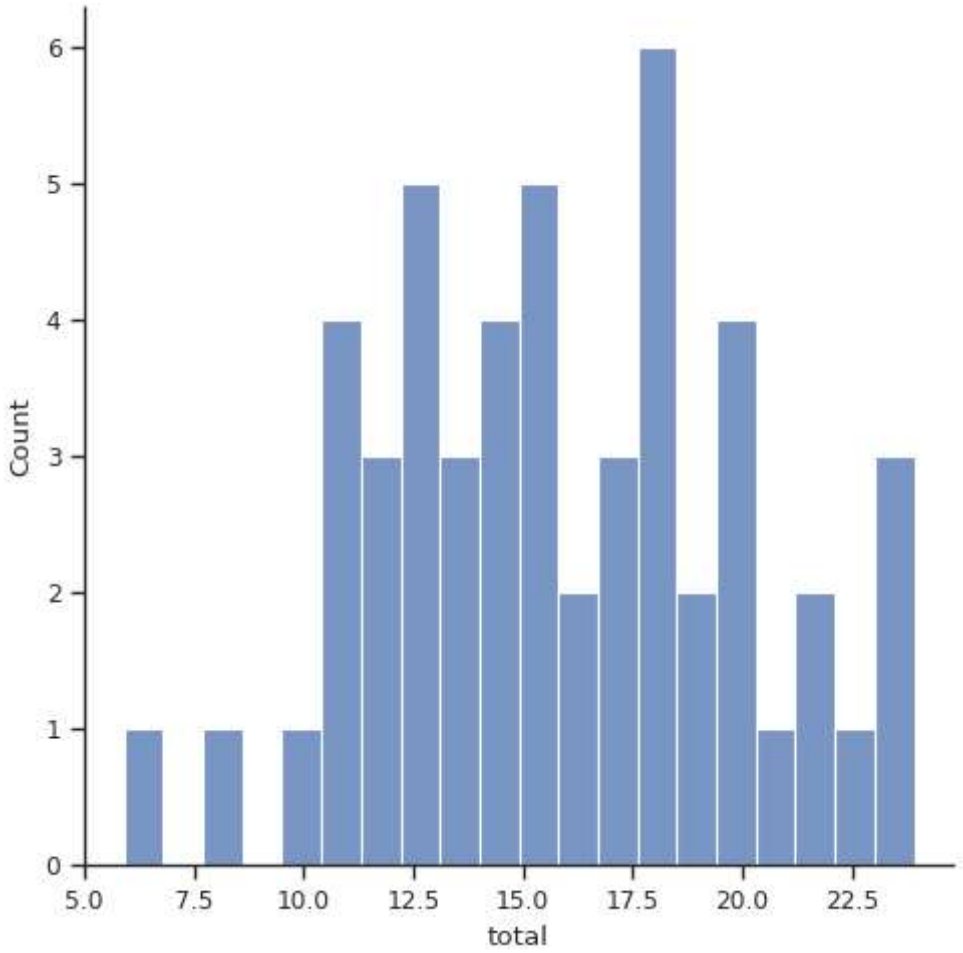
Out[32]:
```
<Axes: xlabel='total', ylabel='not_distracted'>
```



Inference : Seaborn Line Plots depict the relationship between continuous as well as categorical values in a continuous data point format.

In [22]:
```
#Distribution Plot
sns.displot(df['total'],kde=False,bins =20)
```

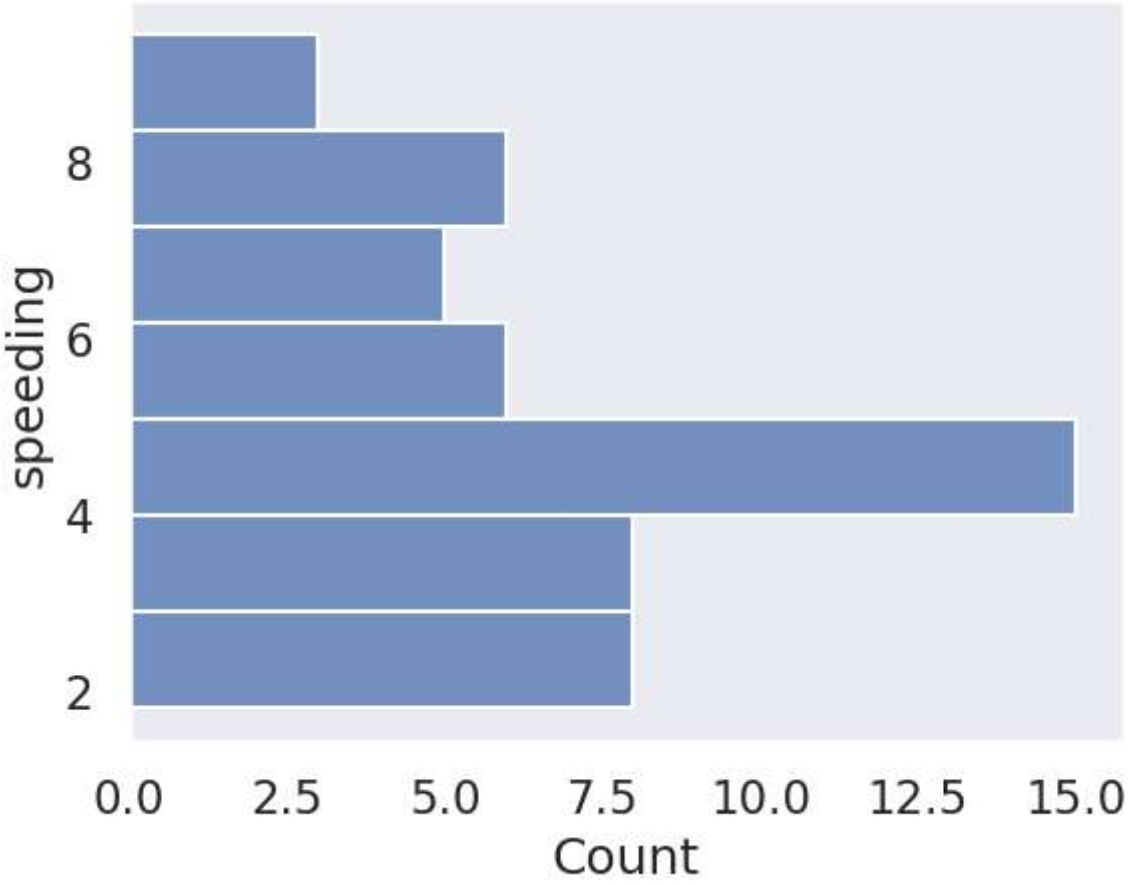
Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x7a16da0342b0>
```



Inference: Distribution plot in seaborn library will give us the histogram of the attribute which we have mentioned. It will also give us the kernel density estimation line which we can set as true or false.

In [33]:
```
#Histogram Plot
sns.histplot(data=df, y="speeding")
```

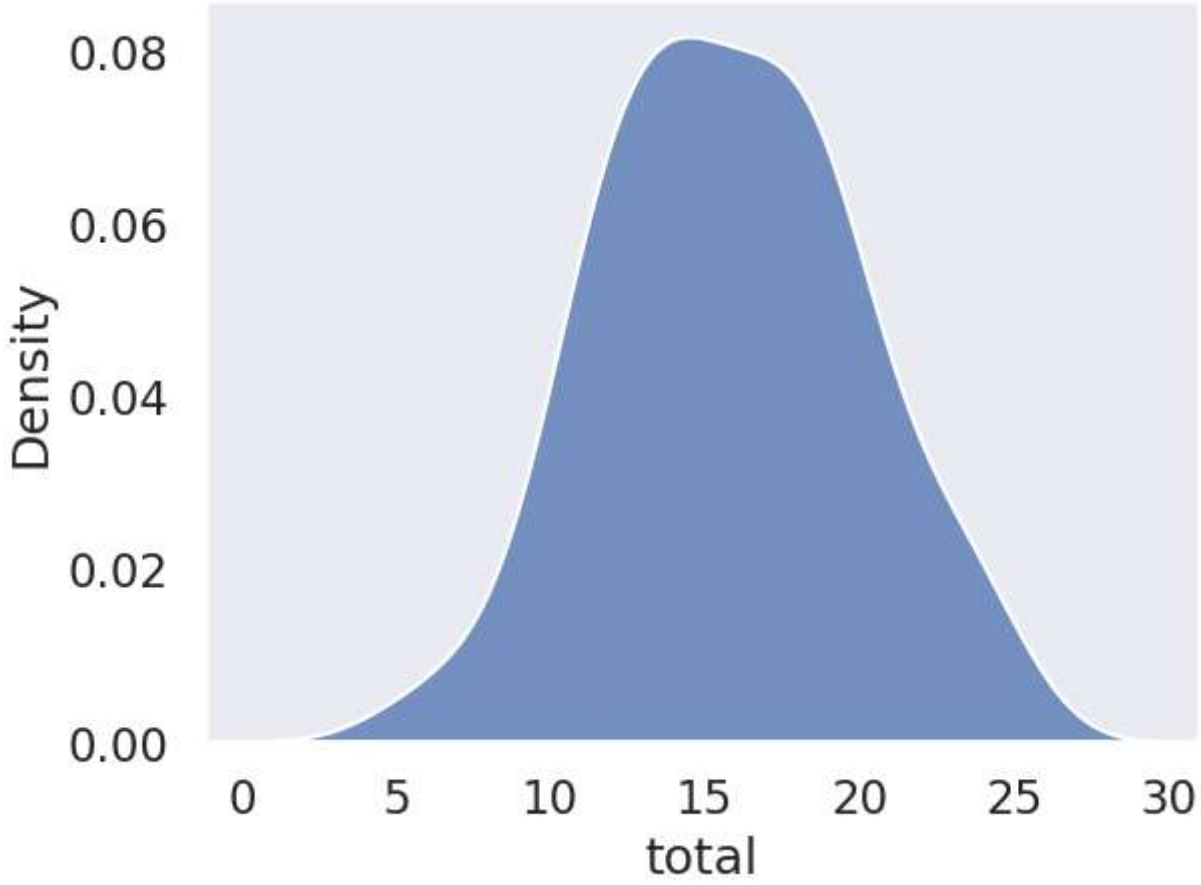
Out[33]:
```
<Axes: xlabel='Count', ylabel='speeding'>
```



Inference: A histogram represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.

In [36]:
```
#kde plot
sns.kdeplot(data=df, x="total",multiple="stack")
```

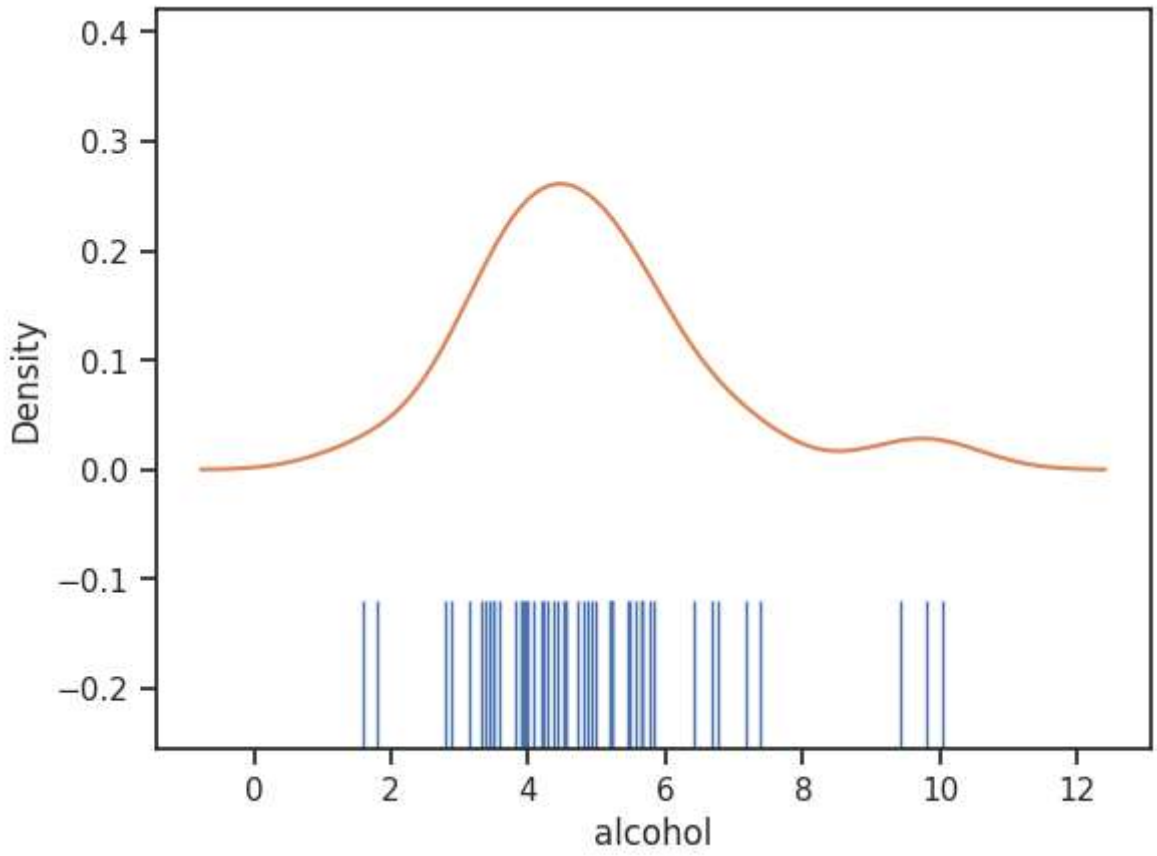
Out[36]:
```
<Axes: xlabel='total', ylabel='Density'>
```



Inference: The kdeplot() function is used to plot the data against a single/univariate variable. It represents the probability distribution of the data values as the area under the plotted curve.

In [16]:
```
#Rug Plot
sns.rugplot(df['alcohol'],height=0.2)
sns.kdeplot(df['alcohol'])
```

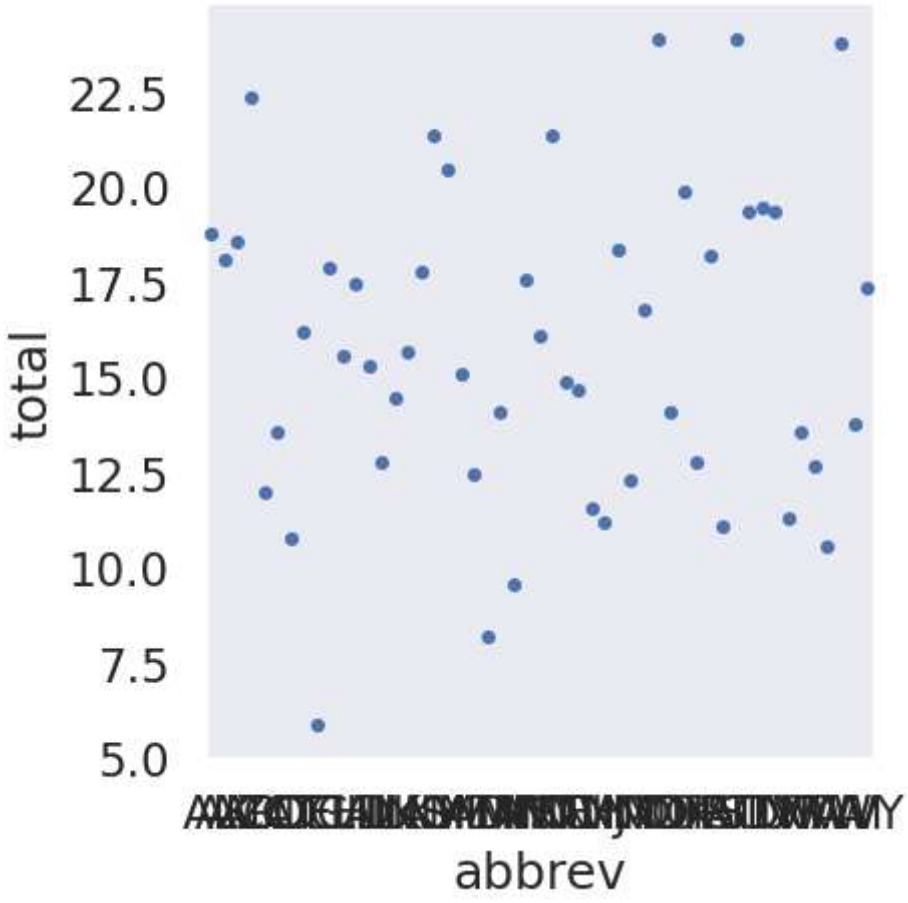
Out[16]:
```
<Axes: xlabel='alcohol', ylabel='Density'>
```



Inference : Rug plot is going to plot a single column of data points in a dataframe as sticks. And with a rug plot we can see more dense amount of lines where the amount is most common.

In [40]:
```
#Cat Plot
sns.catplot(data=df, x="abbrev", y="total")
```

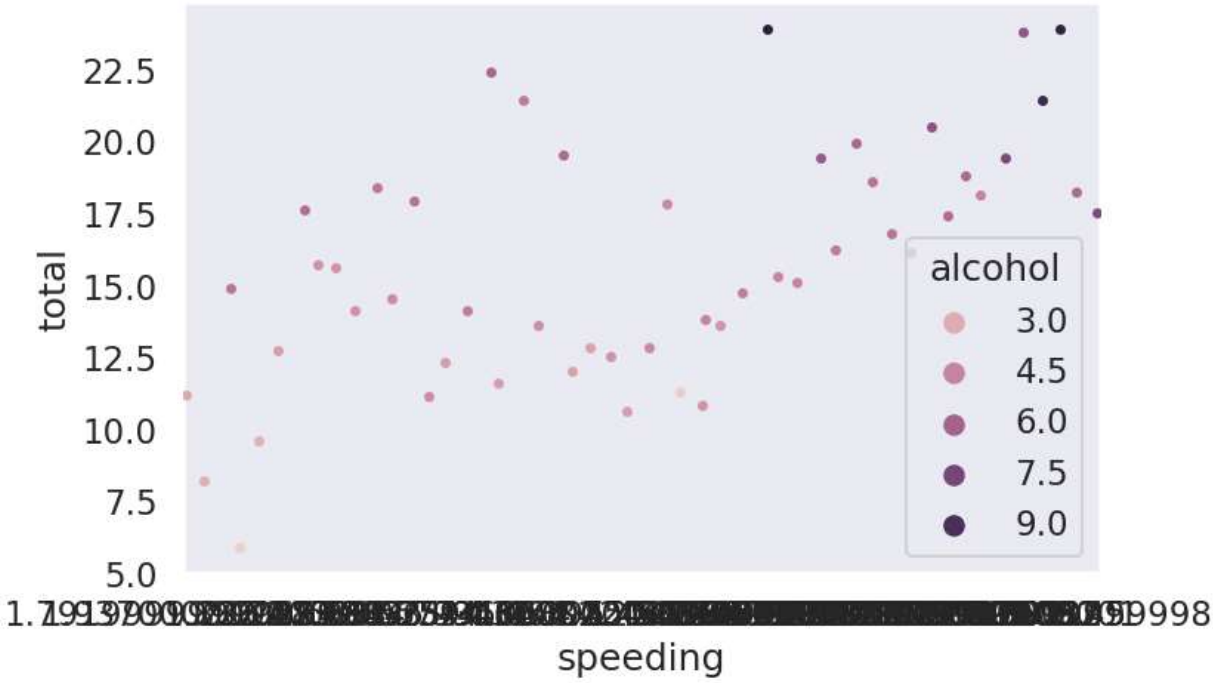
Out[40]:
```
<seaborn.axisgrid.FacetGrid at 0x7a16d684c070>
```



Inference : The catplot() method is used to plot categorical plots.This function gives users access to a number of axes-level functions that illustrate the connection between numerical data and one or more category variables.

In [23]:
```
#Strip Plot
plt.figure(figsize=(8,5))
sns.set_style('dark')
sns.set_context('talk')
sns.stripplot(x='speeding',y='total',data=df ,hue='alcohol',jitter=True,dodge=True)
```

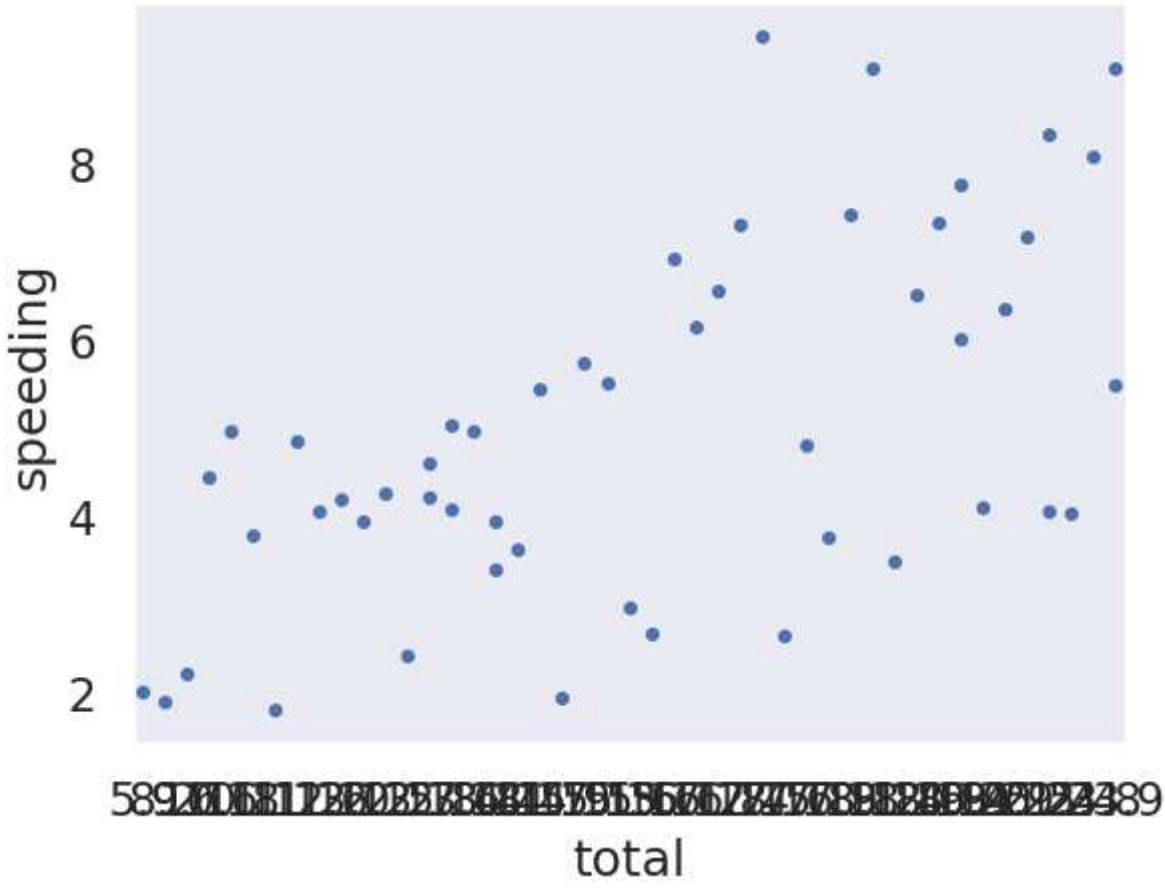
Out[23]:
```
<Axes: xlabel='speeding', ylabel='total'>
```



Inference : A strip plot is a single-axis scatter plot that is used to visualise the distribution of many individual one-dimensional values.

In [43]:
```
#Swarm Plot
sns.swarmplot(data=df, x="total",y="speeding")
```

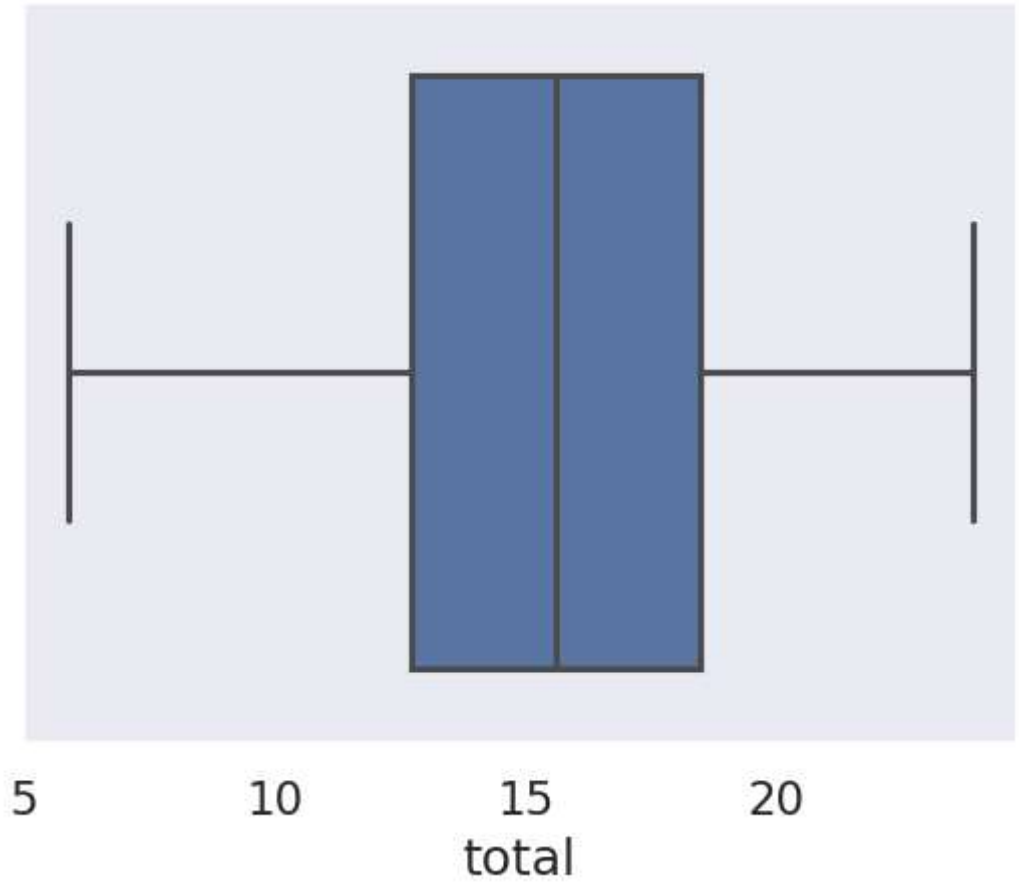
Out[43]:
```
<Axes: xlabel='total', ylabel='speeding'>
```



Inference : A swarm plot is a type of scatter plot that is used for representing categorical values. It is very similar to the strip plot, but it avoids the overlapping of points.

In [45]:
```
# Box Plot
sns.boxplot(data=df, x="total")
```

Out[45]:
```
<Axes: xlabel='total'>
```

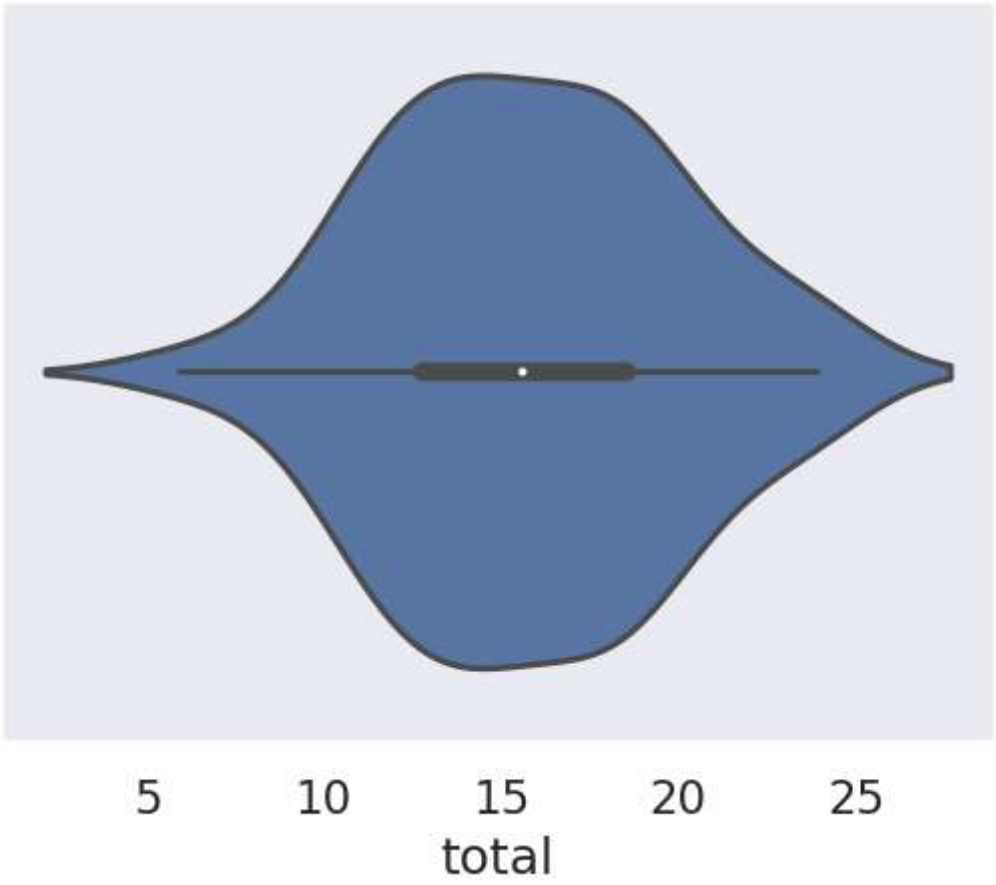


Box plot will allow us to compare different variables. It will show us the quartiles of the data.

The black line (horizontal) in the middle of the bar is the median. The box is going to extend one standard deviation from our median. The black line(vertical) on the bar is called whiskers and they are going to extend to all of the other data asides from what is in our standard deviation

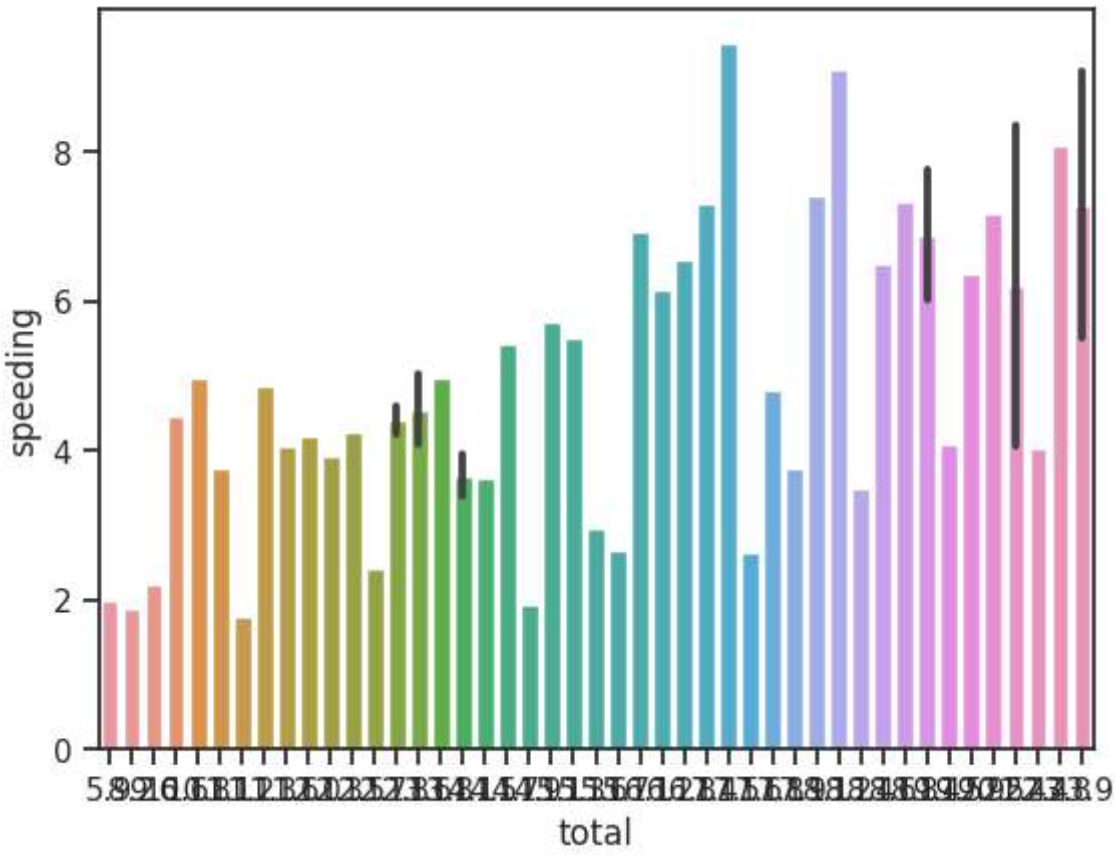
In [46]:
```
#Violin Plot
sns.violinplot(x=df["total"])
```

Out[46]:
```
<Axes: xlabel='total'>
```



Inference : The violinplot() method allows the user to draw a box plot along with a kernel density estimate

In [17]:
```
#Bar Plot
sns.barplot(x = 'total' , y = 'speeding' , data = df)
sns.set_context('paper')
```



Inference :barplot() method is used to show point estimates and confidence intervals as rectangular bars. With the height of each rectangle representing an estimate of central tendency for a numerical variable, a bar plot also shows the degree of uncertainty surrounding that estimate via error bars.

In [18]:
```
crash_mx = df.corr()
crash_mx
```

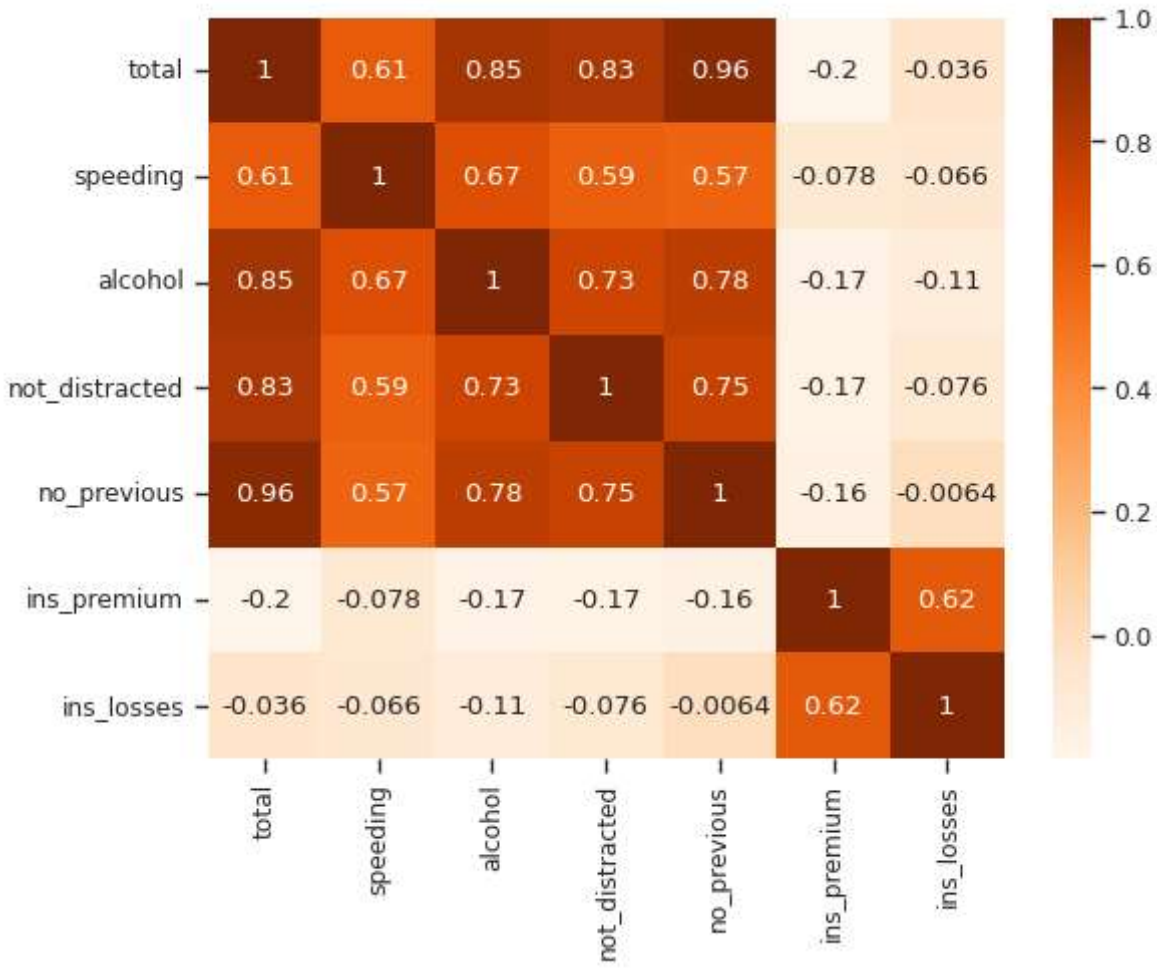
```
<ipython-input-18-c37eb2669dba>:1: FutureWarning: The default value of numeric_only i
n DataFrame.corr is deprecated. In a future version, it will default to False. Select
only valid columns or specify the value of numeric_only to silence this warning.
  crash_mx = df.corr()
```

Out[18]:

	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses
total	1.000000	0.611548	0.852613	0.827560	0.956179	-0.199702	-0.036011
speeding	0.611548	1.000000	0.669719	0.588010	0.571976	-0.077675	-0.065928
alcohol	0.852613	0.669719	1.000000	0.732816	0.783520	-0.170612	-0.112547
not_distracted	0.827560	0.588010	0.732816	1.000000	0.747307	-0.174856	-0.075970
no_previous	0.956179	0.571976	0.783520	0.747307	1.000000	-0.156895	-0.006359
ins_premium	-0.199702	-0.077675	-0.170612	-0.174856	-0.156895	1.000000	0.623116
ins_losses	-0.036011	-0.065928	-0.112547	-0.075970	-0.006359	0.623116	1.000000

In [19]:
```
#Heat Map
sns.set_context('paper')
sns.heatmap(crash_mx,annot=True,cmap='Oranges')
```

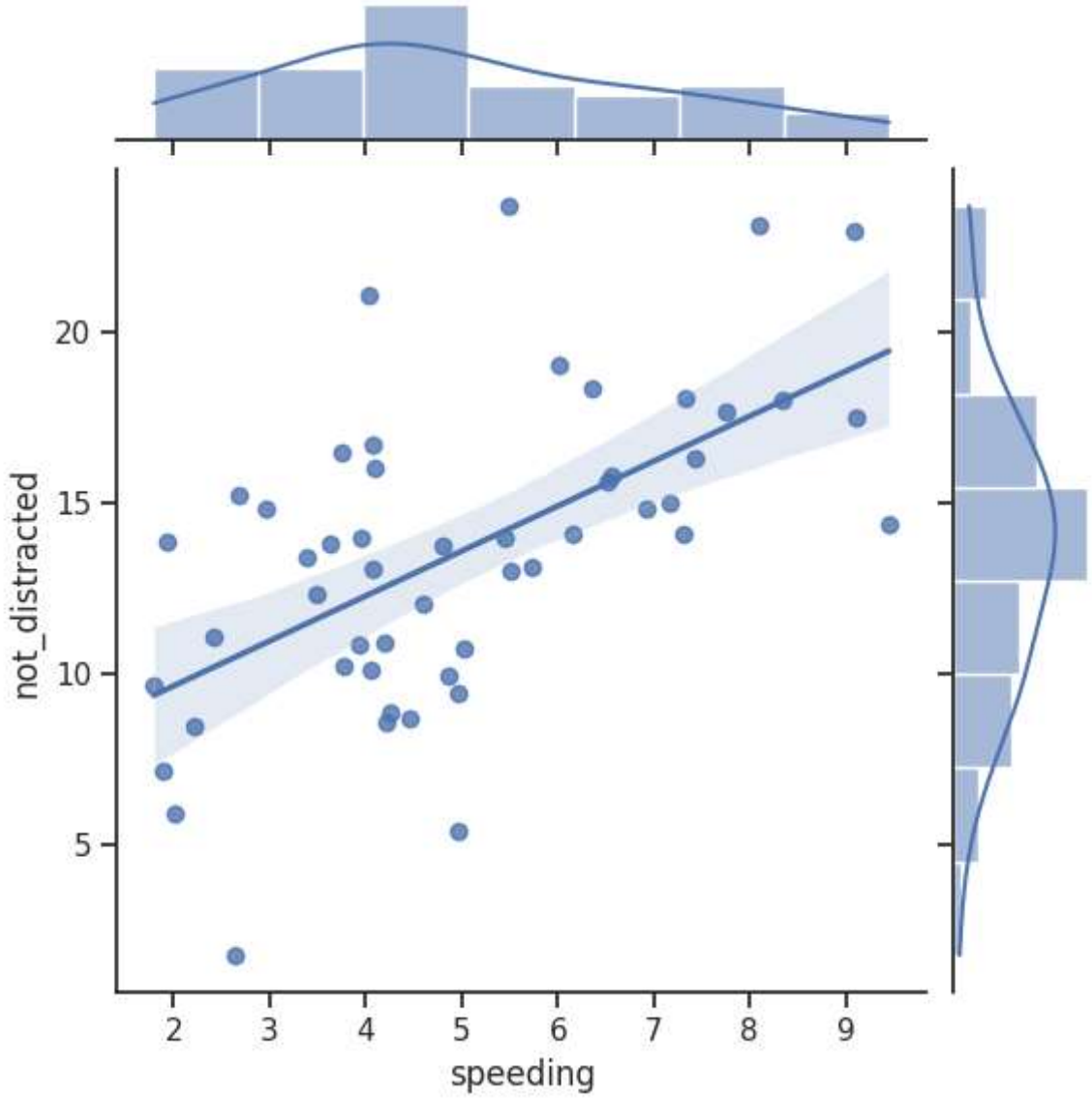
Out[19]:
```
<Axes: >
```



Inference: A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colours.

In [13]:
```
#Joint Plot
sns.jointplot(x='speeding',y='not_distracted', data = df , kind = 'reg')
```

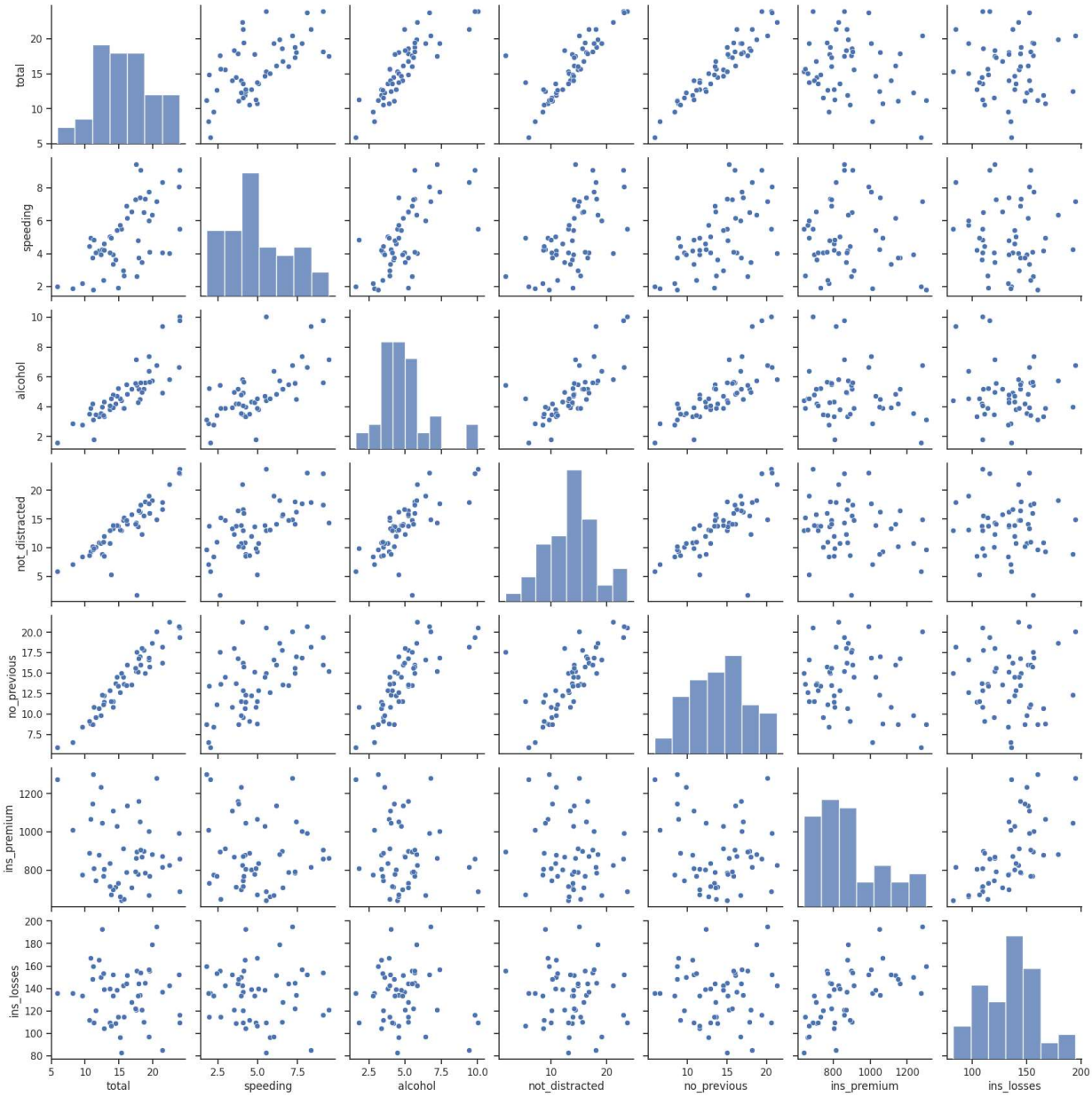
Out[13]:
```
<seaborn.axisgrid.JointGrid at 0x7a16e0ccb880>
```



Inference : Joint Plot is basically used to compare two distribution and it plots a scatter plot by default but we can change it by using kind.

In [14]:
```
#Pair Plot
sns.pairplot(df)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x7a16de825a20>
```



Inference: A pair plot is going to plot the relationships across the entire Dataframe numerical values.

We can also use hue for categorical data. And the plot will be colorized based upon that categorical data