In [2]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\dell\anaconda3\lib\site-package
s (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\dell\anaconda3\lib
\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\users\dell\anaconda3\lib\site-pa
ckages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\dell\anaconda3
\lib\site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\dell\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\dell\anaconda3\lib\site-pa
ckages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dell\anaconda3\lib\si
te-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dell\anaconda3\lib\si
te-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\dell\anaconda3\lib\site
-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\dell\anaconda3\lib\site-p
ackages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\dell\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dell\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\site-pa
ckages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packag
es (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
```

importing seaborn library

In [3]:
```
import seaborn as sns
```

In [4]:
```
print(sns.get_dataset_names())
```

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

In [5]:
```
df=sns.load_dataset('car_crashes')
```

In [6]:
```
df
```

Out[6]:

	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
5	13.6	5.032	3.808	10.744	12.920	835.50	139.91	CO
6	10.8	4.968	3.888	9.396	8.856	1068.73	167.02	CT
7	16.2	6.156	4.860	14.094	16.038	1137.87	151.48	DE
8	5.9	2.006	1.593	5.900	5.900	1273.89	136.05	DC
9	17.9	3.759	5.191	16.468	16.826	1160.13	144.18	FL
10	15.6	2.964	3.900	14.820	14.508	913.15	142.80	GA
11	17.5	9.450	7.175	14.350	15.225	861.18	120.92	HI
12	15.3	5.508	4.437	13.005	14.994	641.96	82.75	ID
13	12.8	4.608	4.352	12.032	12.288	803.11	139.15	IL
14	14.5	3.625	4.205	13.775	13.775	710.46	108.92	IN
15	15.7	2.669	3.925	15.229	13.659	649.06	114.47	IA
16	17.8	4.806	4.272	13.706	15.130	780.45	133.80	KS
17	21.4	4.066	4.922	16.692	16.264	872.51	137.13	KY
18	20.5	7.175	6.765	14.965	20.090	1281.55	194.78	LA
19	15.1	5.738	4.530	13.137	12.684	661.88	96.57	ME
20	12.5	4.250	4.000	8.875	12.375	1048.78	192.70	MD
21	8.2	1.886	2.870	7.134	6.560	1011.14	135.63	MA
22	14.1	3.384	3.948	13.395	10.857	1110.61	152.26	MI
23	9.6	2.208	2.784	8.448	8.448	777.18	133.35	MN
24	17.6	2.640	5.456	1.760	17.600	896.07	155.77	MS
25	16.1	6.923	5.474	14.812	13.524	790.32	144.45	MO
26	21.4	8.346	9.416	17.976	18.190	816.21	85.15	MT
27	14.9	1.937	5.215	13.857	13.410	732.28	114.82	NE
28	14.7	5.439	4.704	13.965	14.553	1029.87	138.71	NV
29	11.6	4.060	3.480	10.092	9.628	746.54	120.21	NH
30	11.2	1.792	3.136	9.632	8.736	1301.52	159.85	NJ
31	18.4	3.496	4.968	12.328	18.032	869.85	120.75	NM
32	12.3	3.936	3.567	10.824	9.840	1234.31	150.01	NY
33	16.8	6.552	5.208	15.792	13.608	708.24	127.82	NC
34	23.9	5.497	10.038	23.661	20.554	688.75	109.72	ND
35	14.1	3.948	4.794	13.959	11.562	697.73	133.52	OH

	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses	abbrev
36	19.9	6.368	5.771	18.308	18.706	881.51	178.86	OK
37	12.8	4.224	3.328	8.576	11.520	804.71	104.61	OR
38	18.2	9.100	5.642	17.472	16.016	905.99	153.86	PA
39	11.1	3.774	4.218	10.212	8.769	1148.99	148.58	RI
40	23.9	9.082	9.799	22.944	19.359	858.97	116.29	SC
41	19.4	6.014	6.402	19.012	16.684	669.31	96.87	SD
42	19.5	4.095	5.655	15.990	15.795	767.91	155.57	TN
43	19.4	7.760	7.372	17.654	16.878	1004.75	156.83	TX
44	11.3	4.859	1.808	9.944	10.848	809.38	109.48	UT
45	13.6	4.080	4.080	13.056	12.920	716.20	109.61	VT
46	12.7	2.413	3.429	11.049	11.176	768.95	153.72	VA
47	10.6	4.452	3.498	8.692	9.116	890.03	111.62	WA
48	23.8	8.092	6.664	23.086	20.706	992.61	152.56	WV
49	13.8	4.968	4.554	5.382	11.592	670.31	106.62	WI
50	17.4	7.308	5.568	14.094	15.660	791.14	122.04	WY

In [7]:
```
df.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
```

In [8]:
```
df.head(5)
```

Out[8]:

	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

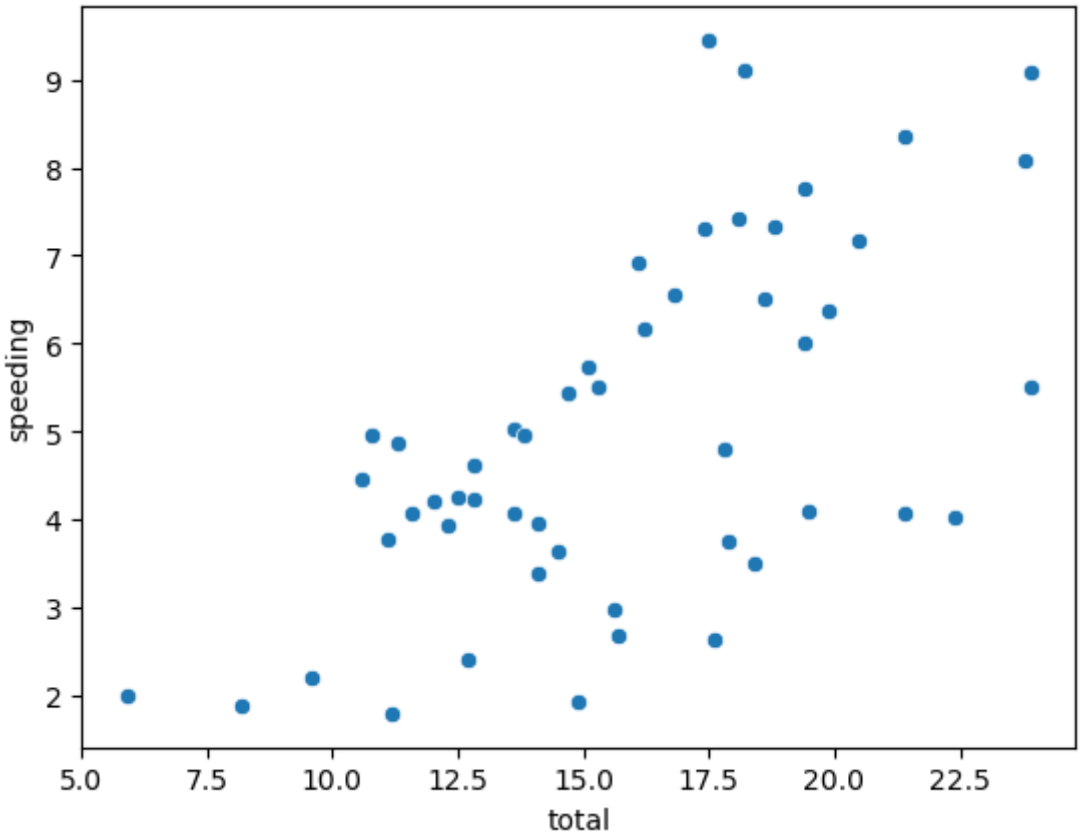
Data Visualization

1)scatterplot

it is also used with several semantic groupings which can help to understand well in graph against categorical data,it can be drawn a two-dimensional graph

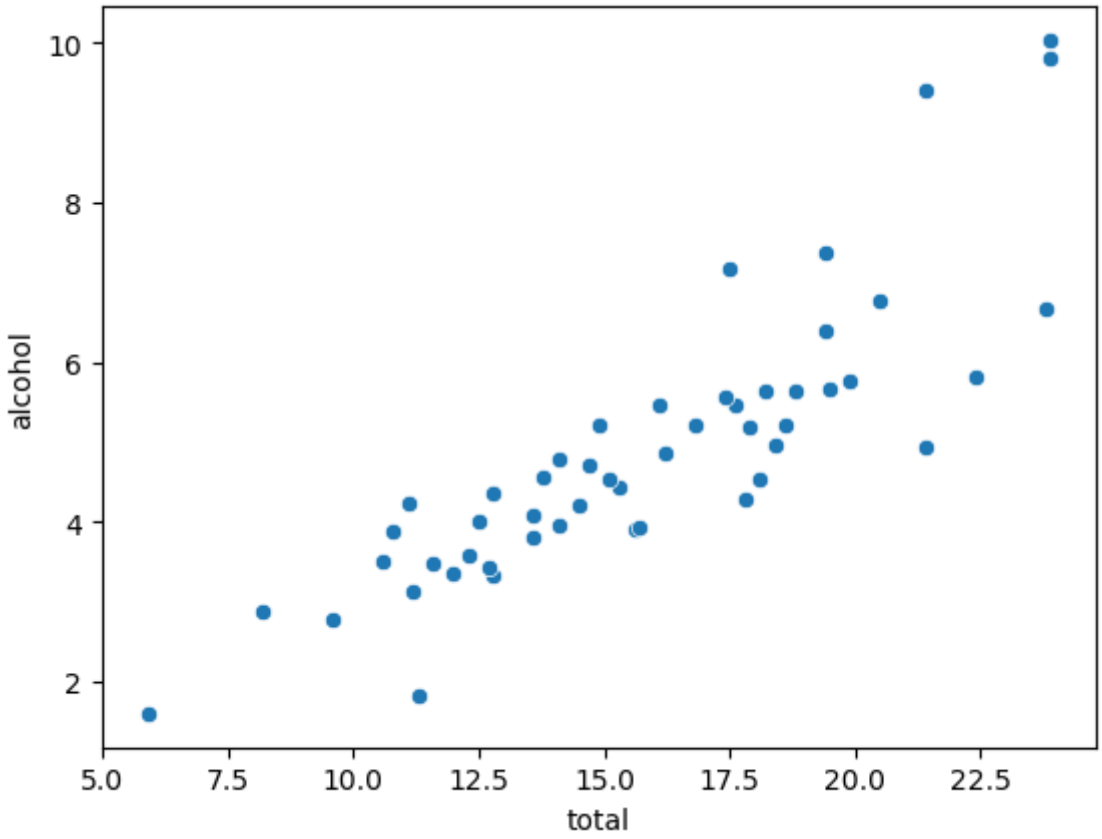
In [10]:
```
sns.scatterplot(x="total",y='speeding',data=df)
```

Out[10]:
```
<Axes: xlabel='total', ylabel='speeding'>
```



In [11]:
```
sns.scatterplot(x="total",y='alcohol',data=df)
```

Out[11]:
```
<Axes: xlabel='total', ylabel='alcohol'>
```



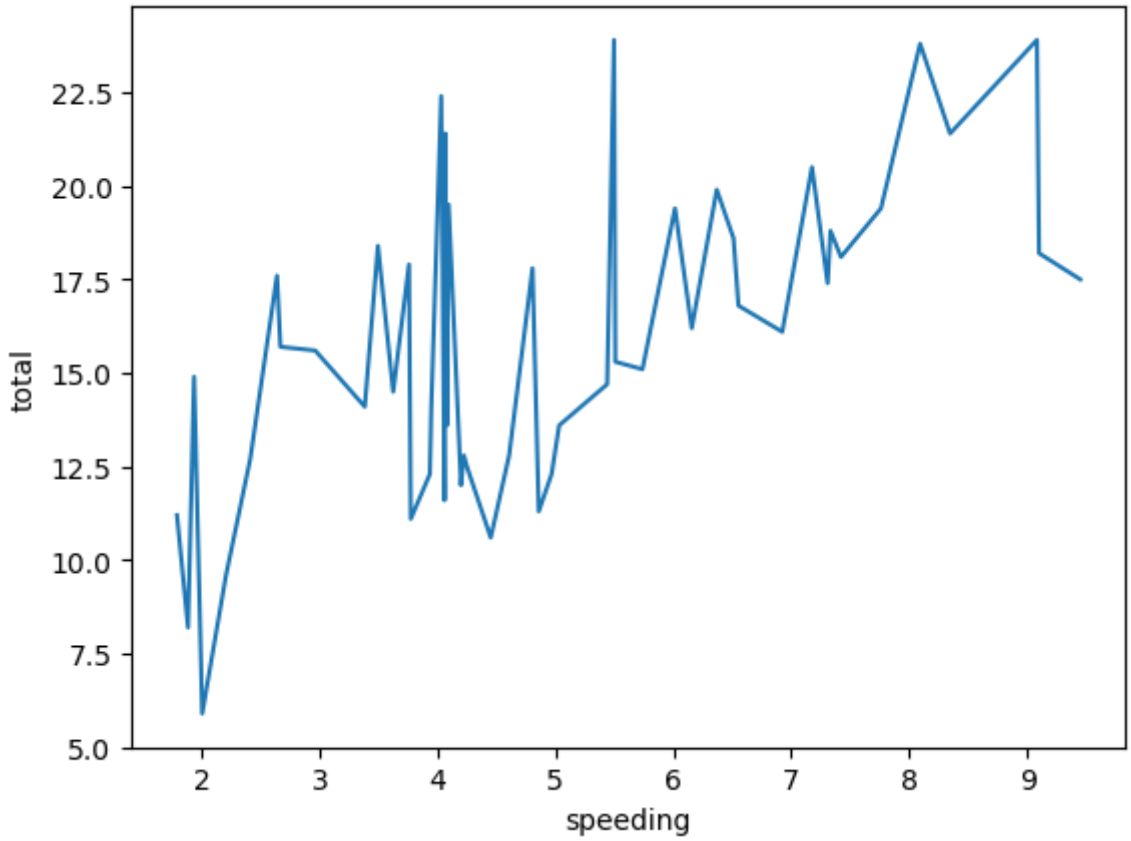
2)lineplot

Lineplot is a plot used to draw a relationship between x and y with the possibility of several semantic groupings.

In [12]:
```
sns.lineplot(x="speeding",y="total",data=df,ci=None)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_7984\899624076.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(x="speeding",y="total",data=df,ci=None)
```

Out[12]:
```
<Axes: xlabel='speeding', ylabel='total'>
```

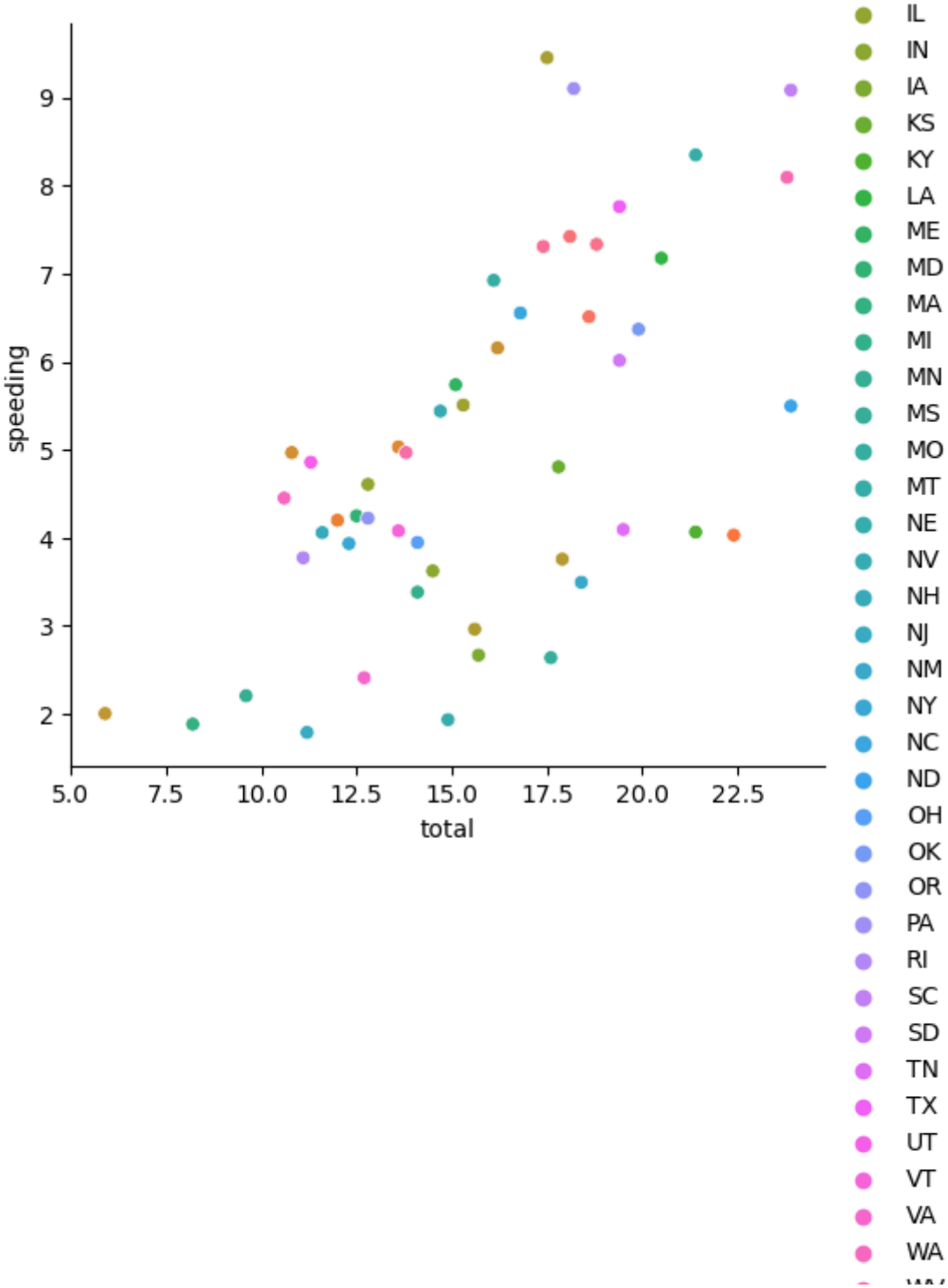


3)replot

This function provides access to several different axes-level functionsthat show the relationship between two variables with semantic mappings of subsets.

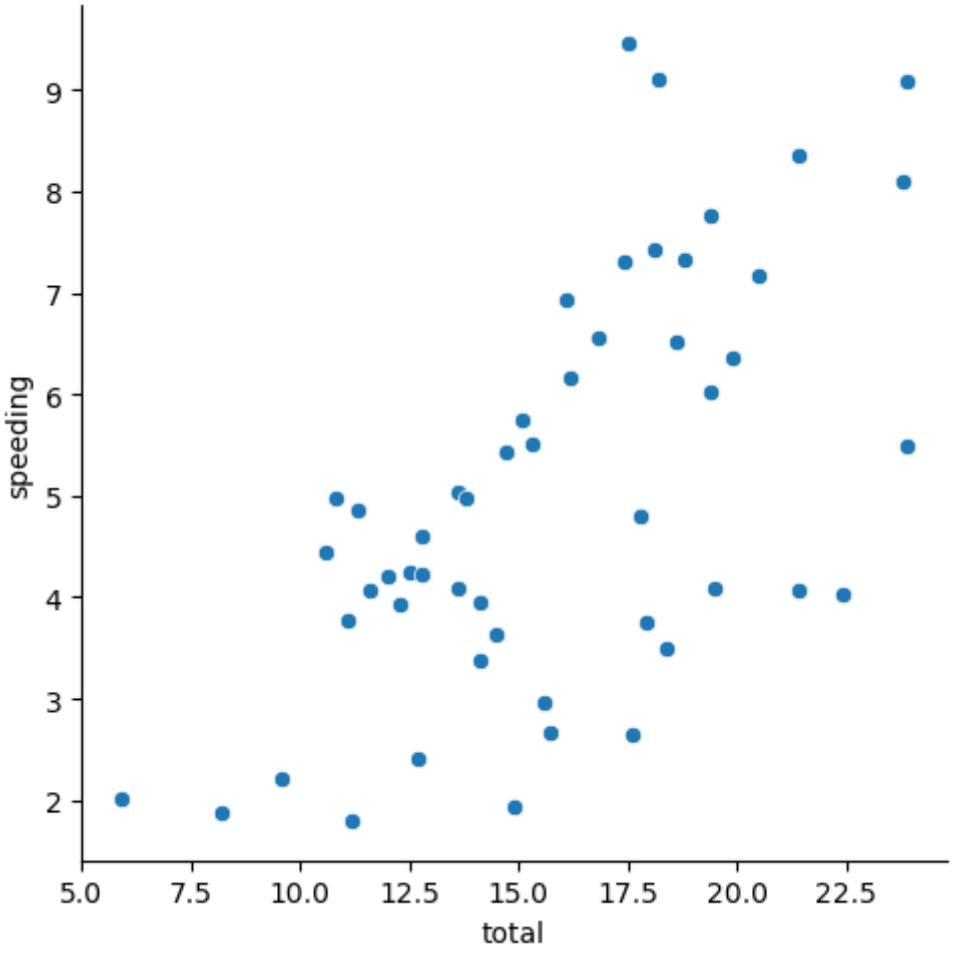
In [14]:
```
sns.relplot(x="total",y="speeding",data=df,hue="abbrev")
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x1ddc856c490>
```


speeding
9
8
7
6
5
4
3
2
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
total
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
SD
TN
TX
UT
VT
VA
WA

In [15]:
```
sns.relplot(x="total",y="speeding",data=df)
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x1ddc8821b90>
```



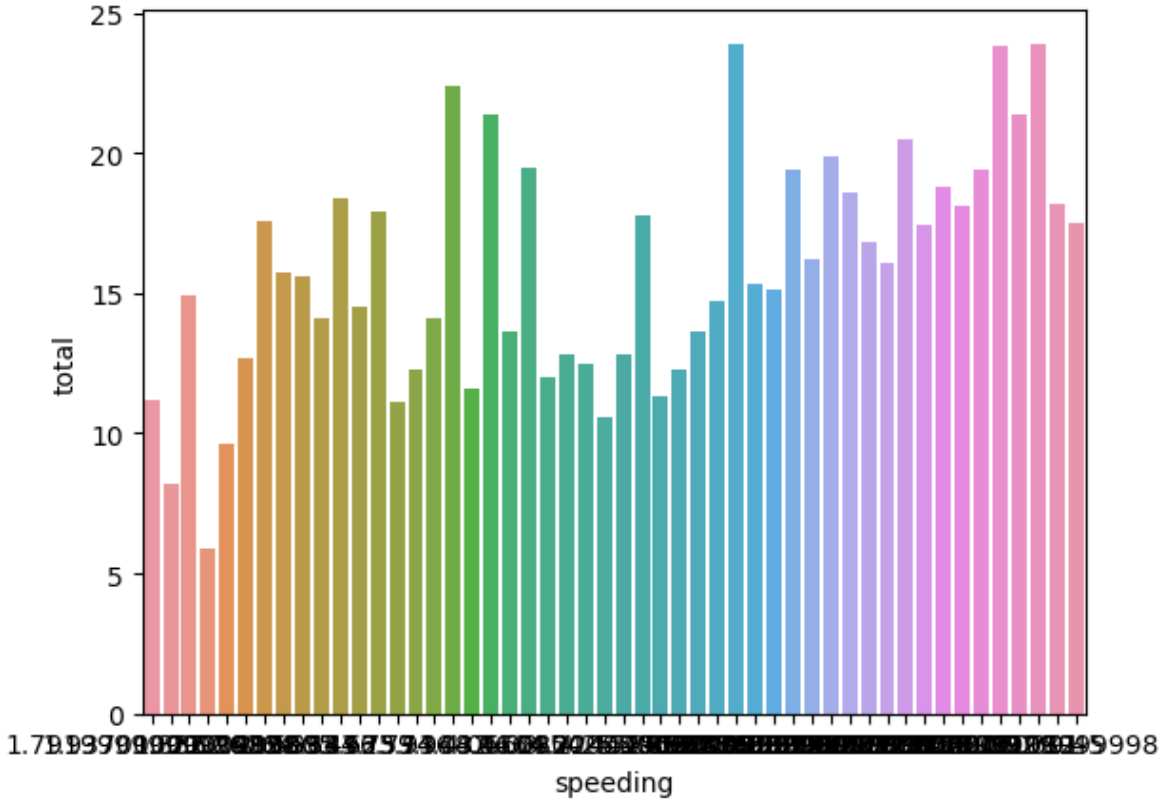
4)Barplot

Barplot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.

In [17]:
```
sns.barplot(data=df,x="speeding",y="total",ci=None)
```

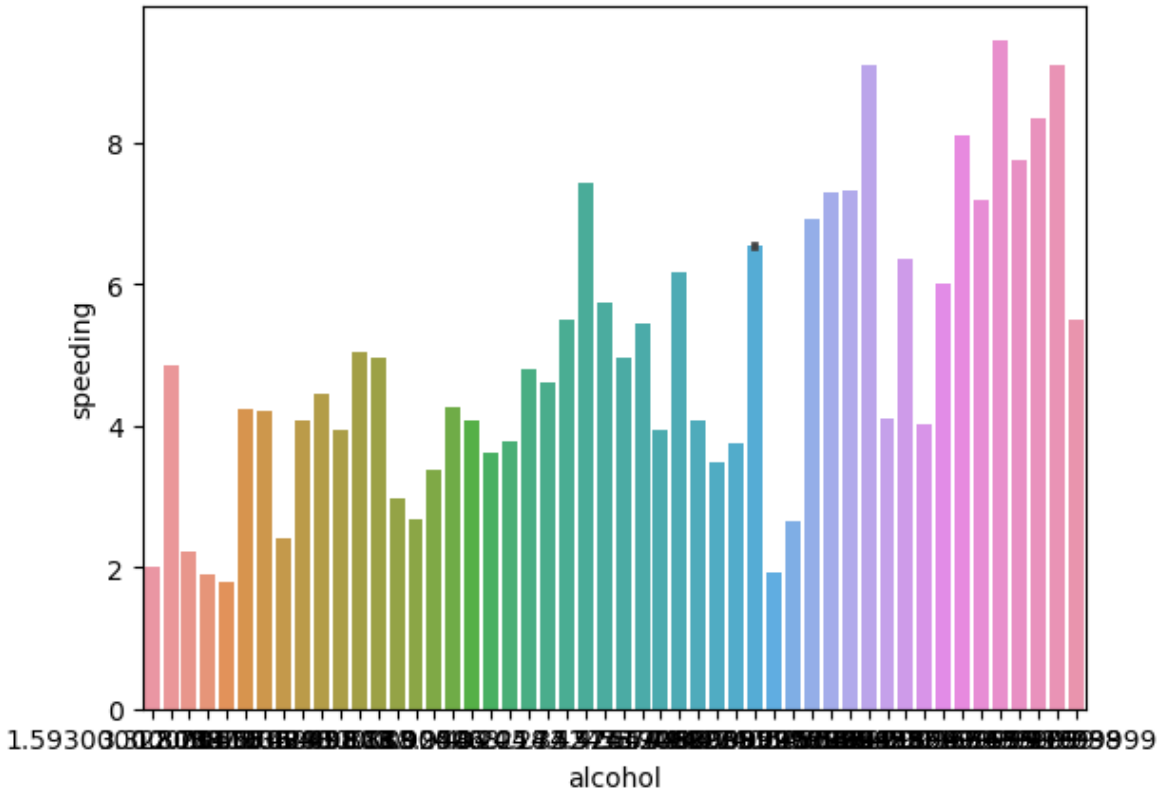
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_7984\1368591.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df,x="speeding",y="total",ci=None)
```

Out[17]:
```
<Axes: xlabel='speeding', ylabel='total'>
```



In [18]:
```
sns.barplot(data=df,x="alcohol",y="speeding")
```

Out[18]:
```
<Axes: xlabel='alcohol', ylabel='speeding'>
```

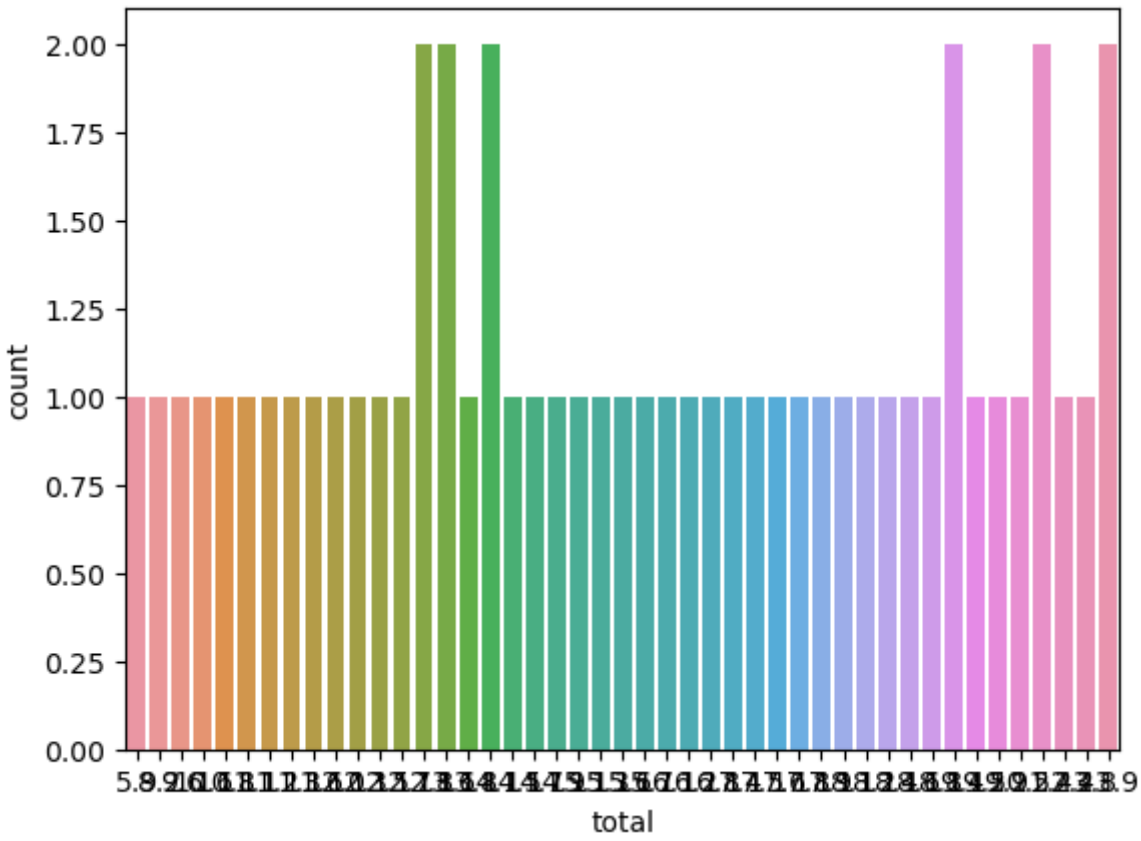


5)Countplot

Count plot used to show the counts of observations in each categorical bin using bars.

In [21]:
```
sns.countplot(x="total",data=df)
```

Out[21]: <Axes: xlabel='total', ylabel='count'>

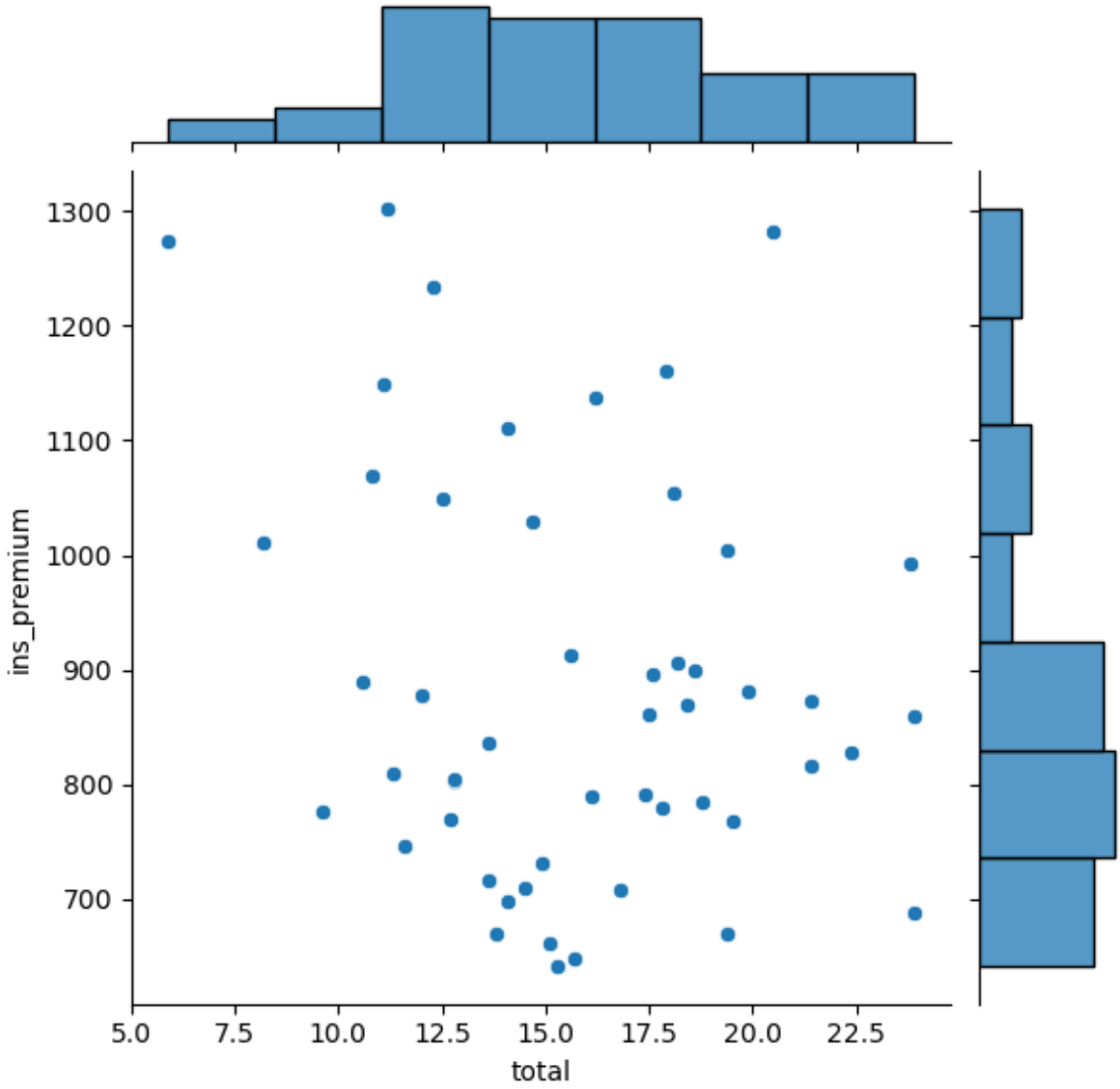


6)Jointplot

The joint plot is a way of understanding the relationship between two variables and the distribution of individuals of each variable.

In [22]:
```
sns.jointplot(x="total",y="ins_premium",data=df)
```

Out[22]: <seaborn.axisgrid.JointGrid at 0x1ddc8a25c90>

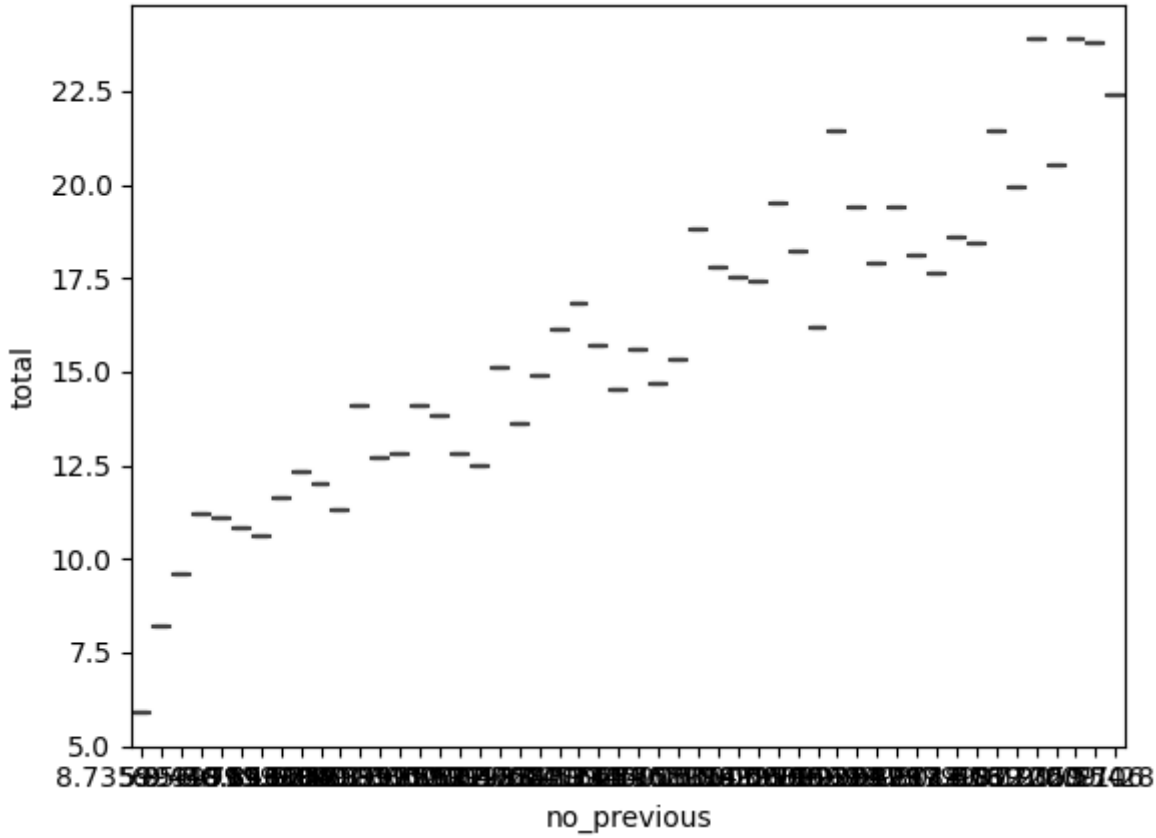


7)Boxplot

A box plot is the visual representation of the depicting groups of numerical data through their quartiles against categorical data.

In [23]:
```
sns.boxplot(x="no_previous",y="total",data=df)
```

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

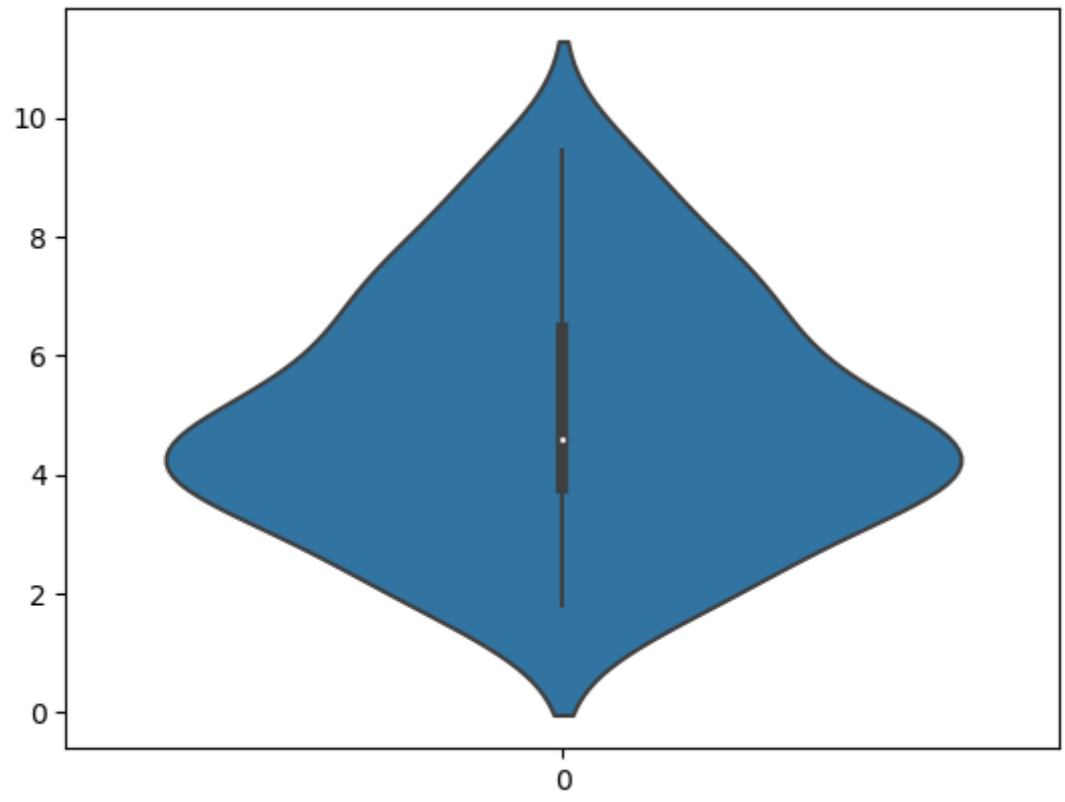


8)Violinplot

A violin plot is similar to a boxplot. It shows several quantitative data across one or more categorical variables such that those distributions can be compared.

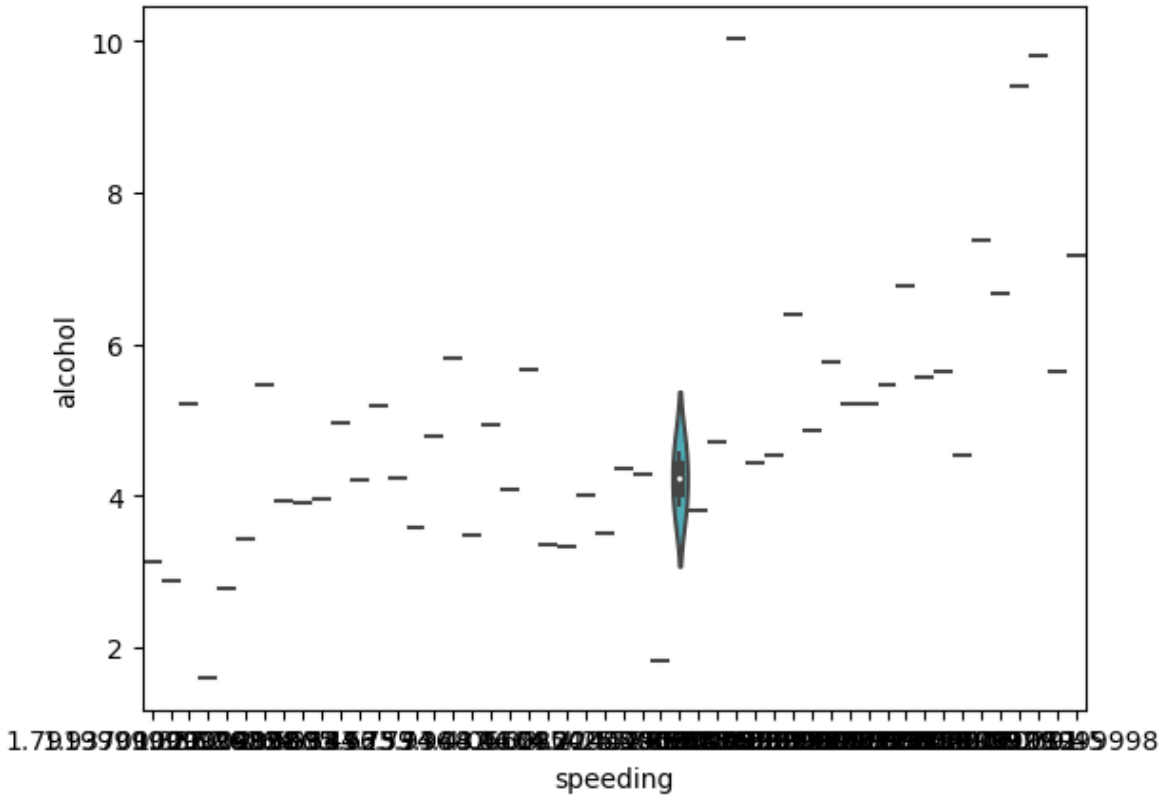
In [25]:
```
sns.violinplot(df['speeding'])
```

Out[25]:
```
<Axes: >
```



In [26]:
```
sns.violinplot(x ="speeding", y ="alcohol",data = df)
```

Out[26]:
```
<Axes: xlabel='speeding', ylabel='alcohol'>
```

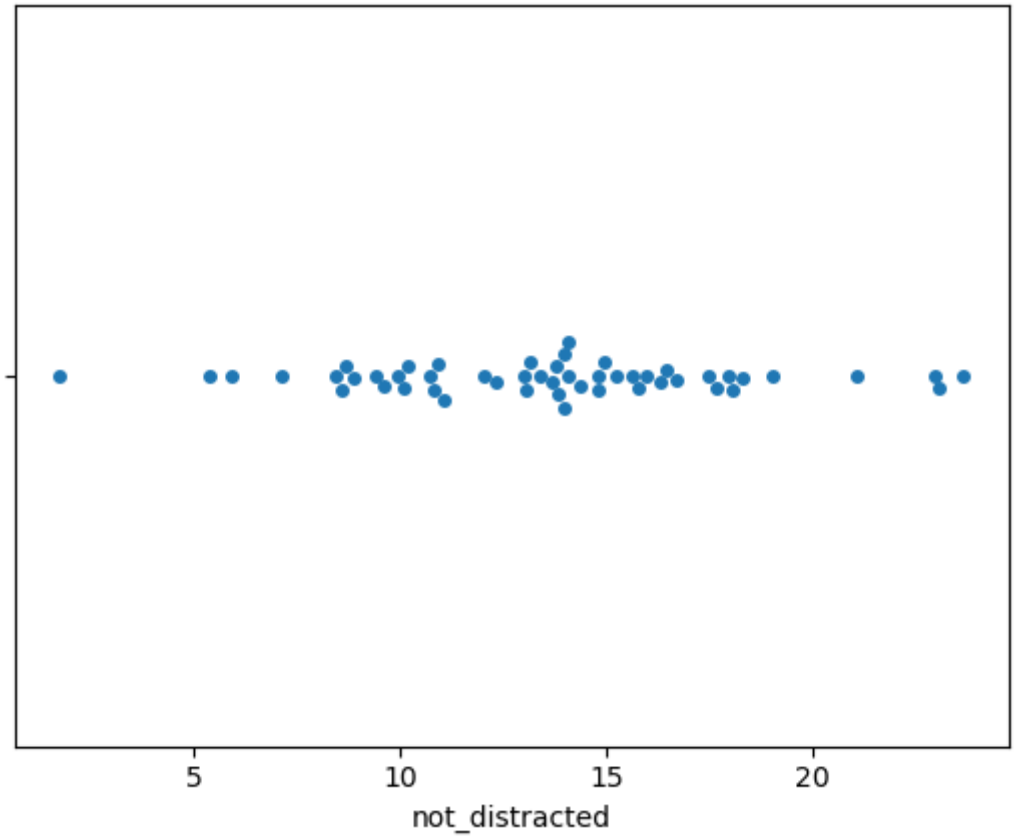


9)Swarmplot

A swarm plot is similar to a strip plot, We can draw a swarm plot with non-overlapping points against categorical data.

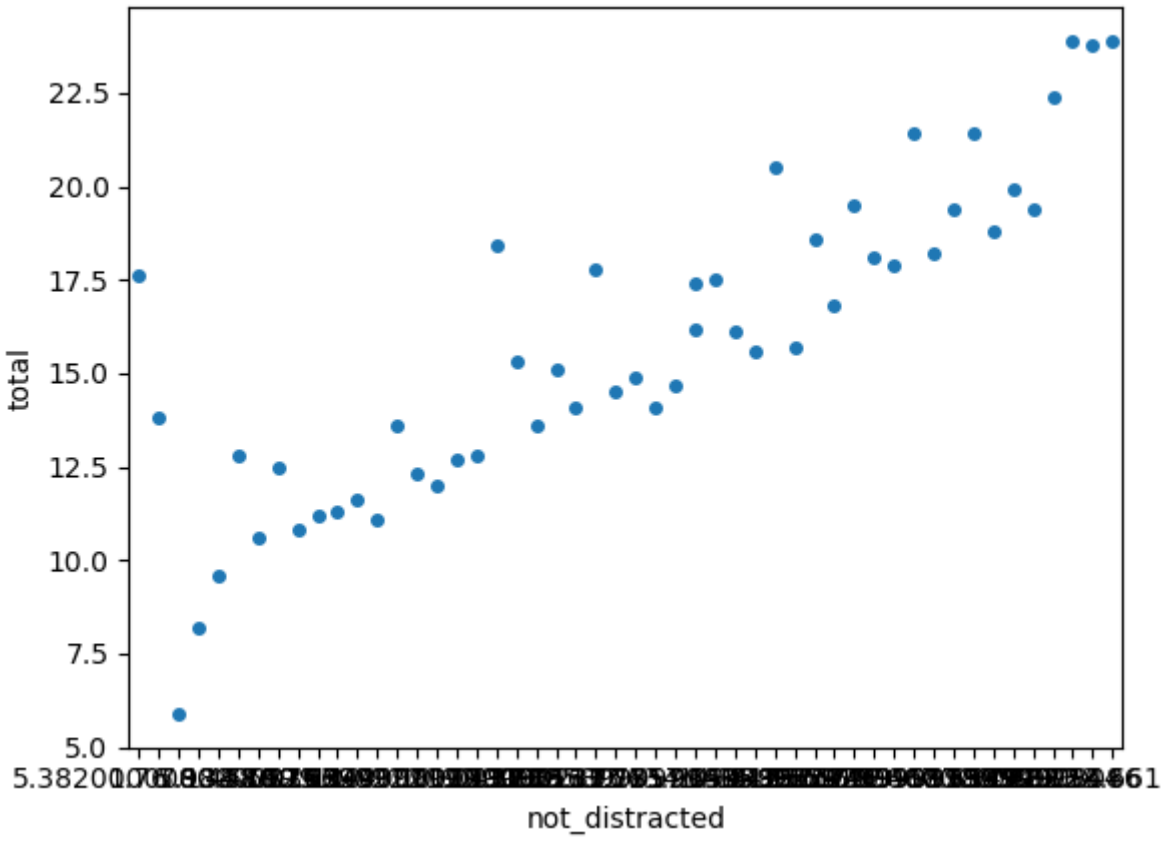
In [27]:
```
sns.swarmplot(x = df["not_distracted"])
```

Out[27]:
```
<Axes: xlabel='not_distracted'>
```



In [28]:
```
sns.swarmplot(x ="not_distracted", y ="total",data = df)
```

Out[28]:
```
<Axes: xlabel='not_distracted', ylabel='total'>
```

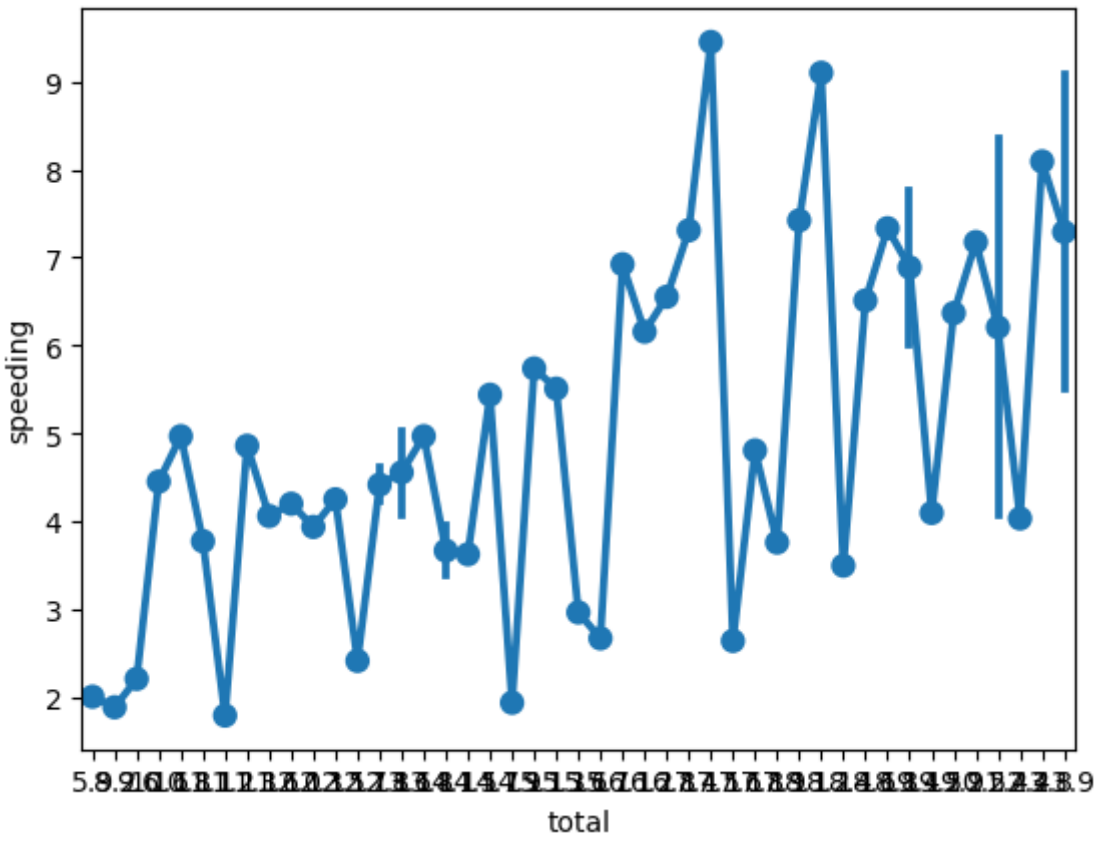


10)Pointplot

Point plot used to show point estimates and confidence intervals using scatter plot glyphs. A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

In [29]:
```
sns.pointplot(x = "total", y = "speeding", data = df)
```

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



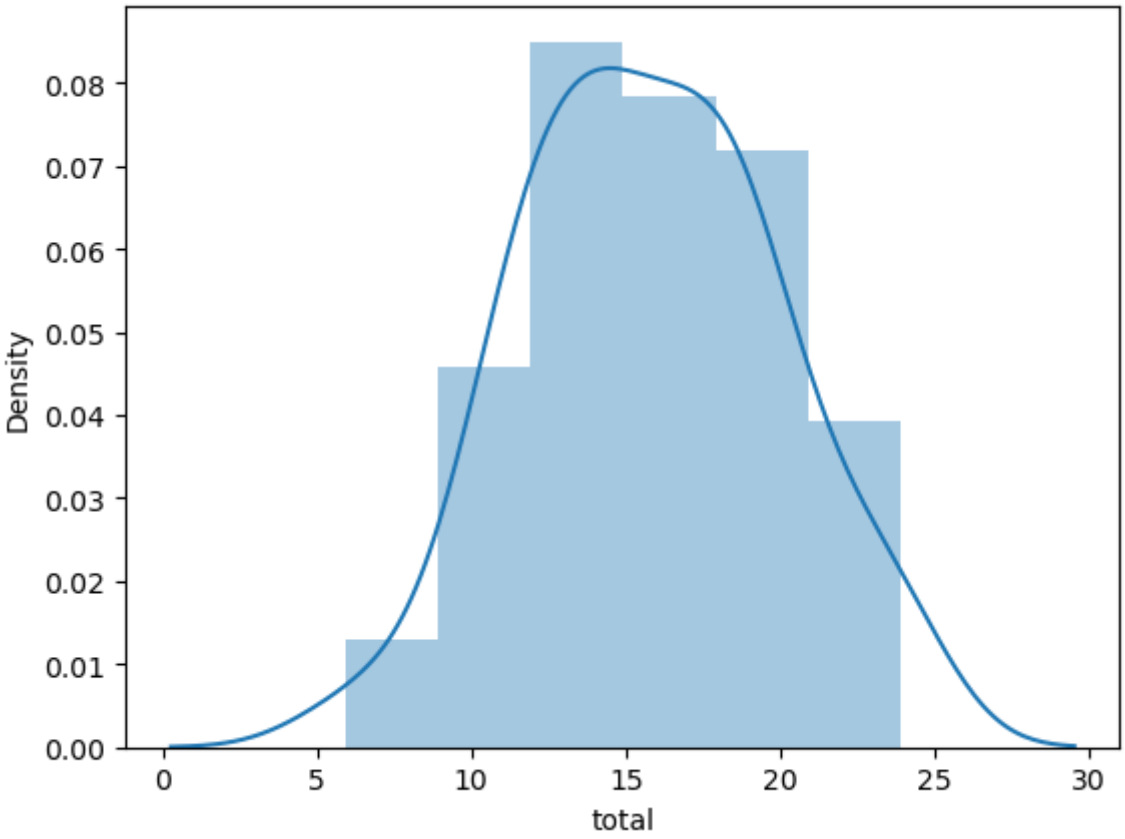
11)Distplot

It is used basically for univariant set of observations and visualizes it through a histogram that is only one observation and hence we choose one particular column of the dataset.

In [30]:
```
sns.distplot( df['total'])
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_7984\1867420185.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['total'])
```

Out[30]:
```
<Axes: xlabel='total', ylabel='Density'>
```


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
0
5
10
15
20
25
30
total

In [ ]: