GAYATRI MOITRA
ASSIGNMENT 2

DATA VISUALISATION TASKS

1.Take car crashes dataset from seaborn library

2.load the dataset

3.Perfrom Data Visualization

4.Inference is must for each and every graph

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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

Out[ ]:

	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

In [ ]:
```
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 [ ]:
```
df.shape
```

Out[ ]:
```
(51, 8)
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
total             0
speeding          0
alcohol           0
not_distracted    0
no_previous       0
ins_premium       0
ins_losses        0
abbrev            0
dtype: int64
```

In [ ]:
```
df.describe()
```

Out[ ]:

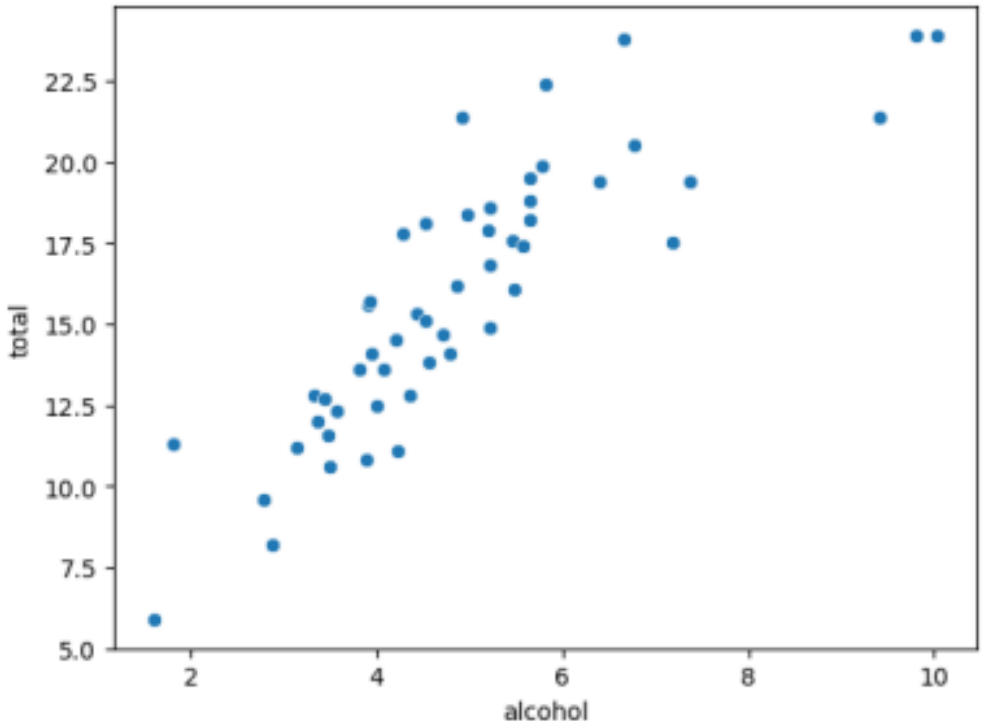
	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses
count	51.000000	51.000000	51.000000	51.000000	51.000000	51.000000	51.000000
mean	15.790196	4.998196	4.886784	13.573176	14.004882	886.957647	134.493137
std	4.122002	2.017747	1.729133	4.508977	3.764672	178.296285	24.835922
min	5.900000	1.792000	1.593000	1.760000	5.900000	641.960000	82.750000
25%	12.750000	3.766500	3.894000	10.478000	11.348000	768.430000	114.645000
50%	15.600000	4.608000	4.554000	13.857000	13.775000	858.970000	136.050000
75%	18.500000	6.439000	5.604000	16.140000	16.755000	1007.945000	151.870000
max	23.900000	9.450000	10.038000	23.661000	21.280000	1301.520000	194.780000

In [ ]:
```
#total -> Number of drivers involved in fatal collisions per billion miles
#speeding -> Percentage Of Drivers Involved In Fatal Collisions Who Were Speeding
#alcohol -> Percentage Of Drivers Involved In Fatal Collisions Who Were Alcohol-Imp
#not_distracted -> Percentage Of Drivers Involved In Fatal Collisions Who Were Not
#no_previous -> Percentage Of Drivers Involved In Fatal Collisions Who Had Not Beer
#ins_premium -> Car Insurance Premiums
#ins_losses -> Losses incurred by insurance companies for collisions per insured dr
```

Scatterplot

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

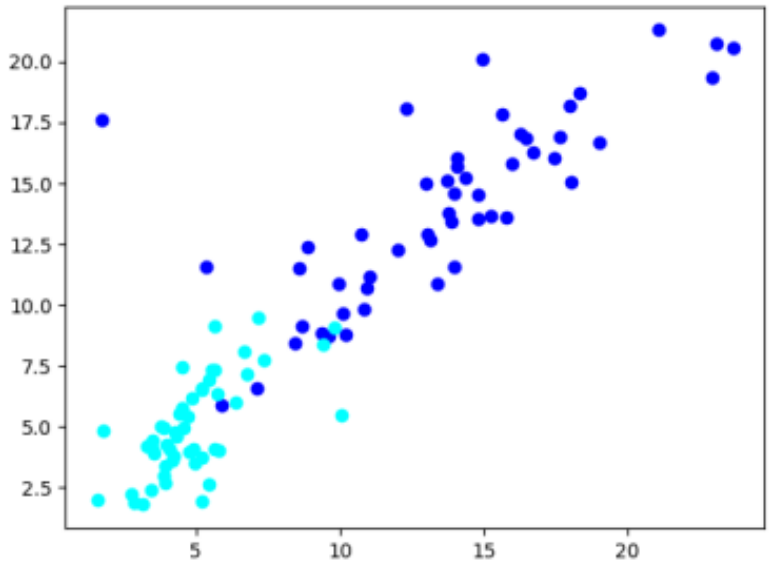
Out[ ]:
```
<Axes: xlabel='alcohol', ylabel='total'>
```



Inference: From the scatterplot it can be stated that when Percentage Of Drivers Involved In Fatal Collisions Who Were Alcohol-Impaired increases , Number of drivers involved in fatal collisions per billion miles also increases.

Comparing two plots

In [ ]:
```
plt.scatter(x="not_distracted",y="no_previous",data=df,color="blue")
plt.scatter(x="alcohol",y="speeding",data=df,color="cyan")
plt.show()
```



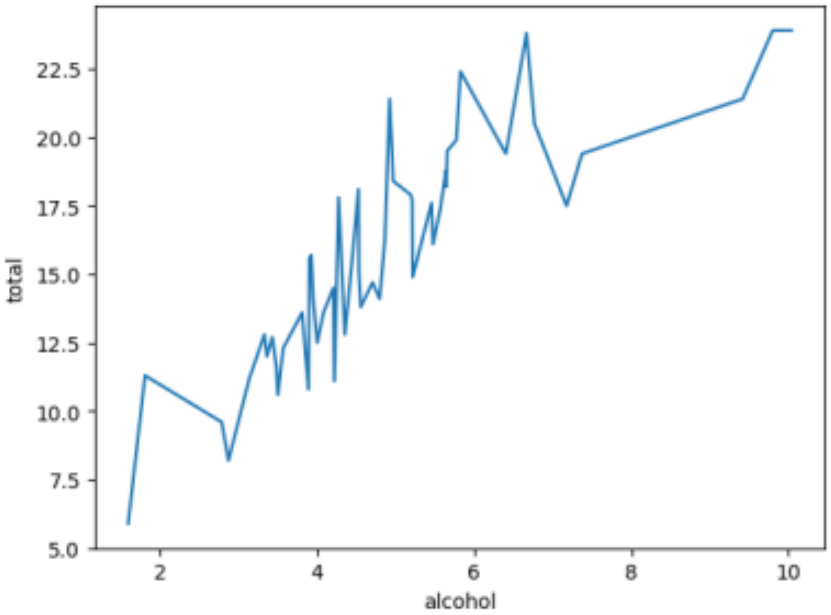
Inference: Comparing two scatterplots, one with accidents caused due to speeding and consumption of alcohol and two with accidents caused when driver was not distracted and drivers having no previous accident records.

Lineplot

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

```
<ipython-input-24-10da8a41c1e4>:1: FutureWarning:

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

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

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



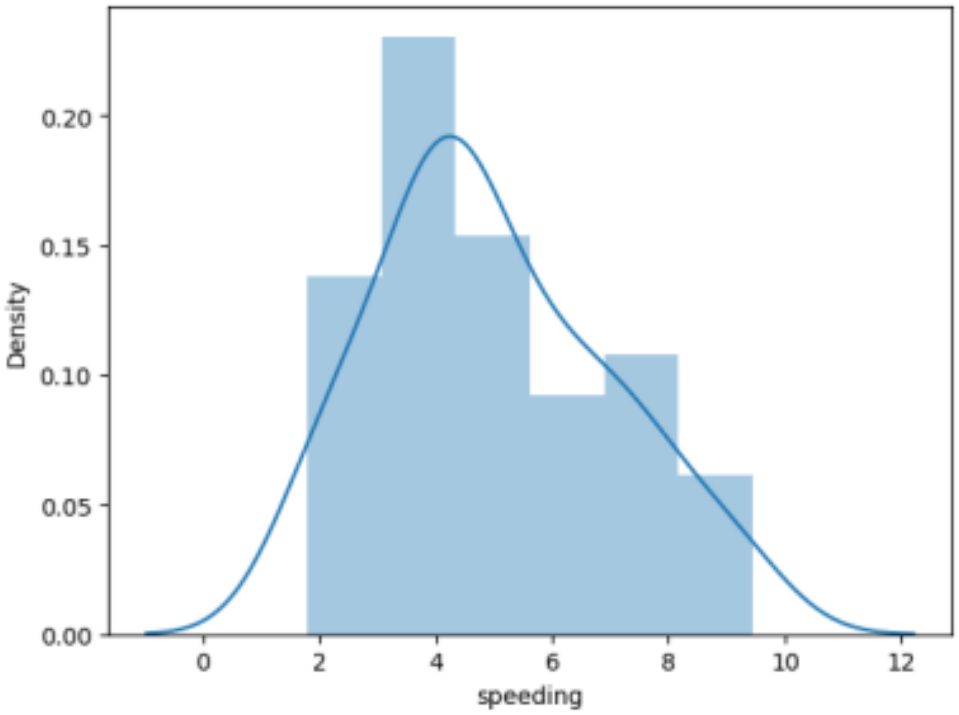
Inference: From the barplot it can be stated that as the number of alcohol impaired driver increases the total number od car crashes gradually increases.

Distribution plot

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

```
<ipython-input-25-8ecb7fd34a3c>: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"])
```

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

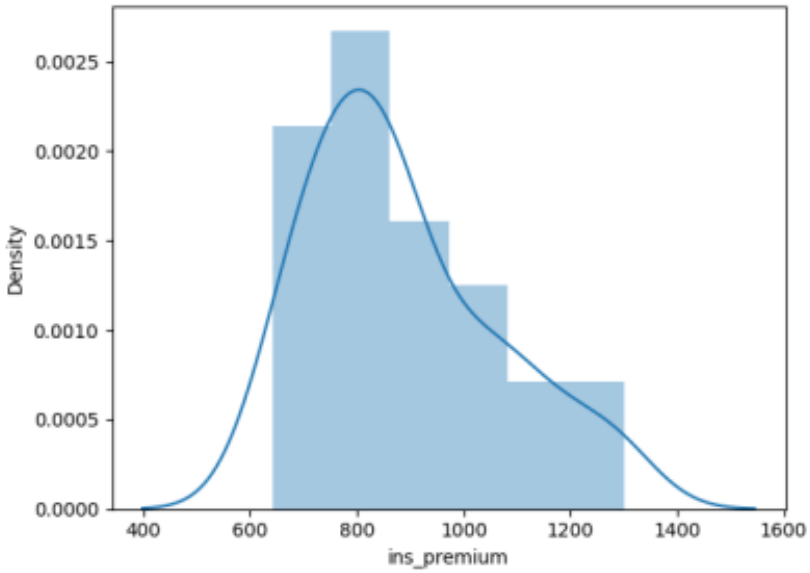


Inference: The above distribution plot shows the normal distribution of the variable "speeding" of the dataset.

In [ ]:
```
sns.distplot(df["ins_premium"])
```

```
<ipython-input-26-8677a75b2d6c>: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["ins_premium"])
```

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

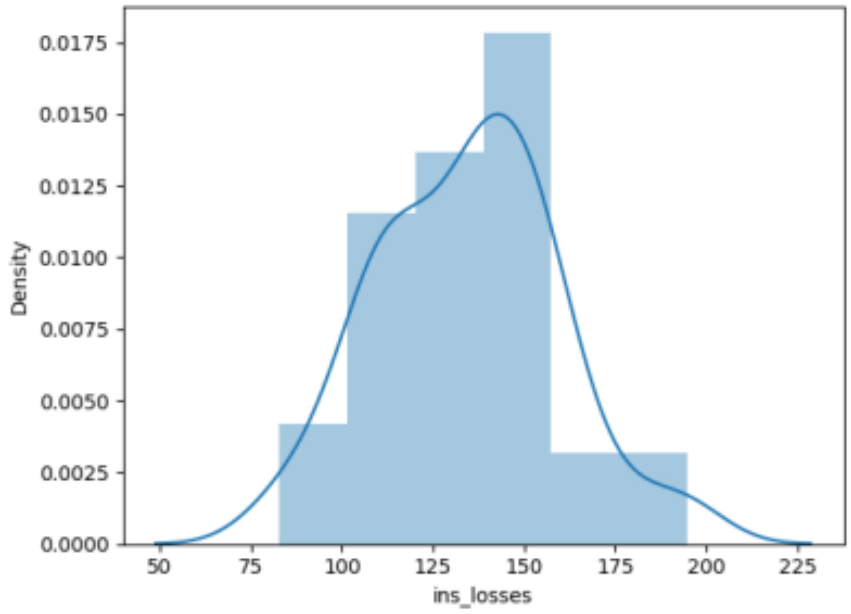


Inference: The above distribution plot shows the normal distribution of the variable "ins_premium" of the dataset.

In [ ]:
```
sns.distplot(df["ins_losses"])
```

```
<ipython-input-27-46fdb0fb15ea>: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["ins_losses"])
```

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

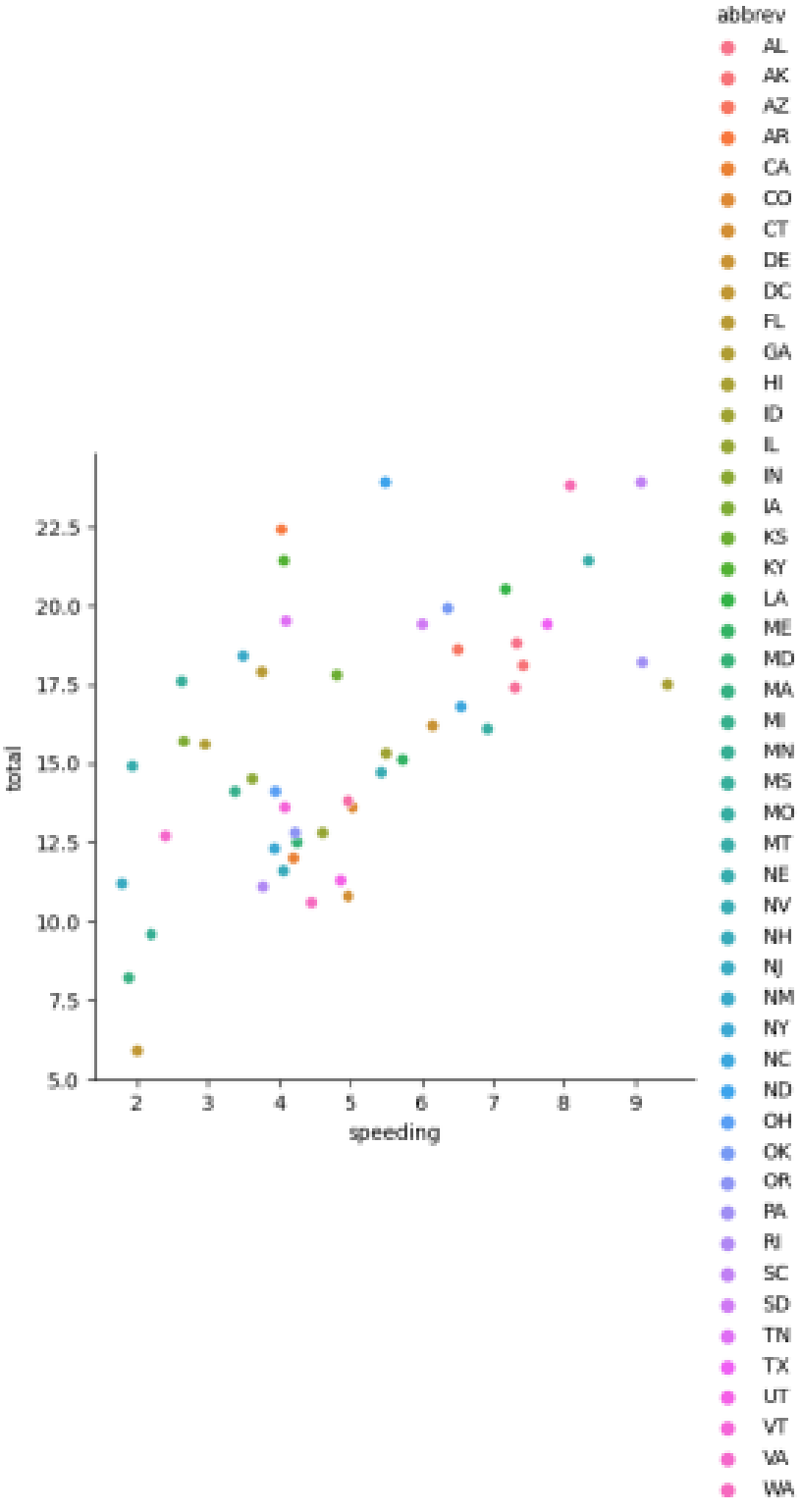


Inference: The above distribution plot shows the normal distribution of the variable "ins_losses" of the dataset.

Relation plot

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

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x79fe03303af0>
```



Inference: The relation plot depicts the proportional relation between crashes cause by high speeding and total crashes via different colour highlighting of the states.

In [ ]:
```
df["abbrev"].value_counts()
```

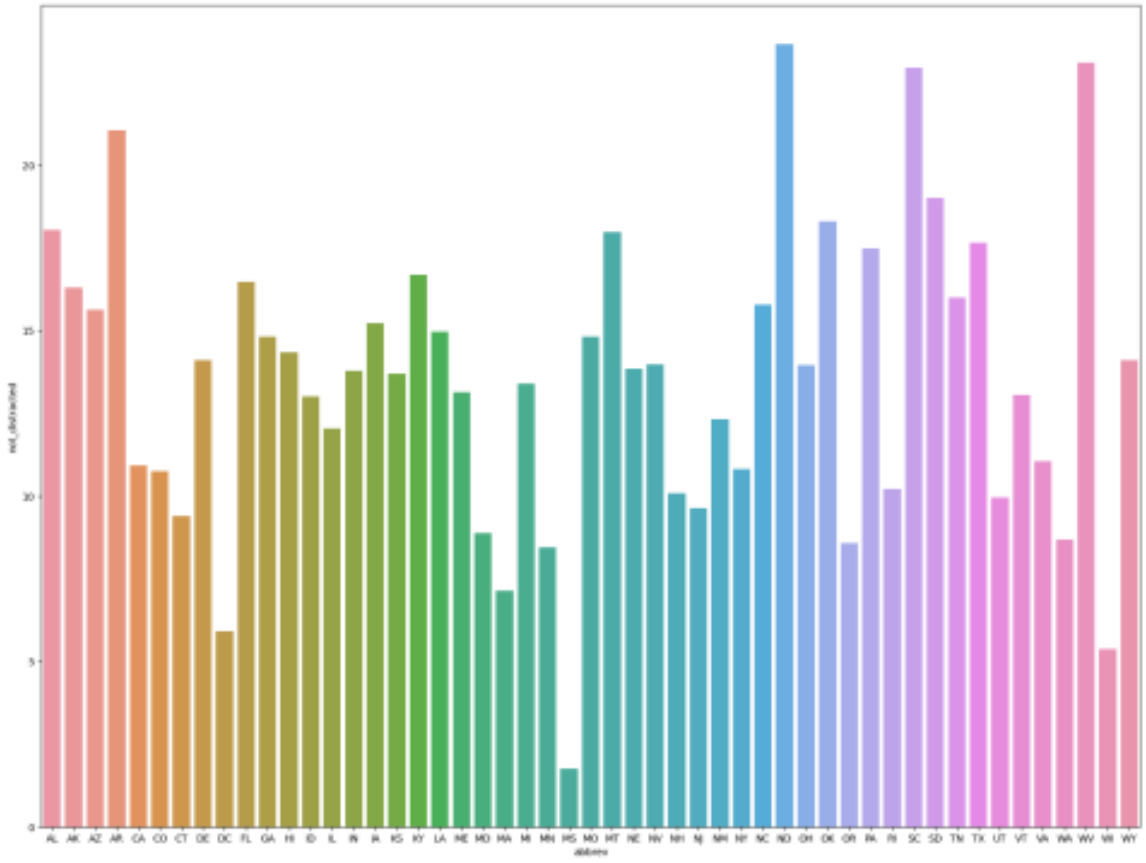
Out[ ]:
```
AL    1
PA    1
NV    1
NH    1
NJ    1
NM    1
NY    1
NC    1
ND    1
OH    1
OK    1
OR    1
RI    1
MT    1
SC    1
SD    1
TN    1
TX    1
UT    1
VT    1
VA    1
WA    1
WV    1
WI    1
NE    1
MD    1
AK    1
ID    1
AZ    1
AR    1
CA    1
CO    1
CT    1
DE    1
DC    1
FL    1
GA    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
```

Barplot

In [ ]:
```
plt.subplots(figsize=(20,15))
sns.barplot(x="abbrev",y="not_distracted",data=df,ci=None)
```

```
<ipython-input-42-e3432450d5c5>:2: FutureWarning:

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

  sns.barplot(x="abbrev",y="not_distracted",data=df,ci=None)
```

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

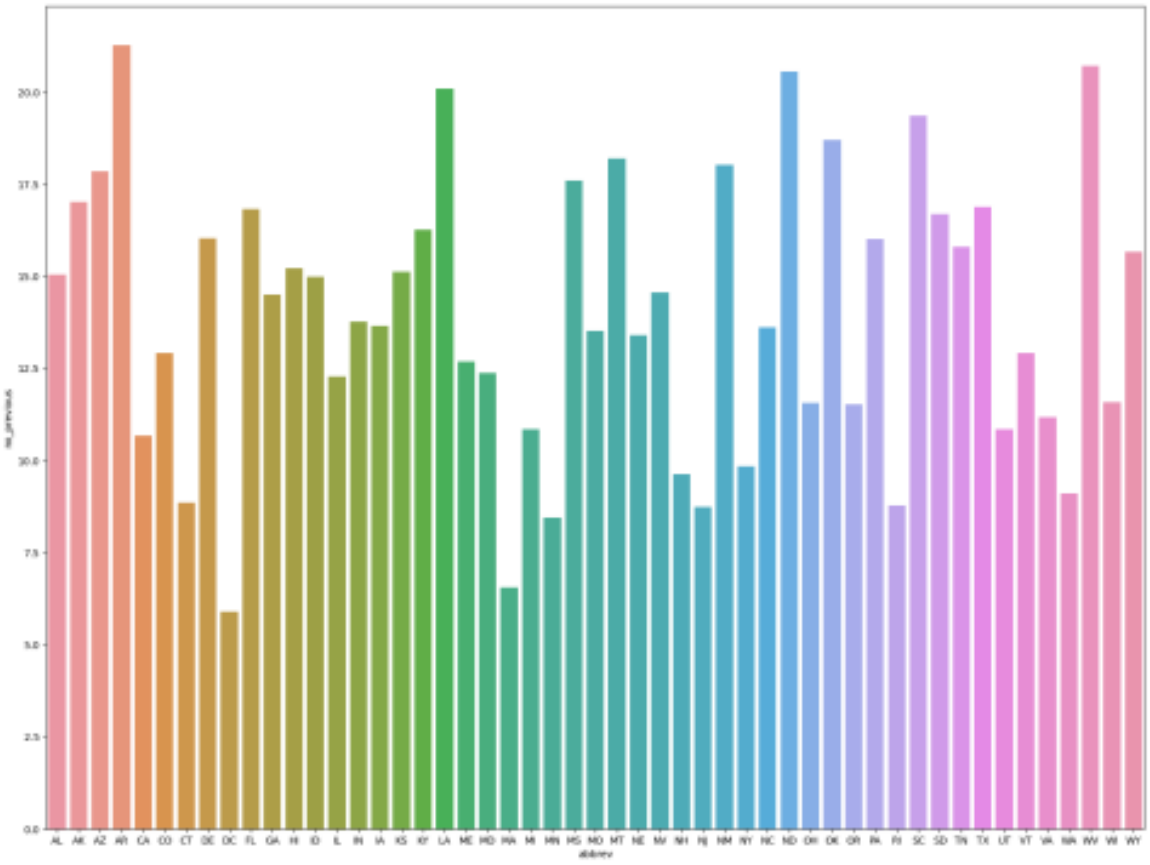


Inference: From the barplot it can be stated that the state abbreviated as ND has maximum Percentage Of Drivers Involved In Fatal Collisions Who Were Not Distracted and the state abbreviated MS has minimum Percentage Of Drivers Involved In Fatal Collisions Who Were Not Distracted.

In [ ]:
```
plt.subplots(figsize=(20,15))
sns.barplot(x="abbrev",y="no_previous",data=df,ci=None)
```

```
<ipython-input-43-1cfdfc35f9a5>:2: FutureWarning:

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

  sns.barplot(x="abbrev",y="no_previous",data=df,ci=None)
```

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

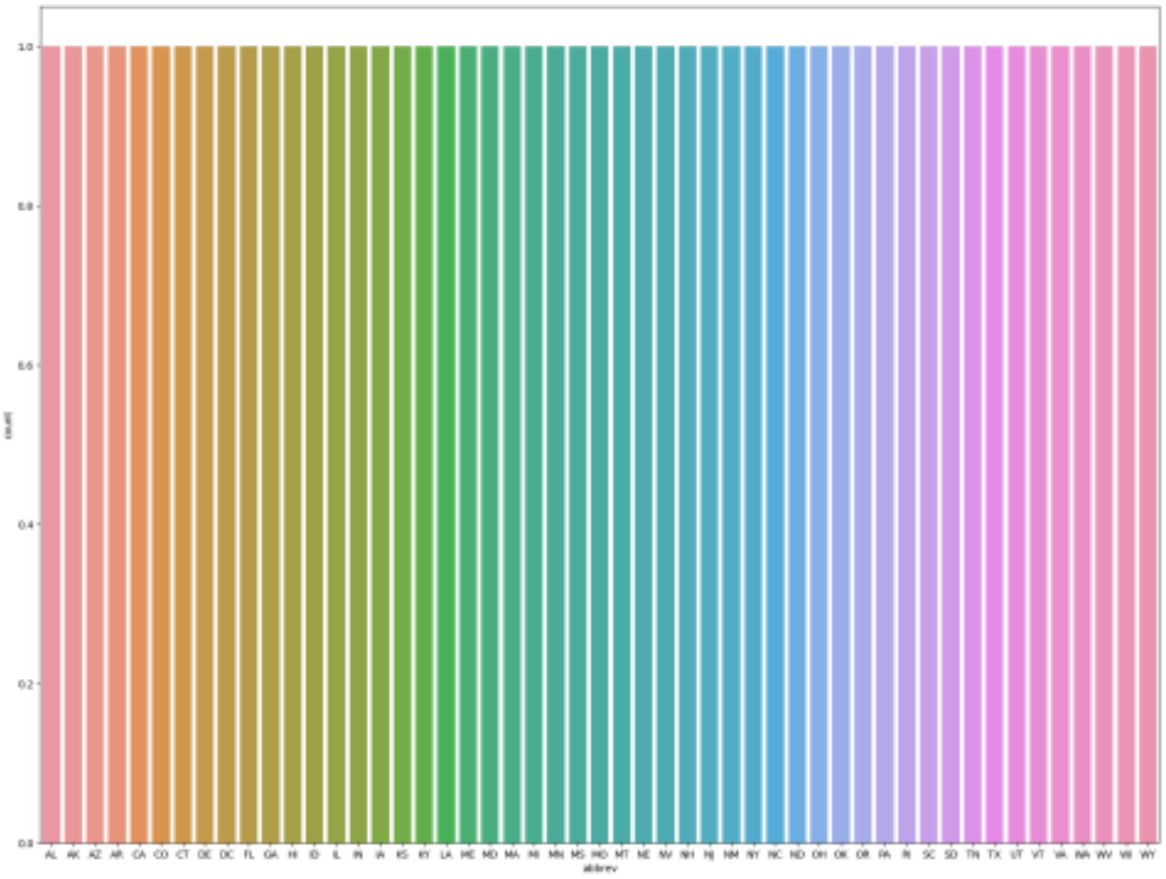


Inference: From the barplot it can be stated that the state abbreviated as AR has maximum Percentage Of Drivers Involved In Fatal Collisions Who Had Not Been Involved In Any Previous Accidents and the state abbreviated DC has minimum Percentage Of Drivers Involved In Fatal Collisions Who Had Not Been Involved In Any Previous Accidents.

Countplot

In [ ]:
```
plt.subplots(figsize=(20,15))
sns.countplot(x="abbrev",data=df)
```

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

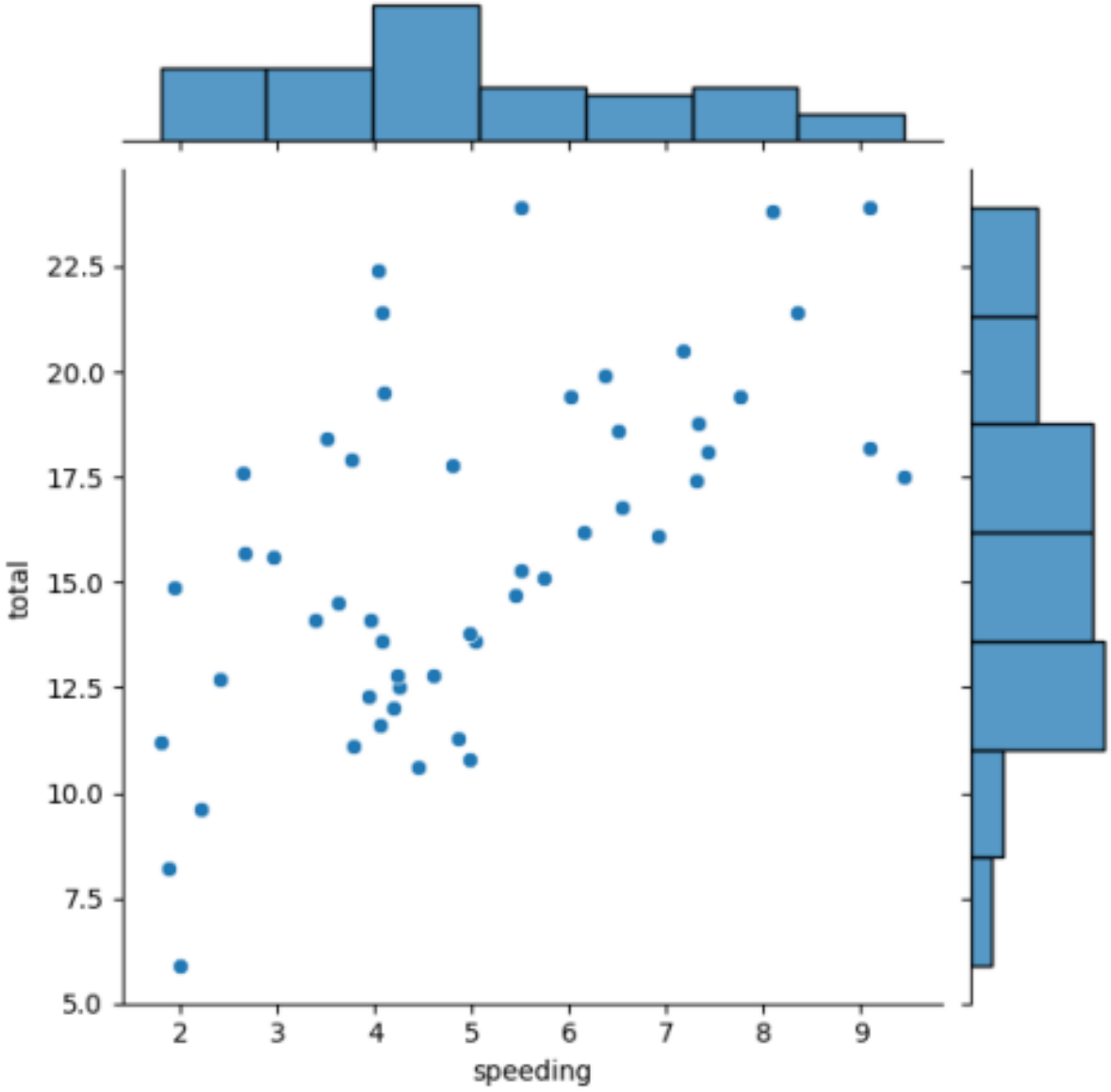


Inference: The above Countplot shows the count of occurence of the states' abbreviation in the dataset.

Jointplot

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

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x79fe00dd3c40>
```



Inference: The relation between car crashes caused by speeding and total number of car crashes is depicted together in the jointplot.

Correlation and heatmap

In [ ]:
```
cor=df.corr()
cor
```

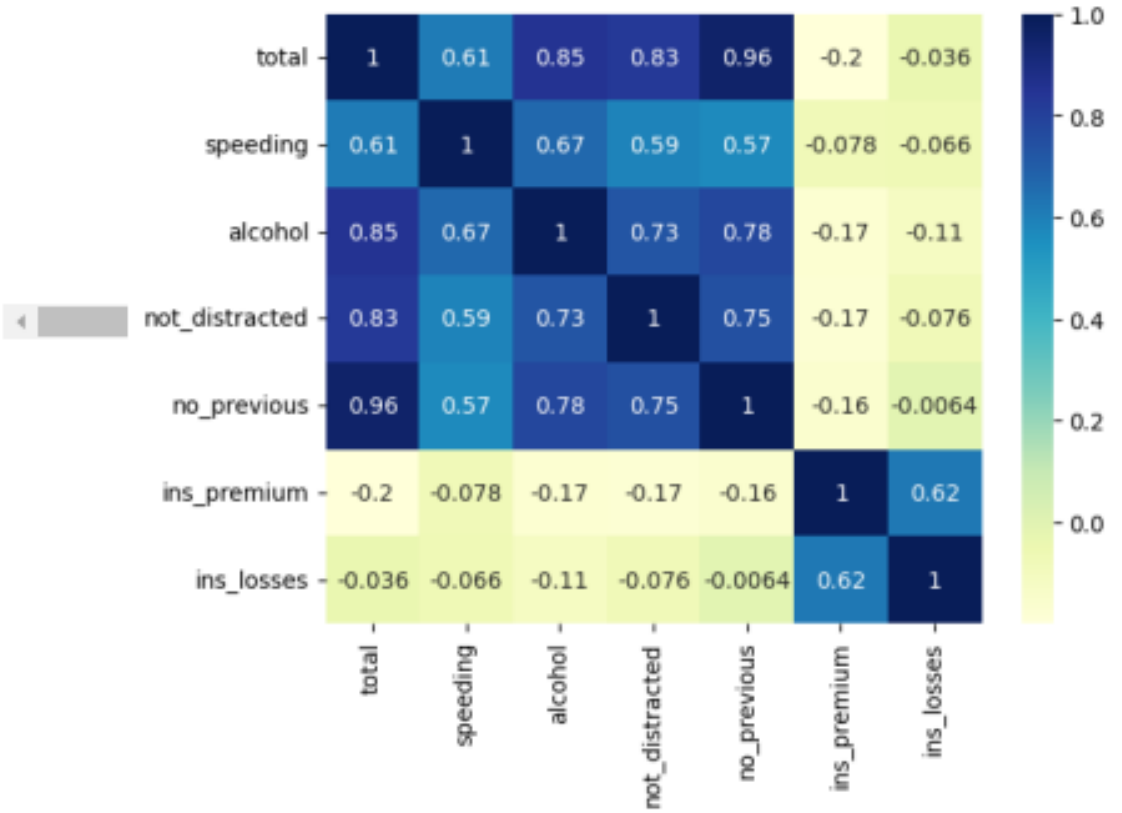
```
<ipython-input-52-7a446f931109>:1: FutureWarning: The default value of numeric_onl
y in DataFrame.corr is deprecated. In a future version, it will default to False.
Select only valid columns or specify the value of numeric_only to silence this war
ning.
  cor=df.corr()
```

Out[ ]:

	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losse
total	1.000000	0.611548	0.852613	0.827560	0.956179	-0.199702	-0.03601
speeding	0.611548	1.000000	0.669719	0.588010	0.571976	-0.077675	-0.06592
alcohol	0.852613	0.669719	1.000000	0.732816	0.783520	-0.170612	-0.11254
not_distracted	0.827560	0.588010	0.732816	1.000000	0.747307	-0.174856	-0.07597
no_previous	0.956179	0.571976	0.783520	0.747307	1.000000	-0.156895	-0.00635
ins_premium	-0.199702	-0.077675	-0.170612	-0.174856	-0.156895	1.000000	0.62311
ins_losses	-0.036011	-0.065928	-0.112547	-0.075970	-0.006359	0.623116	1.00000

In [ ]:
```
sns.heatmap(cor,annot=True,cmap="YlGnBu")
```

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



Inference: The heatmap shows dependencies between two variables via correlation coefficients.