SEABORN

Seaborn is an amazing data visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of the matplotlib library and also closely integrated to the data structures from pandas.

Installing

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

```
Requirement already satisfied: seaborn in c:\users\heman\appdata\roaming\python\p
ython310\site-packages (0.12.1)
Requirement already satisfied: numpy>=1.17 in c:\users\heman\appdata\local\progra
ms\python\python310\lib\site-packages (from seaborn) (1.24.4)
Requirement already satisfied: pandas>=0.25 in c:\users\heman\appdata\roaming\pyt
hon\python310\site-packages (from seaborn) (1.5.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\heman\appdata
\local\programs\python\python310\lib\site-packages (from seaborn) (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\heman\appdata\local\p
rograms\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seabor
n) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\heman\appdata\local\progr
ams\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\heman\appdata\local
\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seabo
rn) (4.33.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\heman\appdata\local
\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seabo
rn) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:\users\heman\appdata\local\pr
ograms\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(21.3)
Requirement already satisfied: pillow>=8 in c:\users\heman\appdata\local\programs
\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.5.
0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\heman\appdata\local\p
rograms\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seabor
n) (3.0.7)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\heman\appdata\loc
al\programs\python\python310\lib\site-packages (from matplotlib!=3.6.1,>=3.1->sea
born) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\heman\appdata\local\progr
ams\python\python310\lib\site-packages (from pandas>=0.25->seaborn) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\heman\appdata\local\programs
\python\python310\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.
```

In [ ]:
```
# Also, make sure you have the following dependencies installed on your computer

# Python 3.6+
# NumPy
# SciPy
# Pandas
# Matplotlib
```

In [ ]:
```
import seaborn as sns
print(sns.get_dataset_names())
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamond
s', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthe
xp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic',
'anagrams', 'anagrams', 'anscombe', 'anscombe', 'attention', 'attention', 'brain_
networks', 'brain_networks', 'car_crashes', 'car_crashes', 'diamonds', 'diamond
s', 'dots', 'dots', 'dowjones', 'dowjones', 'exercise', 'exercise', 'flights', 'f
lights', 'fmri', 'fmri', 'geyser', 'geyser', 'glue', 'glue', 'healthexp', 'health
exp', 'iris', 'iris', 'mpg', 'mpg', 'penguins', 'penguins', 'planets', 'planets',
'seaice', 'seaice', 'taxis', 'taxis', 'tips', 'tips', 'titanic', 'titanic', 'anag
rams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dot
s', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'ir
is', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']
```

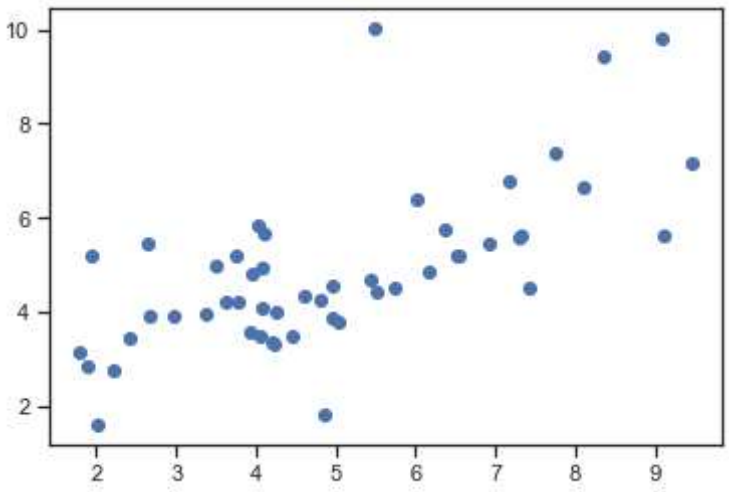
In [ ]:
```
df=sns.load_dataset('car_crashes')
df
```

Out[ ]:

	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses	abb
0	18.8	7.332	5.640	18.048	15.040	784.55	145.08	
1	18.1	7.421	4.525	16.290	17.014	1053.48	133.93	
2	18.6	6.510	5.208	15.624	17.856	899.47	110.35	
3	22.4	4.032	5.824	21.056	21.280	827.34	142.39	
4	12.0	4.200	3.360	10.920	10.680	878.41	165.63	
5	13.6	5.032	3.808	10.744	12.920	835.50	139.91	
6	10.8	4.968	3.888	9.396	8.856	1068.73	167.02	
7	16.2	6.156	4.860	14.094	16.038	1137.87	151.48	
8	5.9	2.006	1.593	5.900	5.900	1273.89	136.05	
9	17.9	3.759	5.191	16.468	16.826	1160.13	144.18	
10	15.6	2.964	3.900	14.820	14.508	913.15	142.80	
11	17.5	9.450	7.175	14.350	15.225	861.18	120.92	
12	15.3	5.508	4.437	13.005	14.994	641.96	82.75	
13	12.8	4.608	4.352	12.032	12.288	803.11	139.15	
14	14.5	3.625	4.205	13.775	13.775	710.46	108.92	
15	15.7	2.669	3.925	15.229	13.659	649.06	114.47	
16	17.8	4.806	4.272	13.706	15.130	780.45	133.80	
17	21.4	4.066	4.922	16.692	16.264	872.51	137.13	
18	20.5	7.175	6.765	14.965	20.090	1281.55	194.78	
19	15.1	5.738	4.530	13.137	12.684	661.88	96.57	
20	12.5	4.250	4.000	8.875	12.375	1048.78	192.70	I
21	8.2	1.886	2.870	7.134	6.560	1011.14	135.63	I
22	14.1	3.384	3.948	13.395	10.857	1110.61	152.26	
23	9.6	2.208	2.784	8.448	8.448	777.18	133.35	N
24	17.6	2.640	5.456	1.760	17.600	896.07	155.77	
25	16.1	6.923	5.474	14.812	13.524	790.32	144.45	N
26	21.4	8.346	9.416	17.976	18.190	816.21	85.15	
27	14.9	1.937	5.215	13.857	13.410	732.28	114.82	
28	14.7	5.439	4.704	13.965	14.553	1029.87	138.71	
29	11.6	4.060	3.480	10.092	9.628	746.54	120.21	
30	11.2	1.792	3.136	9.632	8.736	1301.52	159.85	
31	18.4	3.496	4.968	12.328	18.032	869.85	120.75	N
32	12.3	3.936	3.567	10.824	9.840	1234.31	150.01	

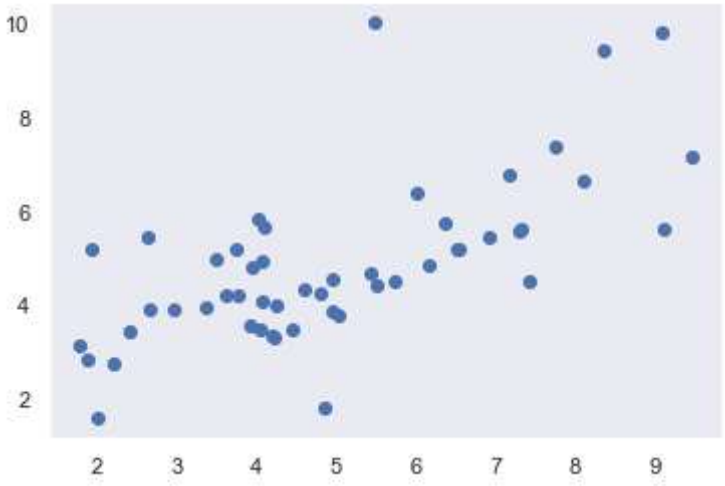
	total	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses	abb
33	16.8	6.552	5.208	15.792	13.608	708.24	127.82	
34	23.9	5.497	10.038	23.661	20.554	688.75	109.72	
35	14.1	3.948	4.794	13.959	11.562	697.73	133.52	
36	19.9	6.368	5.771	18.308	18.706	881.51	178.86	
37	12.8	4.224	3.328	8.576	11.520	804.71	104.61	
38	18.2	9.100	5.642	17.472	16.016	905.99	153.86	
39	11.1	3.774	4.218	10.212	8.769	1148.99	148.58	
40	23.9	9.082	9.799	22.944	19.359	858.97	116.29	
41	19.4	6.014	6.402	19.012	16.684	669.31	96.87	
42	19.5	4.095	5.655	15.990	15.795	767.91	155.57	
43	19.4	7.760	7.372	17.654	16.878	1004.75	156.83	
44	11.3	4.859	1.808	9.944	10.848	809.38	109.48	
45	13.6	4.080	4.080	13.056	12.920	716.20	109.61	
46	12.7	2.413	3.429	11.049	11.176	768.95	153.72	
47	10.6	4.452	3.498	8.692	9.116	890.03	111.62	
48	23.8	8.092	6.664	23.086	20.706	992.61	152.56	
49	13.8	4.968	4.554	5.382	11.592	670.31	106.62	
50	17.4	7.308	5.568	14.094	15.660	791.14	122.04	

In [ ]:
```
# set_style()

from matplotlib import pyplot as plt
plt.scatter(df.speeding,df.alcohol)
sns.set_style("ticks")
plt.show()
```

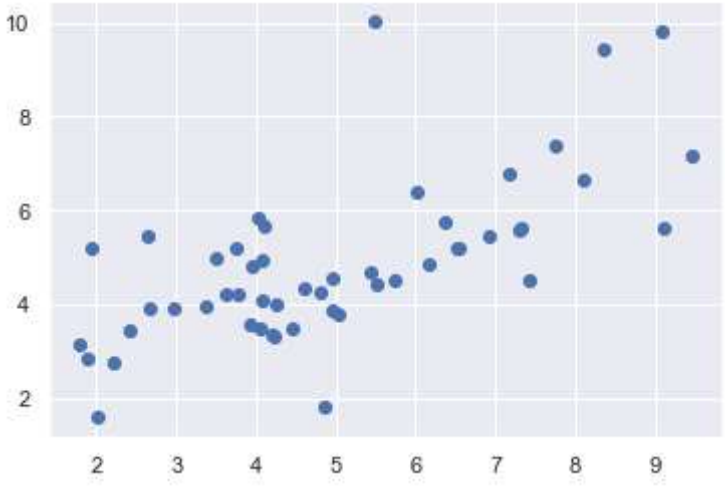


In [ ]:
```
from matplotlib import pyplot as plt
plt.scatter(df.speeding,df.alcohol)
```

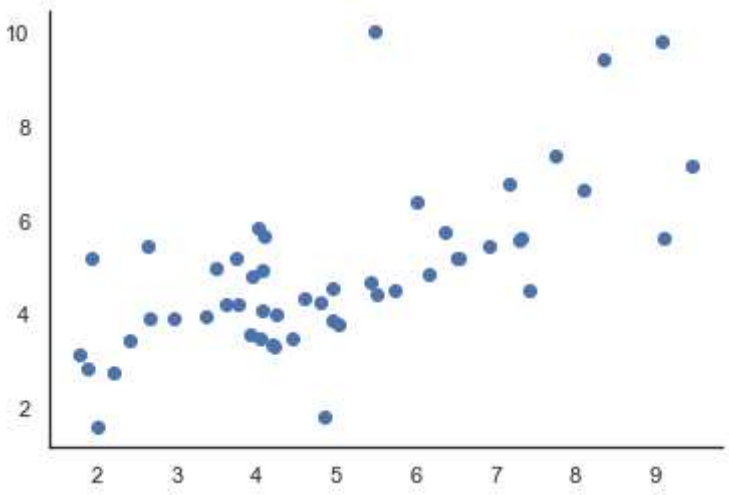
```
sns.set_style("dark")
plt.show()
```



In [ ]:
```
from matplotlib import pyplot as plt
plt.scatter(df.speeding,df.alcohol)
sns.set_style("darkgrid")
plt.show()
```



In [ ]:
```
from matplotlib import pyplot as plt
plt.scatter(df.speeding,df.alcohol)
sns.set_style("white")
#despine
sns.despine()
plt.show()
```

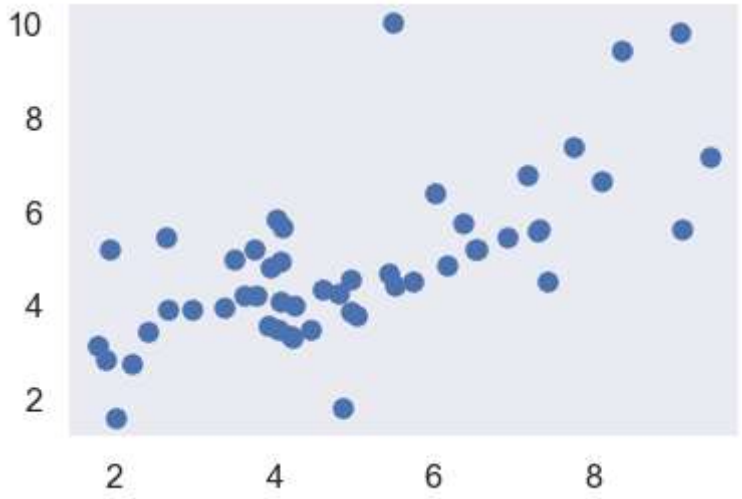


In [ ]:
```
param=sns.axes_style()
param
```

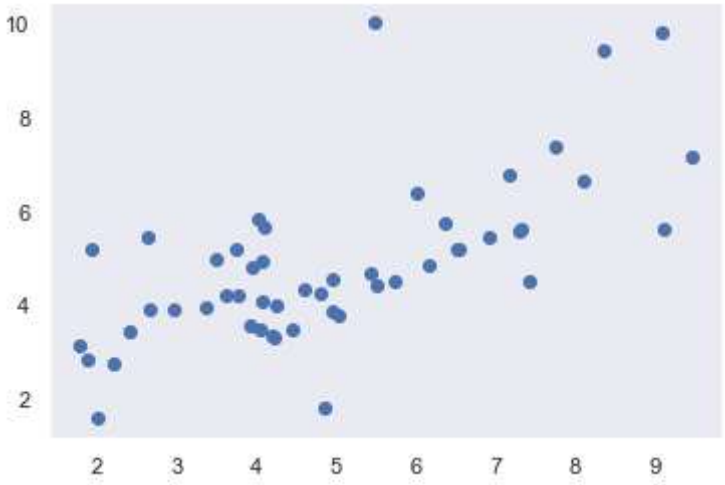
Out[ ]:
```
{'axes.facecolor': '#EAEAF2',
 'axes.edgecolor': 'white',
 'axes.grid': True,
 'axes.axisbelow': True,
 'axes.labelcolor': '.15',
 'figure.facecolor': 'white',
 'grid.color': 'white',
 'grid.linestyle': '-',
 'text.color': '.15',
 'xtick.color': '.15',
 'ytick.color': '.15',
 'xtick.direction': 'out',
 'ytick.direction': 'out',
 'lines.solid_capstyle': <CapStyle.round: 'round'>,
 'patch.edgecolor': 'w',
 'patch.force_edgecolor': True,
 'image.cmap': 'rocket',
 'font.family': ['sans-serif'],
 'font.sans-serif': ['Arial',
  'DejaVu Sans',
  'Liberation Sans',
  'Bitstream Vera Sans',
  'sans-serif'],
 'xtick.bottom': False,
 'xtick.top': False,
 'ytick.left': False,
 'ytick.right': False,
 'axes.spines.left': True,
 'axes.spines.bottom': True,
 'axes.spines.right': True,
 'axes.spines.top': True}
```

set_context() #increase size when go below Paper Notebook Talk Poster

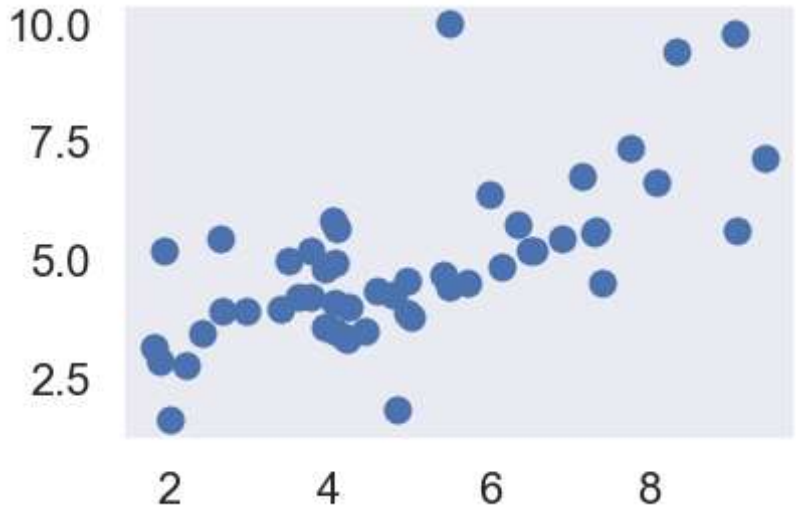
In [ ]:
```
plt.scatter(df.speeding,df.alcohol)
sns.set_style("dark")
sns.set_context("talk")
plt.show()
```



In [ ]:
```
plt.scatter(df.speeding,df.alcohol)
sns.set_style("dark")
sns.set_context("notebook")
plt.show()
```



In [ ]:
```
plt.scatter(df.speeding,df.alcohol)
sns.set_style("dark")
sns.set_context("poster")
plt.show()
```



Palette

In [ ]:
```
# color_palette
sns.palplot(sns.color_palette("deep", 10))
```

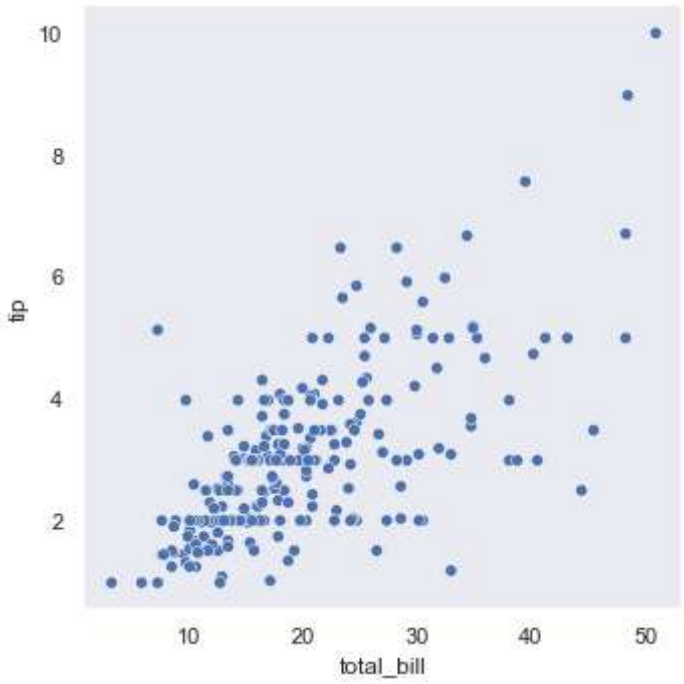
In [ ]:
```
tips = sns.load_dataset("tips")
tips.head()
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [ ]:
```
sns.set_context("notebook")
sns.relplot(data=tips,x="total_bill",y="tip")
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x293a979ef80>
```

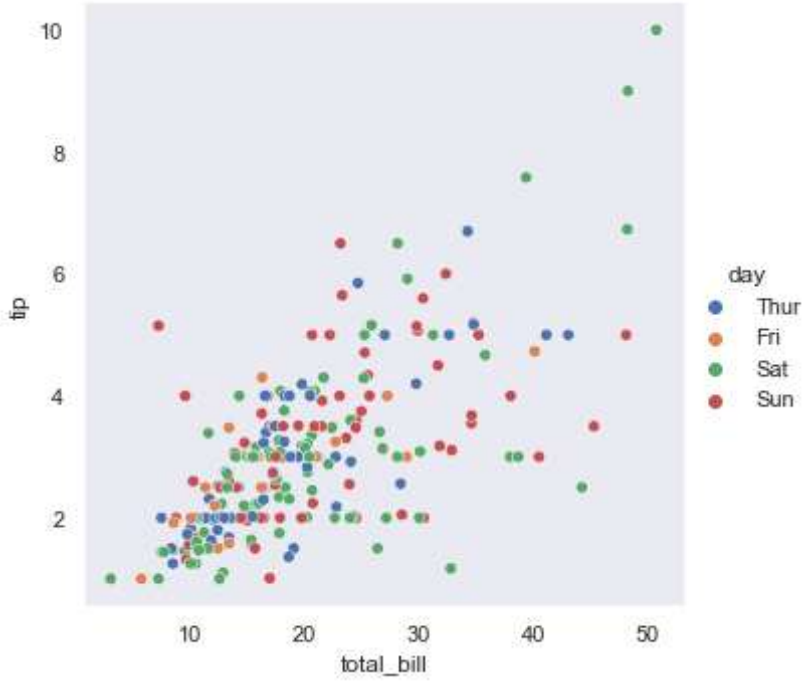


hue semantic

kind=line,scatter,hex,kde,...(default=>scatter) errorbar = None to get just the line without any highlighted portion.
"errorcolor,errorwidth,errorbar" capsize=errorbar cap size

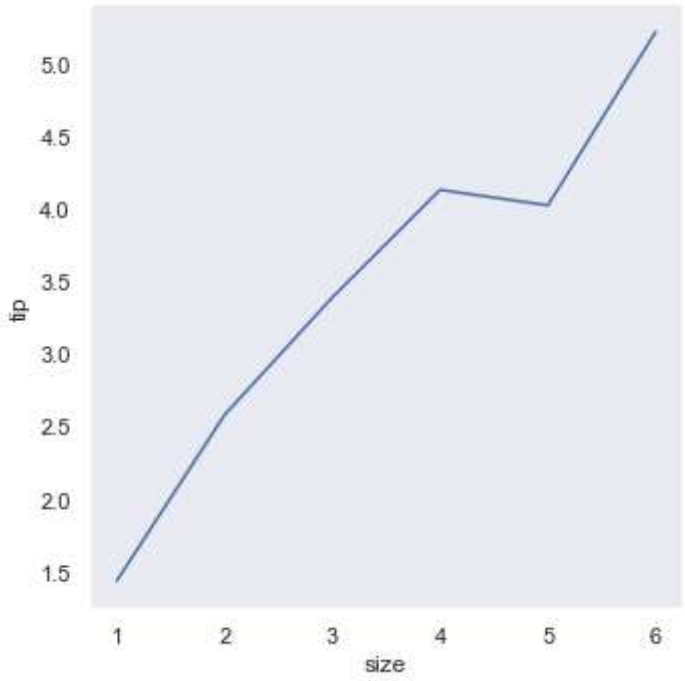
In [ ]:
```
#hue (third variabe=> diff in color)
sns.relplot(data=tips, x="total_bill", y="tip", hue="day")
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x293a979dc90>
```



In [ ]:
```
sns.relplot(data=tips, x="size", y="tip",kind="line",errorbar=None)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x293ab1db490>
```

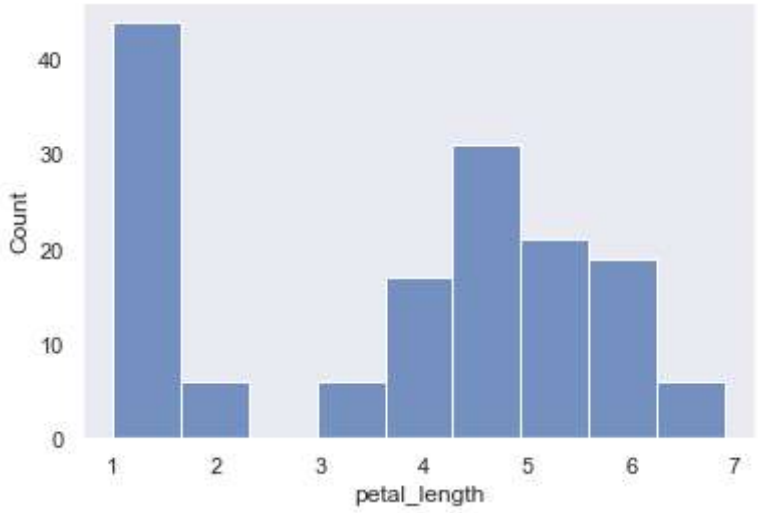


Histogram

Histograms represent the data distribution by forming bins along with the range of the data and then drawing bars to show the number of observations that fall in each bin.In Seaborn we use displot() or histplot() function to plot histograms.

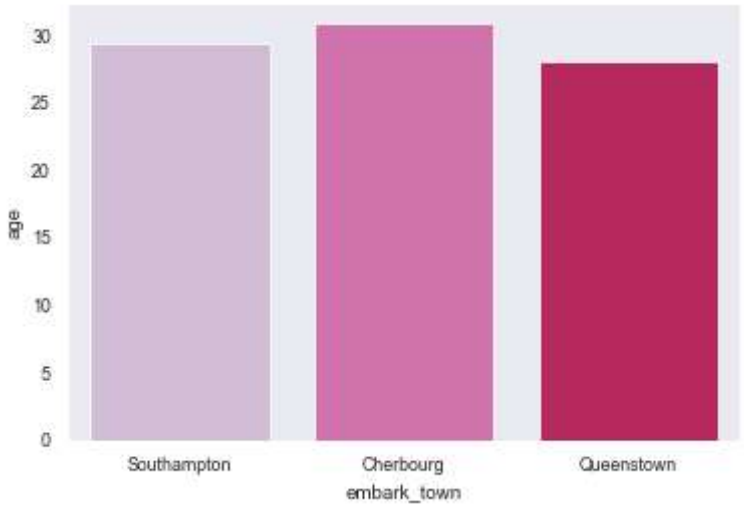
In [ ]:
```
df = sns.load_dataset('iris')
sns.histplot(df['petal_length'])
```

Out[ ]: <Axes: xlabel='petal_length', ylabel='Count'>



Bar Plot

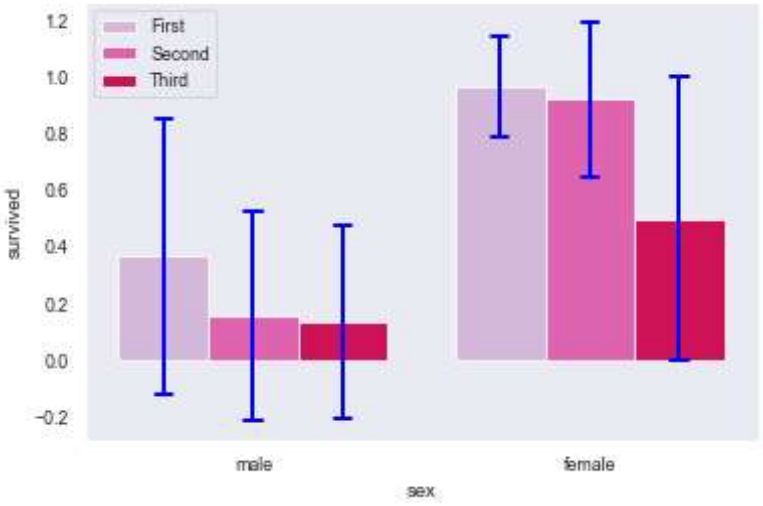
In [ ]:
```
# Vertical barplot

sns.set_context('paper')

# load dataset
titanic = sns.load_dataset('titanic')
# create plot
sns.barplot(x = 'embark_town', y = 'age', data = titanic,
            palette = 'PuRd',errorbar=None
            )
plt.show()
print(titanic.columns)
```




```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

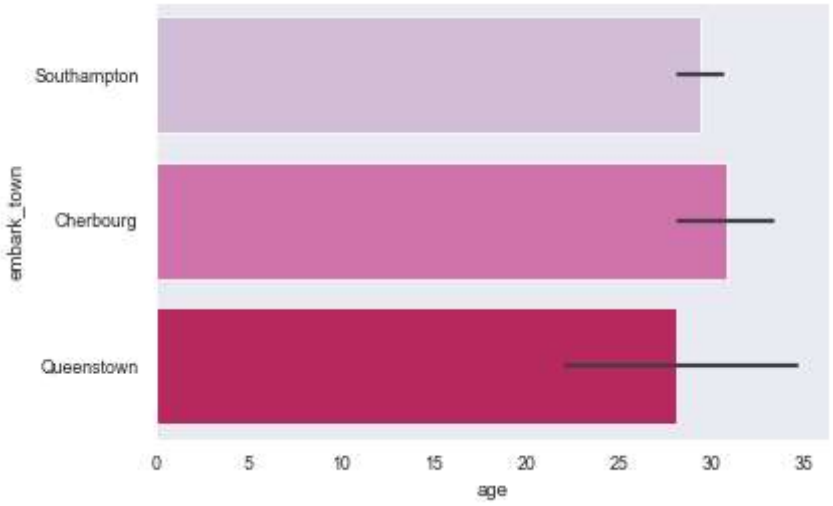
In [ ]:
```
# errorbar = None to get just the line without any highlighted portion.
# "errorcolor,errorwidth,errorbar"
# capsize=errorbar cap size
```

```
sns.barplot(x = 'sex', y = 'survived', hue = 'class', data = titanic,
            palette = 'PuRd',
            order = ['male', 'female'],
            capsize = 0.05,
            saturation = 8,
            errcolor = 'blue', errwidth = 2,
            errorbar= 'sd'
            )
plt.legend()
plt.show()
```



In [ ]:
```
# xaxis=  x
# yaxis=  y
# third variable=  hue
# colors=  palette
# orientation   =orient
```

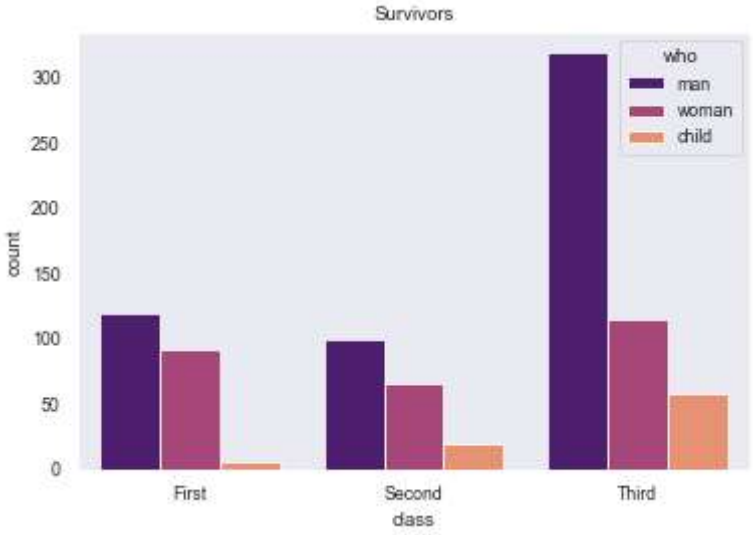
In [ ]:
```
#Horizontal barplot
sns.barplot(x = 'age', y = 'embark_town', data = titanic,
            palette = 'PuRd', orient = 'h',
            )
plt.show()
```



Count plot

The count plot can be thought of as a histogram across a categorical variable

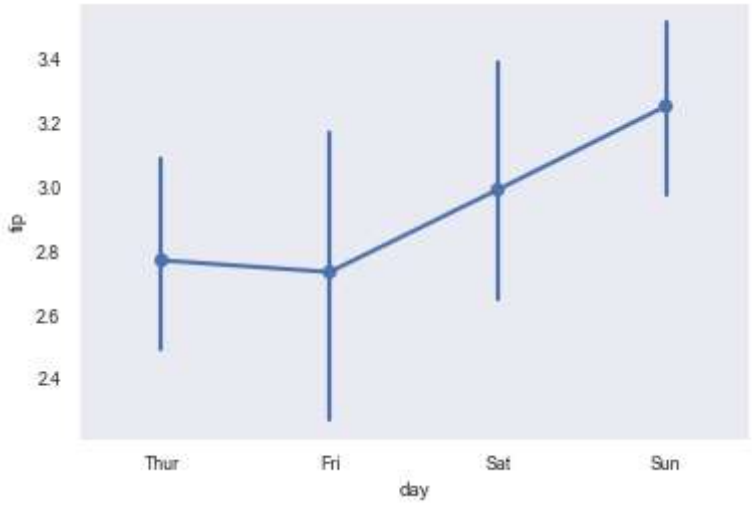
In [ ]:
```
sns.countplot(x = 'class', hue = 'who', data = titanic, palette = 'magma')
plt.title('Survivors')
plt.show()
```



Point Plot

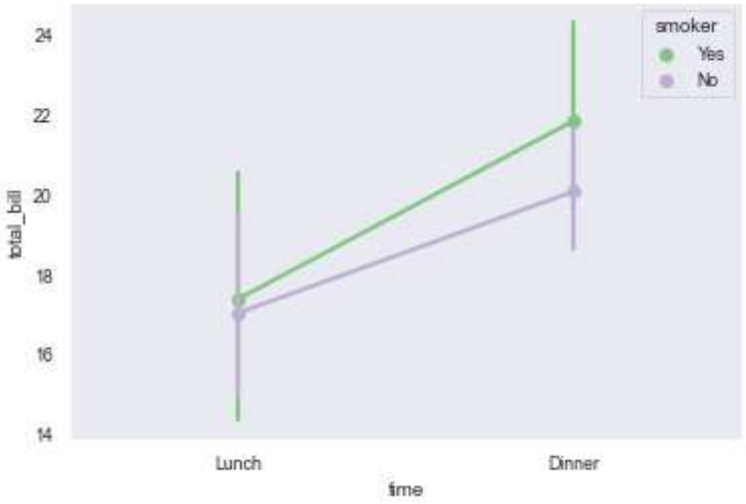
Point plot is used to show point estimates and confidence intervals using scatter plot glyphs. A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars

In [ ]:
```
data = sns.load_dataset("tips")
sns.pointplot(x="day", y="tip", data=data)
plt.show()
```



In [ ]:
```
sns.pointplot(x="time", y="total_bill", hue="smoker",
                    data=data, palette="Accent")
```

Out[ ]:
```
<Axes: xlabel='time', ylabel='total_bill'>
```

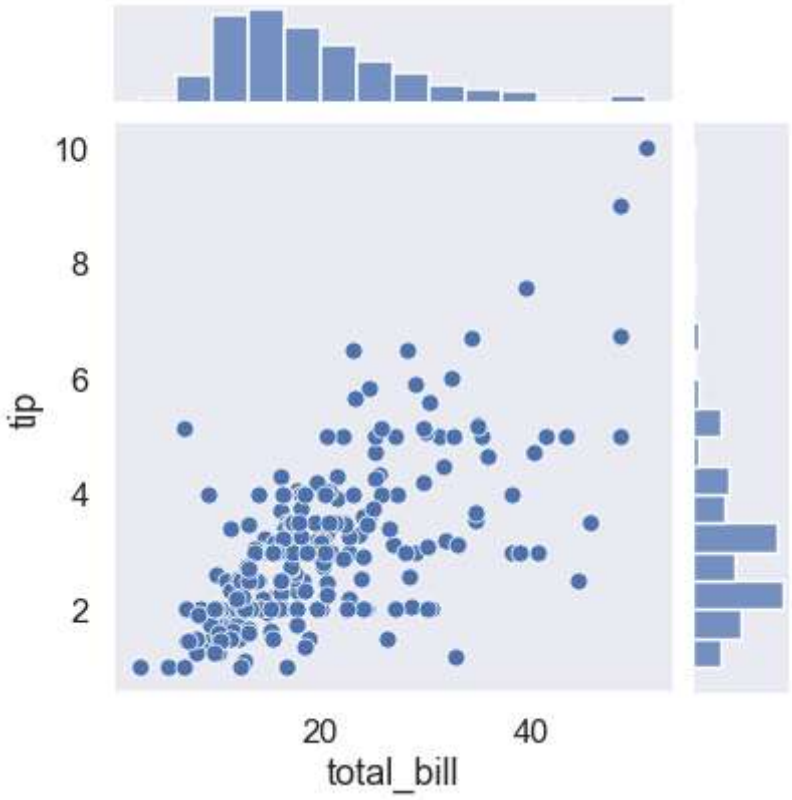


Joint Plot

Joint Plot draws a plot of two variables with bivariate and univariate graphs. It uses the Scatter Plot and Histogram. Joint Plot can also display data using Kernel Density Estimate (KDE) and Hexagons.

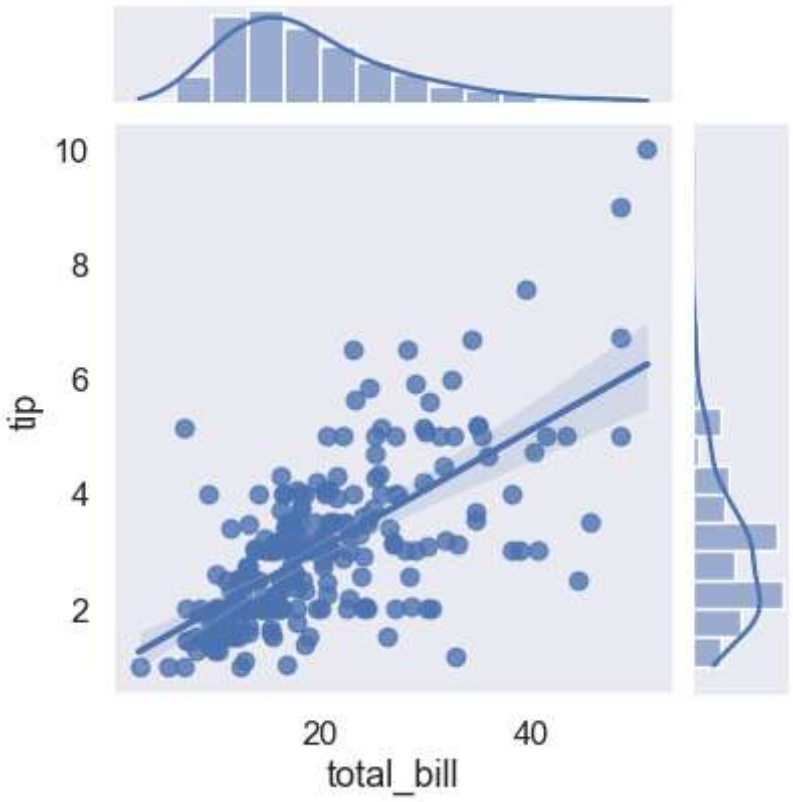
In [ ]:
```
# hex, reg, kde
sns.set_context("talk")
sns.set_style("dark")
tips=sns.load_dataset('tips')
sns.jointplot(x='total_bill', y='tip',data=tips)
```

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x293b0f63dc0>
```



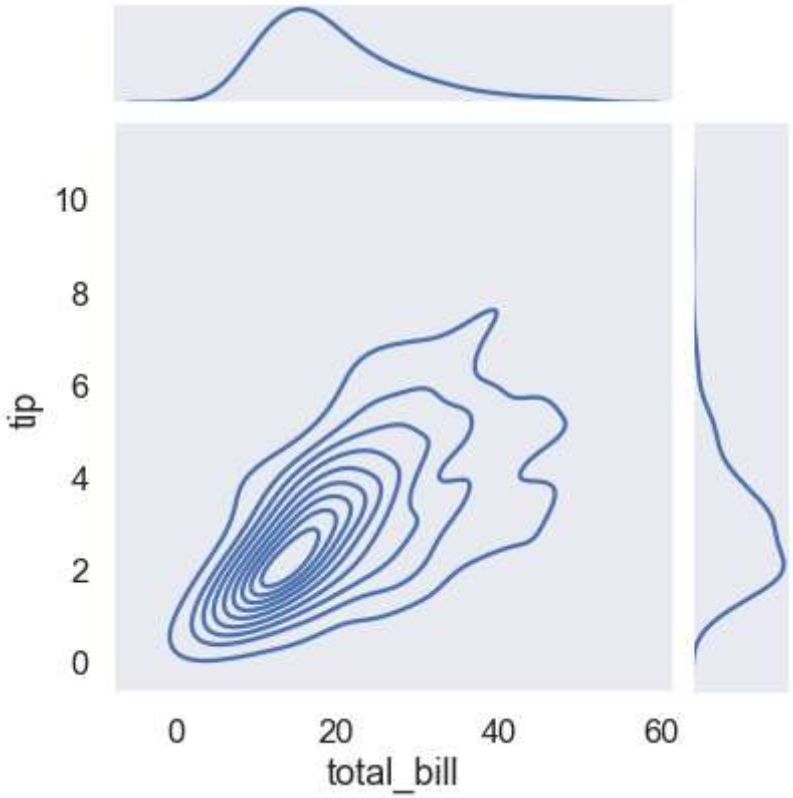
In [ ]:
```
# Add regression line to scatter plot and kernel density estimate to histogram
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg')
```

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x293b10c8790>
```



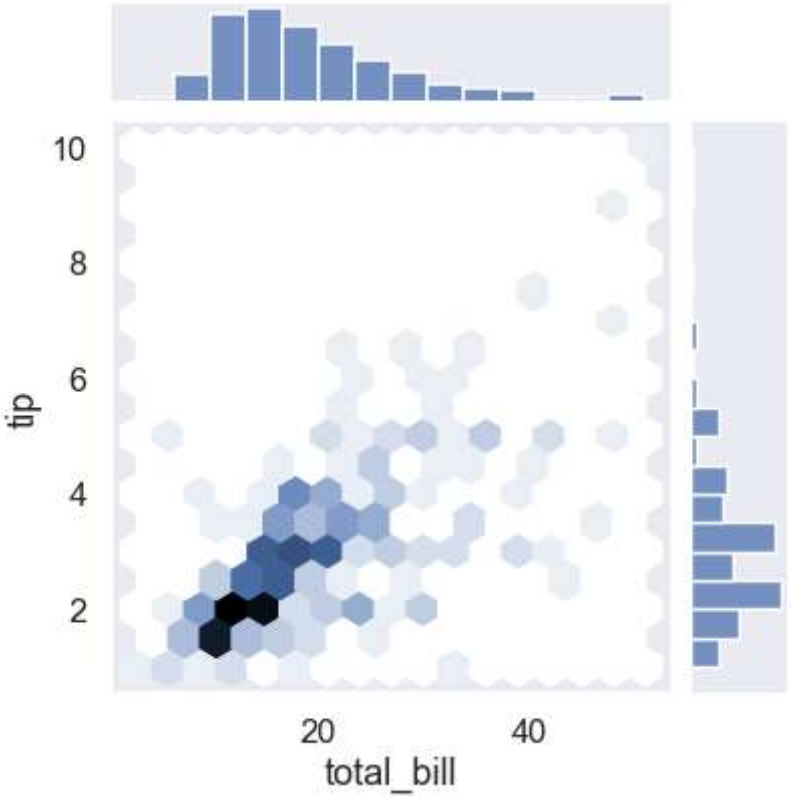
In [ ]:
```
# Display kernel density estimate instead of scatter plot and histogram
sns.set_style("dark")
sns.jointplot(x='total_bill', y='tip', data=tips, kind='kde')
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x293b10b5840>



In [ ]:
```
# Display hexagons instead of points in scatter plot
sns.jointplot(x='total_bill', y='tip', data=tips, kind='hex')
```

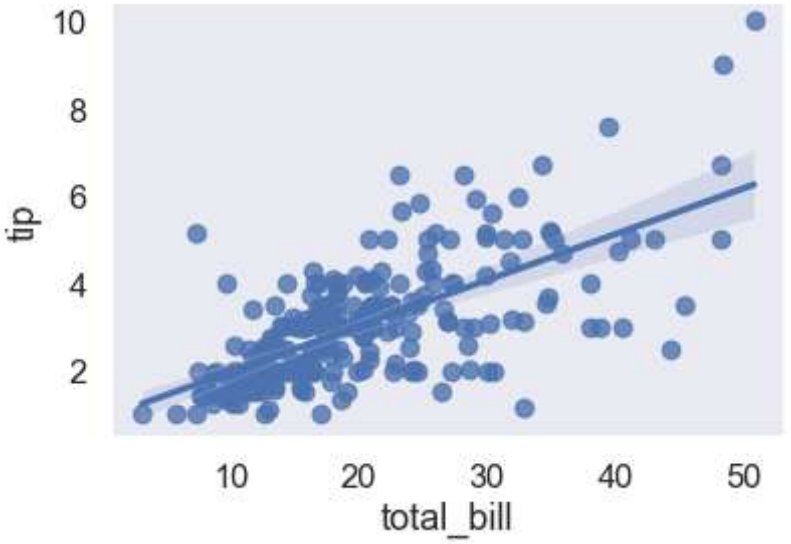
Out[ ]: <seaborn.axisgrid.JointGrid at 0x293affae8c0>



Regplot

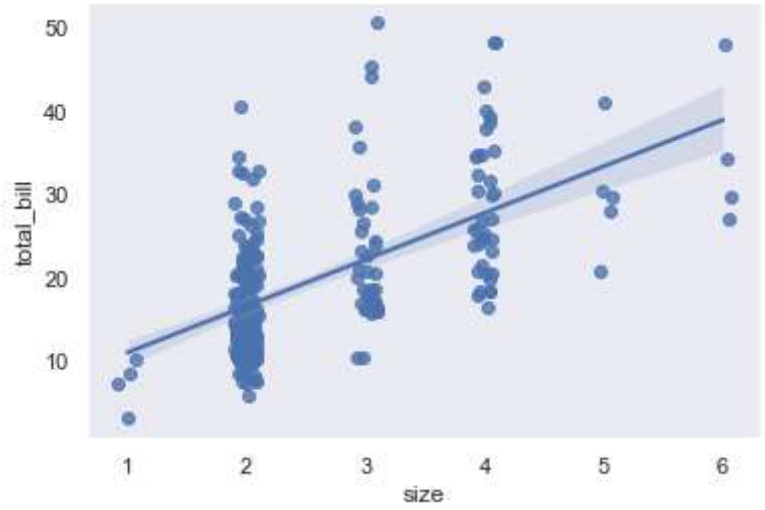
Regplot is one of the functions in Seaborn that are used to visualize the linear relationship as determined through regression. Also, you'll see a slightly shaded portion around the regression line which indicates how much the pints are scattered around a certain area. Here are few of the examples Now we will plot a discrete x variable and add some jitter. Here you can see that the areas where points are more densely populated have less shaded portion around the regression line and shaded portion is more spread where the points are more scattered.

In [ ]:
```
tips = sns.load_dataset("tips")
ax = sns.regplot(x="total_bill", y="tip", data=tips)
```



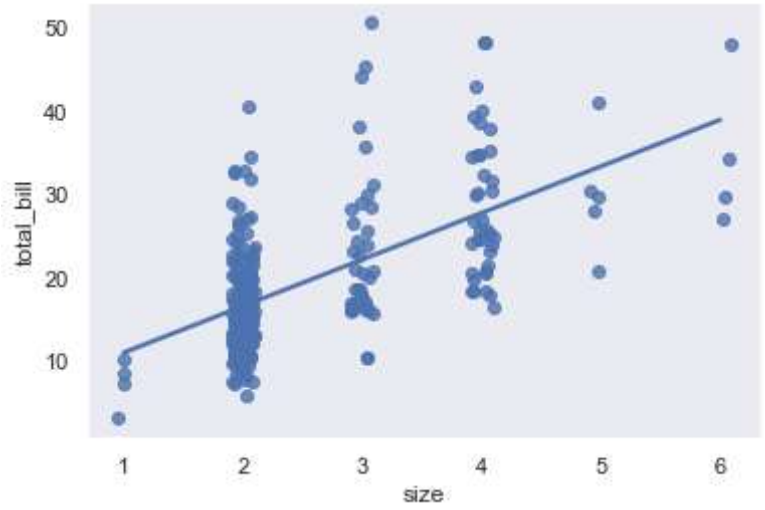
In [ ]:
```
sns.set_context("notebook")
sns.regplot(x="size", y="total_bill", data=tips, x_jitter=0.1)
```

Out[ ]:
```
<Axes: xlabel='size', ylabel='total_bill'>
```



In [ ]:
```
sns.regplot(x="size", y="total_bill", data=tips, x_jitter=0.1,ci=None
        )
```

Out[ ]:
```
<Axes: xlabel='size', ylabel='total_bill'>
```



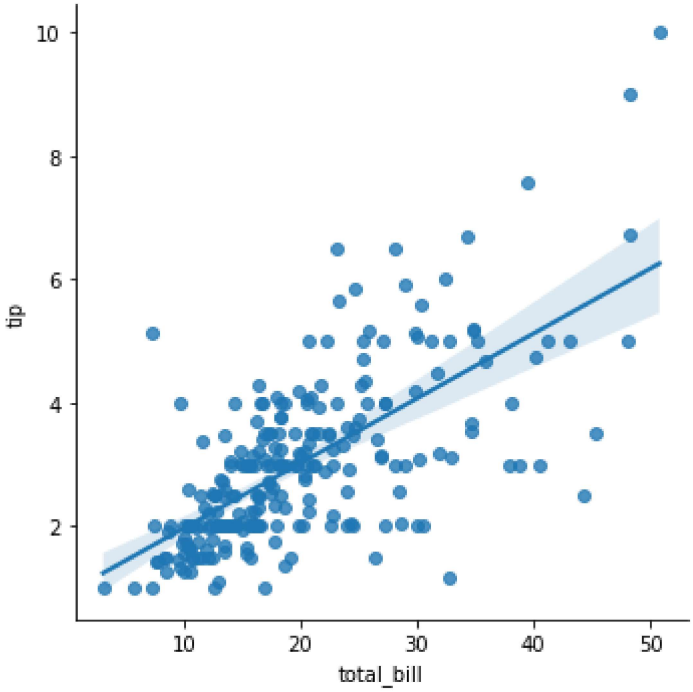
Seaborn Part 2

Lm Plot

The regplot function performs a simple linear regression model fit and plot whereas the lmplot function combines regplot and FacetGrid.The FacetGrid class helps in visualizing the distribution of one variable as well as the relationship between multiple variables separately within subsets of your dataset using multiple panels. It is further important to note that lmplot() is more computationally intensive and is intended as a convenient interface to fit regression models across conditional subsets of a dataset.

In [ ]:
```
import seaborn as sns
tips = sns.load_dataset("tips")
sns.lmplot(x="total_bill", y="tip", data=tips)
```

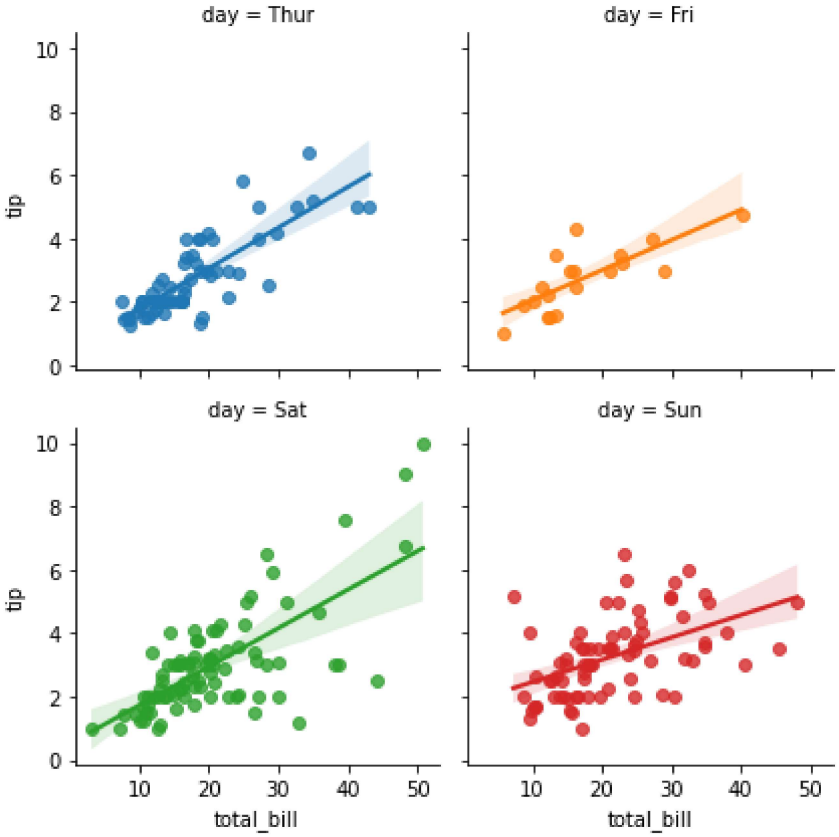
Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x206654af610>
```



In [ ]:
```
# col_warp=>numbers of column
# height=>size of the sub graph
```

In [ ]:
```
sns.lmplot(x="total_bill", y="tip", col="day", hue="day",
                data=tips, col_wrap=2, height=3)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x206676b89a0>
```

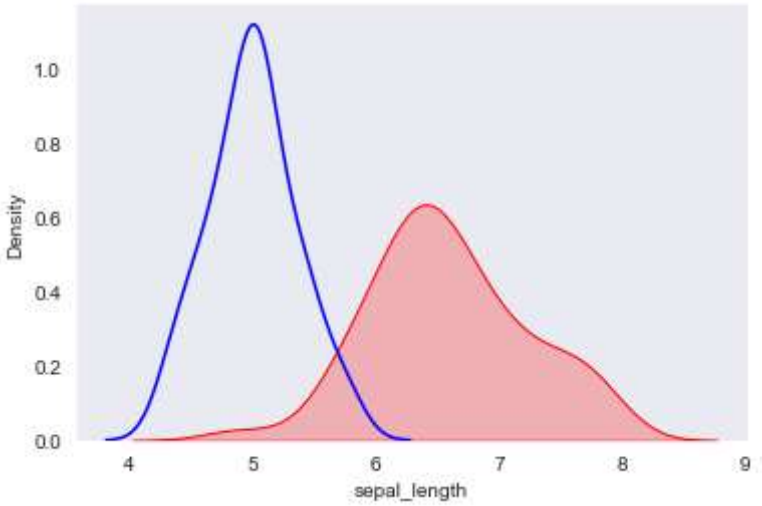


KDE plot

KDE plot is a Kernel Density Estimate that is used for visualizing the Probability Density of the continuous or non-parametric data variables.we can plot for the univariate or multiple variables altogether.

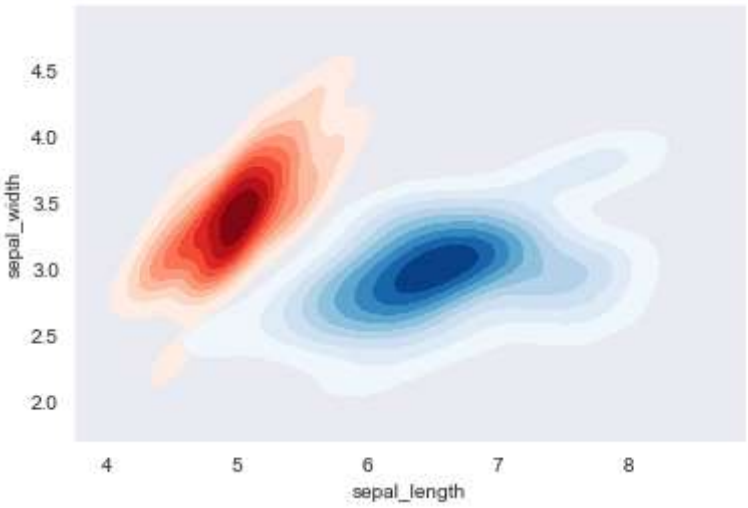
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style("dark")
iris = sns.load_dataset("iris")
sns.kdeplot(iris.loc[(iris['species']=='setosa'),
            'sepal_length'], color='b', fill=None)
sns.kdeplot(iris.loc[(iris['species']=='virginica'),
            'sepal_length'], color='r', fill=True)
```

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



In [ ]:
```
import seaborn as sns
from pandas import Series, DataFrame  # Import for clarity

# Assuming you have loaded the iris dataset

iris_setosa = iris.query("species=='setosa'")
iris_virginica = iris.query("species=='virginica'")

# Plotting KDE for Iris Setosa
sns.kdeplot(y=iris_setosa['sepal_width'],x=iris_setosa['sepal_length'],
            color='r', fill=True, label='Iris_Setosa',cmap="Reds")

# Plotting KDE for Iris Virginica
sns.kdeplot(y=iris_virginica['sepal_width'], x=iris_virginica['sepal_length'],
            color='b', fill=True, label='Iris_Virginica',
             cmap="Blues")
```

Out[ ]:
```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```

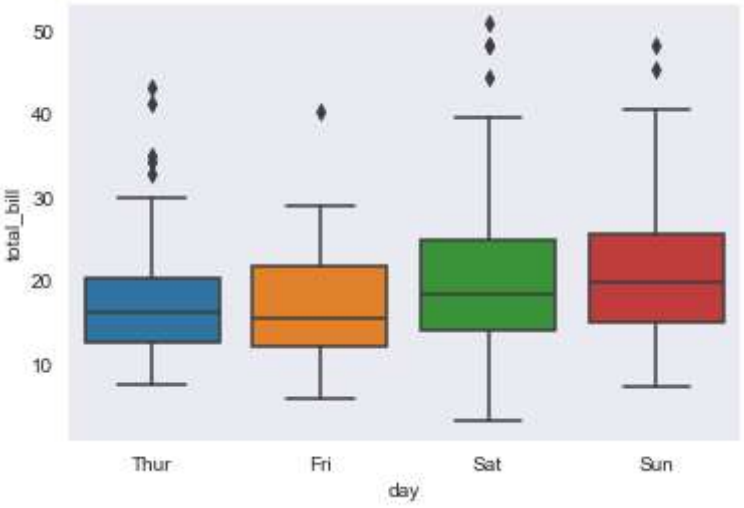


Box Plot

Boxplot is also used for detecting the outlier in a data set.The small diamond shape of the box plot is outlier data. horizontal line- min and max First Quartile or 25% Median (Second Quartile) or 50% Third Quartile or 75% horizontal line of the rectangle shape of the box plot

In [ ]:
```
tips = sns.load_dataset("tips")
sns.boxplot(x="day", y="total_bill", data=tips)
```

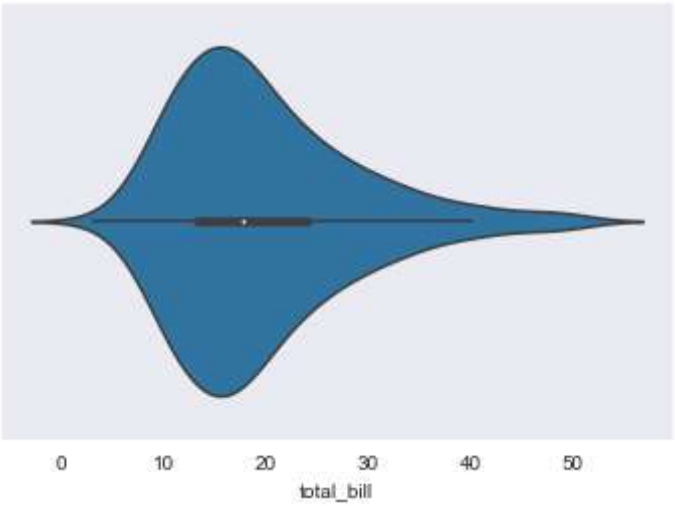
Out[ ]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



Violin Plot

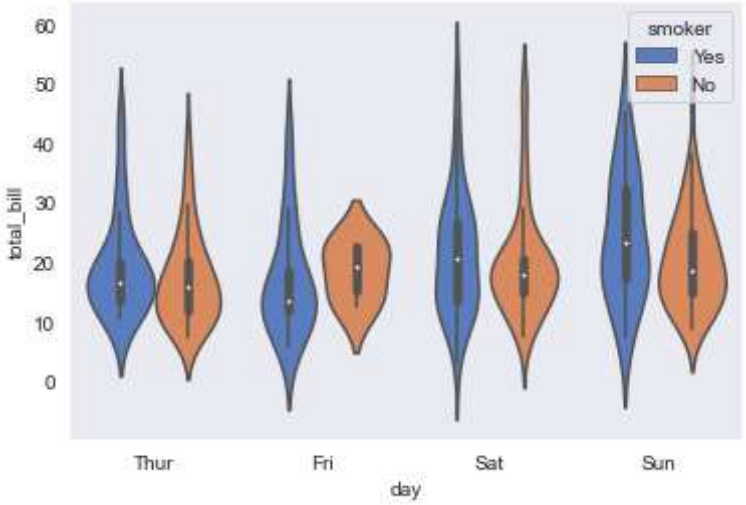
They are essentially a box plot with a kernel density estimate (KDE) overlaid along with the range of the box and reflected to make it look nice. Unlike a box plot, in which all of the plot components correspond to actual data points, the violin plot features a kernel density estimation of the underlying distribution.

In [ ]:
```
import seaborn as sns
tips = sns.load_dataset("tips")
ax = sns.violinplot(x=tips["total_bill"])
```



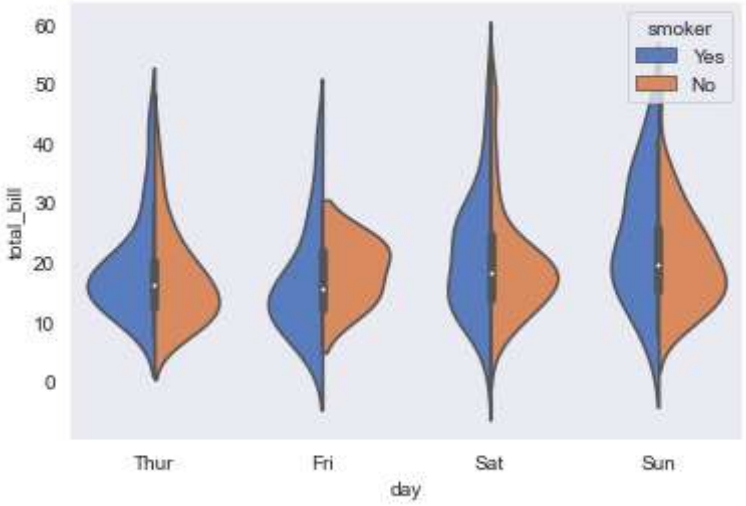
In [ ]:
```
sns.violinplot(x="day", y="total_bill", hue="smoker",
                    data=tips, palette="muted")
```

Out[ ]: <Axes: xlabel='day', ylabel='total_bill'>



In [ ]:
```
sns.violinplot(x="day", y="total_bill", hue="smoker",
                    data=tips, palette="muted", split=True)
```

Out[ ]: <Axes: xlabel='day', ylabel='total_bill'>



Heatmap

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

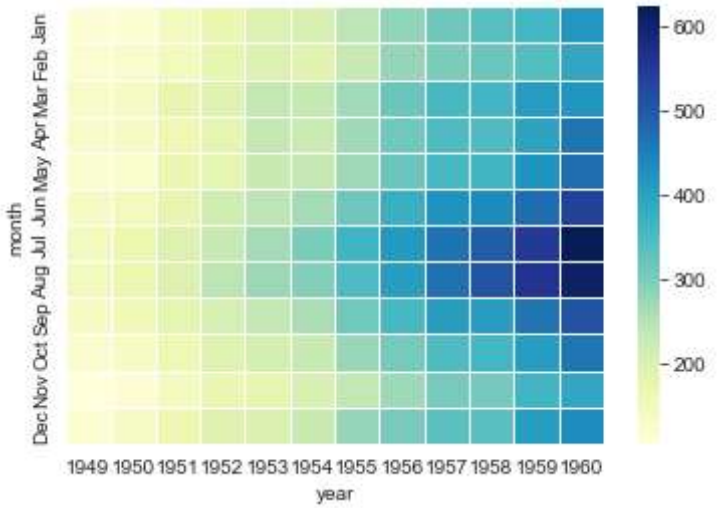
In [ ]:
```
flights=sns.load_dataset("flights")
flights = flights.pivot("month", "year", "passengers",)
print(flights)
```

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

```
C:\Users\heman\AppData\Local\Temp\ipykernel_20552\2606158732.py:2: FutureWarning:
In a future version of pandas all arguments of DataFrame.pivot will be keyword-on
ly.
  flights = flights.pivot("month", "year", "passengers")
```

In [ ]:
```
sns.heatmap(flights,linewidths=.9)
```

Out[ ]:
```
<Axes: xlabel='year', ylabel='month'>
```

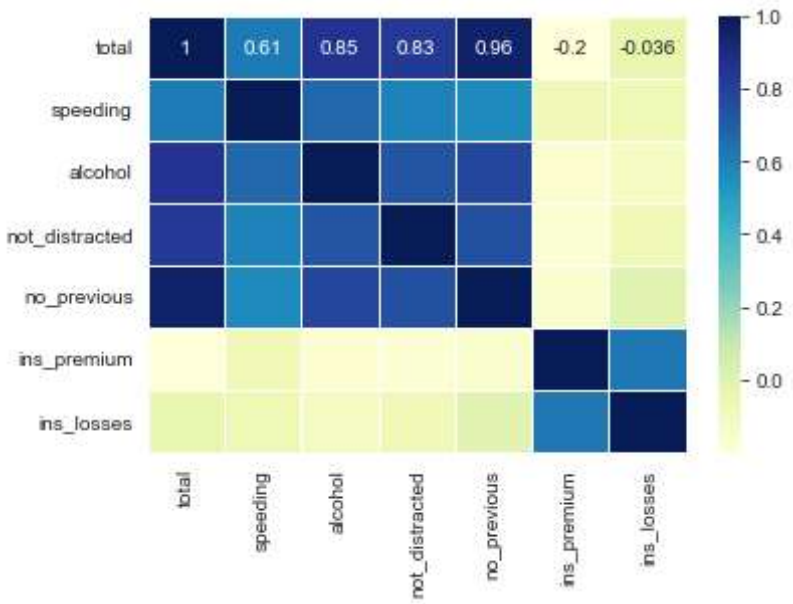


In [ ]:
```
car_crashes = sns.load_dataset("car_crashes")
corr=car_crashes.corr()
print(corr)
sns.heatmap(corr,annot=True,linewidths=.5,cmap="YlGnBu")
```

```
                    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
```

```
C:\Users\heman\AppData\Local\Temp\ipykernel_20552\3320747515.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
rsion, it will default to False. Select only valid columns or specify the value o
f numeric_only to silence this warning.
  corr=car_crashes.corr()
```

Out[ ]: <Axes: >



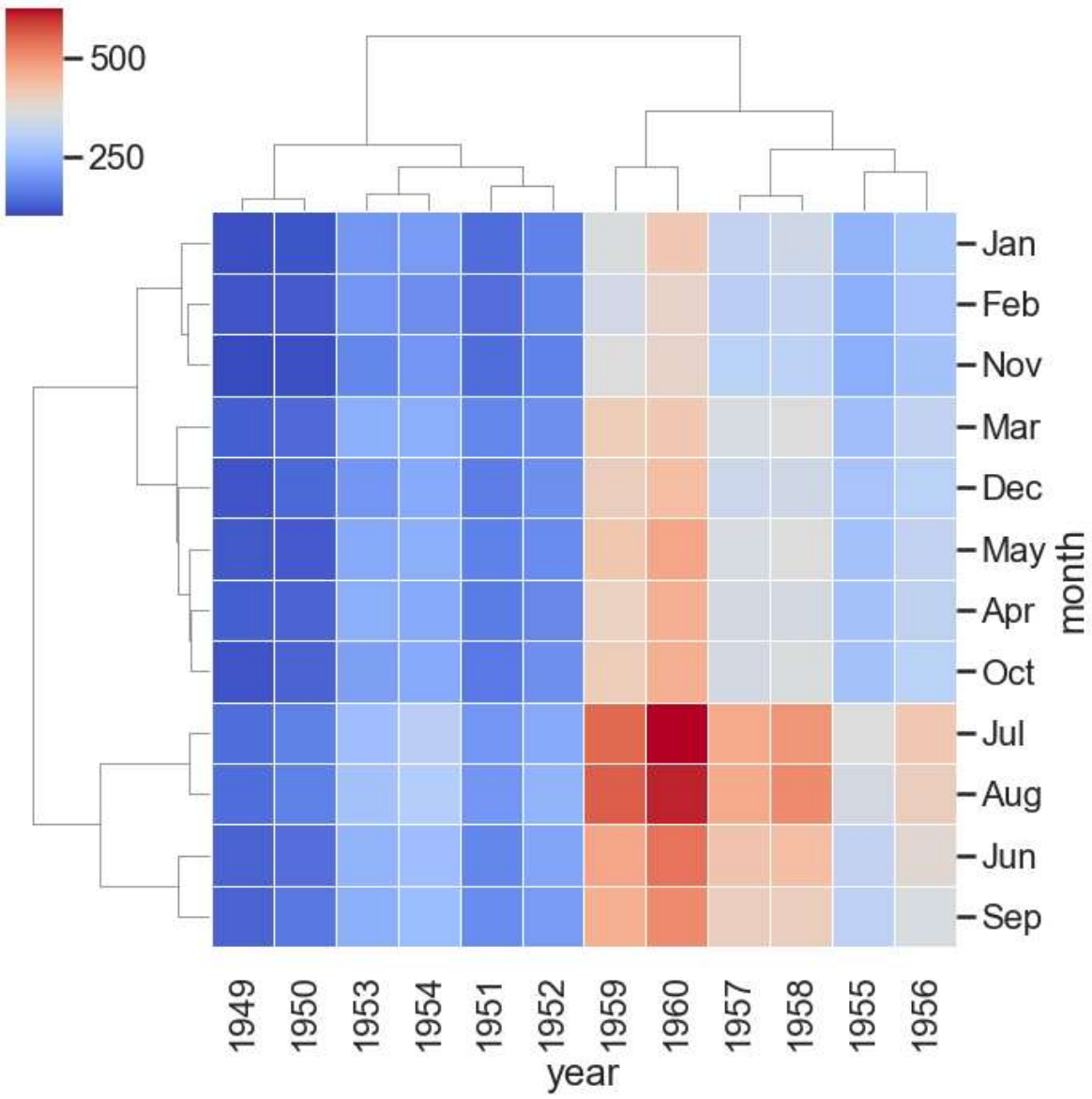
Cluster map

Cluster map method plots a matrix dataset as a hierarchically-clustered heatmap. It uses hierarchical clusters to order data by similarity. This reorganizes the data for the rows and columns and displays similar content next to one another for even more depth of understanding the data.

In [ ]:
```
sns.set_context("poster")
flights=sns.load_dataset("flights")
flights = flights.pivot("month", "year", "passengers")
sns.clustermap(flights,linewidths=.5,cmap="coolwarm")
```

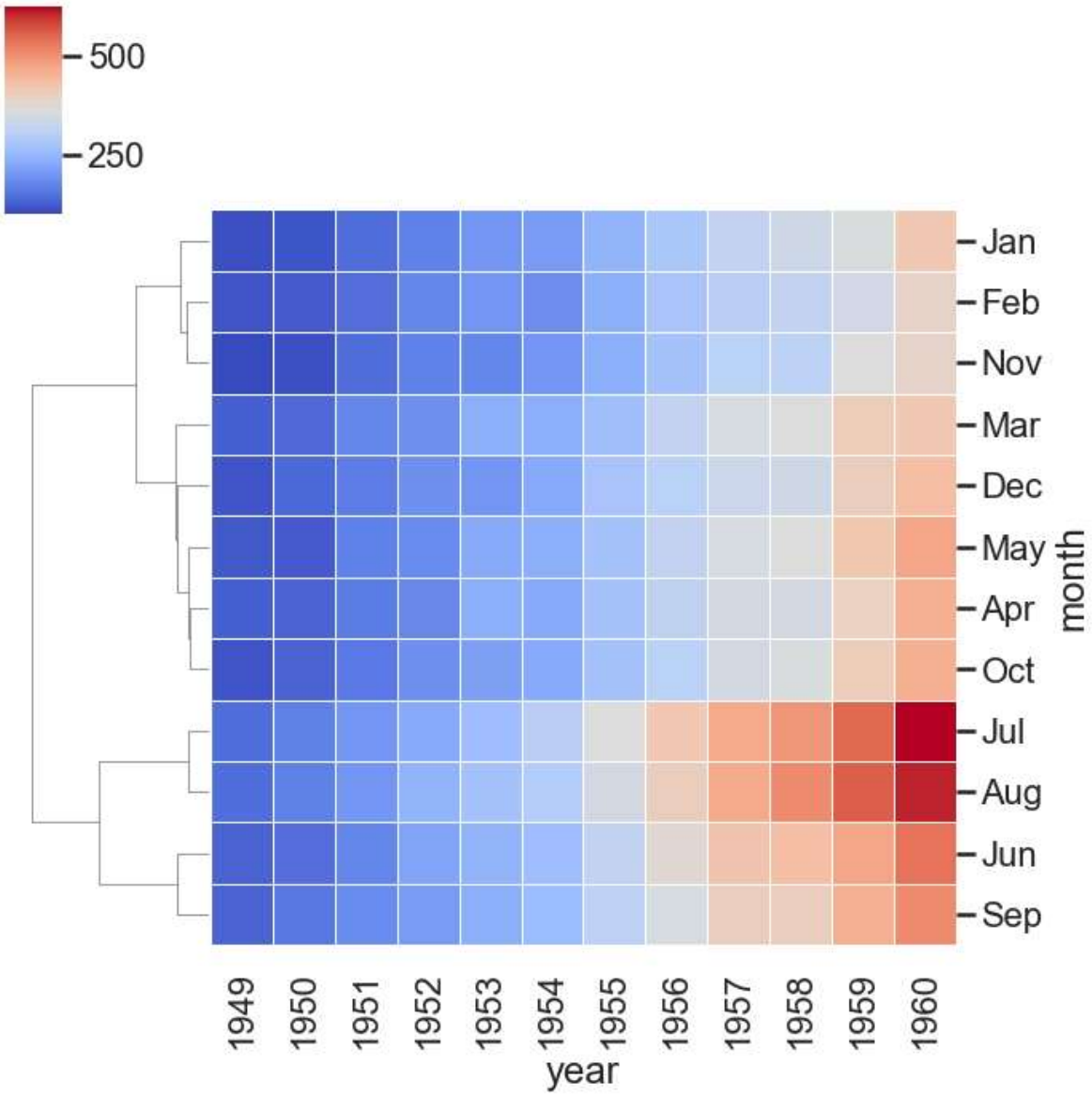
```
C:\Users\heman\AppData\Local\Temp\ipykernel_20552\4046513125.py:3: FutureWarning:
In a future version of pandas all arguments of DataFrame.pivot will be keyword-on
ly.
  flights = flights.pivot("month", "year", "passengers")
```

Out[ ]: <seaborn.matrix.ClusterGrid at 0x20677f9dde0>



In [ ]:
```
sns.set_context("poster")
flights=sns.load_dataset("flights")
flights = flights.pivot(index="month",columns= "year", values="passengers")
sns.clustermap(flights,linewidths=.6,cmap="coolwarm",col_cluster=False)
```

Out[ ]: <seaborn.matrix.ClusterGrid at 0x2067f1bce80>

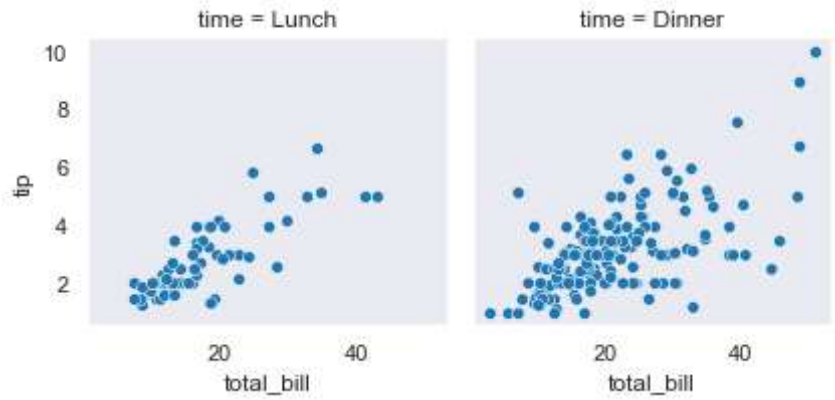


Facetgrid

The advantage of using Facet is, we can input another variable into the plot. The above plot is divided into two plots based on a third variable called ﷺdietﷺ using the ﷺcolﷺ parameter. We can also one more parameter ﷺrowﷺ which can help to add one more variable to our plot

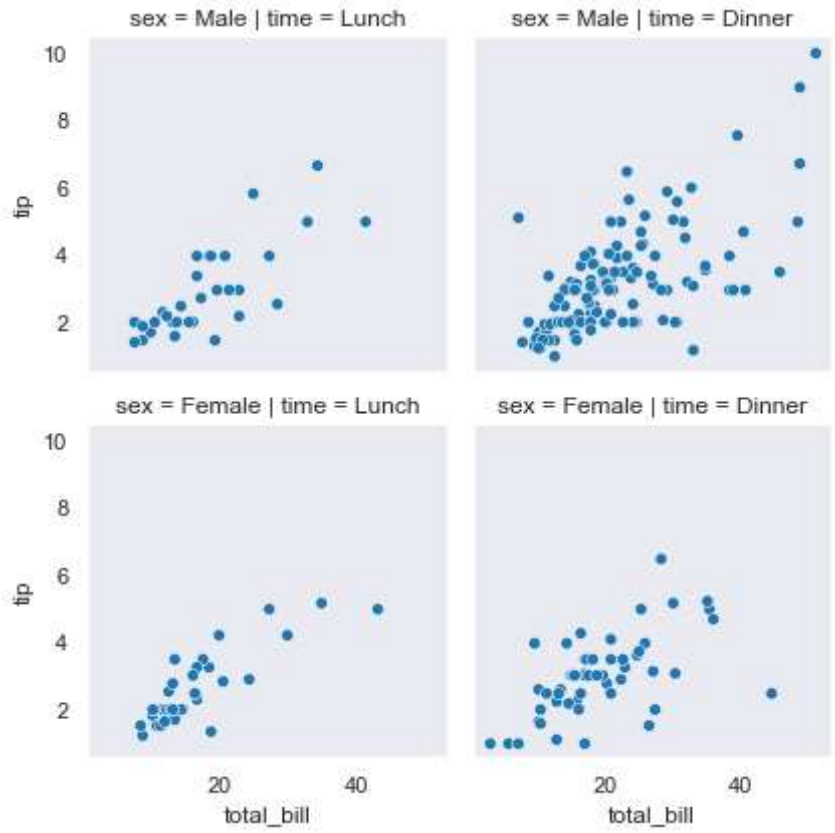
In [ ]:
```
sns.set_context("notebook")
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time")
g.map(sns.scatterplot, "total_bill", "tip")
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x2067dba9b70>
```



In [ ]:
```
g = sns.FacetGrid(tips, col="time", row="sex")
g.map(sns.scatterplot, "total_bill", "tip")
```

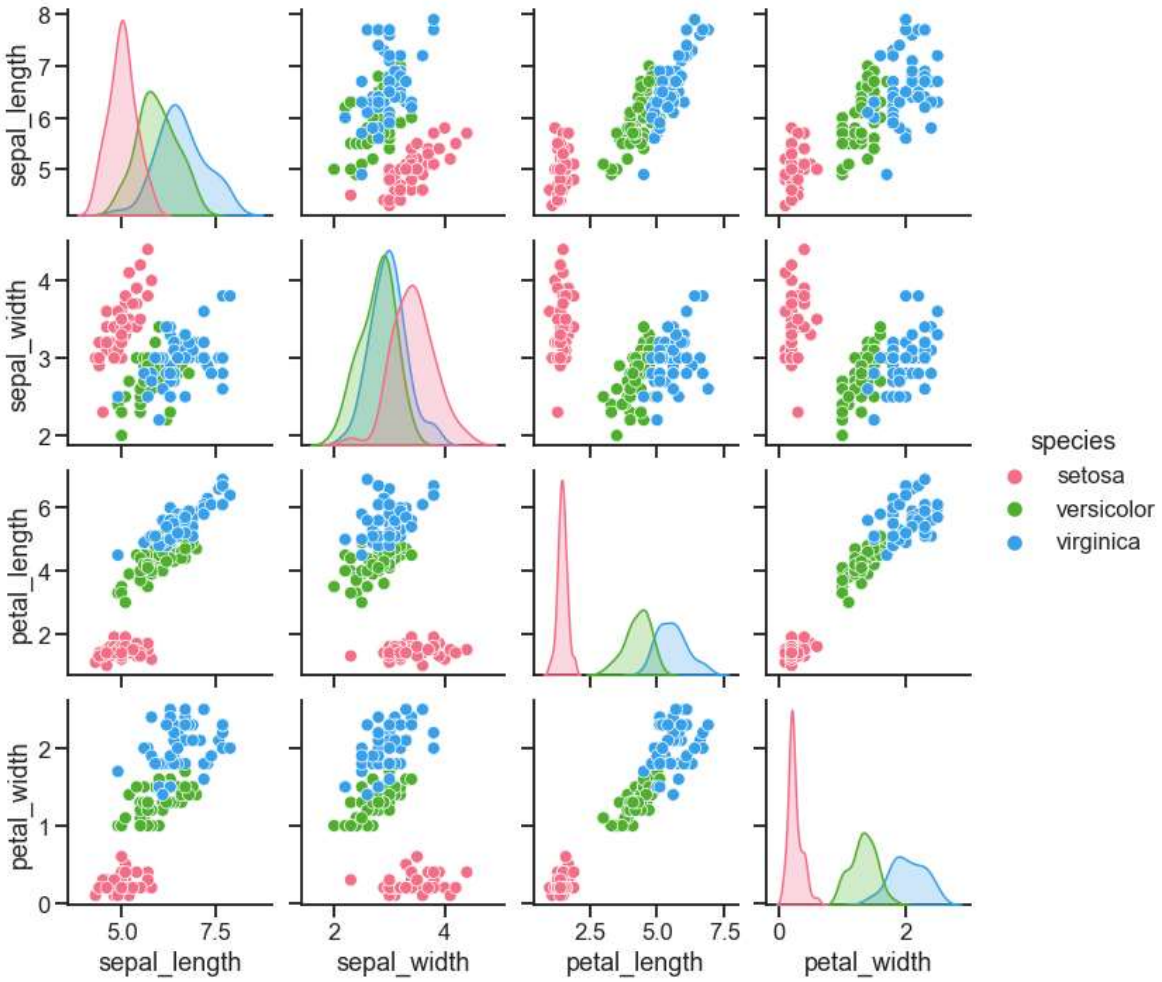
Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x2067f6417e0>
```



Pair Plot

Pair Plots are a really simple way to visualize relationships between each variable. It produces a matrix of relationships between each variable in your data for an instant examination

In [ ]:
```
df = sns.load_dataset('iris')
sns.set_style("ticks")
sns.set_context("talk")
sns.pairplot(df,hue = 'species',diag_kind = "kde",kind = "scatter",palette = "hu
plt.show()
```


sepal_length
sepal_width
petal_length
petal_width
5.0
7.5
2
4
2.5
5.0
7.5
0
2
species
setosa
versicolor
virginica

In [ ]: