Seaborn Part 2

Lm Plot

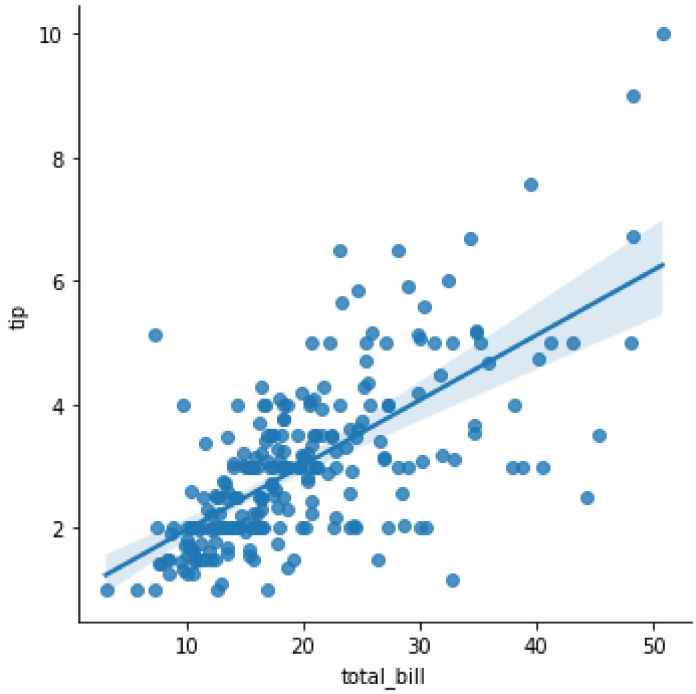
The regplot function performs a simple linear regression model fit and plot whereas the lmplot function combines regplot and FacetGrid.

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 [1]:
```
import seaborn as sns
tips = sns.load_dataset("tips")
sns.lmplot(x="total_bill", y="tip", data=tips)
```

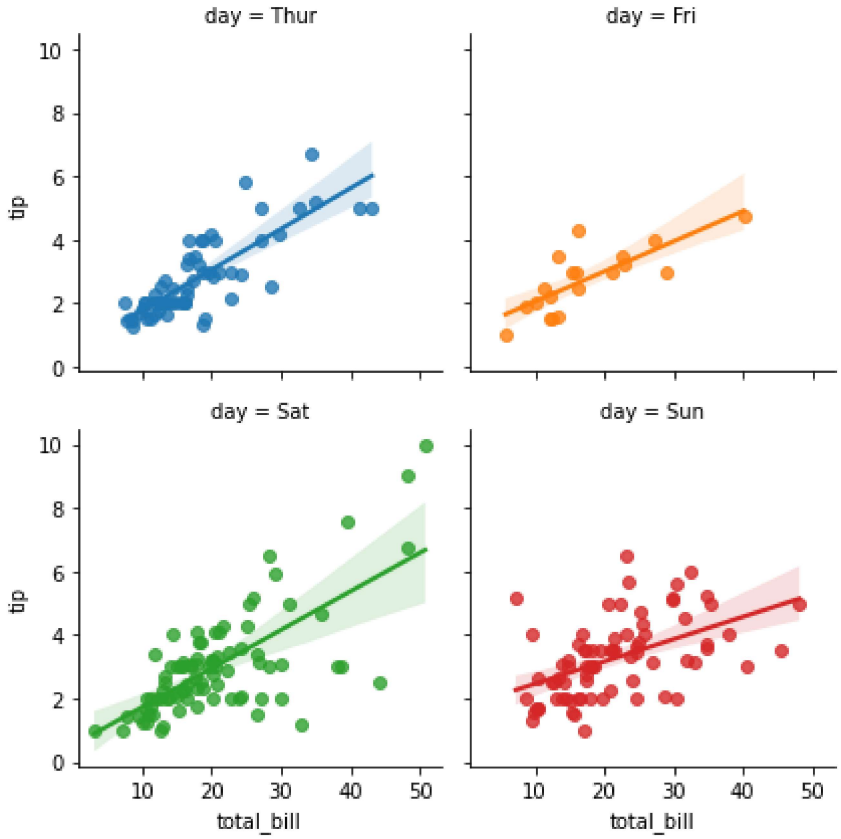
Out[1]:
```
<seaborn.axisgrid.FacetGrid at 0x206654af610>
```



In [2]:
```
# col_warp=>numbers of column
# height=>size of the sub graph
```

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

Out[3]:
```
<seaborn.axisgrid.FacetGrid at 0x206676b89a0>
```

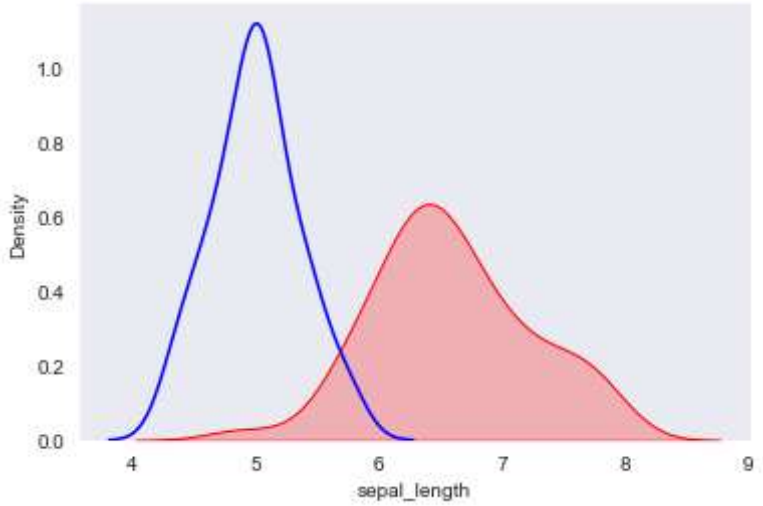


KDE plot

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.

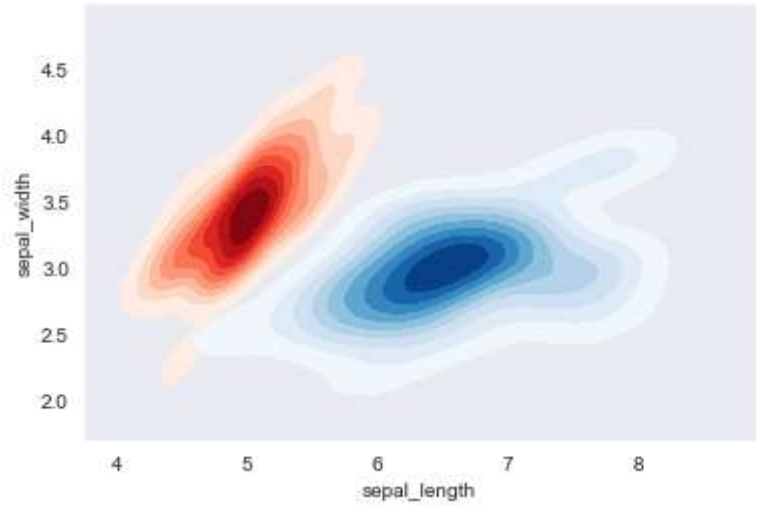
In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style("dark")
iris = sns.load_dataset("iris")
sns.kdeplot(iris.loc[(iris['species']=='setosa'),
            'sepal_length'], color='b', fill=None)
sns.kdeplot(iris.loc[(iris['species']=='virginica'),
            'sepal_length'], color='r', fill=True)
```

Out[4]:
```
<Axes: xlabel='sepal_length', ylabel='Density'>
```



In [11]:
```
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[11]: <Axes: xlabel='sepal_length', ylabel='sepal_width'>

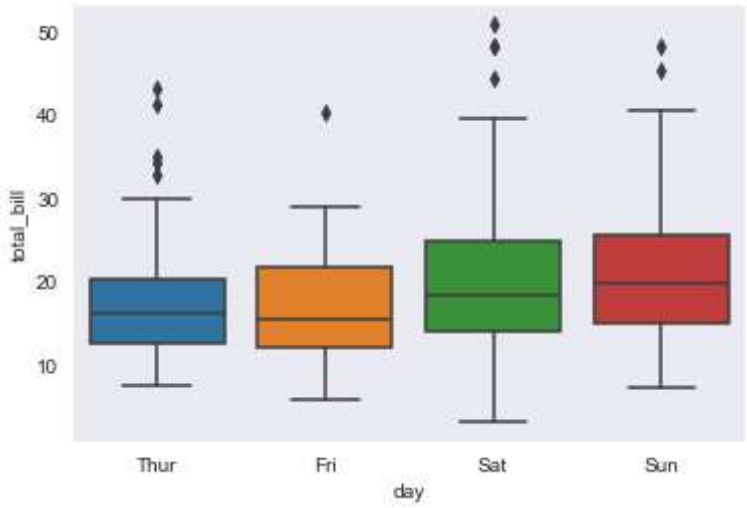


Box Plot

```
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 [12]:
```
tips = sns.load_dataset("tips")
sns.boxplot(x="day", y="total_bill", data=tips)
```

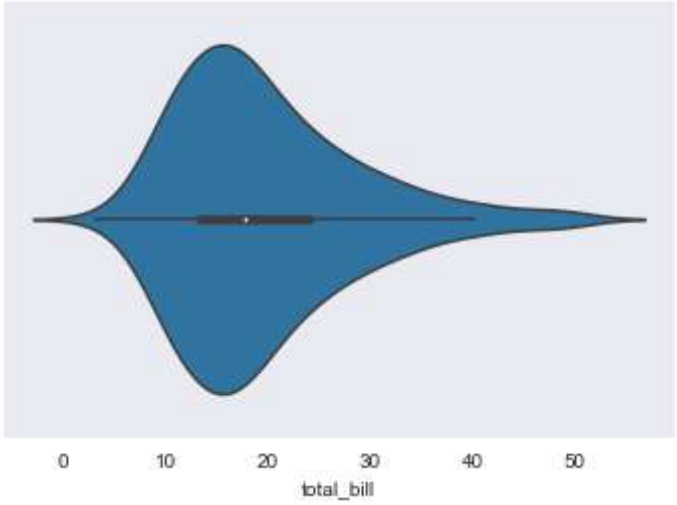
Out[12]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



Violin Plot

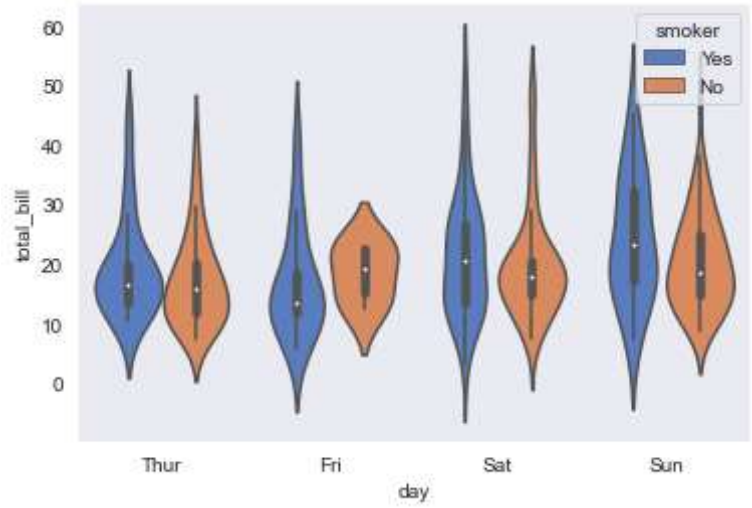
They are essentially a box plot with a kernel density estimate (KDE) overlaid along with the range of the box and reflected to make it look nice. Unlike a box plot, in which all of the plot components correspond to actual data points, the violin plot features a kernel density estimation of the underlying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

In [13]:
```
import seaborn as sns
tips = sns.load_dataset("tips")
ax = sns.violinplot(x=tips["total_bill"])
```



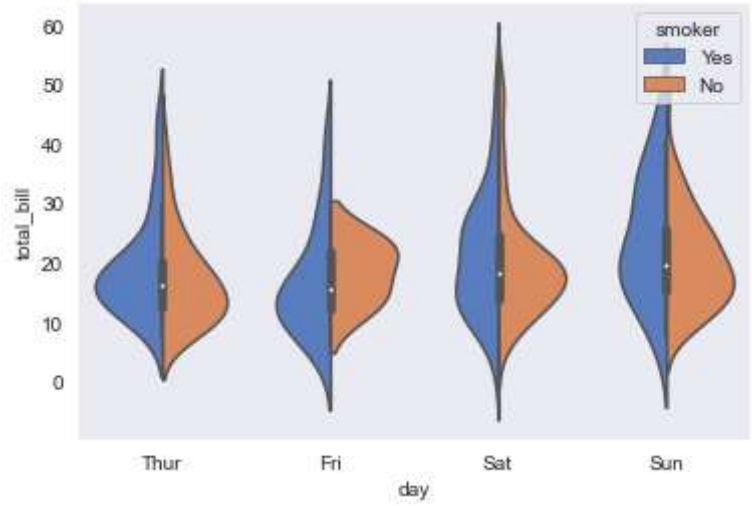
In [14]:
```
sns.violinplot(x="day", y="total_bill", hue="smoker",
                    data=tips, palette="muted")
```

Out[14]: <Axes: xlabel='day', ylabel='total_bill'>



In [15]:
```
sns.violinplot(x="day", y="total_bill", hue="smoker",
                    data=tips, palette="muted", split=True)
```

Out[15]: <Axes: xlabel='day', ylabel='total_bill'>



Heatmap

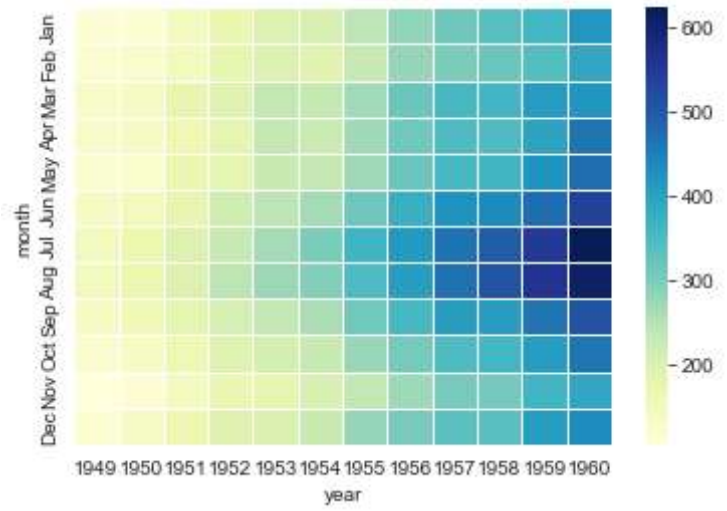
A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colours.

In [17]:
```
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  19
60
month
Jan      112   115   145   171   196   204   242   284   315   340   360   4
17
Feb      118   126   150   180   196   188   233   277   301   318   342   3
91
Mar      132   141   178   193   236   235   267   317   356   362   406   4
19
Apr      129   135   163   181   235   227   269   313   348   348   396   4
61
May      121   125   172   183   229   234   270   318   355   363   420   4
72
Jun      135   149   178   218   243   264   315   374   422   435   472   5
35
Jul      148   170   199   230   264   302   364   413   465   491   548   6
22
Aug      148   170   199   242   272   293   347   405   467   505   559   6
06
Sep      136   158   184   209   237   259   312   355   404   404   463   5
08
Oct      119   133   162   191   211   229   274   306   347   359   407   4
61
Nov      104   114   146   172   180   203   237   271   305   310   362   3
90
Dec      118   140   166   194   201   229   278   306   336   337   405   4
32

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

In [22]:
```
sns.heatmap(flights,linewidths=.9)
```

Out[22]:
```
<Axes: xlabel='year', ylabel='month'>
```

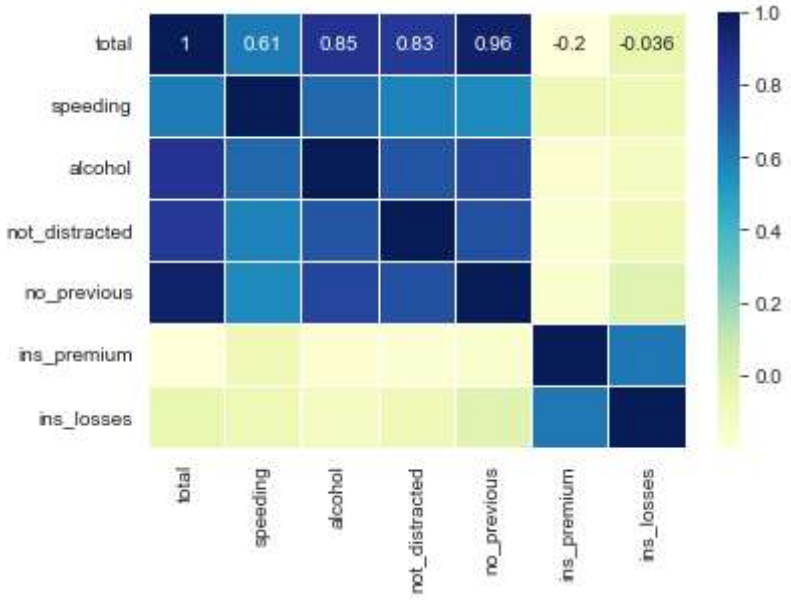


In [25]:
```
car_crashes = sns.load_dataset("car_crashes")
corr=car_crashes.corr()
print(corr)
sns.heatmap(corr,annot=True,linewidths=.5,cmap="YlGnBu")
```

```
                   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: FutureWa
rning: The default value of numeric_only in DataFrame.corr is deprecated. I
n a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  corr=car_crashes.corr()
```

Out[25]: <Axes: >



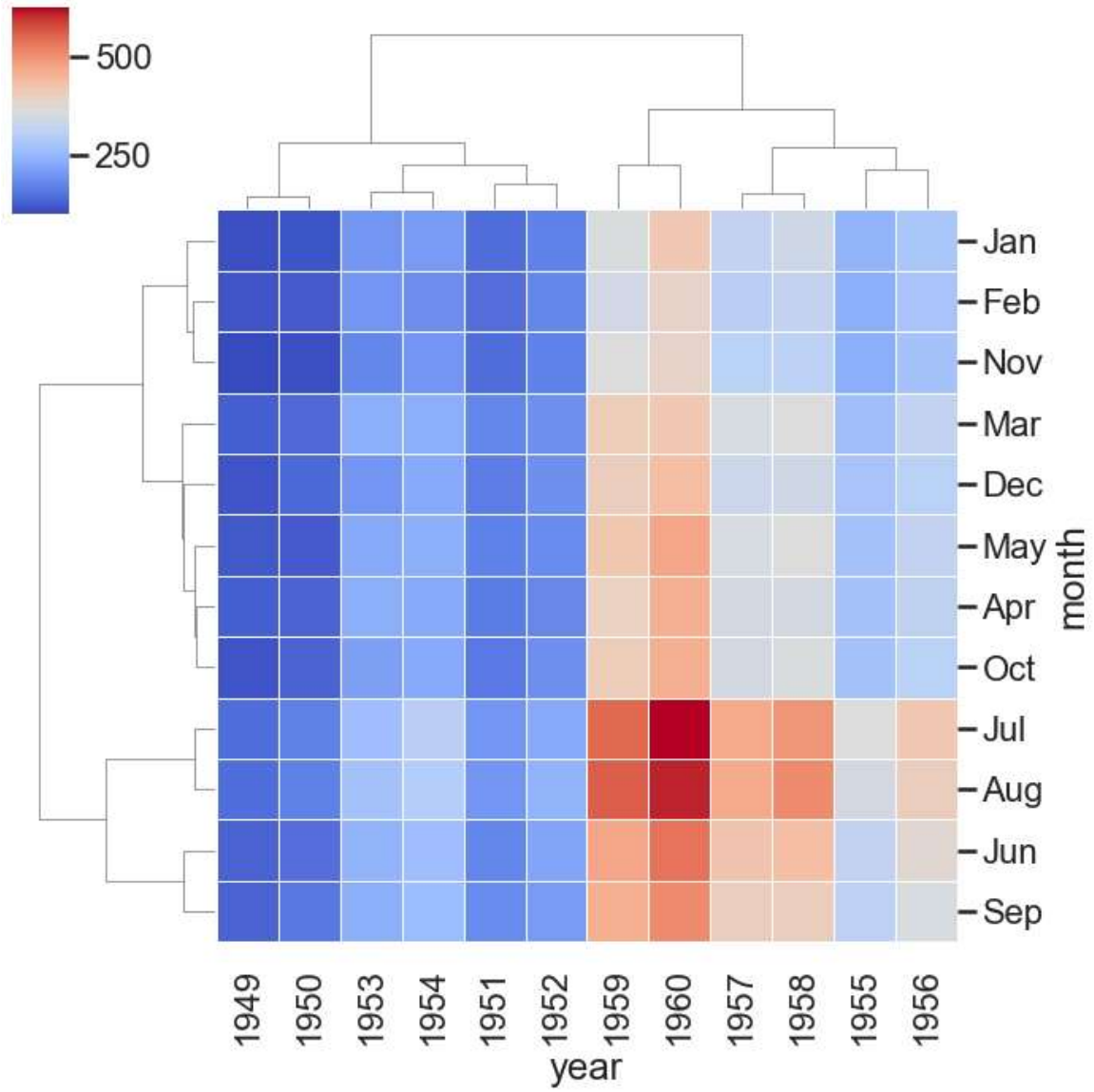
Cluster map

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 [30]:
```
sns.set_context("poster")
flights=sns.load_dataset("flights")
flights = flights.pivot("month", "year", "passengers")
sns.clustermap(flights,linewidths=.5,cmap="coolwarm")
```

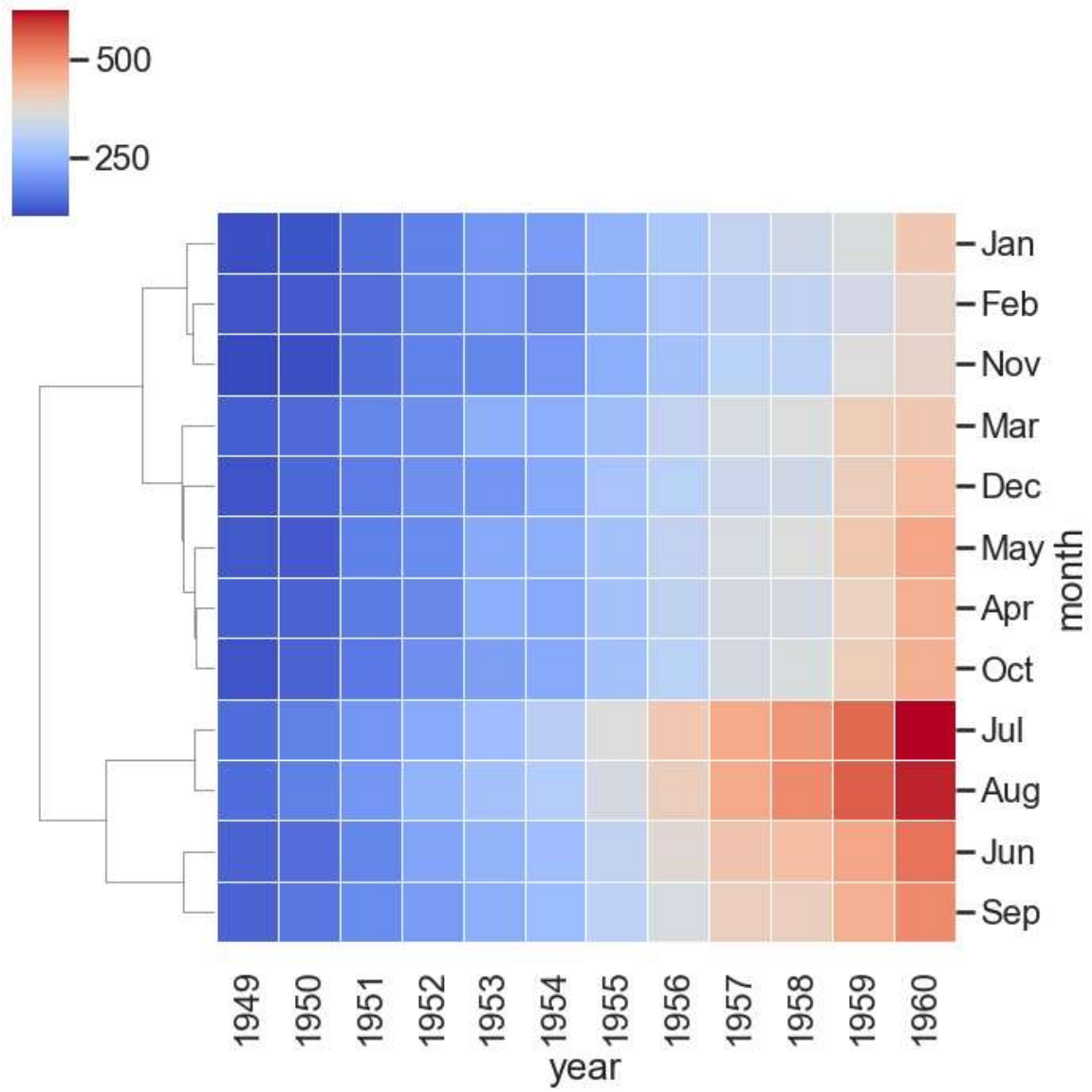
```
C:\Users\heman\AppData\Local\Temp\ipykernel_20552\4046513125.py:3: FutureWa
rning: In a future version of pandas all arguments of DataFrame.pivot will
be keyword-only.
  flights = flights.pivot("month", "year", "passengers")
```

Out[30]: <seaborn.matrix.ClusterGrid at 0x20677f9dde0>



In [43]:
```
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[43]:
```
<seaborn.matrix.ClusterGrid at 0x2067f1bce80>
```

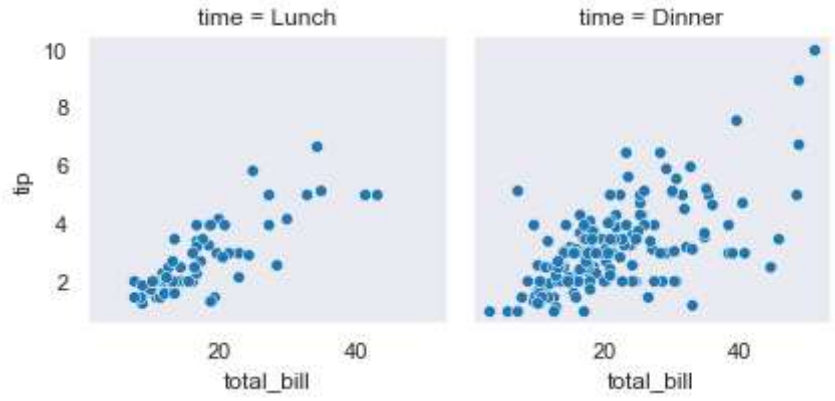


Facetgrid

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 ‘diet’ using the ‘col’ parameter. We can also one more parameter “row” which can help to add one more variable to our plot

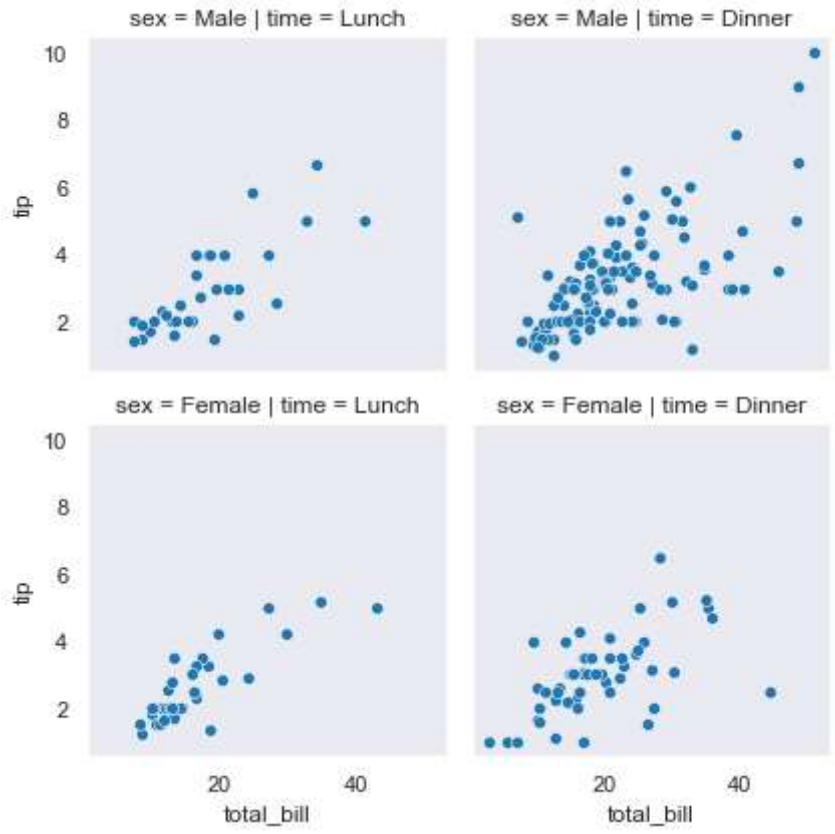
In [45]:
```
sns.set_context("notebook")
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time")
g.map(sns.scatterplot, "total_bill", "tip")
```

Out[45]:
```
<seaborn.axisgrid.FacetGrid at 0x2067dba9b70>
```



In [46]:
```
g = sns.FacetGrid(tips, col="time", row="sex")
g.map(sns.scatterplot, "total_bill", "tip")
```

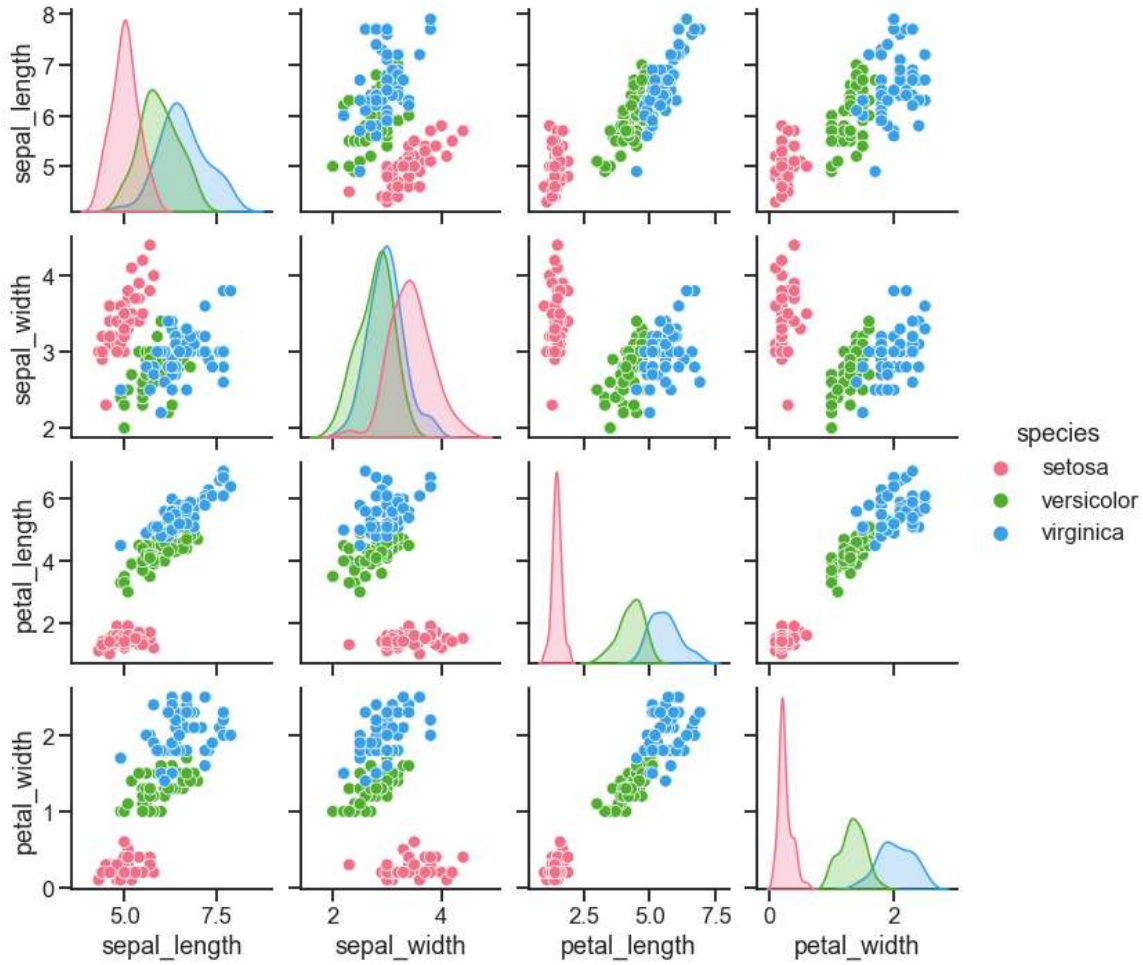
Out[46]:
```
<seaborn.axisgrid.FacetGrid at 0x2067f6417e0>
```



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 [48]:
```
df = sns.load_dataset('iris')
sns.set_style("ticks")
sns.set_context("talk")
sns.pairplot(df,hue = 'species',diag_kind = "kde",kind = "scatter",palette =
plt.show()
```



In [ ]: