In [1]:
```
import seaborn as sns
```

In [2]:
```
df = sns.load_dataset("tips")
df.head()
```

Out[2]:

| | 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 |

### Regression Plot

- It creates a regression line between two parameters and helps to visulize their linear realtionship
- Implot

In [3]:
```
print(dir(sns))
```

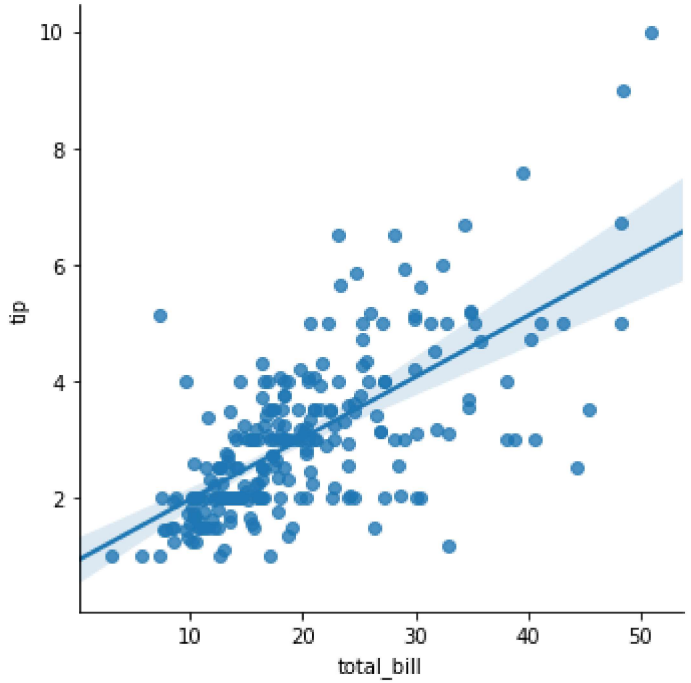
```
['FacetGrid', 'JointGrid', 'PairGrid', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__n
ame__', '__package__', '__path__', '__spec__', '__version__', '_orig_rc_params', 'algorithms', 'axes_style',
'axisgrid', 'barplot', 'blend_palette', 'boxenplot', 'boxplot', 'categorical', 'catplot', 'choose_colorbrewer_
palette', 'choose_cubehelix_palette', 'choose_dark_palette', 'choose_diverging_palette', 'choose_light_palett
e', 'clustermap', 'cm', 'color_palette', 'colors', 'countplot', 'crayon_palette', 'crayons', 'cubehelix_palett
e', 'dark_palette', 'desaturate', 'despine', 'distplot', 'distributions', 'diverging_palette', 'dogplot', 'ext
ernal', 'factorplot', 'get_dataset_names', 'heatmap', 'hls_palette', 'husl_palette', 'jointplot', 'kdeplot',
'light_palette', 'lineplot', 'lmplot', 'load_dataset', 'lvplot', 'matrix', 'miscplot', 'mpl', 'mpl_palette',
'pairplot', 'palettes', 'palplot', 'plotting_context', 'pointplot', 'rcmod', 'regplot', 'regression', 'relatio
nal', 'relplot', 'reset_defaults', 'reset_orig', 'residplot', 'rugplot', 'saturate', 'scatterplot', 'set', 'se
t_color_codes', 'set_context', 'set_hls_values', 'set_palette', 'set_style', 'stripplot', 'swarmplot', 'timese
ries', 'tsplot', 'utils', 'violinplot', 'widgets', 'xkcd_palette', 'xkcd_rgb']
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
```

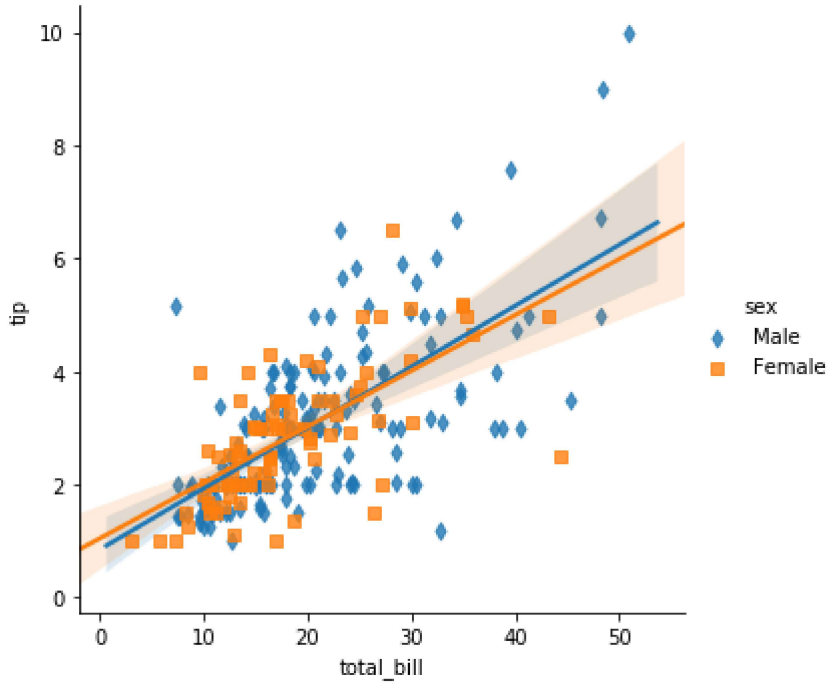
In [5]:
```
sns.lmplot(x="total_bill",y="tip",data=df)
```

Out[5]:
```
<seaborn.axisgrid.FacetGrid at 0x14a7d40aa90>
```



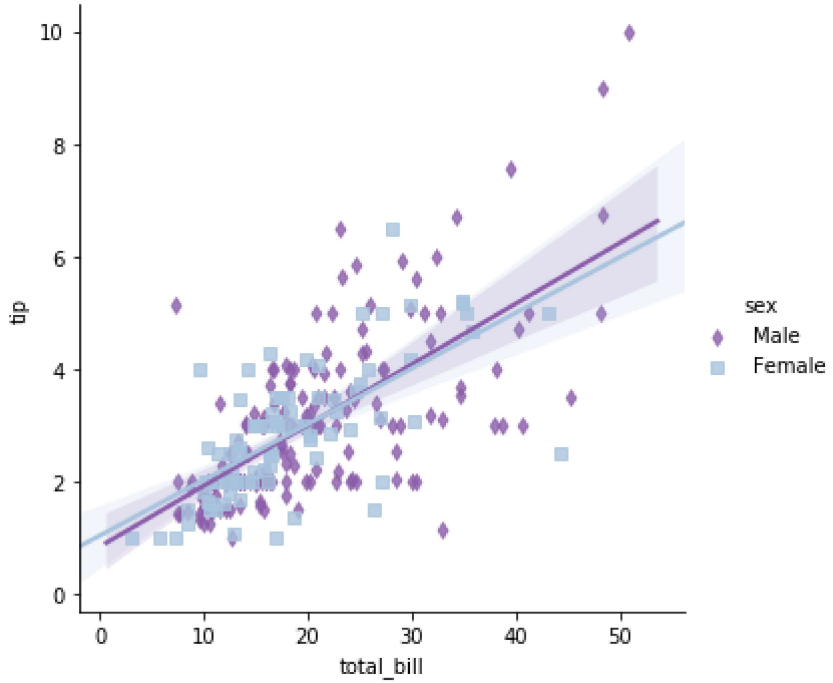
In [7]:
```
sns.lmplot(x="total_bill",y="tip",data=df,hue="sex",markers=["d","s"])
```

Out[7]:
```
<seaborn.axisgrid.FacetGrid at 0x14a7e7ed940>
```



In [13]:
```
sns.lmplot(x="total_bill",y="tip",data=df,hue="sex",markers=["d","s"],palette="BuPu_r",scatter_kws={"s":30})
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x14a7ece83c8>
```

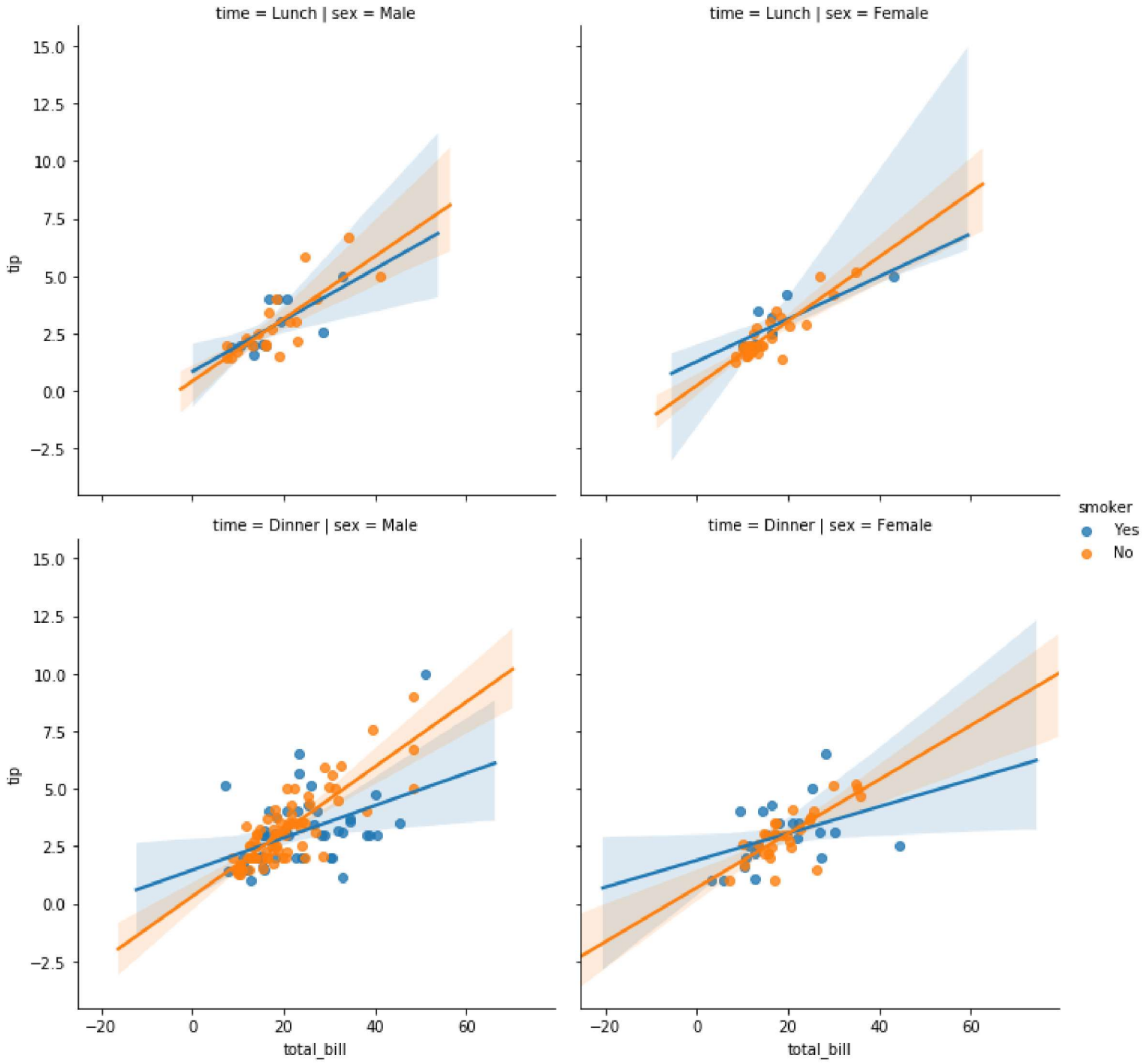


In [14]:
```
help(sns.lmplot)
```

```
e, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, ord
er=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None, y_partial=None, truncate=Fals
e, x_jitter=None, y_jitter=None, scatter_kws=None, line_kws=None, size=None)
    Plot data and regression model fits across a FacetGrid.

    This function combines :func:`regplot` and :class:`FacetGrid`. It is
    intended as a convenient interface to fit regression models across
    conditional subsets of a dataset.

    When thinking about how to assign variables to different facets, a general
    rule is that it makes sense to use ``hue`` for the most important
    comparison, followed by ``col`` and ``row``. However, always think about
    your particular dataset and the goals of the visualization you are
    creating.

    There are a number of mutually exclusive options for estimating the
    regression model. See the :ref:`tutorial <regression_tutorial>` for more
    information.

    The parameters to this function span most of the options in
```

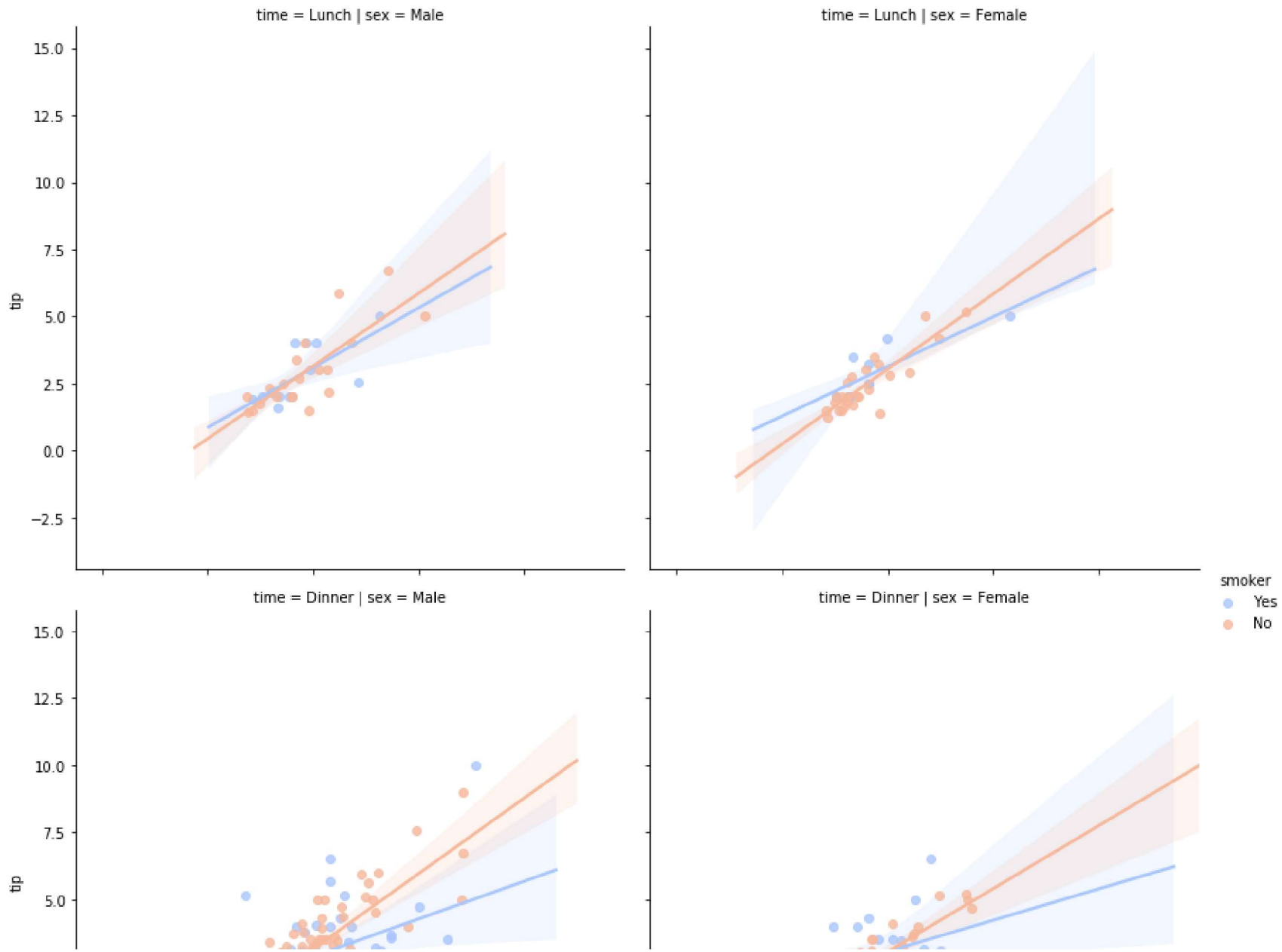
In [15]:
```
# Displaying Multiple plots
sns.lmplot(x="total_bill",y="tip",data=df,col="sex",row="time",hue="smoker")
```

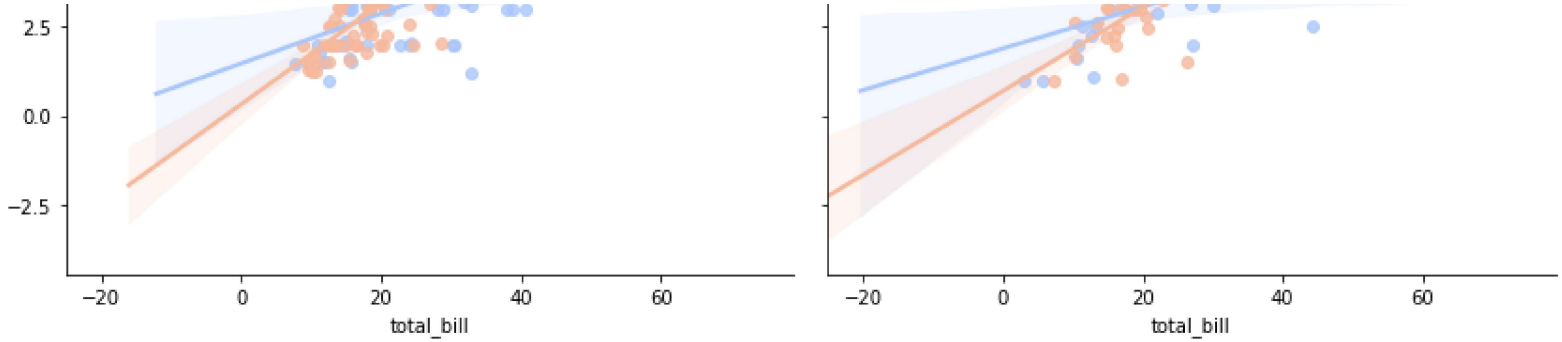
Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x14a7edb0550>
```


time = Lunch | sex = Male
time = Lunch | sex = Female
time = Dinner | sex = Male
time = Dinner | sex = Female
tip
total_bill
smoker
Yes
No

In [16]:
```
# Displaying Multiple plots
sns.lmplot(x="total_bill",y="tip",data=df,col="sex",row="time",hue="smoker",height=6,palette="coolwarm")
```

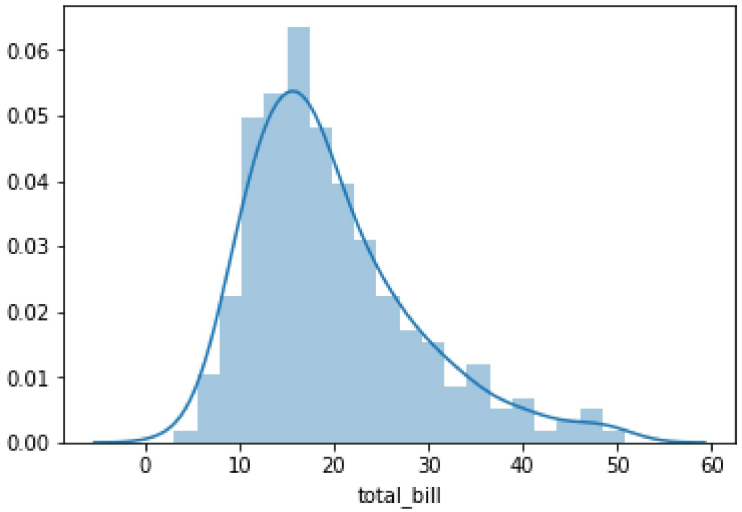
Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x14a7f181978>
```






In [24]:
```
# bin represent how many falls in data
sns.distplot(df["total_bill"],bins=20)
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a7fa3f1d0>
```




In [ ]:
```
# Matrix plot and heatmap
```

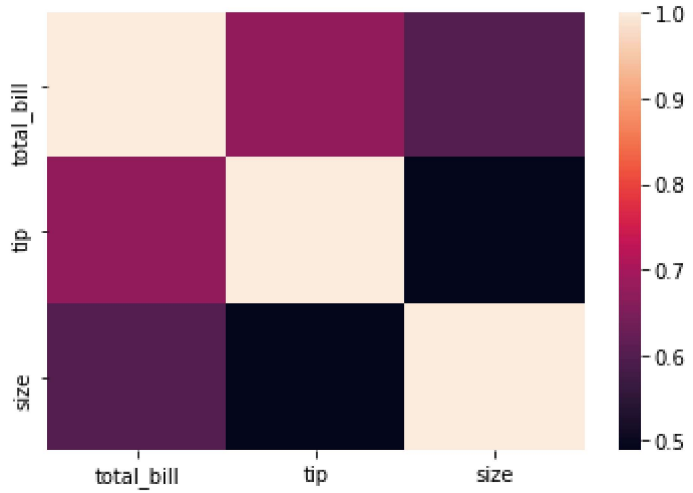
In [25]:
```
tc = df.corr()
tc
```

Out[25]:

| | total_bill | tip | size |
|---|---|---|---|
| **total_bill** | 1.000000 | 0.675734 | 0.598315 |
| **tip** | 0.675734 | 1.000000 | 0.489299 |
| **size** | 0.598315 | 0.489299 | 1.000000 |

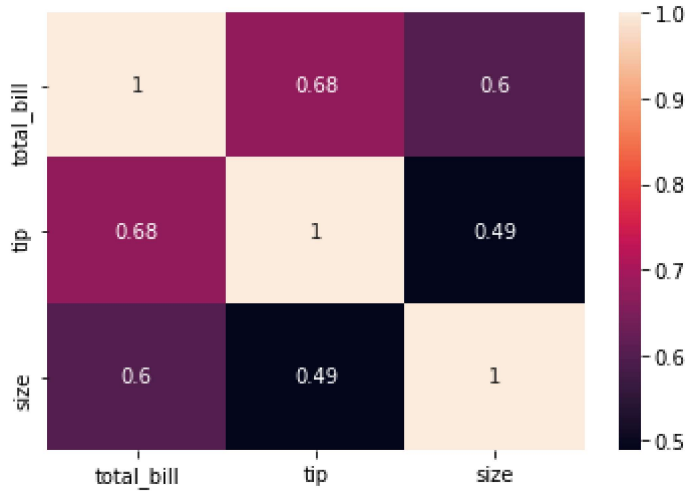
In [26]:
```
sns.heatmap(tc)
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a7fb28208>
```



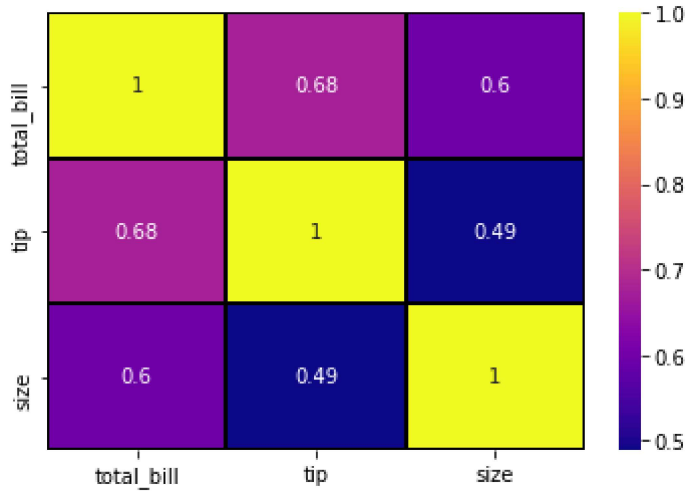
In [27]:
```
sns.heatmap(tc,annot=True)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a7fbc1780>
```



In [28]:
```
sns.heatmap(tc,annot=True,cmap="plasma",linecolor="black",linewidth=1)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a7ff8d208>
```



In [30]:
```
sns.get_dataset_names()
```

```
C:\Users\Alekhya\Anaconda3\lib\site-packages\seaborn\utils.py:376: UserWarning: No parser was explicitly speci
fied, so I'm using the best available HTML parser for this system ("lxml"). This usually isn't a problem, but
if you run this code on another system, or in a different virtual environment, it may use a different parser a
nd behave differently.

The code that caused this warning is on line 376 of the file C:\Users\Alekhya\Anaconda3\lib\site-packages\seab
orn\utils.py. To get rid of this warning, pass the additional argument 'features="lxml"' to the BeautifulSoup
constructor.

  gh_list = BeautifulSoup(http)
```

Out[30]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

In [29]:
```
s = sns.load_dataset("iris")
s.head()
```

Out[29]:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |

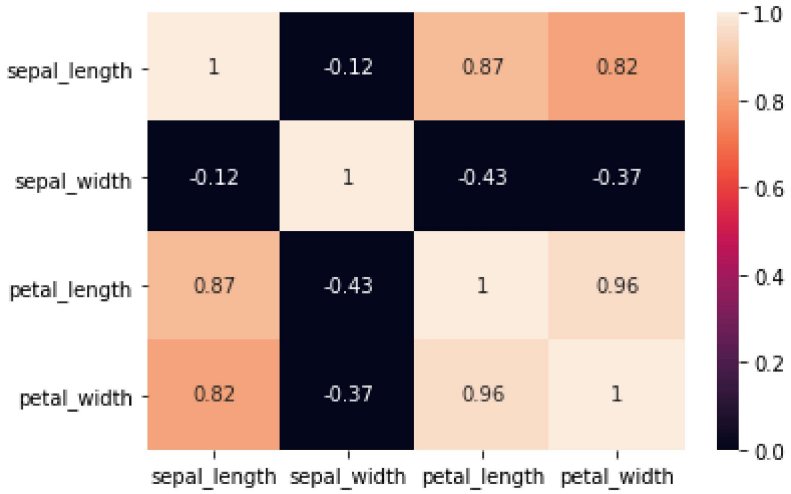
In [31]:
```
s1 = s.corr()
s1
```

Out[31]:

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| **sepal_length** | 1.000000 | -0.117570 | 0.871754 | 0.817941 |
| **sepal_width** | -0.117570 | 1.000000 | -0.428440 | -0.366126 |
| **petal_length** | 0.871754 | -0.428440 | 1.000000 | 0.962865 |
| **petal_width** | 0.817941 | -0.366126 | 0.962865 | 1.000000 |

In [58]:
```
sns.heatmap(s1,annot=True,vmin=0,vmax=1)
```

Out[58]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a03c4c2b0>
```



In [37]:
```
help(sns.heatmap)
```

```
    ----------
    data : rectangular dataset
        2D dataset that can be coerced into an ndarray. If a Pandas DataFrame
        is provided, the index/column information will be used to label the
        columns and rows.
    vmin, vmax : floats, optional
        Values to anchor the colormap, otherwise they are inferred from the
        data and other keyword arguments.
    cmap : matplotlib colormap name or object, or list of colors, optional
        The mapping from data values to color space. If not provided, the
        default will depend on whether ``center`` is set.
    center : float, optional
        The value at which to center the colormap when plotting divergant data.
        Using this parameter will change the default ``cmap`` if none is
        specified.
    robust : bool, optional
        If True and ``vmin`` or ``vmax`` are absent, the colormap range is
        computed with robust quantiles instead of the extreme values.
    annot : bool or rectangular dataset, optional
        If True, write the data value in each cell. If an array-like with the
```

In [38]:
```
flights = sns.load_dataset("flights")
flights.head()
```

Out[38]:

| | year | month | passengers |
|---|---|---|---|
| **0** | 1949 | January | 112 |
| **1** | 1949 | February | 118 |
| **2** | 1949 | March | 132 |
| **3** | 1949 | April | 129 |
| **4** | 1949 | May | 121 |

In [44]:
```
f = flights.pivot("year","month","passengers")
f.head()
```

Out[44]:

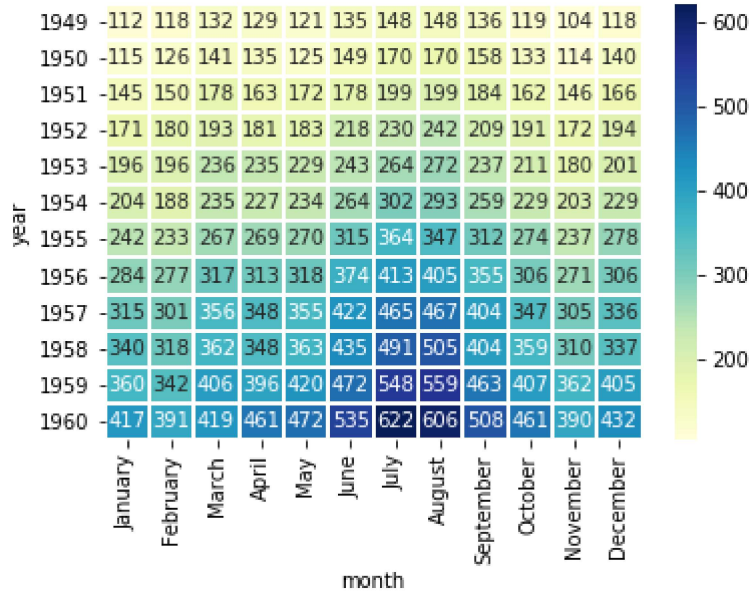
| month / year | January | February | March | April | May | June | July | August | September | October | November | December |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1949** | 112 | 118 | 132 | 129 | 121 | 135 | 148 | 148 | 136 | 119 | 104 | 118 |
| **1950** | 115 | 126 | 141 | 135 | 125 | 149 | 170 | 170 | 158 | 133 | 114 | 140 |
| **1951** | 145 | 150 | 178 | 163 | 172 | 178 | 199 | 199 | 184 | 162 | 146 | 166 |
| **1952** | 171 | 180 | 193 | 181 | 183 | 218 | 230 | 242 | 209 | 191 | 172 | 194 |
| **1953** | 196 | 196 | 236 | 235 | 229 | 243 | 264 | 272 | 237 | 211 | 180 | 201 |

In [45]:
```
f.shape
```

Out[45]:
```
(12, 12)
```

In [48]:
```
sns.heatmap(f,annot=True,cmap="YlGnBu",linewidths=1,fmt="d")
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a02b3db38>
```



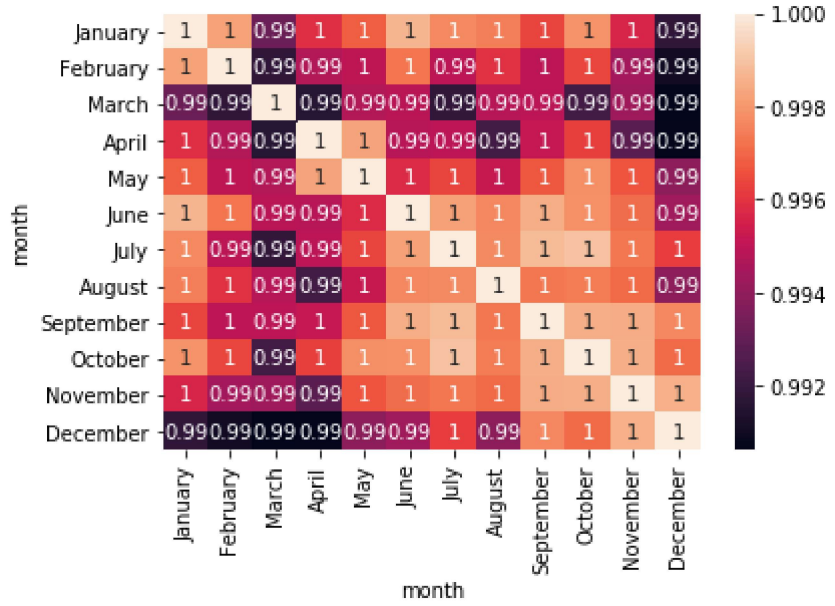
In [49]:
```
f1 = f.corr()
f1
```

Out[49]:

| month | January | February | March | April | May | June | July | August | September | October | November | December |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **month** | | | | | | | | | | | | |
| **January** | 1.000000 | 0.998247 | 0.993215 | 0.995918 | 0.996823 | 0.998700 | 0.997717 | 0.997479 | 0.996372 | 0.997920 | 0.995648 | 0.991821 |
| **February** | 0.998247 | 1.000000 | 0.991813 | 0.994752 | 0.995039 | 0.997296 | 0.994991 | 0.995954 | 0.995011 | 0.996401 | 0.994475 | 0.991053 |
| **March** | 0.993215 | 0.991813 | 1.000000 | 0.991589 | 0.994742 | 0.994923 | 0.991882 | 0.994852 | 0.994969 | 0.992312 | 0.994334 | 0.990635 |
| **April** | 0.995918 | 0.994752 | 0.991589 | 1.000000 | 0.998290 | 0.994910 | 0.994981 | 0.992308 | 0.995188 | 0.996277 | 0.992867 | 0.990684 |
| **May** | 0.996823 | 0.995039 | 0.994742 | 0.998290 | 1.000000 | 0.995804 | 0.996357 | 0.995219 | 0.996724 | 0.997837 | 0.996645 | 0.993936 |
| **June** | 0.998700 | 0.997296 | 0.994923 | 0.994910 | 0.995804 | 1.000000 | 0.998211 | 0.997674 | 0.998527 | 0.997856 | 0.997113 | 0.994374 |
| **July** | 0.997717 | 0.994991 | 0.991882 | 0.994981 | 0.996357 | 0.998211 | 1.000000 | 0.997756 | 0.998808 | 0.998987 | 0.997343 | 0.996202 |
| **August** | 0.997479 | 0.995954 | 0.994852 | 0.992308 | 0.995219 | 0.997674 | 0.997756 | 1.000000 | 0.997334 | 0.997429 | 0.997102 | 0.993931 |
| **September** | 0.996372 | 0.995011 | 0.994969 | 0.995188 | 0.996724 | 0.998527 | 0.998808 | 0.997334 | 1.000000 | 0.998562 | 0.998478 | 0.997680 |
| **October** | 0.997920 | 0.996401 | 0.992312 | 0.996277 | 0.997837 | 0.997856 | 0.998987 | 0.997429 | 0.998562 | 1.000000 | 0.998563 | 0.997108 |
| **November** | 0.995648 | 0.994475 | 0.994334 | 0.992867 | 0.996645 | 0.997113 | 0.997343 | 0.997102 | 0.998478 | 0.998563 | 1.000000 | 0.998601 |
| **December** | 0.991821 | 0.991053 | 0.990635 | 0.990684 | 0.993936 | 0.994374 | 0.996202 | 0.993931 | 0.997680 | 0.997108 | 0.998601 | 1.000000 |

In [50]:
```
sns.heatmap(f1,annot=True)
```

Out[50]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a013dc668>
```



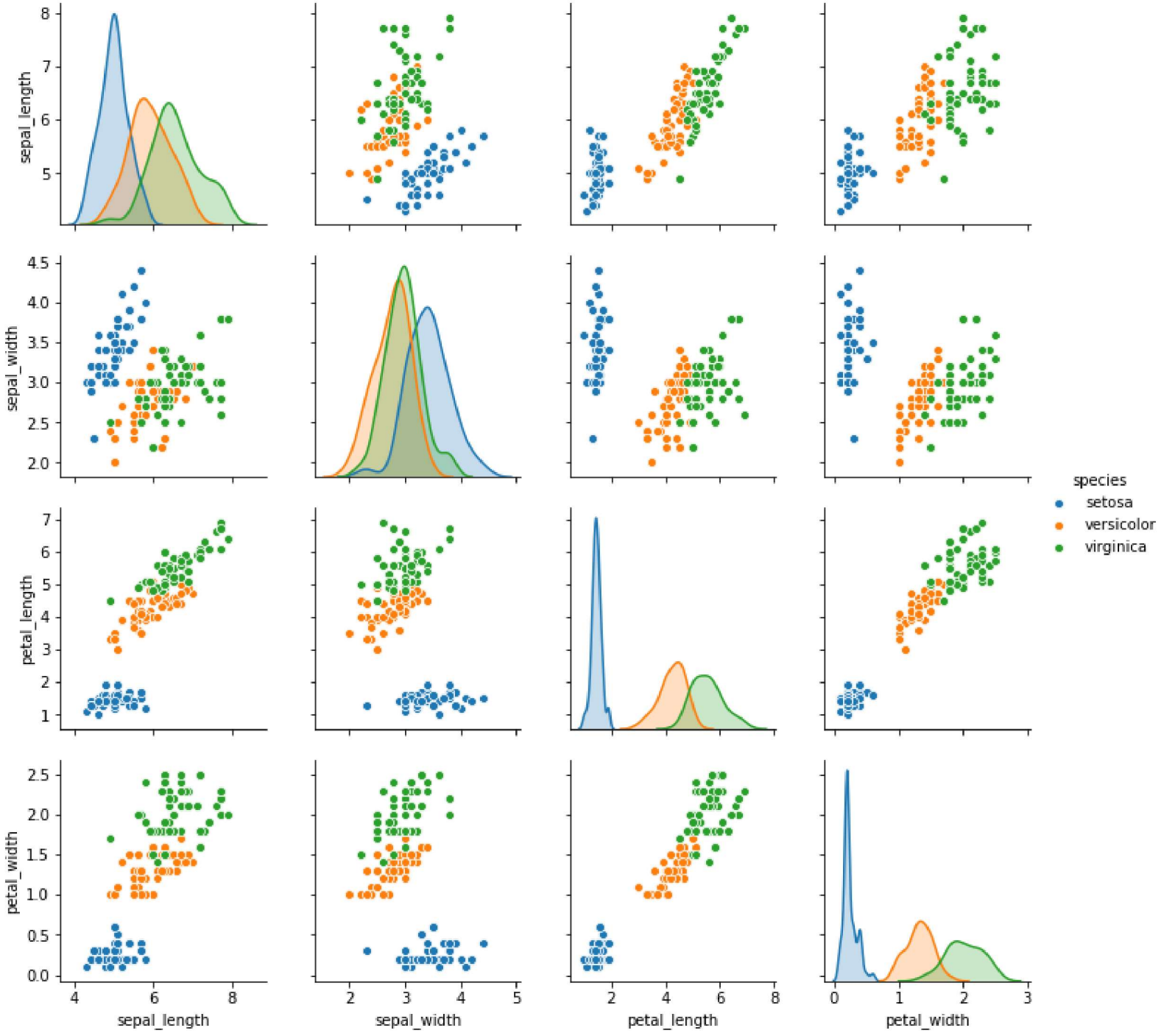
In [53]:
```
s.head()
```

Out[53]:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |

In [52]:
```
#pairplot
sns.pairplot(s,hue="species")
```

Out[52]:
```
<seaborn.axisgrid.PairGrid at 0x14a01491cf8>
```


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

In [54]:
```
help(sns.heatmap)
```

```
o', yticklabels='auto', mask=None, ax=None, **kwargs)
    Plot rectangular data as a color-encoded matrix.

    This is an Axes-level function and will draw the heatmap into the
    currently-active Axes if none is provided to the ``ax`` argument.  Part of
    this Axes space will be taken and used to plot a colormap, unless ``cbar``
    is False or a separate Axes is provided to ``cbar_ax``.

    Parameters
    ----------
    data : rectangular dataset
        2D dataset that can be coerced into an ndarray. If a Pandas DataFrame
        is provided, the index/column information will be used to label the
        columns and rows.
    vmin, vmax : floats, optional
        Values to anchor the colormap, otherwise they are inferred from the
        data and other keyword arguments.
    cmap : matplotlib colormap name or object, or list of colors, optional
        The mapping from data values to color space. If not provided, the
        default will depend on whether ``center`` is set.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: