Data Visualization using Seaborn

```
* Seaborn is a Python data visualization library based on matplotlib
* It provides a high-level interface for drawing attractive and informative statist
ical graphics
* Seaborn is a library for making statistical graphics in Python
* Applications - used in visualising data in Machine learning, data Science
               - statistical aggregation to produce informative plots
```

Color specifications

- **hue is shade of color & color appearance parameter**
- **saturation is intensity of a colour**
- **lightness - fairness**

lightness n saturation max value is 1

1. **Color Palette**
   - **This function provides an interface to many (though not all) of the possible ways you can generate colors in seaborn**
2. **Plotting with categorical data**

   Categorical scatterplots:

   ```
   * stripplot() (with kind="strip"; the default)

   * swarmplot() (with kind="swarm")
   ```

   Categorical distribution plots:

   ```
   * boxplot() (with kind="box")

   * violinplot() (with kind="violin")
   ```

3. **Joint plot**
4. **Pairplot**
5. **Heatmap**

In [1]:

```
import seaborn as sns
```

2. Plotting with categorical data

- **catplot()**

Categorical scatterplots:

1. strip plot

2. Swarm plot

In [2]:

```
sns.get_dataset_names()
```

```
C:\Users\lavan\anaconda3\lib\site-packages\seaborn\utils.py:384: GuessedAtParserWarning:
No parser was explicitly specified, 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 and behave differe
ntly.

The code that caused this warning is on line 384 of the file C:\Users\lavan\anaconda3\lib
\site-packages\seaborn\utils.py. To get rid of this warning, pass the additional argument
'features="lxml"' to the BeautifulSoup constructor.

  gh_list = BeautifulSoup(http)
```

Out[2]:

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

In [3]:

```
iris = sns.load_dataset("iris")
iris.head()
```

Out[3]:

	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 [4]:

```
iris.shape
```

Out[4]:

```
(150, 5)
```

In [5]:

```
iris["species"].value_counts()
```

Out[5]:

```
virginica     50
versicolor    50
setosa        50
Name: species, dtype: int64
```

In [6]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [7]:

```
iris.describe()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Catplot()

In [8]:

```
iris.columns
```

Out[8]:

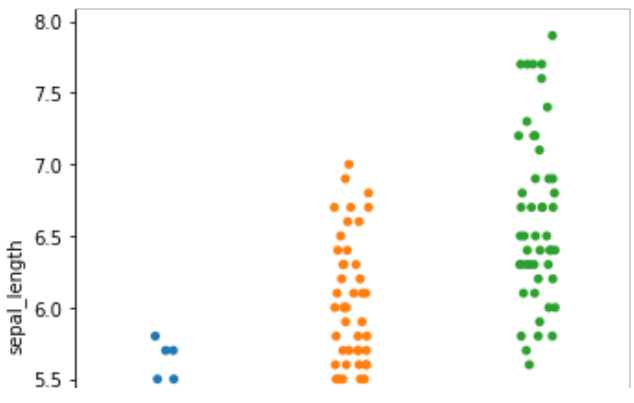
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

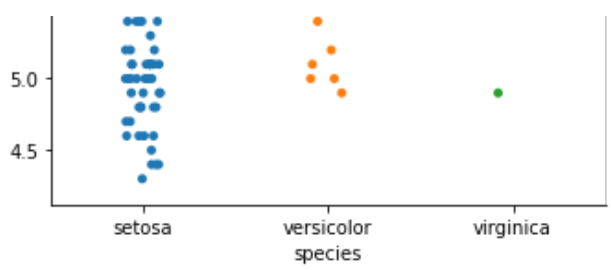
In [9]:

```
sns.catplot(x = "species", y = "sepal_length", data = iris)
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x24570b641f0>
```





Swarm plot

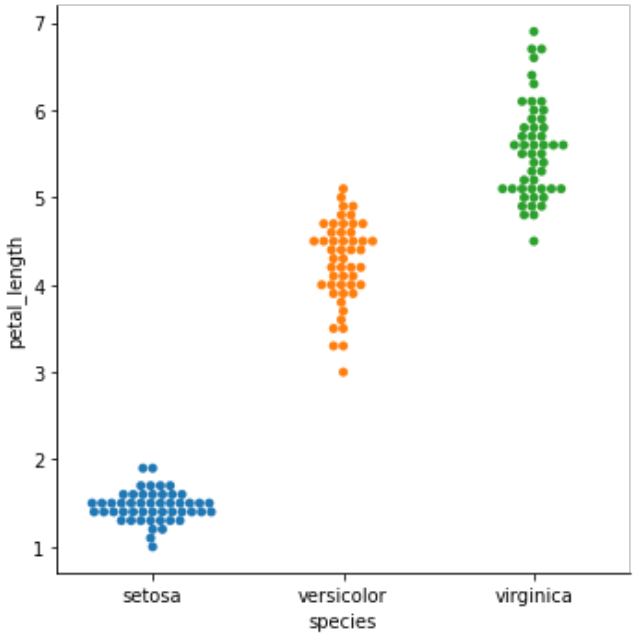
- **Cat data**
- **it adjustas the points along with categorial data**
- **it prevents the overlaping between the data points**

In [10]:

```
sns.catplot(x = "species" , y = "petal_length", data = iris, kind = "swarm")
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x245708df400>
```



Strip plot

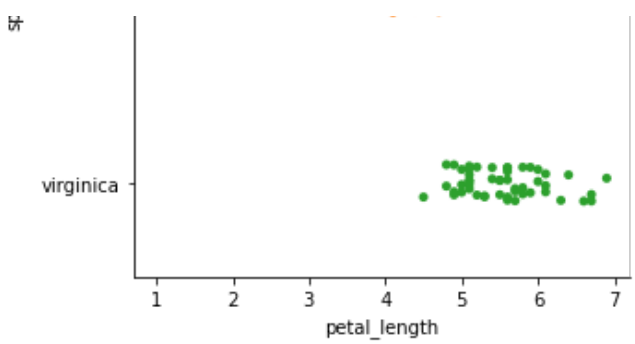
- **it gives relationship between both cat data & numeric data**
- **scatter plot**

In [12]:

```
sns.catplot(y = "species" , x = "petal_length", data = iris, kind = "strip")
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x24570d19df0>
```

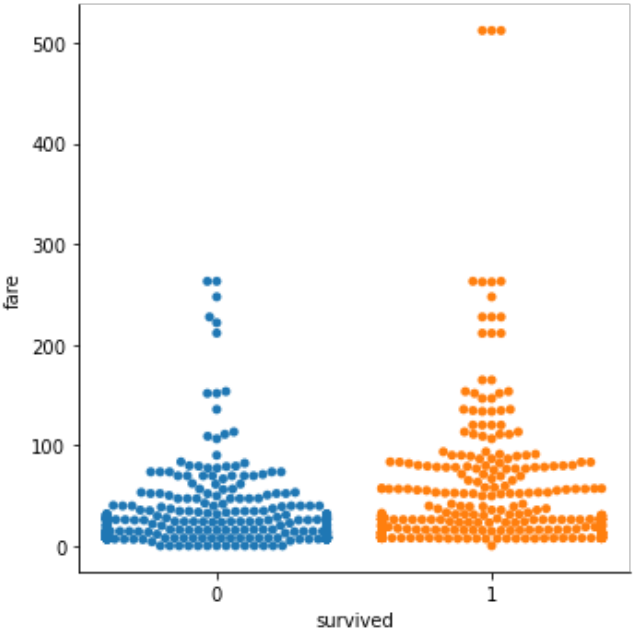


In [14]:

```
titanic = sns.load_dataset('titanic')
sns.catplot(x = 'survived', y ='fare' , data = titanic, kind = 'swarm')
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x245708e6e50>
```



3. Boxplot

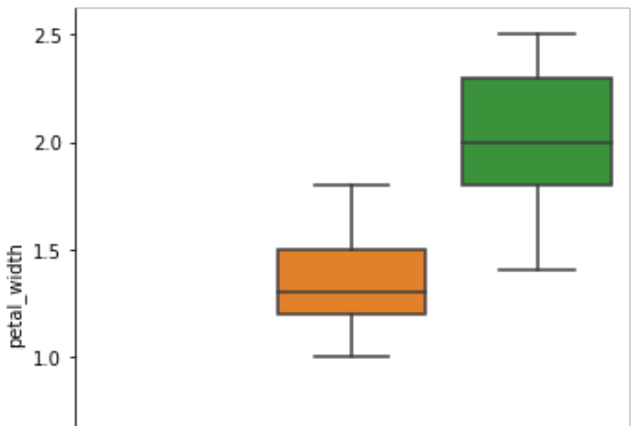
- **mainly used to find relation b/w cat data**
- **the data points gives distribution**
- **min value, 25% quantile, median, 75% quantile, IQR, max values and extrame values (outliers)**

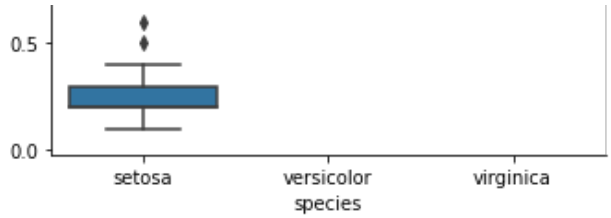
In [15]:

```
sns.catplot(x = "species", y = "petal_width", kind = "box", data = iris)
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x245707a5fd0>
```



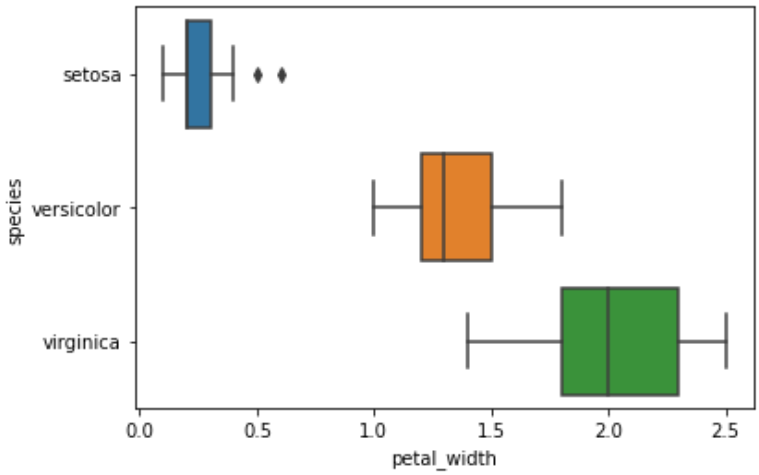


In [16]:

```
sns.boxplot(y = "species", x = "petal_width",  data = iris)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2456eafaee0>
```

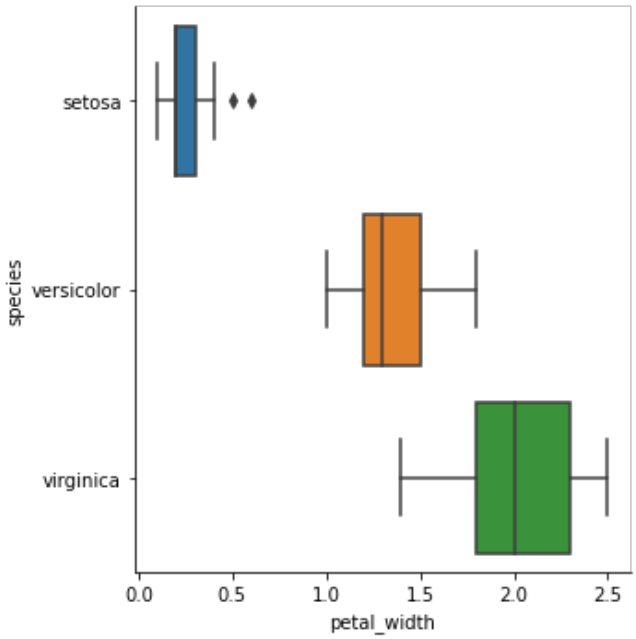


In [19]:

```
sns.catplot(y = "species", x = "petal_width", kind = "box", data = iris, orient = "h")
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x24570a8fd60>
```

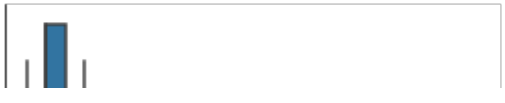


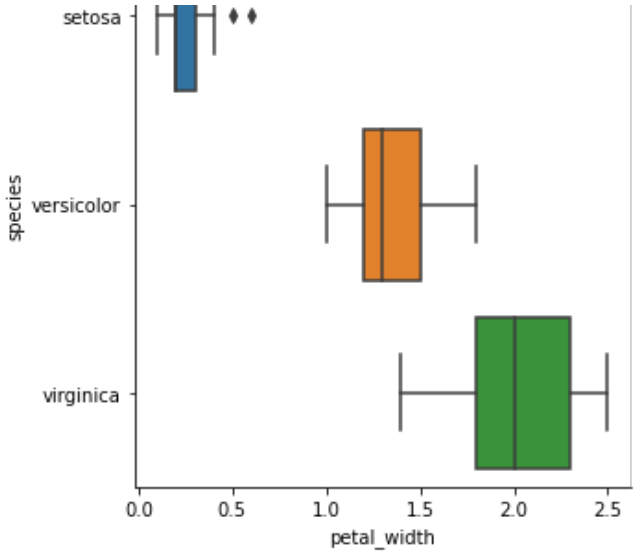
In [21]:

```
sns.catplot(y = "species", x = "petal_width", kind = "box", data = iris, orient = "h")
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x245726ee730>
```



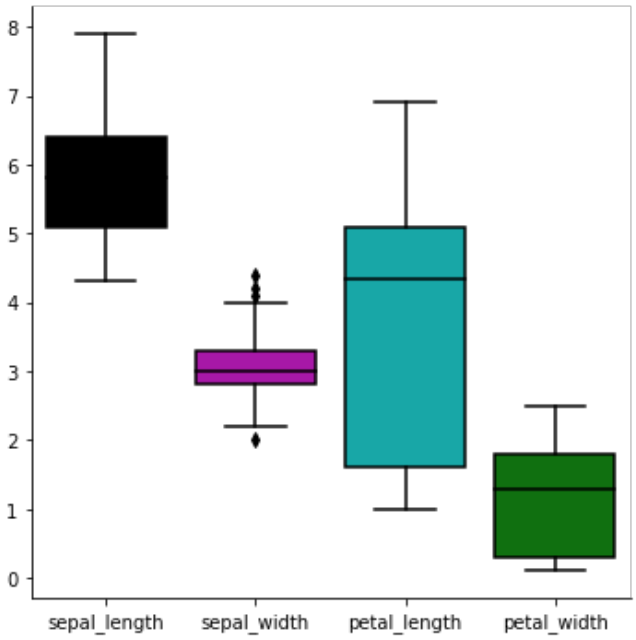


In [23]:

```
sns.catplot(data = iris, kind = "box", palette = ["black", "m","c","g"])
```

Out[23]:

```
<seaborn.axisgrid.FacetGrid at 0x2456eaaacd0>
```



4. violinplot

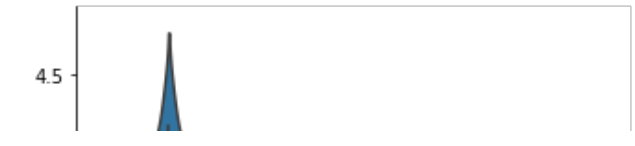
- **used to find Distribution**
- **cat data**
- **gives more info than box plot**
- **same as box**
- **it's the combination of both both plot and KDE**
- **KDE -- it given continues data**

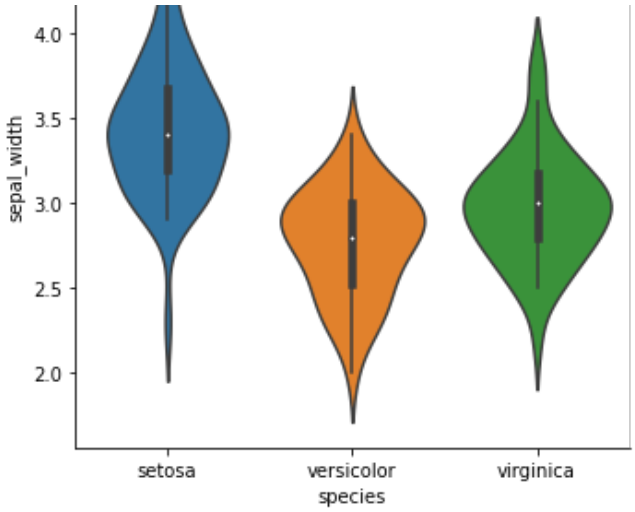
In [25]:

```
sns.catplot(x = "species", y = "sepal_width", data = iris, kind = "violin")
```

Out[25]:

```
<seaborn.axisgrid.FacetGrid at 0x24572627880>
```



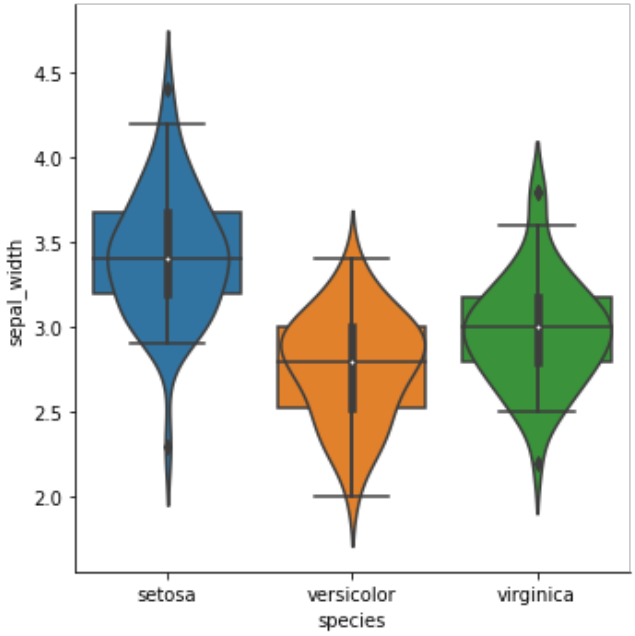


In [27]:

```
sns.catplot(x = "species", y = "sepal_width", data = iris, kind = "violin")
sns.boxplot(x = "species", y = "sepal_width", data = iris)
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24570099b20>
```

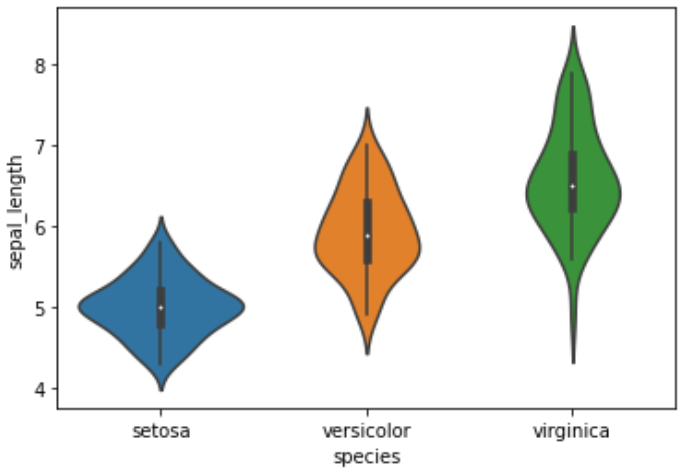


In [28]:

```
sns.violinplot(x = iris.species, y = iris.sepal_length)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2457291adc0>
```



5. Bar plot

- **finding relationship between cat data & numerical data**
- **it is used for multiple observations in each category**

In [30]:

```
titanic.head()
```

Out[30]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alo
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	Fa
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	Tı
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	Tı

In [31]:

```
titanic.shape
```

Out[31]:

```
(891, 15)
```

In [32]:

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.6+ KB
```

In [33]:

```
titanic["embarked"].value_counts()
```

Out[33]:

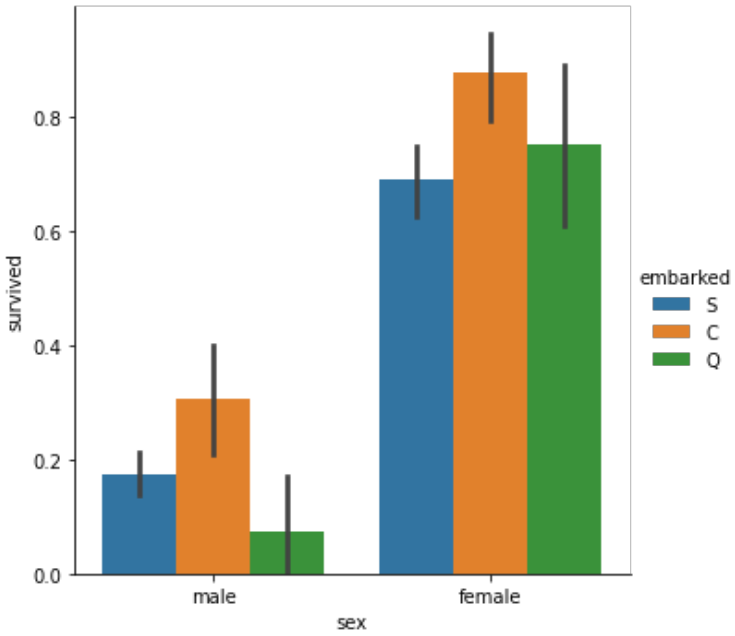
```
S    644
C    168
Q     77
Name: embarked, dtype: int64
```

In [36]:

```
sns.catplot(x = "sex", y = "survived" ,hue = "embarked", kind = "bar", data = titanic)
```

Out[36]:

```
<seaborn.axisgrid.FacetGrid at 0x245729b99a0>
```



6. Joint plot

- **used to combine two plots**
- **it given more info about dataset(statastics)**
- **default it gives scatter plot**

In [37]:

```
iris.head()
```

Out[37]:

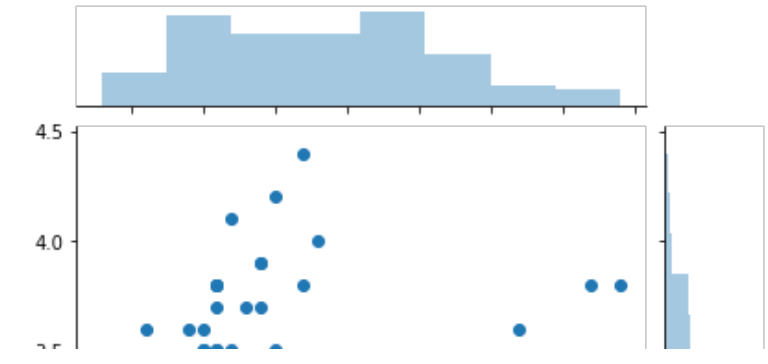
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

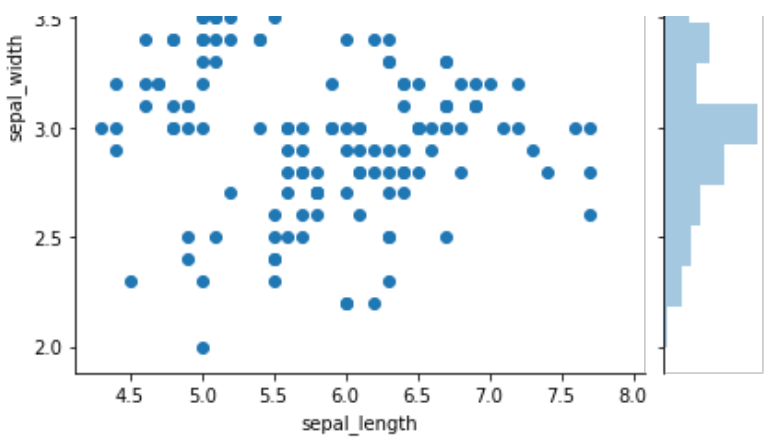
In [38]:

```
sns.jointplot( x= "sepal_length", y = "sepal_width", data = iris)
```

Out[38]:

```
<seaborn.axisgrid.JointGrid at 0x24573d0cc40>
```



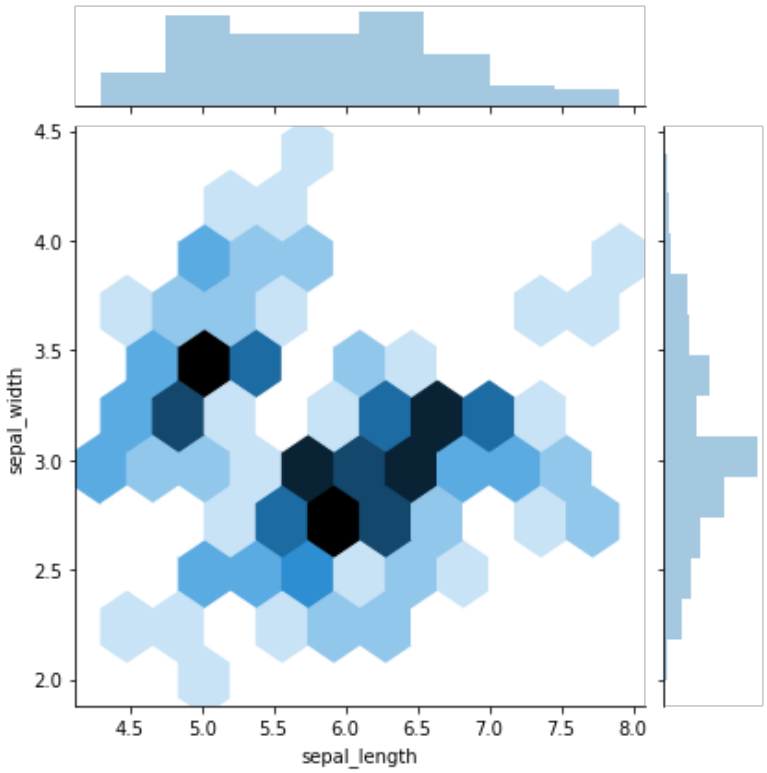


In [40]:

```
sns.jointplot( x= "sepal_length", y = "sepal_width", data = iris , kind = "hex")
```

Out[40]:

```
<seaborn.axisgrid.JointGrid at 0x24573b849a0>
```



In [39]:

```
help(sns.jointplot)
```

```
Help on function jointplot in module seaborn.axisgrid:

jointplot(x, y, data=None, kind='scatter', stat_func=None, color=None, height=6, ratio=5,
space=0.2, dropna=True, xlim=None, ylim=None, joint_kws=None, marginal_kws=None, annot_kw
s=None, **kwargs)
    Draw a plot of two variables with bivariate and univariate graphs.

    This function provides a convenient interface to the :class:`JointGrid`
    class, with several canned plot kinds. This is intended to be a fairly
    lightweight wrapper; if you need more flexibility, you should use
    :class:`JointGrid` directly.

    Parameters
    ----------
    x, y : strings or vectors
        Data or names of variables in ``data``.
    data : DataFrame, optional
        DataFrame when ``x`` and ``y`` are variable names.
    kind : { "scatter" | "reg" | "resid" | "kde" | "hex" }, optional
        Kind of plot to draw.
```

```
    kind of plot to draw.
stat_func : callable or None, optional
    *Deprecated*
color : matplotlib color, optional
    Color used for the plot elements.
height : numeric, optional
    Size of the figure (it will be square).
ratio : numeric, optional
    Ratio of joint axes height to marginal axes height.
space : numeric, optional
    Space between the joint and marginal axes
dropna : bool, optional
    If True, remove observations that are missing from ``x`` and ``y``.
{x, y}lim : two-tuples, optional
    Axis limits to set before plotting.
{joint, marginal, annot}_kws : dicts, optional
    Additional keyword arguments for the plot components.
kwargs : key, value pairings
    Additional keyword arguments are passed to the function used to
    draw the plot on the joint Axes, superseding items in the
    ``joint_kws`` dictionary.

Returns
-------
grid : :class:`JointGrid`
    :class:`JointGrid` object with the plot on it.

See Also
--------
JointGrid : The Grid class used for drawing this plot. Use it directly if
            you need more flexibility.

Examples
--------

Draw a scatterplot with marginal histograms:

.. plot::
    :context: close-figs

    >>> import numpy as np, pandas as pd; np.random.seed(0)
    >>> import seaborn as sns; sns.set(style="white", color_codes=True)
    >>> tips = sns.load_dataset("tips")
    >>> g = sns.jointplot(x="total_bill", y="tip", data=tips)

Add regression and kernel density fits:

.. plot::
    :context: close-figs

    >>> g = sns.jointplot("total_bill", "tip", data=tips, kind="reg")

Replace the scatterplot with a joint histogram using hexagonal bins:

.. plot::
    :context: close-figs

    >>> g = sns.jointplot("total_bill", "tip", data=tips, kind="hex")

Replace the scatterplots and histograms with density estimates and align
the marginal Axes tightly with the joint Axes:

.. plot::
    :context: close-figs

    >>> iris = sns.load_dataset("iris")
    >>> g = sns.jointplot("sepal_width", "petal_length", data=iris,
    ...                   kind="kde", space=0, color="g")

Draw a scatterplot, then add a joint density estimate:

.. plot::
    :context: close-figs
```

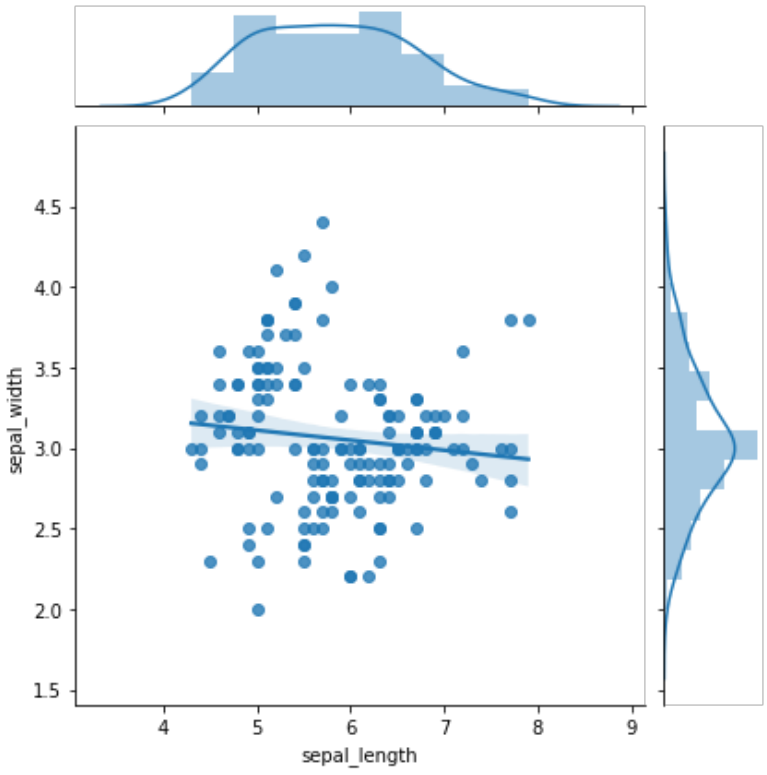
```
    :context: close-figs

    >>> g = (sns.jointplot("sepal_length", "sepal_width",
    ...                    data=iris, color="k")
    ...          .plot_joint(sns.kdeplot, zorder=0, n_levels=6))

Pass vectors in directly without using Pandas, then name the axes:

.. plot::
    :context: close-figs

    >>> x, y = np.random.randn(2, 300)
    >>> g = (sns.jointplot(x, y, kind="hex")
    ...          .set_axis_labels("x", "y"))

Draw a smaller figure with more space devoted to the marginal plots:

.. plot::
    :context: close-figs

    >>> g = sns.jointplot("total_bill", "tip", data=tips,
    ...                   height=5, ratio=3, color="g")

Pass keyword arguments down to the underlying plots:

.. plot::
    :context: close-figs

    >>> g = sns.jointplot("petal_length", "sepal_length", data=iris,
    ...                   marginal_kws=dict(bins=15, rug=True),
    ...                   annot_kws=dict(stat="r"),
    ...                   s=40, edgecolor="w", linewidth=1)
```

In [41]:

```
sns.jointplot( x= "sepal_length", y = "sepal_width", data = iris , kind = "reg")
```

Out[41]:

```
<seaborn.axisgrid.JointGrid at 0x24573e455e0>
```

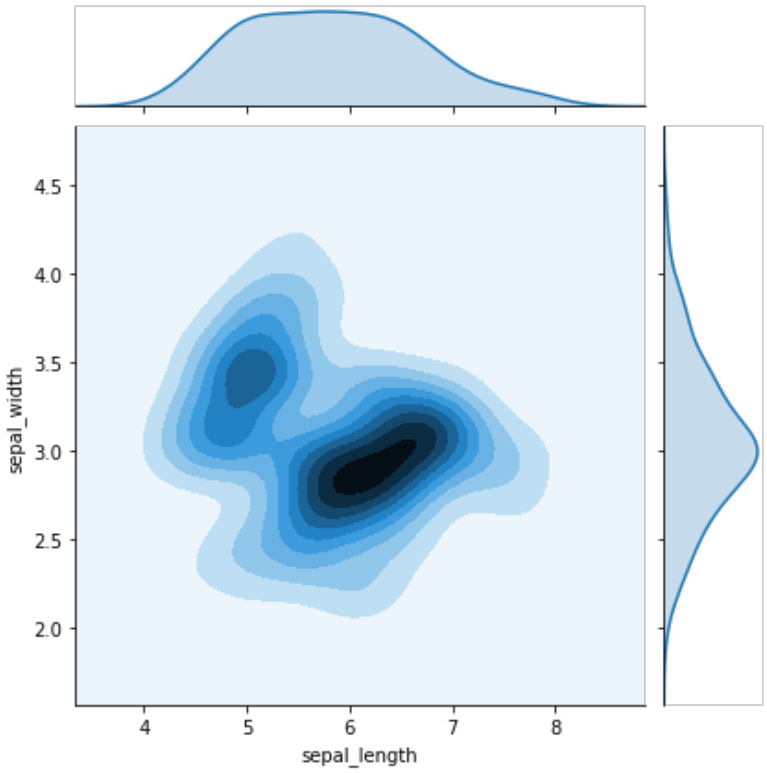


In [42]:

```
sns.jointplot( x= "sepal_length", y = "sepal_width", data = iris , kind = "kde")
```

Out[42]:

```
<seaborn.axisgrid.JointGrid at 0x2457407d370>
```



7. Pairplot

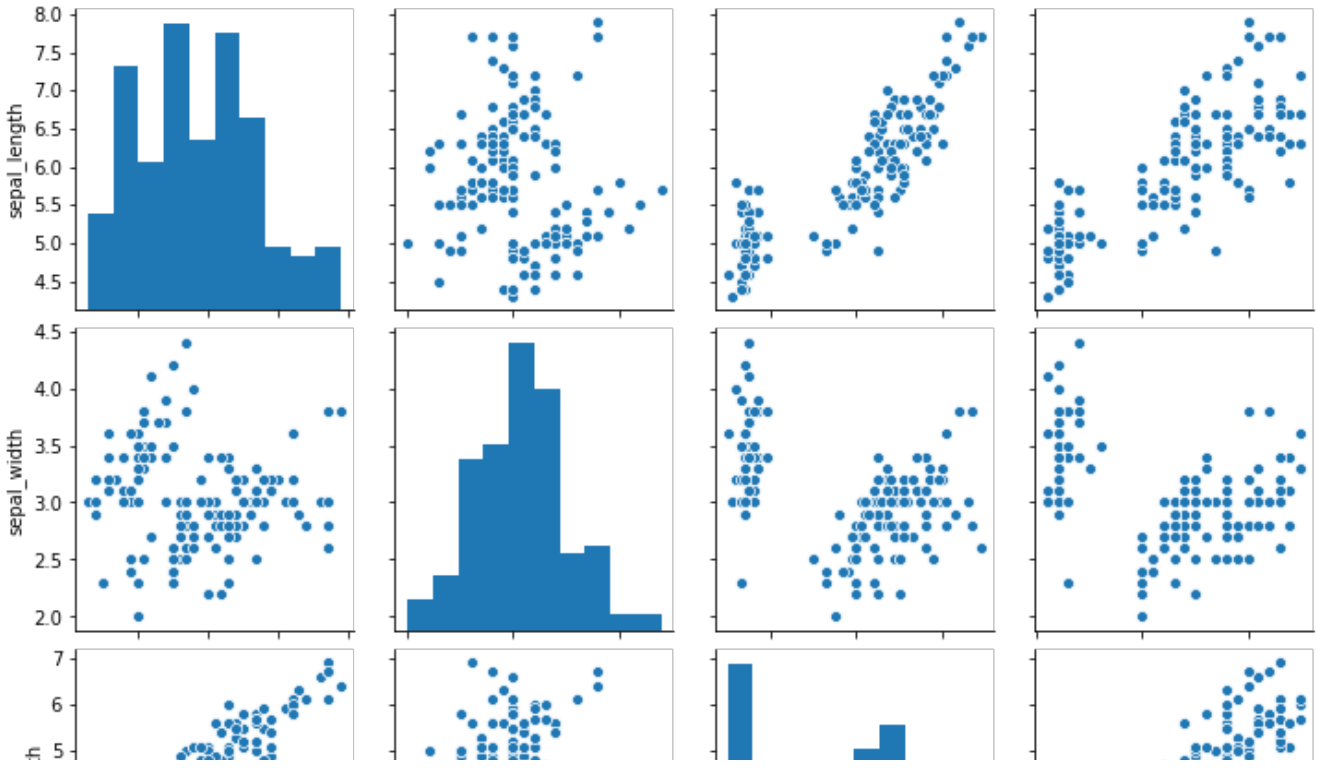
- **it gives multiple plots at a time of the entire data**
- **used to find the distibution**
- **it gives pairwise relation of every feature**
- **also called as Matrics plot**
- **default scatter matrics**

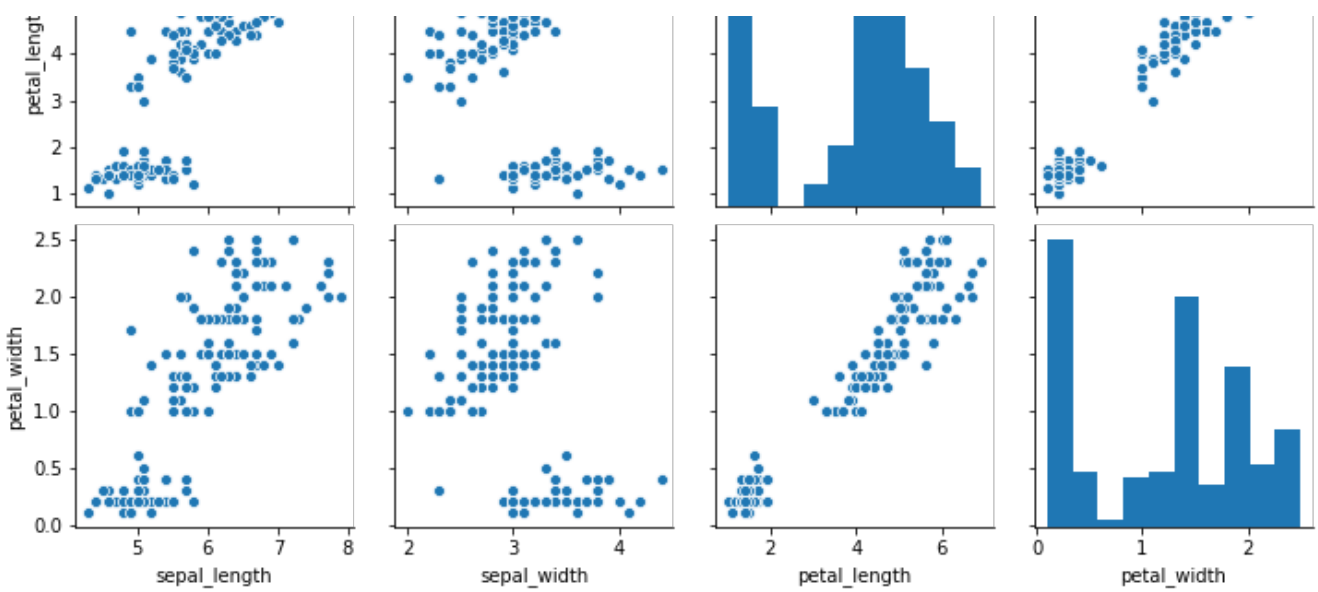
In [43]:

```
sns.pairplot(data = iris, kind = "scatter")
```

Out[43]:

```
<seaborn.axisgrid.PairGrid at 0x245740f6df0>
```



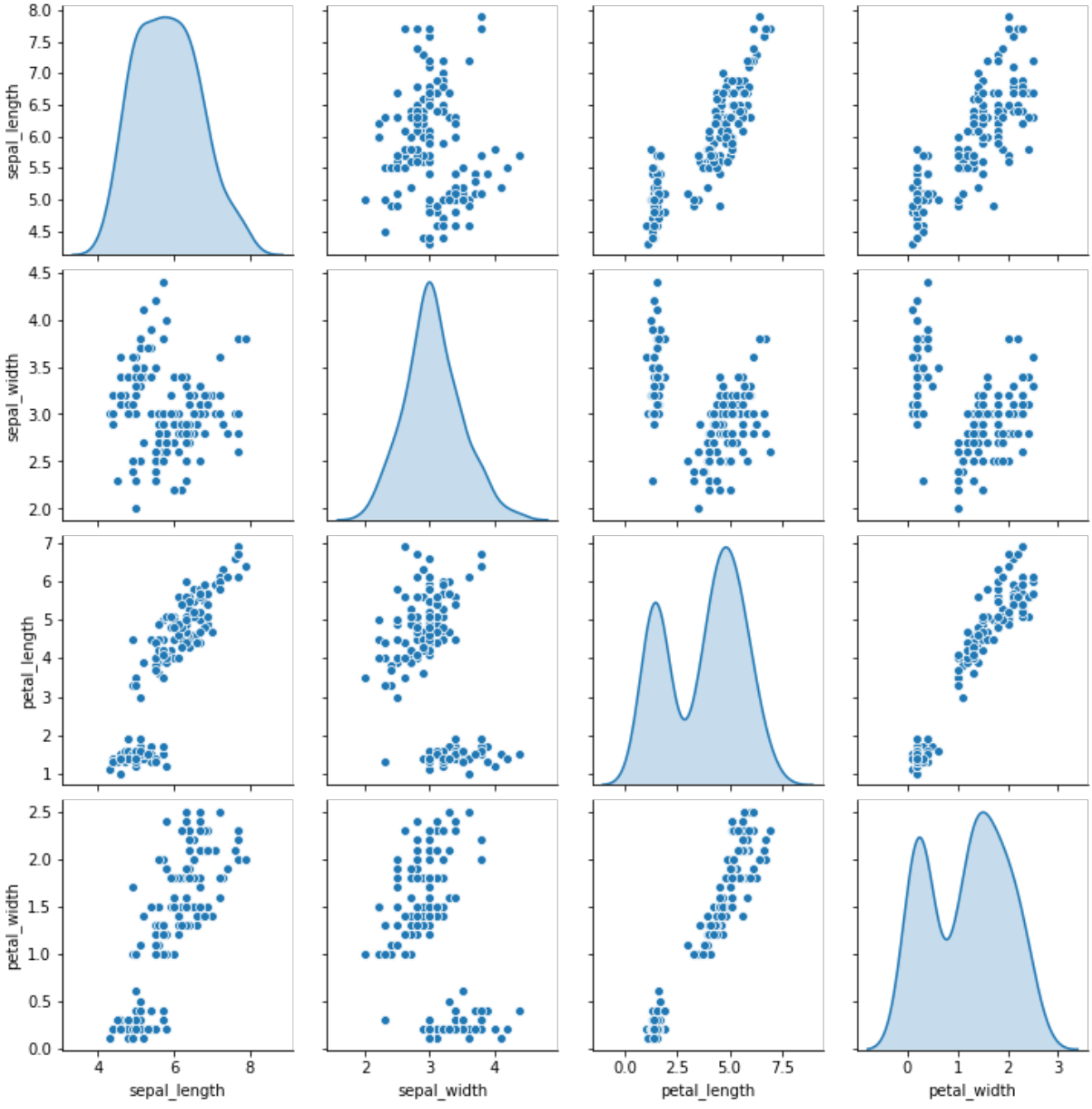


In [44]:

```
sns.pairplot(data = iris, kind = "scatter", diag_kind = "kde")
```

Out[44]:

```
<seaborn.axisgrid.PairGrid at 0x2457481b640>
```

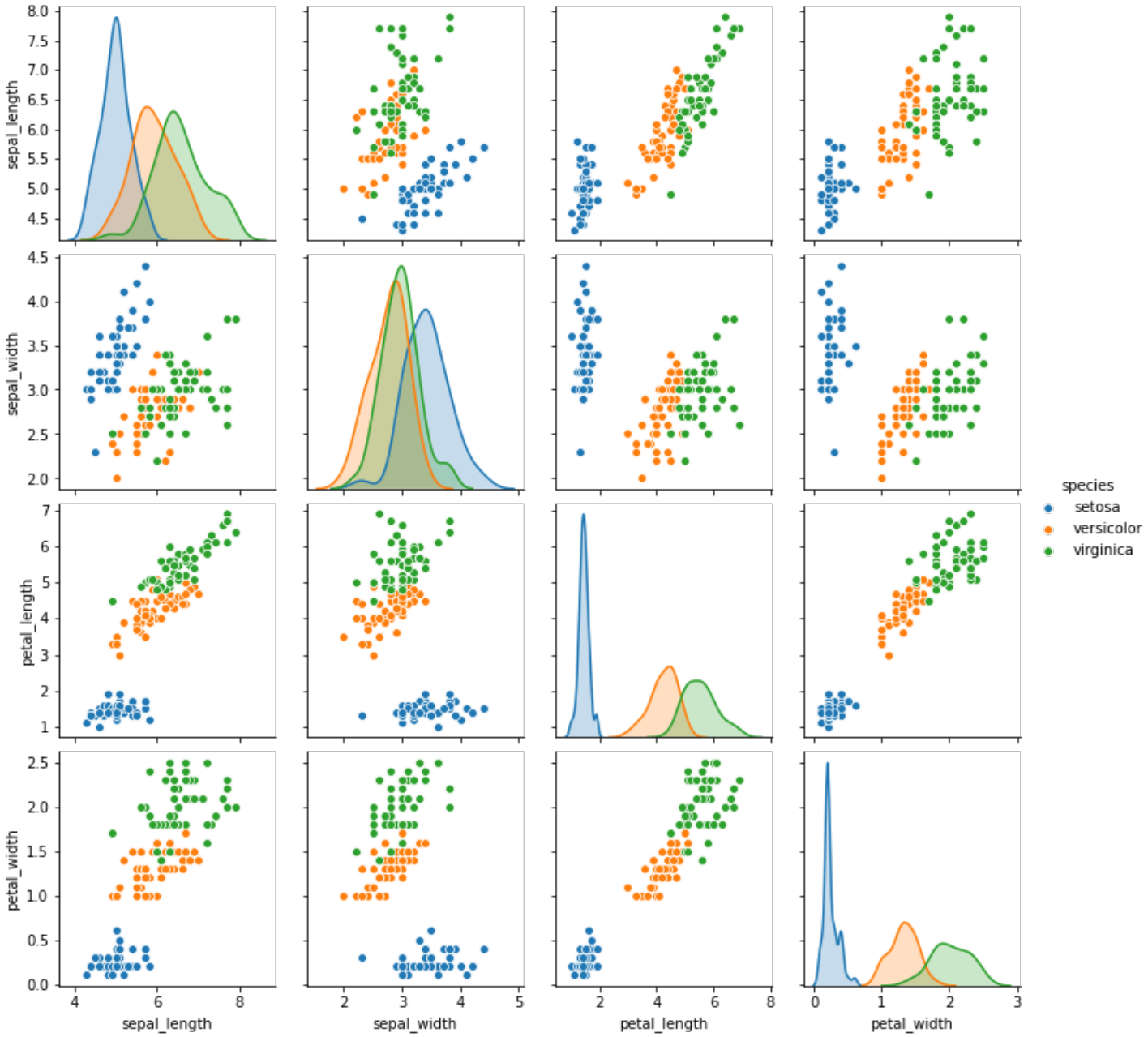


In [45]:

```
sns.pairplot(data = iris, kind = "scatter", diag_kind = "kde", hue = "species")
```

Out[45]:

```
<seaborn.axisgrid.PairGrid at 0x24576163e80>
```

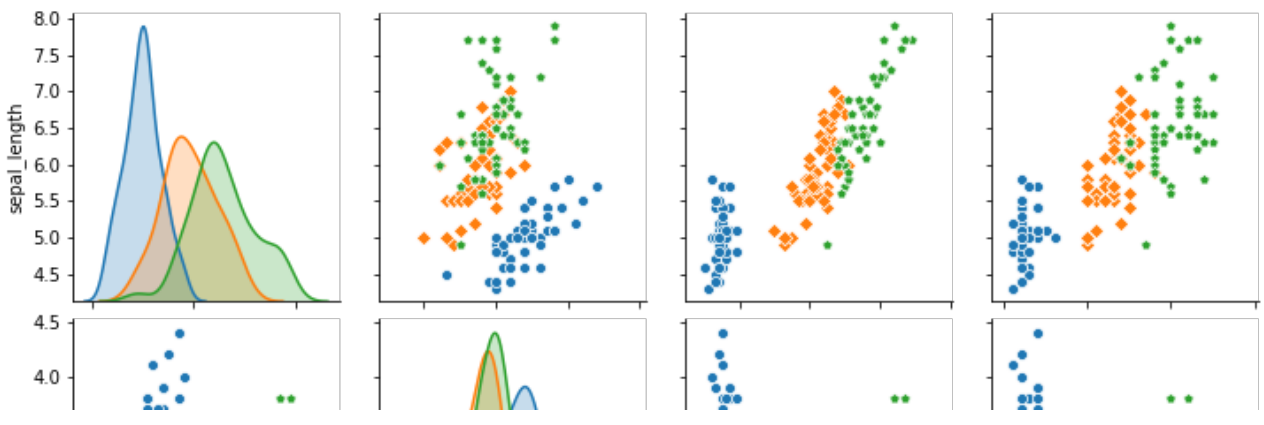


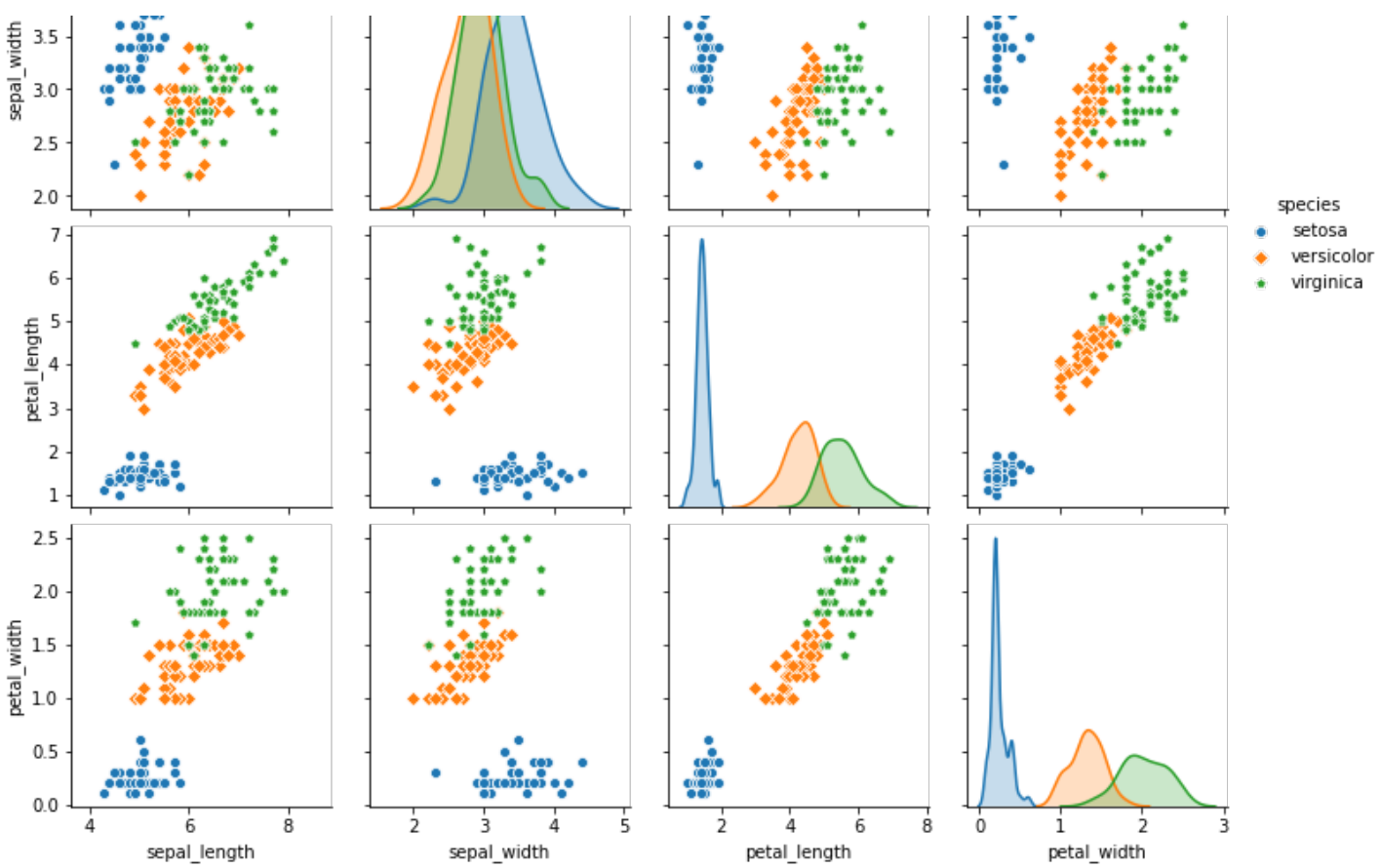
In [46]:

```
sns.pairplot(data = iris, kind = "scatter", diag_kind = "kde", hue = "species", markers
= ["o","D","p"])
```

Out[46]:

```
<seaborn.axisgrid.PairGrid at 0x24576e187c0>
```





8. Heatmap

- It's used to find corelation between each and every feature
  - mutual connection between features
- this is only for numerical type data
- it gives matrics format

In [47]:

```
tips = sns.load_dataset("tips")
tips.head()
```

Out[47]:

	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 [48]:

```
Cor = tips.corr()
Cor
```

Out[48]:

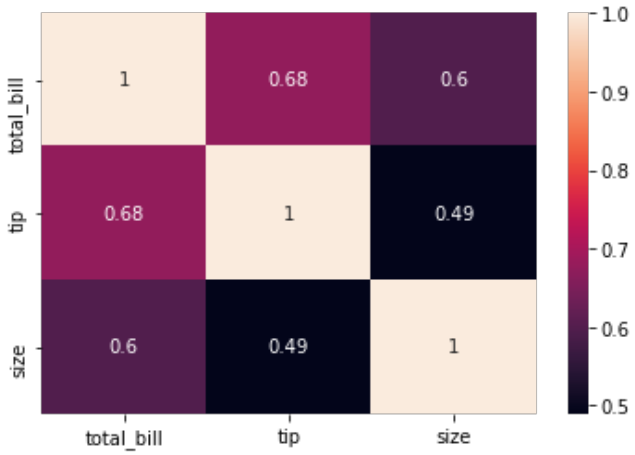
	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [50]:

```
sns.heatmap(Cor, annot = True)
```

Out[50]:

<matplotlib.axes._subplots.AxesSubplot at 0x24577876910>

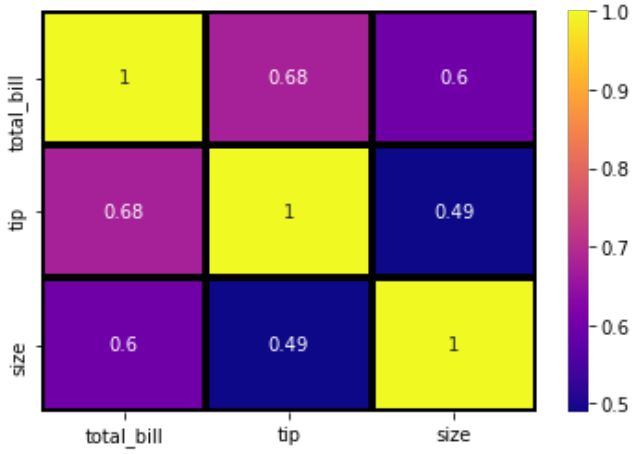


In [53]:

```
sns.heatmap(Cor, annot = True, cmap = "plasma", linecolor = "black", linewidth = 3)
```

Out[53]:

<matplotlib.axes._subplots.AxesSubplot at 0x24577d32ca0>



In [54]:

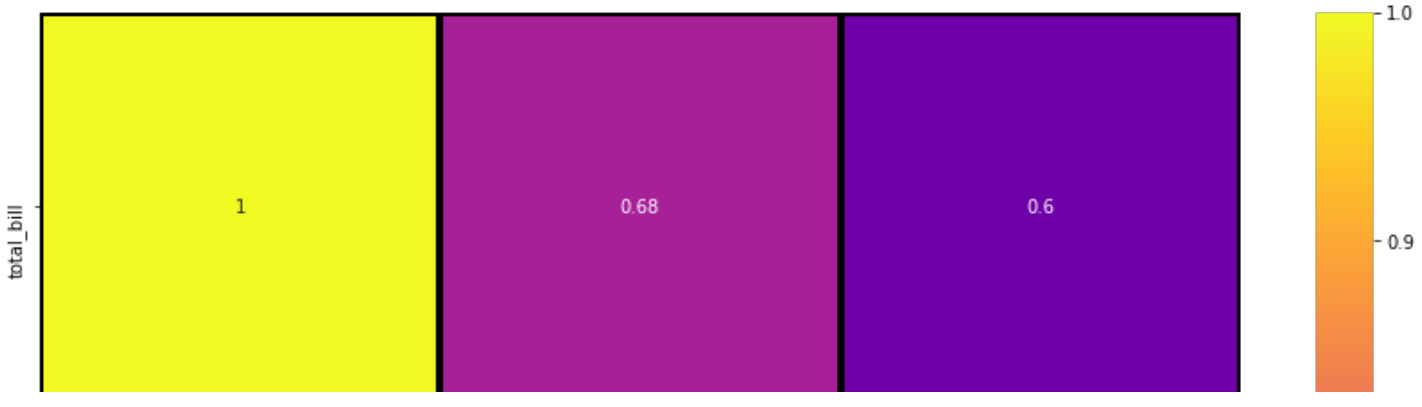
```
import matplotlib.pyplot as plt
```

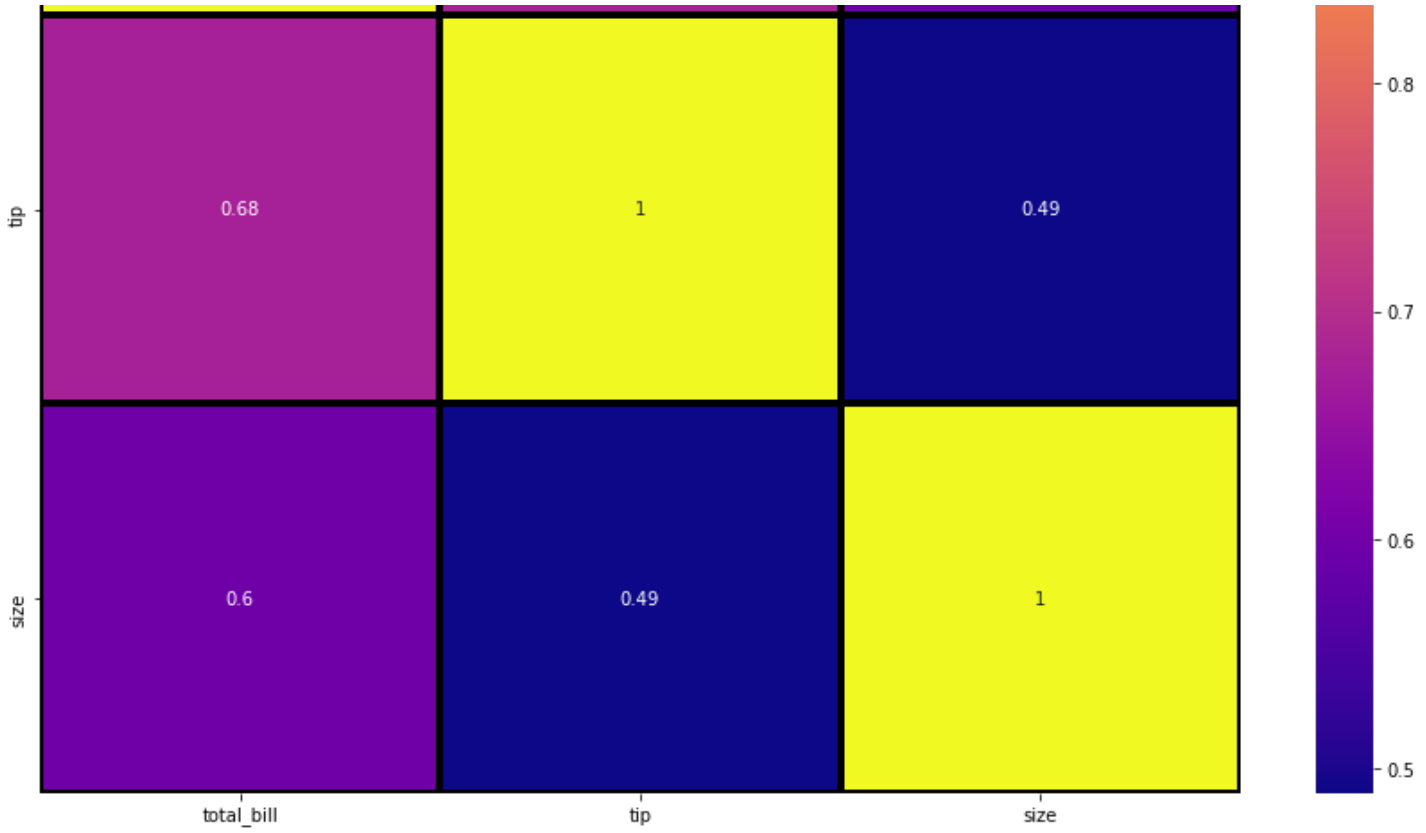
In [56]:

```
plt.figure(figsize = (15,12))
sns.heatmap(Cor, annot = True, cmap = "plasma", linecolor = "black", linewidth = 3)
```

Out[56]:

<matplotlib.axes._subplots.AxesSubplot at 0x24577b87d90>





9. Count plot

- **frequency count**

In [57]:

```
titanic.head()
```

Out[57]:

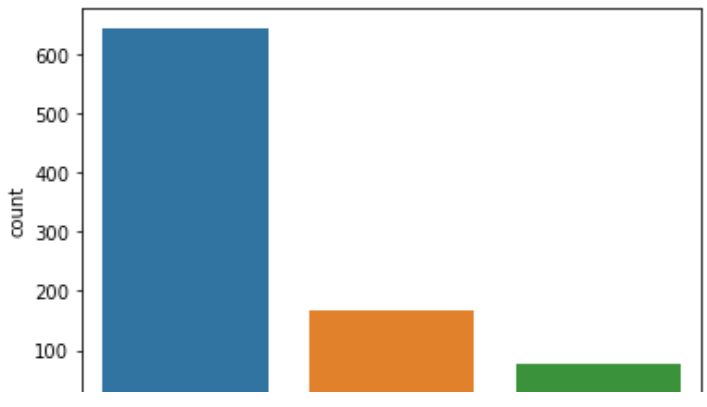
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alo
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	Fa
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	Tr
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	Tr

In [59]:

```
sns.countplot(x = "embarked", data = titanic)
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24578fcec70>
```



In [60]:

```
titanic["embarked"].value_counts()
```

Out[60]:

```
S    644
C    168
Q     77
Name: embarked, dtype: int64
```

10. Point plot

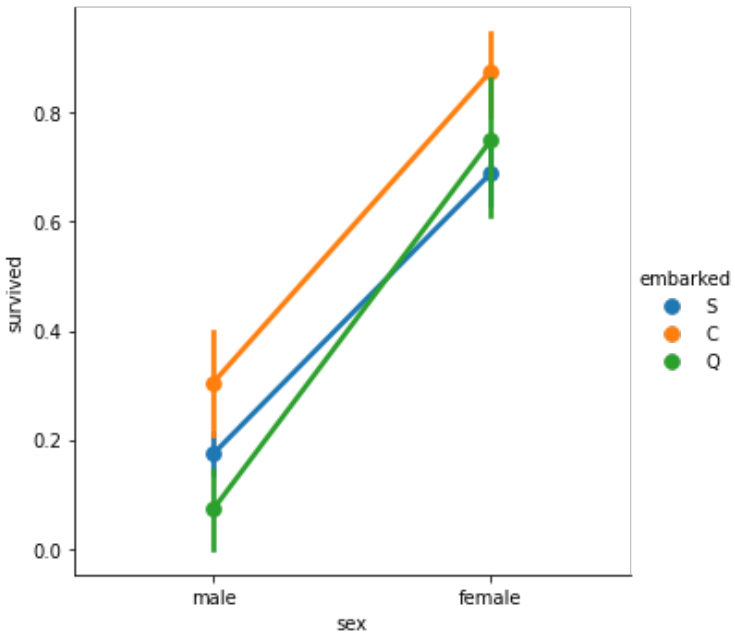
- **same as bar**

In [61]:

```
sns.catplot(x = "sex", y = "survived" ,hue = "embarked", kind = "point", data = titanic)
```

Out[61]:

```
<seaborn.axisgrid.FacetGrid at 0x24577616070>
```



In [ ]: