Seaborn

- Like *Matplotlib*, Seaborn is another 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*.
- Actually, Seaborn compliments and extends Matplotlib.

Seaborn vs Matplotlib

- Seaborn helps resolve the two major problems/frustrations faced by Matplotlib:
  - Default Matplotlib parameters (default style is more attractive in Seaborn).
  - Working with data frames (Seaborn works well with NumPy and Pandas data structures).

Installing Seaborn

- Using Pip:
  - `pip install seaborn`
- Using Conda:
  - `conda install seaborn`

Dependencies

Consider the following dependencies of Seaborn:

- Python
- NumPy
- Pandas
- Matplotlib

Import Seaborn

In [1]:
```
"""
 import convention "sns" comes from the fictional character
 Samuel Norman “Sam” Seaborn on the TV serial drama The West Wing.
 It’s an inside joke by the core developer of Seaborn, namely, Michael Waskom.
"""
import seaborn as sns
```

Seaborn Version

In [2]:
```
sns.__version__
```

Out[2]:
```
'0.11.1'
```

Settings & Import Other Libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# To display plots inside notebook
%matplotlib inline
```

Built-in Datasets

- Seaborn comes with a number of buit-in datasets in the library which downloaded automatically when Seaborn is installed.
- You can use any of these datasets for your learning by loading them.

In [4]:
```
# Available datasets comes with Seaborn
sns.get_dataset_names()
```

Out[4]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

Load Built-in Datasets

- By default, dataset load as Pandas `DataFrame`

In [5]:
```
# For example,
# Load built-in "tips" dataset from Seaborn
tips = sns.load_dataset("tips")
tips
```

Out[5]:

	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
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [6]:

```
# Load built-in "iris" dataset from Seaborn
iris = sns.load_dataset("iris")
iris
```

Out[6]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [7]:

```
# Load built-in "flights" dataset from Seaborn
flights = sns.load_dataset("flights")
flights
```

Out[7]:

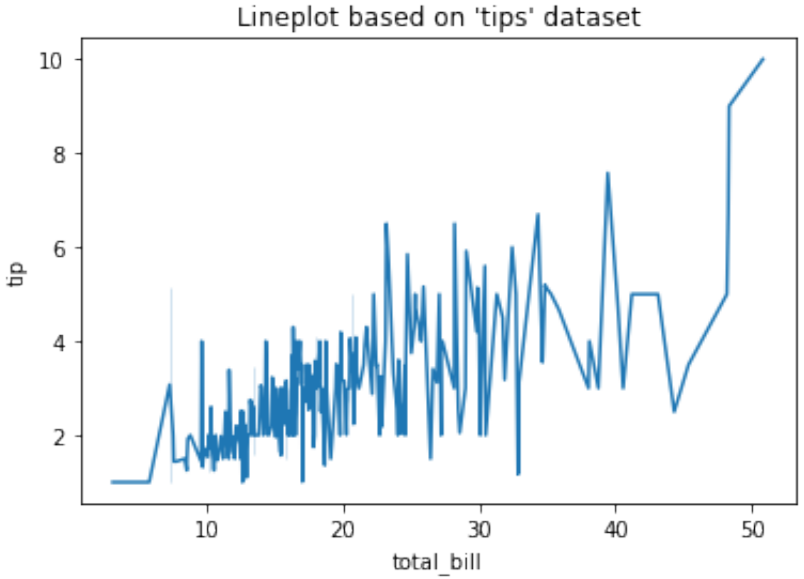
	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121
...	...	...	...
139	1960	Aug	606
140	1960	Sep	508
141	1960	Oct	461
142	1960	Nov	390
143	1960	Dec	432

144 rows × 3 columns

Lineplot

In [8]:

```
sns.lineplot(x="total_bill", y="tip", data=tips)

# Setting title of the plot
plt.title("Lineplot based on 'tips' dataset");
```

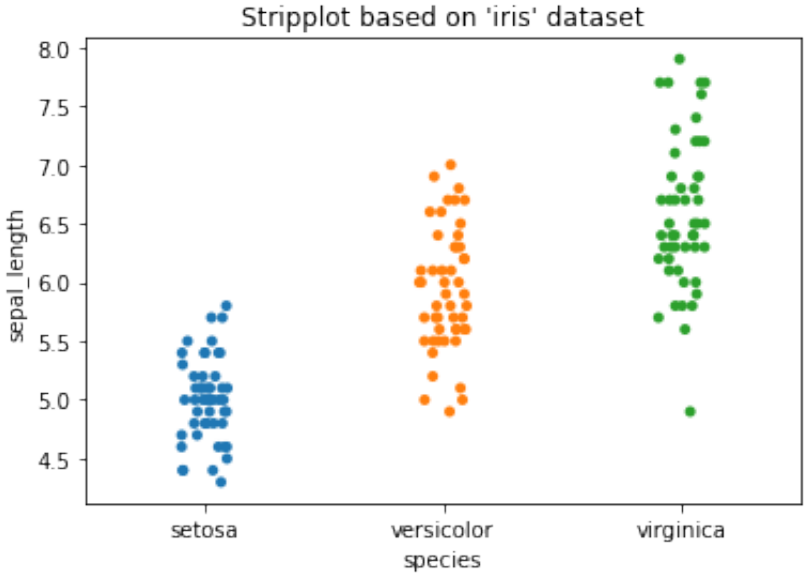


- Unlike Matplotlib, labels are set automatically in Seaborn.
- Plot `title` can be set by using the Matplotlib function.

Stripplot

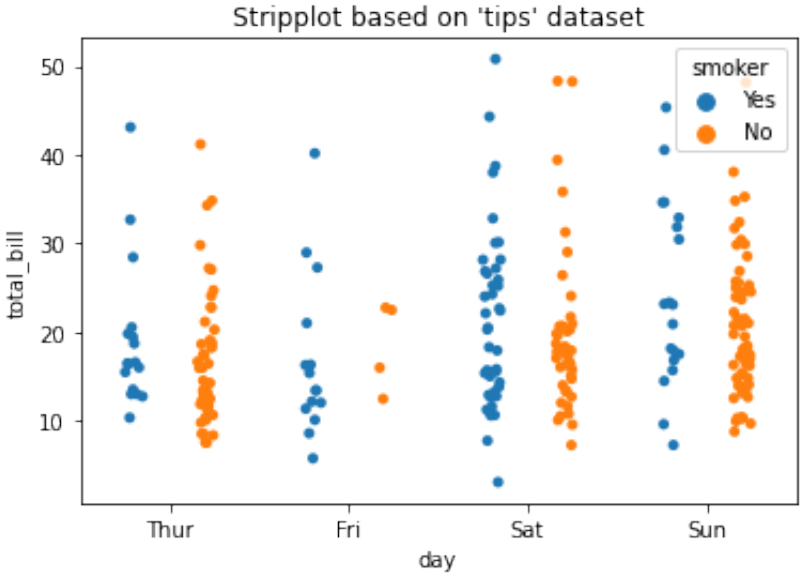
- Stripplot is one kind of scatter plot of categorical data.
- That means, it creates a scatter plot based on the category.

In [9]:
```
sns.stripplot(x='species', y='sepal_length', data=iris)

plt.title("Stripplot based on 'iris' dataset");
```



In [10]:

```
sns.stripplot(x='day', y='total_bill', data=tips,
              jitter=True,
              hue='smoker',
              dodge=True
             )

plt.title("Stripplot based on 'tips' dataset");
```

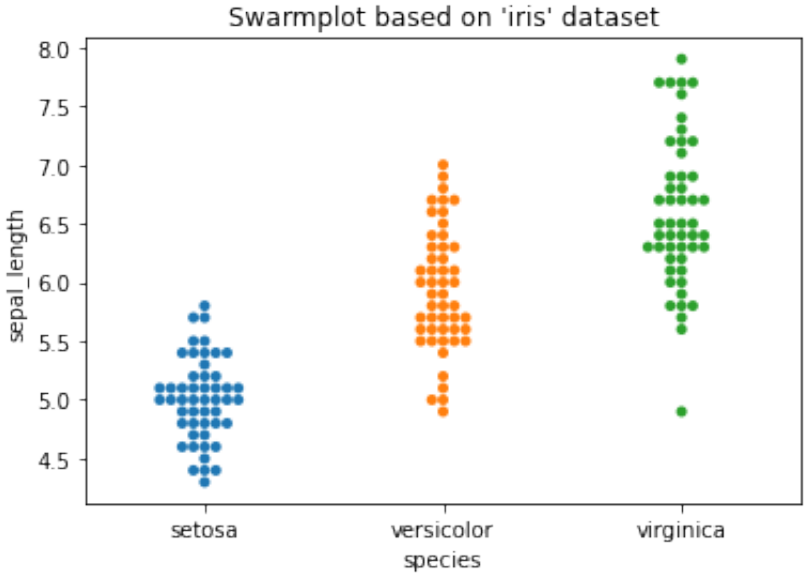


- `jitter` parameter is used to add an amount of jitter which can be useful when you have many points and they overlap so that it is easier to see the distribution.
- `hue` is used to provide an additional categorical separation.
- `dodge=True` is used to draw separate stripplots based on the category specified by the `hue` parameter.

Swarmplot

- *Swarmplot* is very much similar to *stripplot* but it does not allow overlapping of markers. That means, it causes jittering in the markers of the plot so that graph can easily be read without information loss as seen.

In [11]:
```
sns.swarmplot(x='species', y='sepal_length', data=iris)

plt.title("Swarmplot based on 'iris' dataset");
```

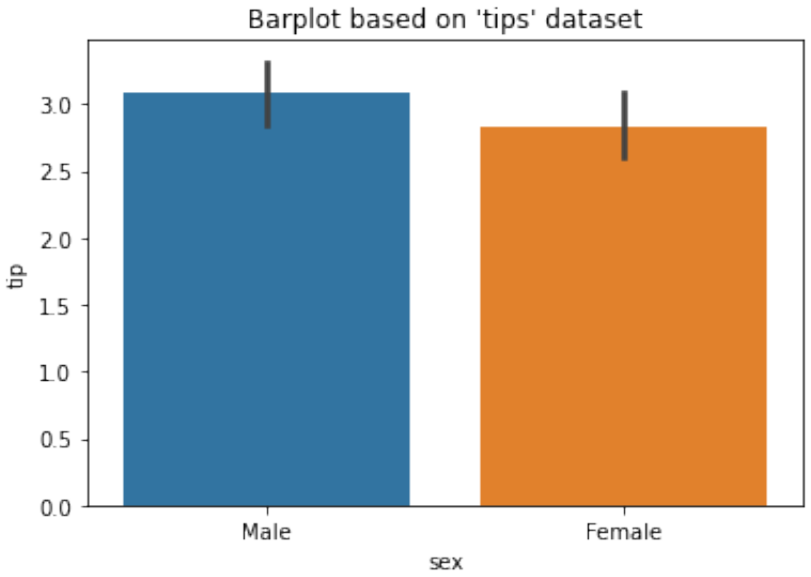


Barplot

- *Barplot* is basically used to aggregate the categorical data according to some methods and by default it's the mean.
- It can also be understood as a visualization of the group by action.

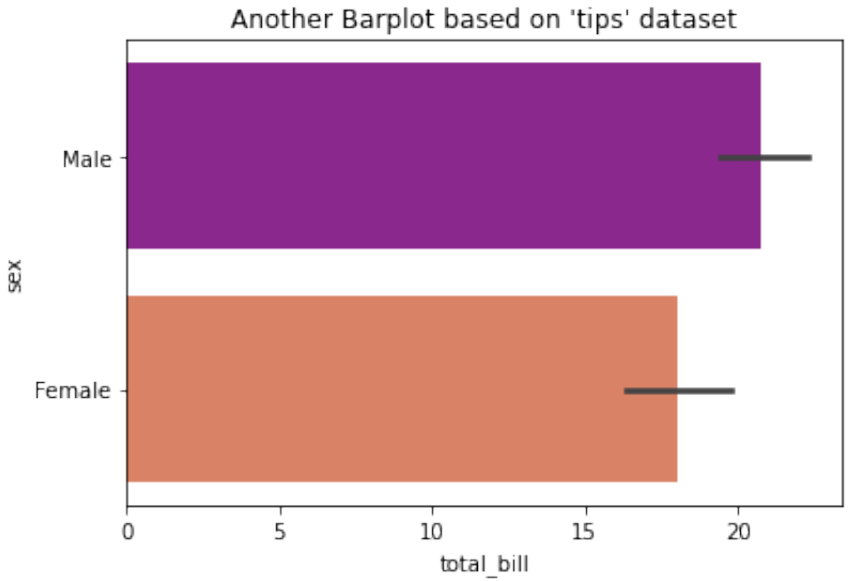
In [12]:

```
sns.barplot(x="sex", y="tip", data=tips)

plt.title("Barplot based on 'tips' dataset");
```



In [13]:

```
sns.barplot(x="total_bill", y="sex", data=tips,
            palette="plasma")

plt.title("Another Barplot based on 'tips' dataset");
```

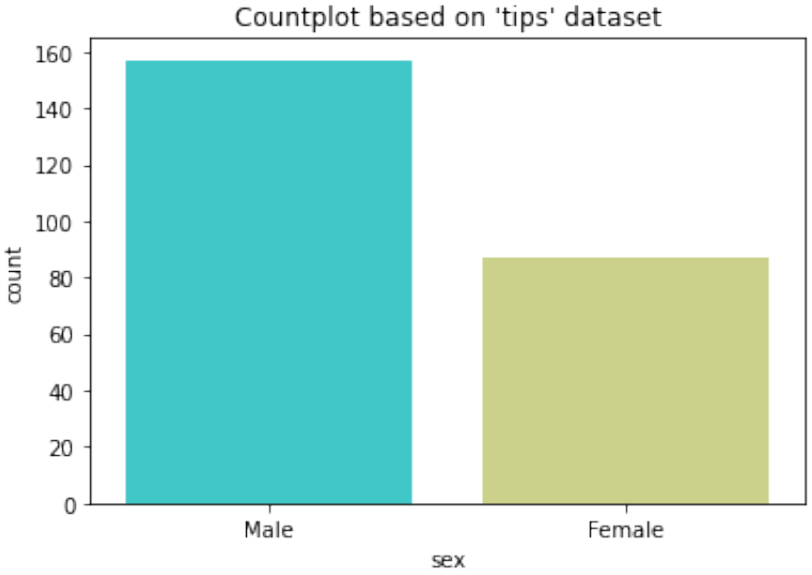


- `palette` is used to set the color of the plot.

Countplot

- *Countplot* basically counts the categories and returns a count of their occurrences.
- It is one of the simplest plots provided by the seaborn library.

In [14]:
```
sns.countplot(x="sex", data=tips,
              palette="rainbow")

plt.title("Countplot based on 'tips' dataset");
```

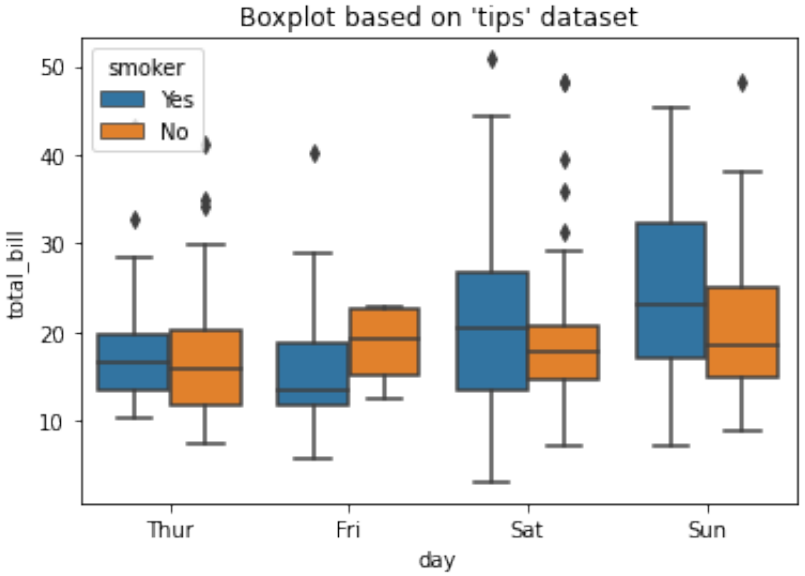


Boxplot

- Box Plot is the visual representation of the depicting groups of numerical data through their quartiles.
- Boxplot is also used to detect the outlier in the data set.

In [15]:

```
sns.boxplot(x="day", y="total_bill", data=tips,
            hue="smoker")

plt.title("Boxplot based on 'tips' dataset");
```



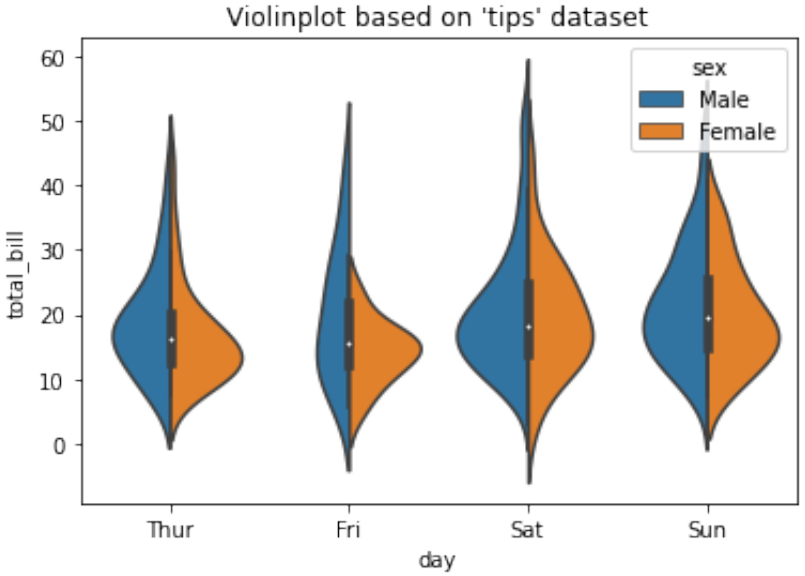
- `hue` parameter is used to further add a categorical separation.
- As we cab see in the above boxplot, the people who do not smoke had a higher bill on Friday as compared to the people who smoked.

Violinplot

- Violinplot is similar to the boxplot except that it provides a higher, more advanced visualization and uses the kernel density estimation to give a better description about the data distribution.

In [16]:

```
sns.violinplot(x='day', y='total_bill', data=tips,
               hue='sex',
               split=True
              )

plt.title("Violinplot based on 'tips' dataset");
```

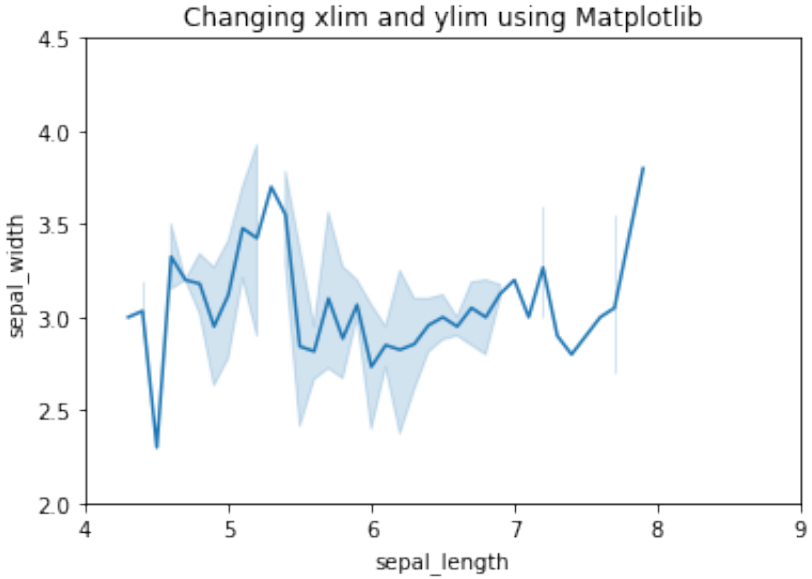


- `hue` parameter is used to separate the data further using the `sex` category.
- `split=True` is used to draw half of a violin for each level. This can make it easier to directly compare the distributions.

Setting `xlim` and `ylim` using Matplotlib Function

In [17]:

```
sns.lineplot(x='sepal_length', y='sepal_width', data=iris)

plt.xlim(4, 9)    # Setting xlim
plt.ylim(2, 4.5)  # Setting ylim
plt.title("Changing xlim and ylim using Matplotlib");
```

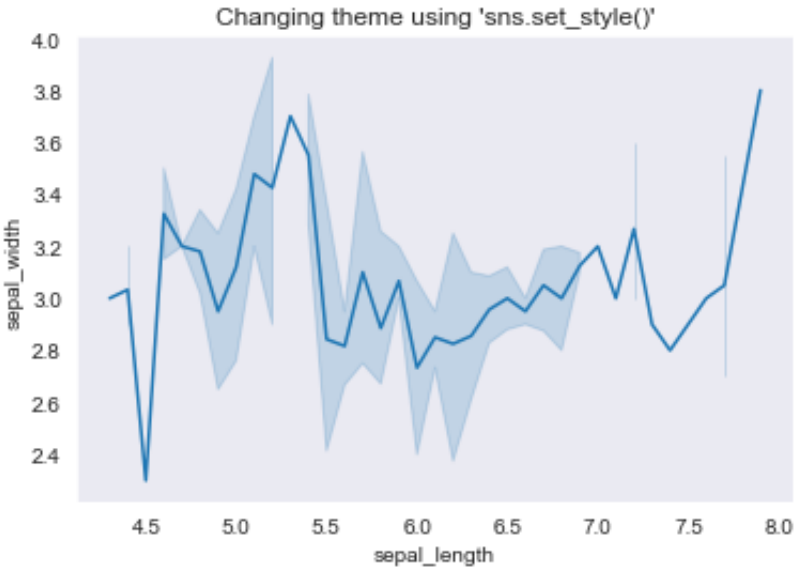


Changing Figure Aesthetic

- `set_style()` method is used to set the aesthetic of the plot, which affects things like the color of the axes, whether the grid is active or not, or other aesthetic elements.
- There are five themes available in Seaborn:
  - darkgrid
  - whitegrid
  - dark
  - white
  - ticks

In [18]:

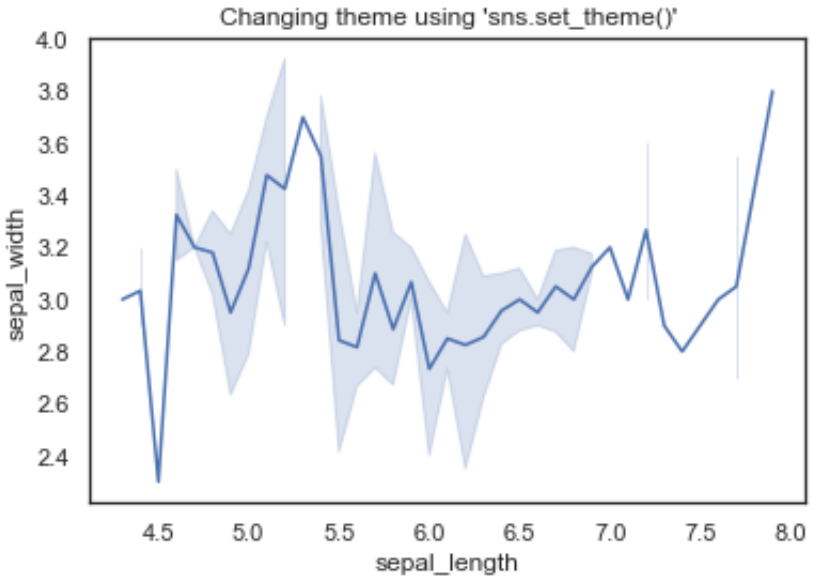
```
# Changing the theme to "dark"
sns.set_style("dark")

# Draw a line plot
sns.lineplot(x='sepal_length', y='sepal_width', data=iris)

plt.title("Changing theme using 'sns.set_style()'");
```



- `set_theme()` method can also be used to chage the theme by providing `style` parameter.

In [19]:

```
# Changing the theme to "white"
sns.set_theme(style="white")

# Draw a line plot
sns.lineplot(x='sepal_length', y='sepal_width', data=iris)

plt.title("Changing theme using 'sns.set_theme()'");
```

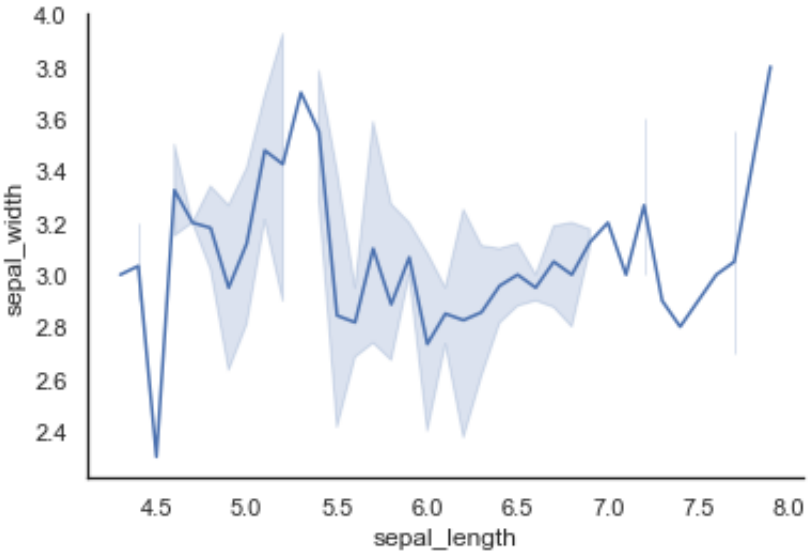


Removing Spine

- `despine()` method removes spines from the plot.

In [20]:

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

# Removing the spines
sns.despine()   # by default, top=True, left=True
```

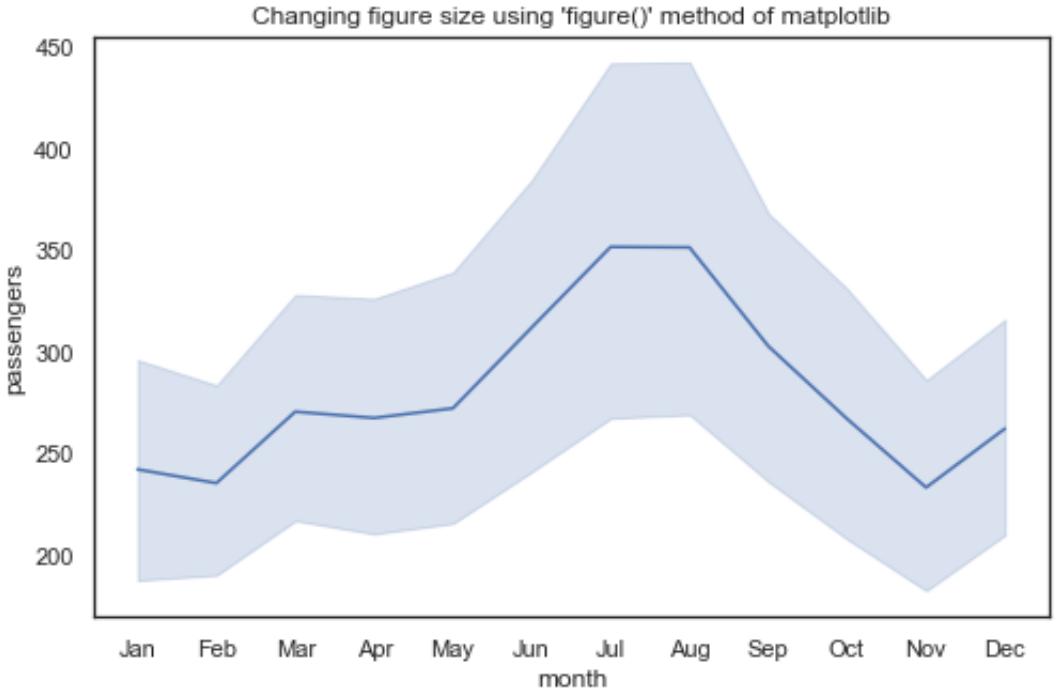


Changing Figure Size

- Figure size can be changed using the `figure()` method of Matplotlib.
- `figure()` method creates a new figure of the specified size passed in the `figsize` parameter.

In [21]:

```
# changing the figure size
plt.figure(figsize = (8, 5))

sns.lineplot(x="month", y="passengers", data=flights)

plt.title("Changing figure size using 'figure()' method of matplotlib");
```

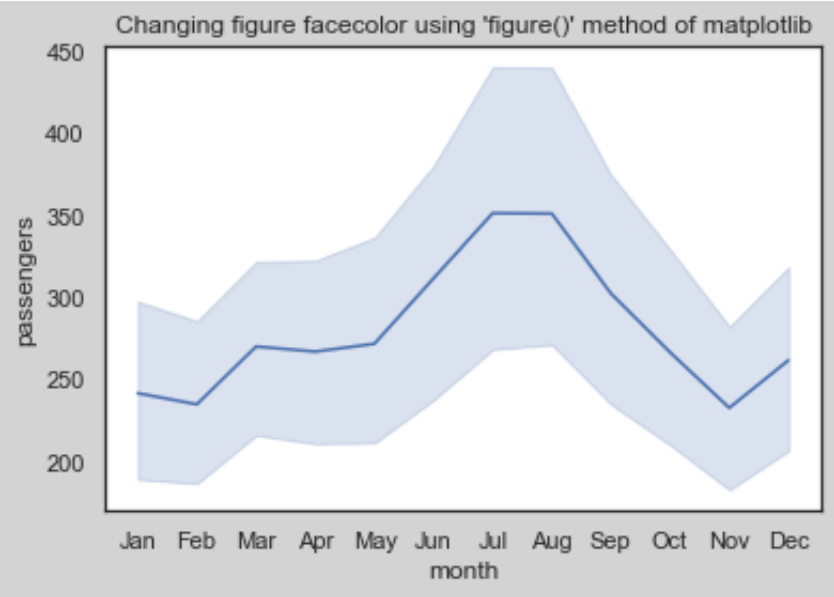


Changing Figure Facecolor

- Like changing the figure size, figure facecolor can also be changed by passing `facecolor` parameter to the `figure()` method of Matplotlib.

In [22]:

```
# changing the figure facecolor
plt.figure(facecolor="lightgray")

sns.lineplot(x="month", y="passengers", data=flights)

plt.title("Changing figure facecolor using 'figure()' method of matplotlib");
```

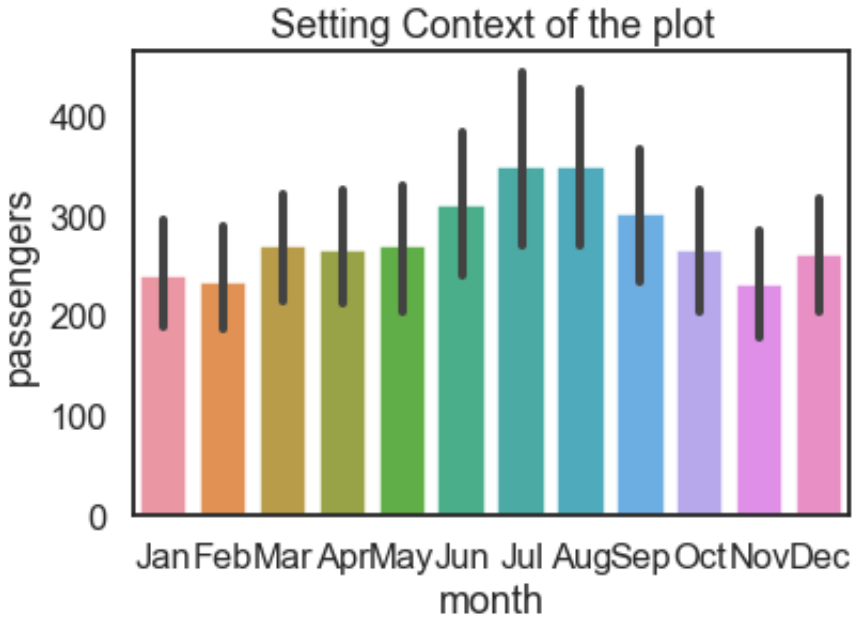


Scaling the plots

- Seaborn also allows us to control the scale of the elements or the plot by using the `set_context()` function.
- This affects things like the size of the labels, lines, and other elements of the plot, but not the overall style.
- There are four preset templates for contexts, based on relative size, the contexts are named as follows:
  - Paper
  - Notebook
  - Talk
  - Poster
- By default, context is set to `notebook`.

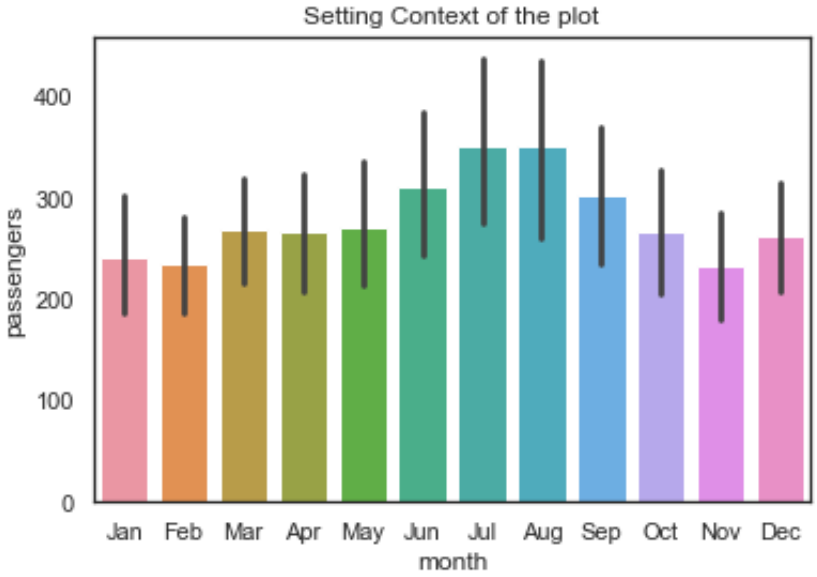
In [23]:

```
# changing context to "talk"
sns.set_context("talk")
bar_plot = sns.barplot(x='month', y='passengers', data=flights)

bar_plot.set_title("Setting Context of the plot");
```



In [24]:

```
# changing context back to "notebook"
sns.set_context("notebook")
bar_plot = sns.barplot(x='month', y='passengers', data=flights)

bar_plot.set_title("Setting Context of the plot");
```

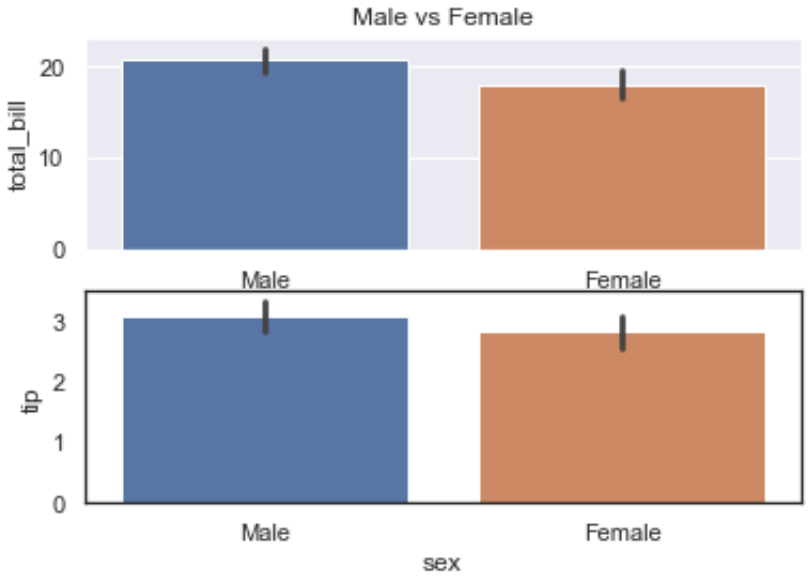


Setting the Style Temporarily

- `axes_style()` method is used to set the style temporarily. It is used along with the `with` statement.

In [25]:

```
with sns.axes_style('darkgrid'):
    plt.subplot(211)
    sns.barplot(x='sex', y='total_bill', data=tips)
    plt.title("Male vs Female")

plt.subplot(212)
sns.barplot(x='sex', y='tip', data=tips);
```

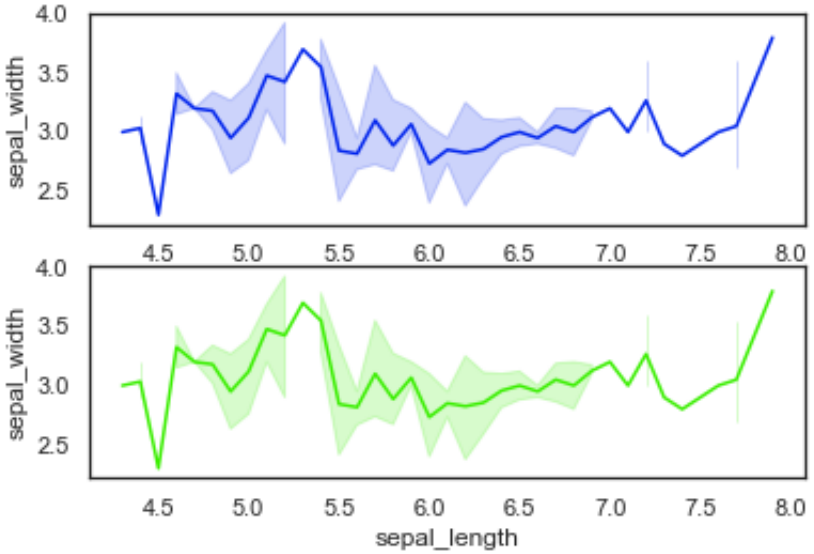


Setting the default Color Palette

- `set_palette()` method is used to set the default color palette for all the plots.

In [26]:

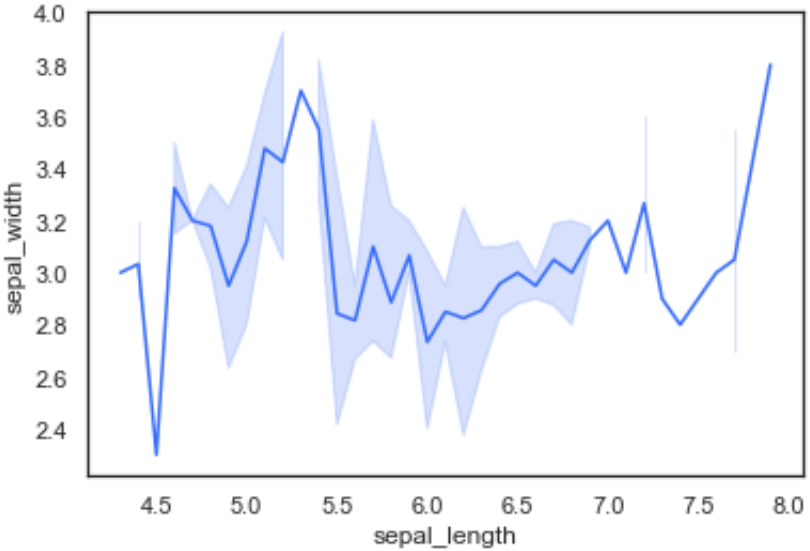
```
# Changing color palette
sns.set_palette('winter')
plt.subplot(211)
sns.lineplot(x='sepal_length', y='sepal_width', data=iris)

# Changing color palette
sns.set_palette('prism')
plt.subplot(212)
sns.lineplot(x='sepal_length', y='sepal_width', data=iris);
```



Setting the Color Palette Temporarily

In [27]:

```
# using color_palette() with "with" statement
with sns.color_palette('rainbow'):
    sns.lineplot(x='sepal_length', y='sepal_width', data=iris);
```

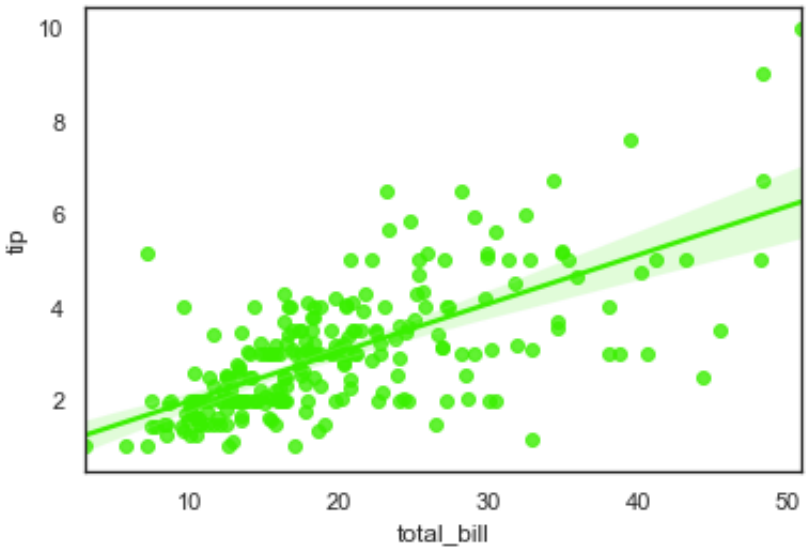


Draw Linear Regression Model

- The regression models are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.
- Regression plots as the name suggests creates a regression line between two parameters and helps to visualize their linear relationships.
- There are two main functions in Seaborn to visualize a linear relationship determined through regression: `regplot()` and `lmplot()`.
- Note: The difference between both the function is that `regplot()` accepts the `x`, `y` variables in different format inlcuding NumPy arrays, Pandas objects, whereas, the `lmplot()` only accepts the value as strings.

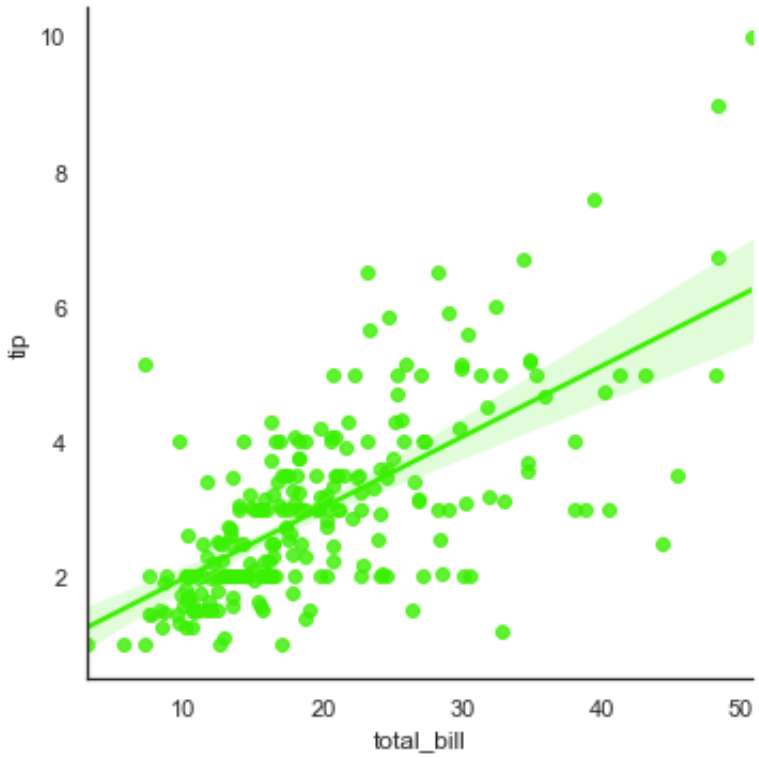
In [28]:

```
# Linear Regression Model using regplot()
sns.regplot(x="total_bill", y="tip", data=tips);
```



In [29]:

```
# Linear Regression Model using lmplot()
sns.lmplot(x="total_bill", y="tip", data=tips);
```


10
8
6
4
2
tip
10
20
30
40
50
total_bill