Seaborn

- Seaborn is a python data visualization libary based on matplotlib.It provides a high level interface for drawing attractive and informative statistical graphics.

In [5]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\alekhya\anaconda3\lib\site-packages (0.9.0)
Requirement already satisfied: pandas>=0.15.2 in c:\users\alekhya\anaconda3\lib\site-packages (from seaborn)
(0.24.2)
Requirement already satisfied: numpy>=1.9.3 in c:\users\alekhya\anaconda3\lib\site-packages (from seaborn) (1.
16.2)
Requirement already satisfied: scipy>=0.14.0 in c:\users\alekhya\anaconda3\lib\site-packages (from seaborn)
(1.2.1)
Requirement already satisfied: matplotlib>=1.4.3 in c:\users\alekhya\anaconda3\lib\site-packages (from seabor
n) (3.0.3)
Requirement already satisfied: pytz>=2011k in c:\users\alekhya\anaconda3\lib\site-packages (from pandas>=0.15.
2->seaborn) (2018.9)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\alekhya\anaconda3\lib\site-packages (from pa
ndas>=0.15.2->seaborn) (2.8.0)
Requirement already satisfied: cycler>=0.10 in c:\users\alekhya\anaconda3\lib\site-packages (from matplotlib>=
1.4.3->seaborn) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\alekhya\anaconda3\lib\site-packages (from matplot
lib>=1.4.3->seaborn) (1.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\alekhya\anaconda3\lib\site
-packages (from matplotlib>=1.4.3->seaborn) (2.3.1)
Requirement already satisfied: six>=1.5 in c:\users\alekhya\anaconda3\lib\site-packages (from python-dateutil>
=2.5.0->pandas>=0.15.2->seaborn) (1.12.0)
Requirement already satisfied: setuptools in c:\users\alekhya\anaconda3\lib\site-packages (from kiwisolver>=1.
0.1->matplotlib>=1.4.3->seaborn) (40.8.0)
Note: you may need to restart the kernel to use updated packages.
```

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

In [2]:
```
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[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

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]:
```
help(sns.colors)
```

```
Help on package seaborn.colors in seaborn:

NAME
    seaborn.colors

PACKAGE CONTENTS
    crayons
    xkcd_rgb

DATA
    crayons = {'Almond': '#EFDECD', 'Antique Brass': '#CD9575', 'Apricot':...
    xkcd_rgb = {'acid green': '#8ffe09', 'adobe': '#bd6c48', 'algae': '#54...

FILE
    c:\users\alekhya\anaconda3\lib\site-packages\seaborn\colors\__init__.py
```

In [6]:
```
help(sns.crayon_palette)
```

```
Help on function crayon_palette in module seaborn.palettes:

crayon_palette(colors)
    Make a palette with color names from Crayola crayons.

    Colors are taken from here:
    https://en.wikipedia.org/wiki/List_of_Crayola_crayon_colors (https://en.wikipedia.org/wiki/List_of_Crayola
_crayon_colors)

    This is just a simple wrapper around the ``seaborn.crayons`` dictionary.

    Parameters
    ----------
    colors : list of strings
        List of keys in the ``seaborn.crayons`` dictionary.

    Returns
    -------
    palette : seaborn color palette
        Returns the list of colors as rgb tuples in an object that behaves like
        other seaborn color palettes.

    See Also
    --------
    xkcd_palette : Make a palette with named colors from the XKCD color survey.
```

In [7]:
```
sns.axes_style()
```

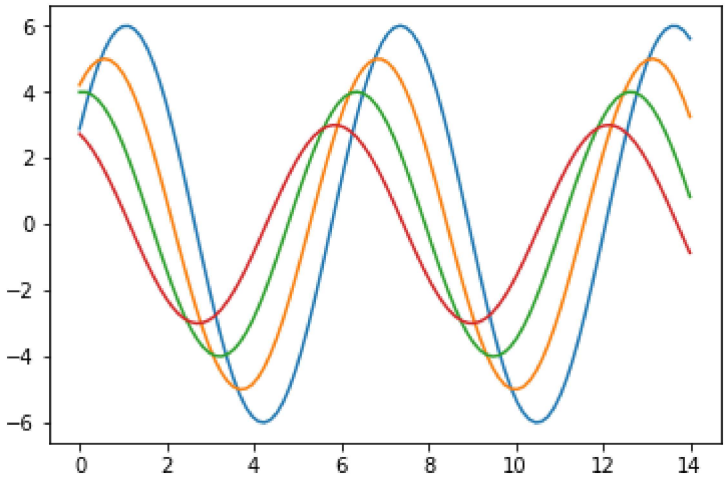
Out[7]:
```
{'axes.facecolor': 'white',
 'axes.edgecolor': 'black',
 'axes.grid': False,
 'axes.axisbelow': 'line',
 'axes.labelcolor': 'black',
 'figure.facecolor': (1, 1, 1, 0),
 'grid.color': '#b0b0b0',
 'grid.linestyle': '-',
 'text.color': 'black',
 'xtick.color': 'black',
 'ytick.color': 'black',
 'xtick.direction': 'out',
 'ytick.direction': 'out',
 'lines.solid_capstyle': 'projecting',
 'patch.edgecolor': 'black',
 'image.cmap': 'viridis',
 'font.family': ['sans-serif'],
 'font.sans-serif': ['DejaVu Sans',
  'Bitstream Vera Sans',
  'Computer Modern Sans Serif',
  'Lucida Grande',
  'Verdana',
  'Geneva',
  'Lucid',
  'Arial',
  'Helvetica',
  'Avant Garde',
  'sans-serif'],
 'patch.force_edgecolor': False,
 'xtick.bottom': True,
 'xtick.top': False,
 'ytick.left': True,
 'ytick.right': False,
 'axes.spines.left': True,
 'axes.spines.bottom': True,
 'axes.spines.right': True,
 'axes.spines.top': True}
```

Builtin Themes

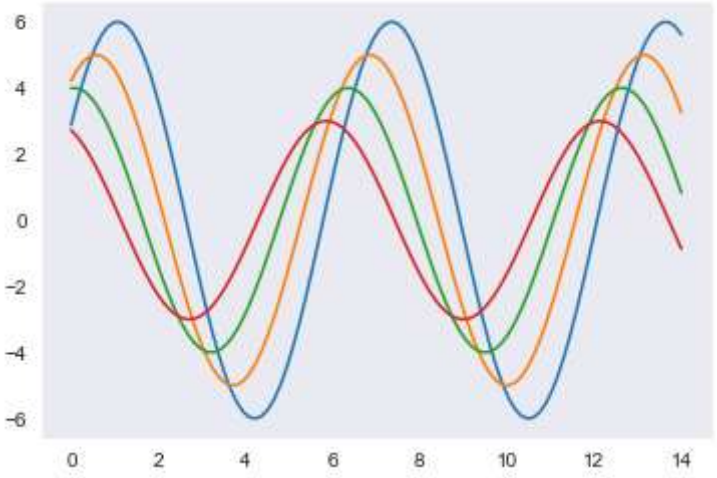
- Darkgrid
- Whitegrid
- dark
- white
- ticks

In [8]:
```
import matplotlib.pyplot as plt
```

In [9]:
```
import numpy as np
x = np.linspace(0,14,100)
for i in range(1,5):
    plt.plot(x,np.sin(x+i*0.5)*(7-i))
plt.show()
```



In [12]:
```
x = np.linspace(0,14,100)
for i in range(1,5):
    plt.plot(x,np.sin(x+i*0.5)*(7-i))
    sns.set_style("darkgrid")
plt.show()
```



In [13]:
```
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[13]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

In [14]:
```
df = sns.load_dataset("tips")
df.head()
```

Out[14]:

	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

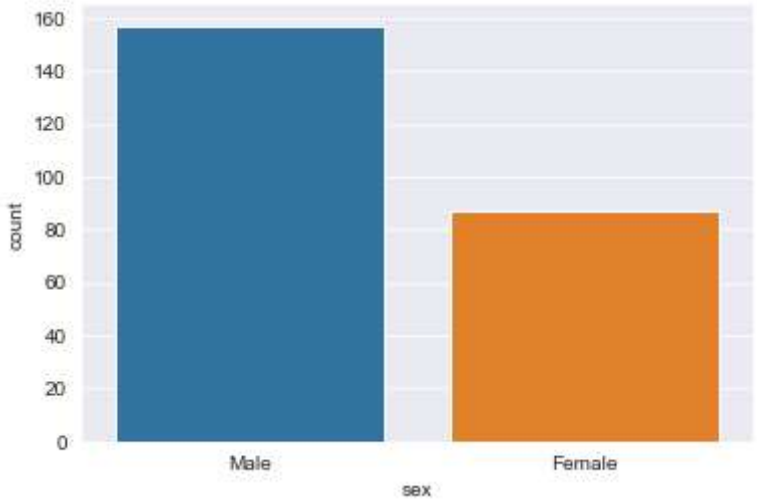
In [16]:
```
df.shape
```

Out[16]:
```
(244, 7)
```

Count plot

In [17]:
```
sns.countplot(x="sex",data=df)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2568f354358>
```

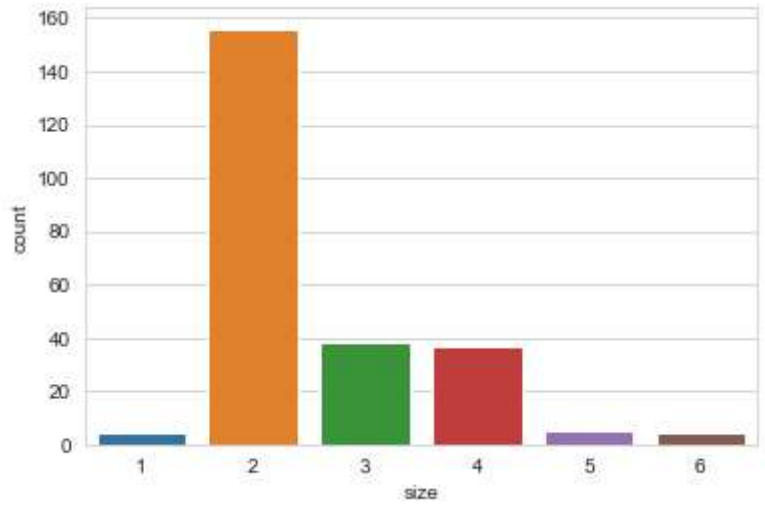


In [18]:
```
df.columns
```

Out[18]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
```

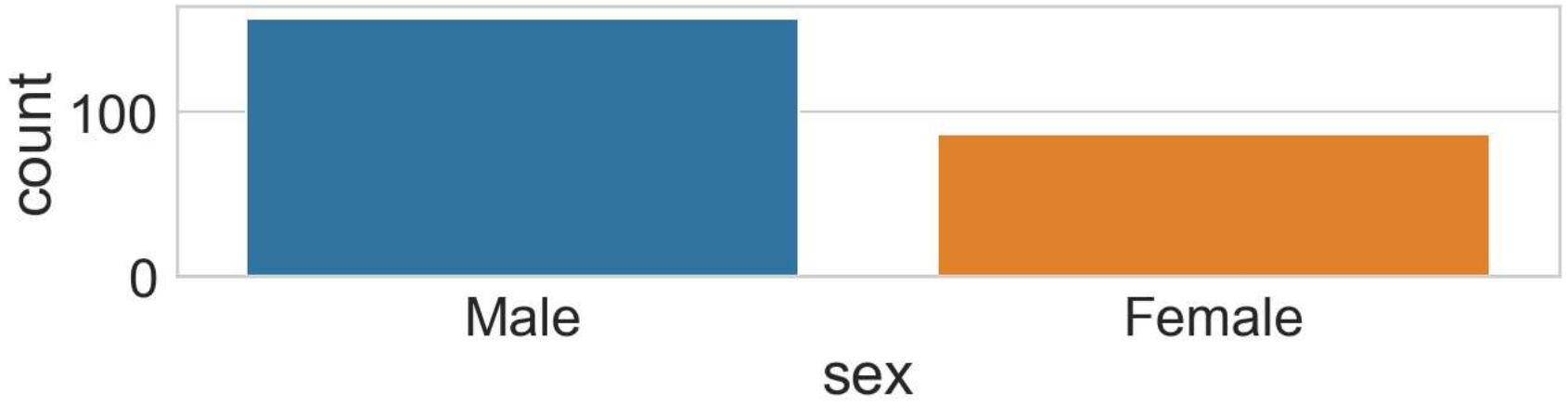
In [21]:
```
sns.set_style("whitegrid")
sns.countplot(x = "size",data = df)
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2568f44e400>
```



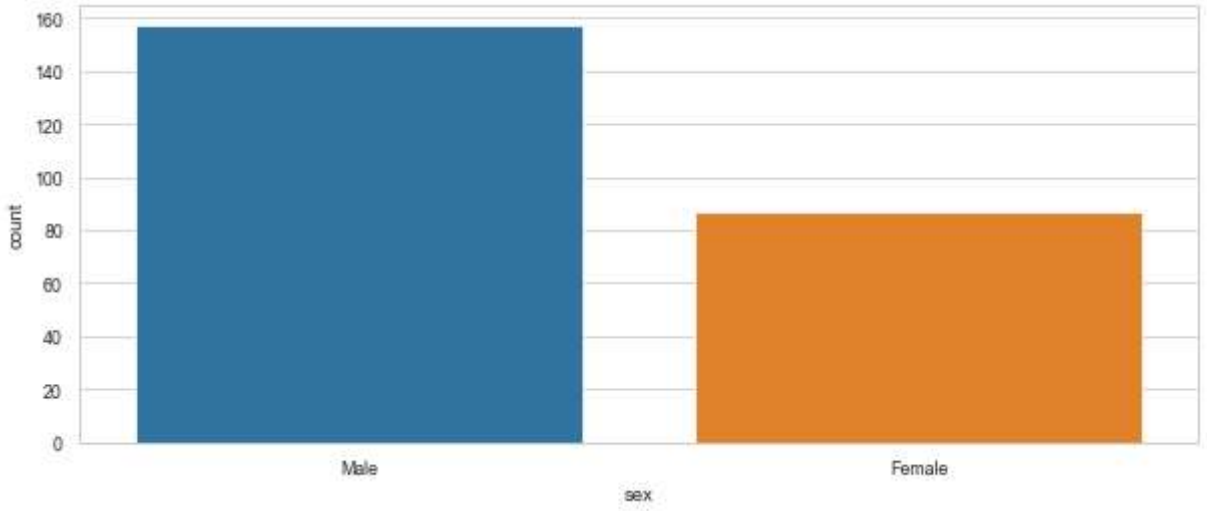
In [22]:
```
# Scale and context
sns.set_context("poster",font_scale=2) #poster,paper,notebook,talk
plt.figure(figsize=(20,4))
sns.countplot(x="sex",data=df)
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x2568efcb438>



In [26]:
```
sns.set_context('paper',font_scale=1)#poster,paper,notebook,talk
plt.figure(figsize=(10,4))
sns.countplot(x='sex',data=df)
```

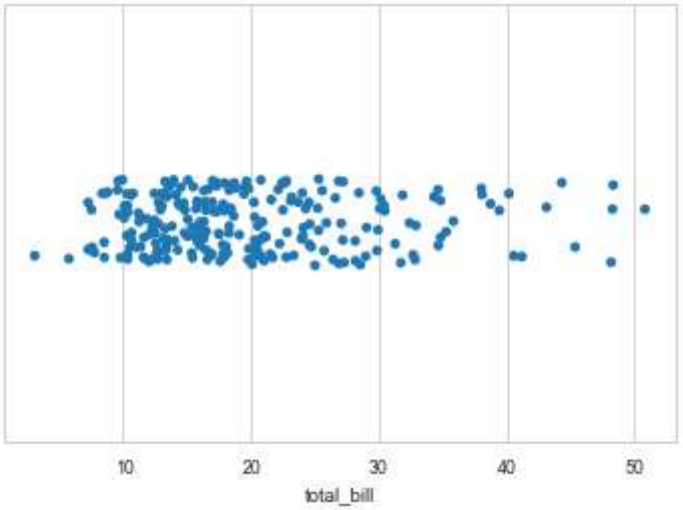
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2568f47efd0>



stripplot

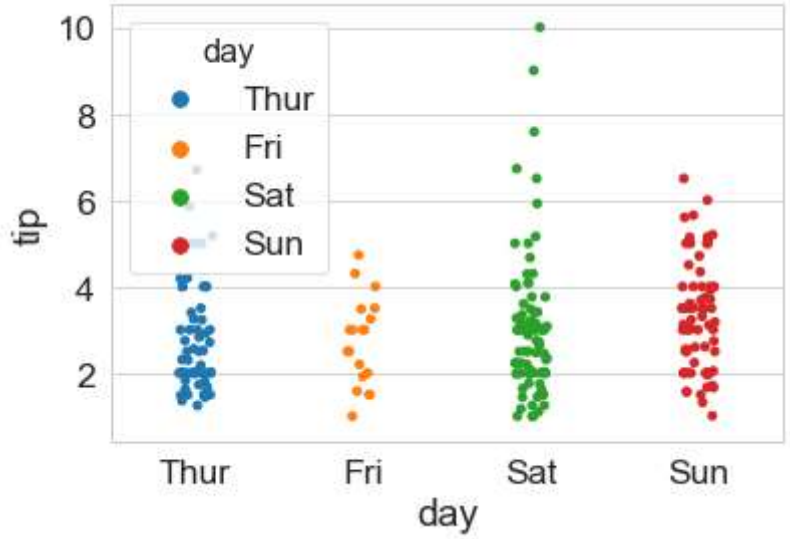
In [28]:
```
sns.stripplot(x="total_bill",data=df)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2568ee905f8>
```



In [34]:
```
sns.set_context("paper",font_scale=2)
sns.stripplot(x="day",y="tip",hue="day",data=df)
```

Out[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2568f81d8d0>
```

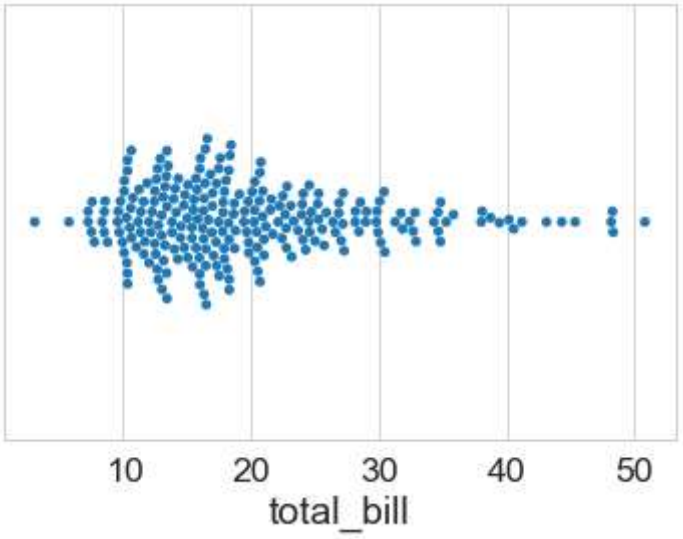


Swarm plot

- swarmplot is smilar to strip plot,locations of points are adjusted automatically to avoid the overlapping

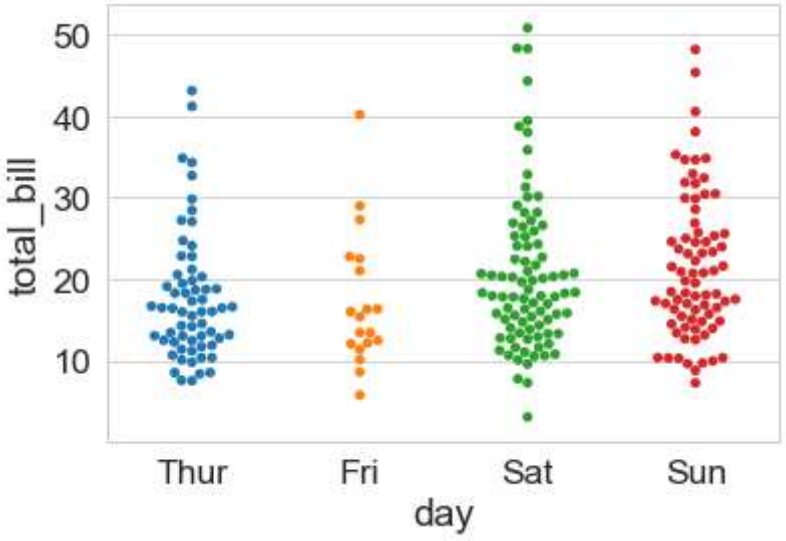
In [35]:
```
sns.swarmplot(x="total_bill",data=df)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2568f891b70>
```



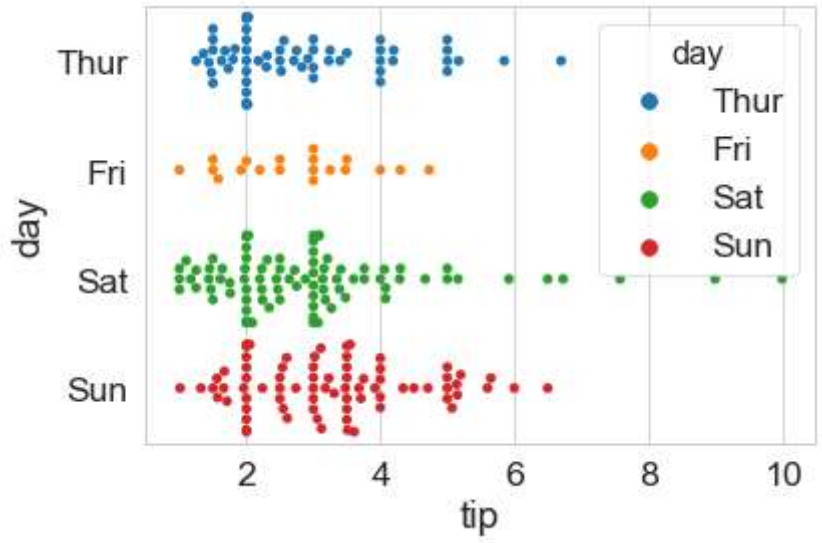
In [36]:
```
sns.swarmplot(x="day",y="total_bill",data=df)
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2568f8dadd8>
```



In [38]:
```
sns.swarmplot(x="tip",y="day",hue="day",data=df)
```

Out[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25690b37a90>
```

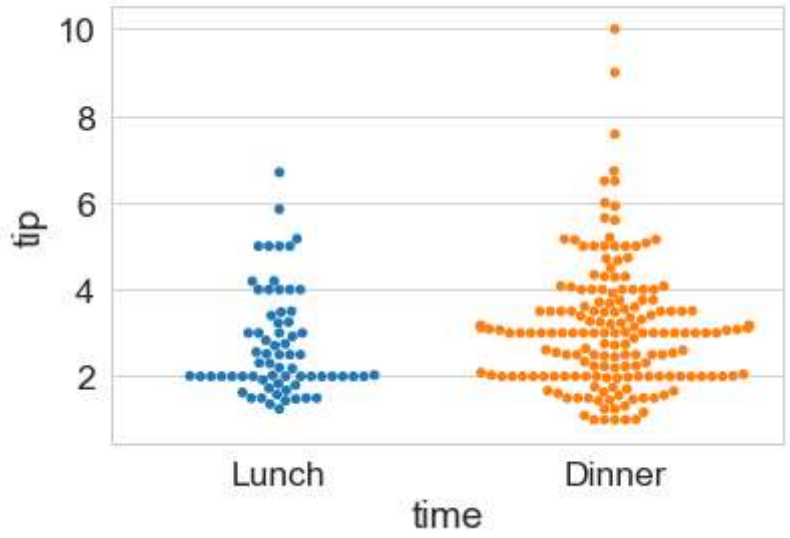


In [40]:
```
df.columns
```

Out[40]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
```

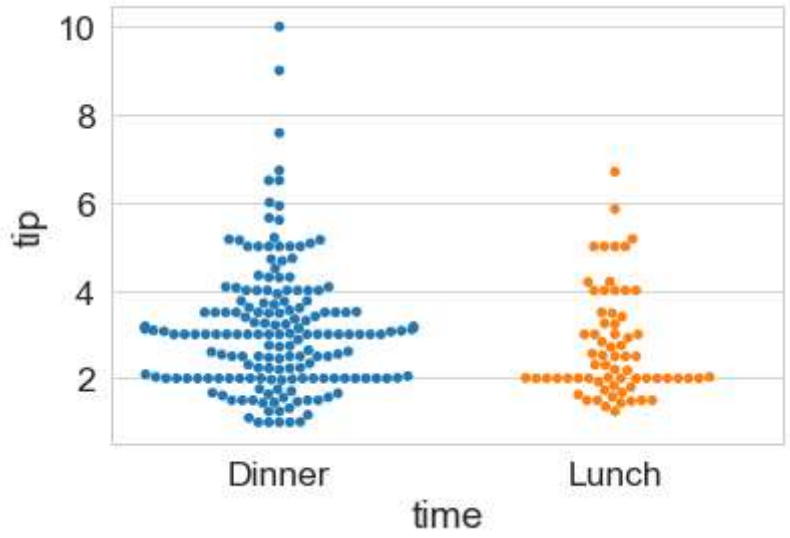
In [41]:
```
sns.swarmplot(x="time",y="tip",data=df)
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25690bcd390>
```



In [42]:
```
sns.swarmplot(x="time",y="tip",data=df,order=["Dinner","Lunch"])
```

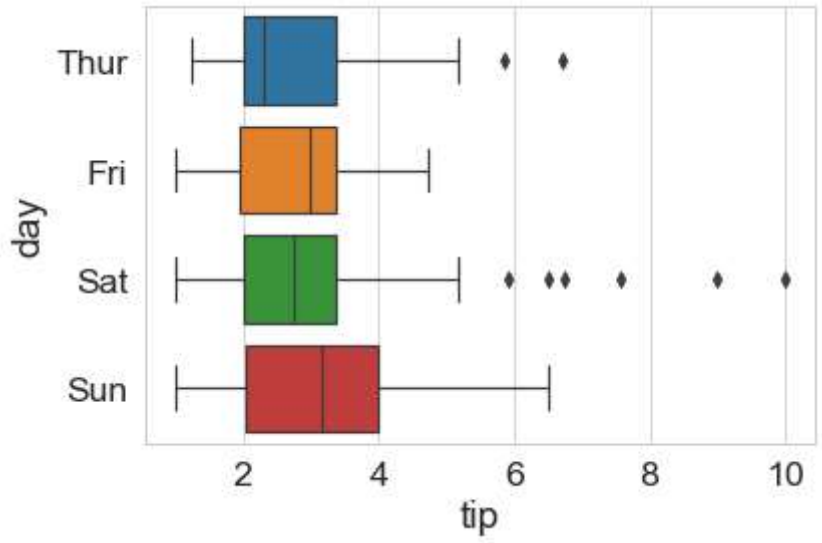
Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25690c36a20>
```



Box plot ot box-and- whisker plot

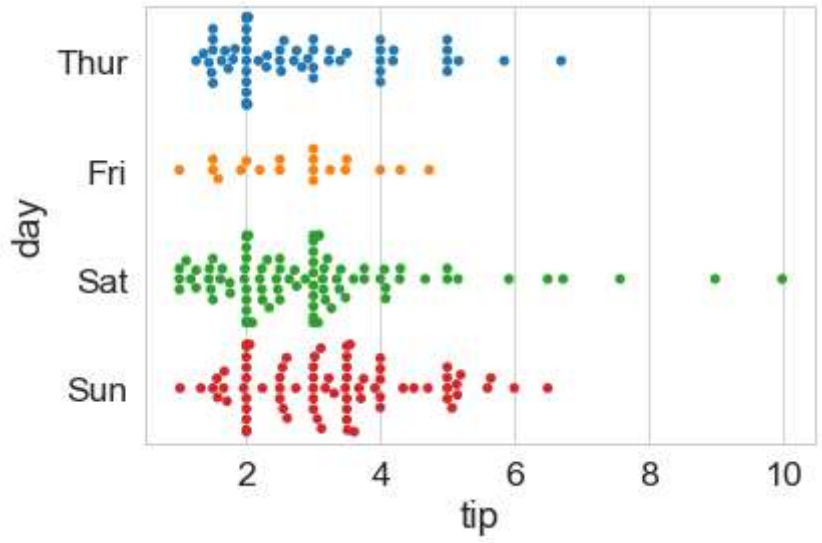
In [44]:
```
sns.boxplot(x="tip",y="day",data=df)
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25690cb2b38>
```



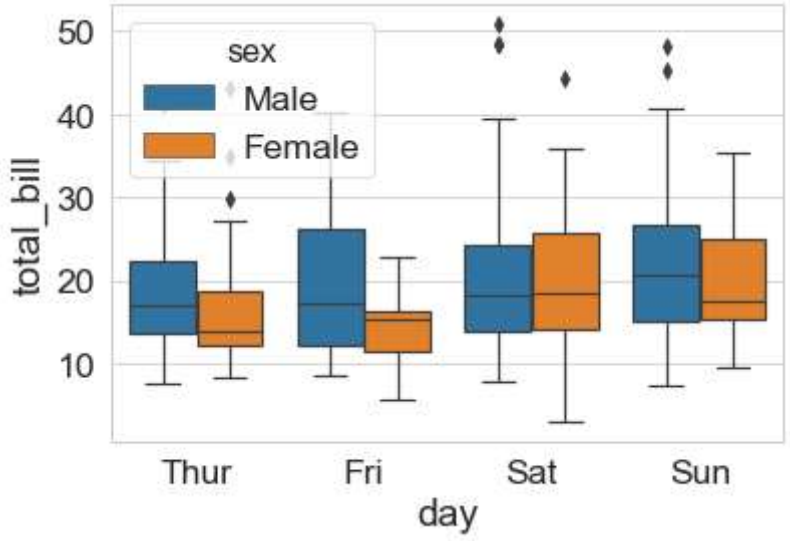
In [45]:
```
sns.swarmplot(x="tip",y="day",data=df)
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25690d31978>
```



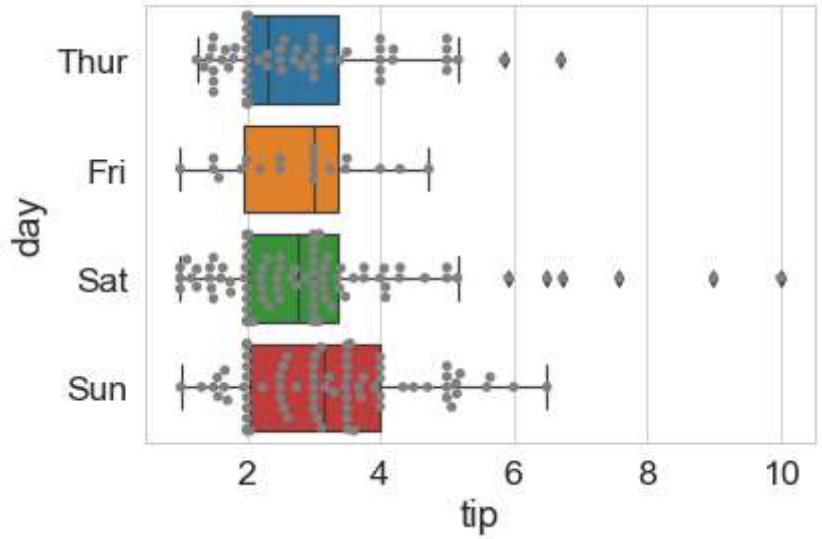
In [47]:
```
sns.boxplot(x="day",y="total_bill",hue="sex",data=df)
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25690e13128>
```



In [52]:
```
sns.boxplot(x="tip",y="day",data=df)
sns.swarmplot(x="tip",y="day",data=df,color="0.5") # color= 0 to 1
```

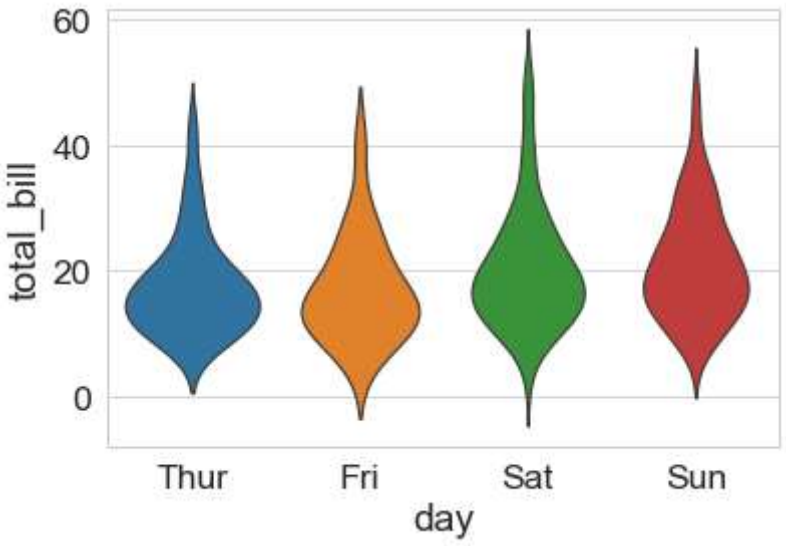
Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x256910ae588>
```



Violin plot

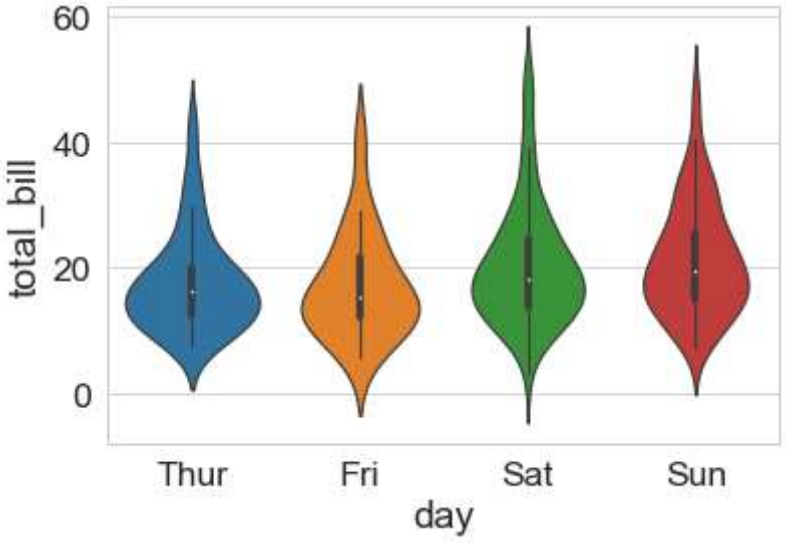
In [55]:
```
sns.violinplot(x="day",y="total_bill",data=df,inner=None)
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2569119dac8>
```



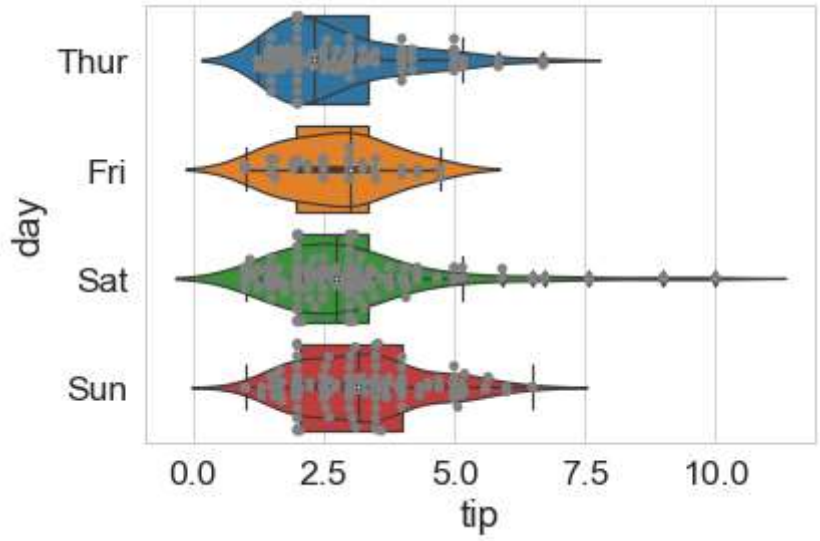
In [56]:
```
sns.violinplot(x="day",y="total_bill",data=df)
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x256911ed588>
```



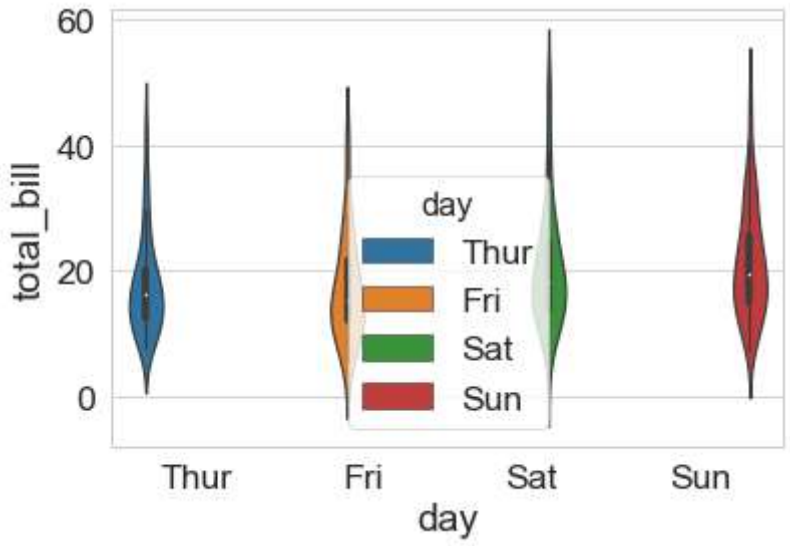
In [61]:
```
sns.boxplot(x="tip",y="day",data=df)
sns.swarmplot(x="tip",y="day",data=df,color="0.5") # color= 0 to 1
sns.stripplot(x="tip",y="day",data=df,color="0.5")
sns.violinplot(x="tip",y="day",data=df)
```

Out[61]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25692426d30>
```



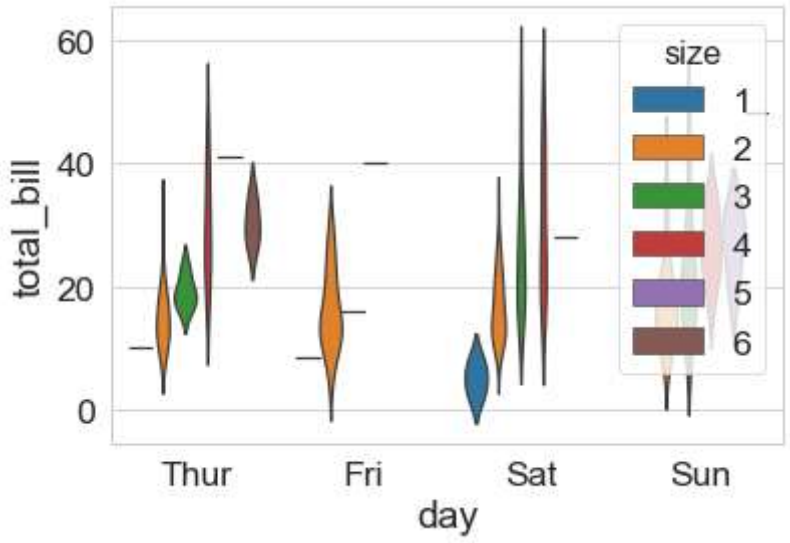
In [62]:
```
sns.violinplot(x="day",y="total_bill",hue="day",data=df)
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x256924e70b8>
```



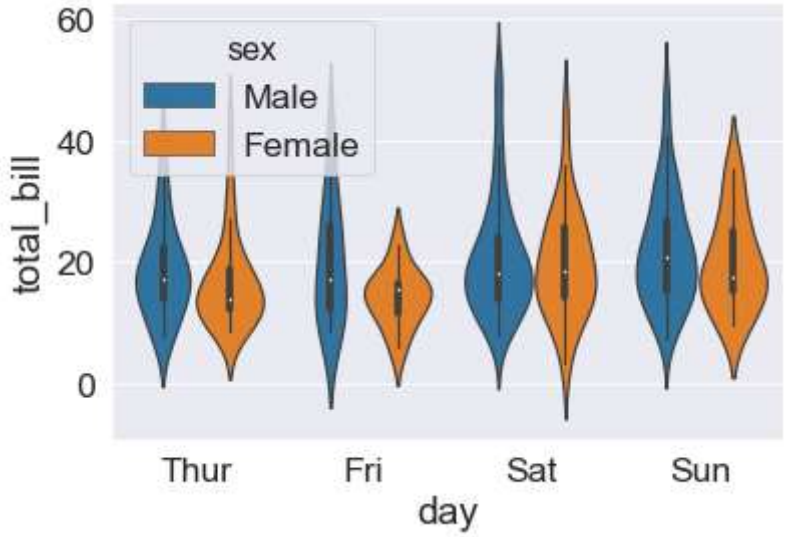
In [63]:
```
sns.violinplot(x='day',y='total_bill',data=df,inner=None,hue='size')
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25692568978>
```



In [66]:
```
sns.set_style("darkgrid")
sns.violinplot(x='day',y='total_bill',data=df,hue='sex')
```

Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x2569268b7b8>



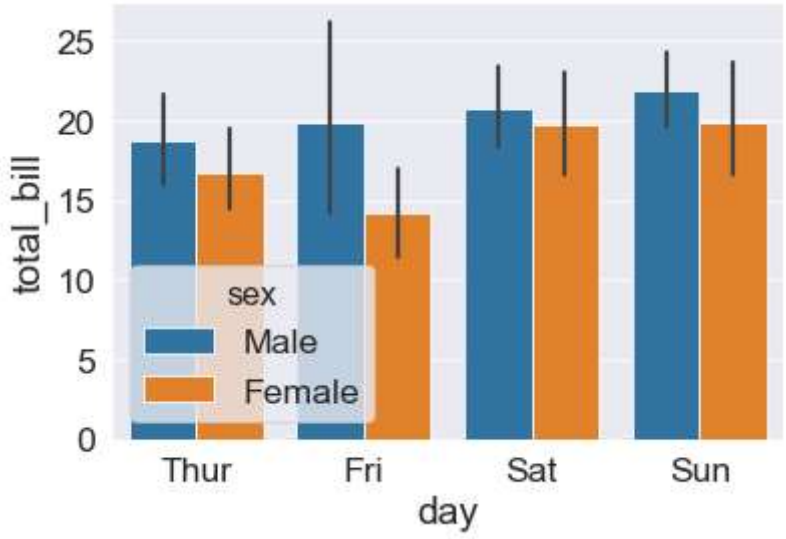
Barplot

In [67]:
```
sns.barplot(x="day",y="total_bill",data=df)
```

Out[67]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25692735898>
```

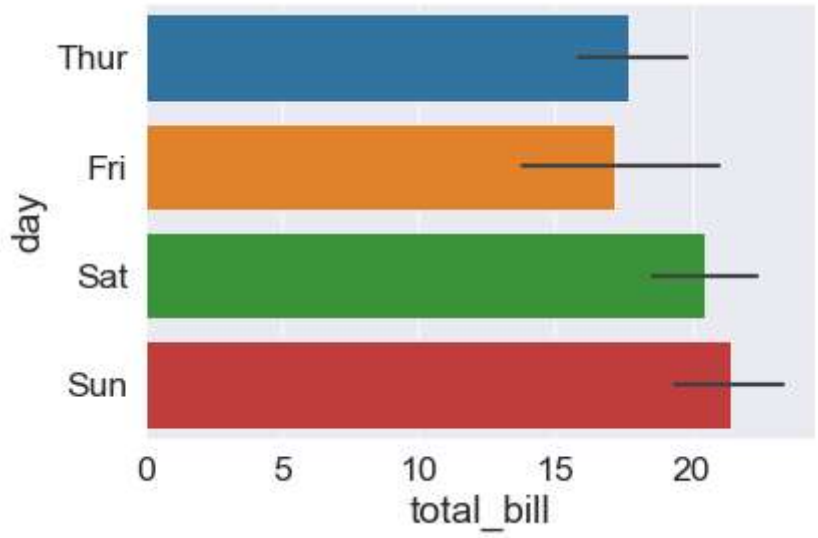
In [68]:
```
sns.barplot(x="day",y="total_bill",data=df,hue="sex")
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2569107e198>
```



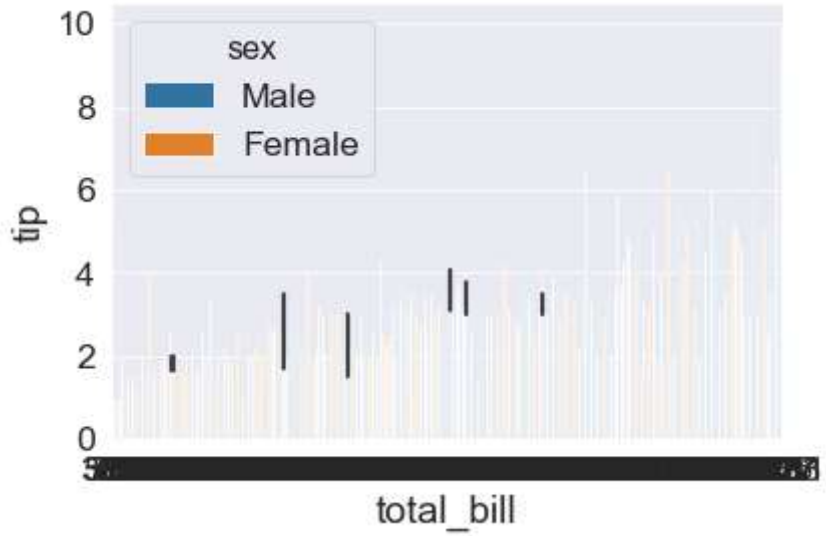
In [70]:
```
sns.barplot(x="total_bill",y="day",data=df)
```

Out[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2569276d588>
```



In [71]:
```
sns.barplot(x="total_bill",y="tip",hue="sex",data=df)
```

Out[71]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x256927dd0b8>
```



In [ ]:
```

```