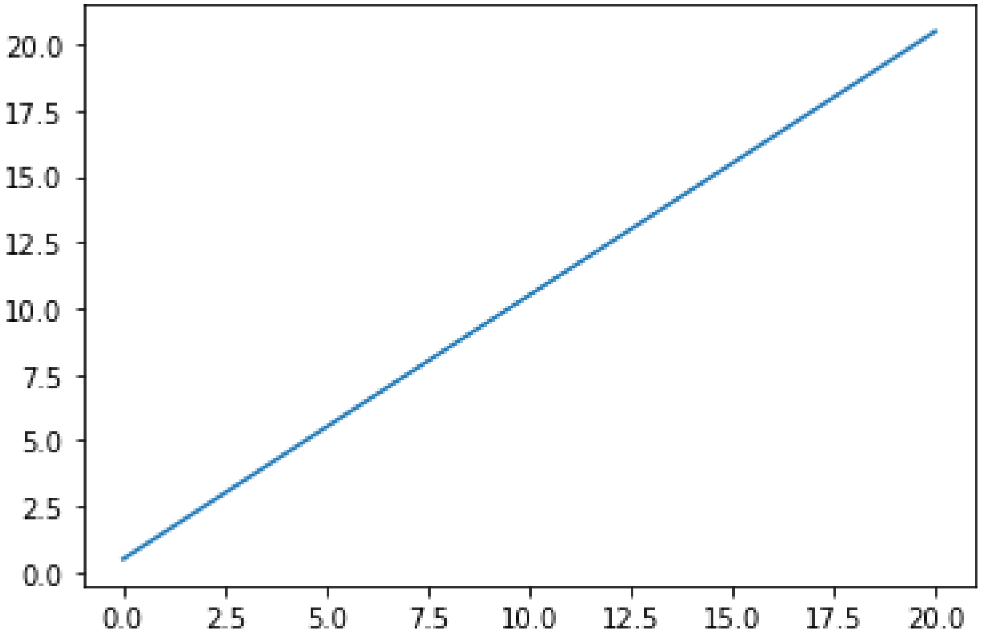
ch5-data-visualization

June 16, 2022

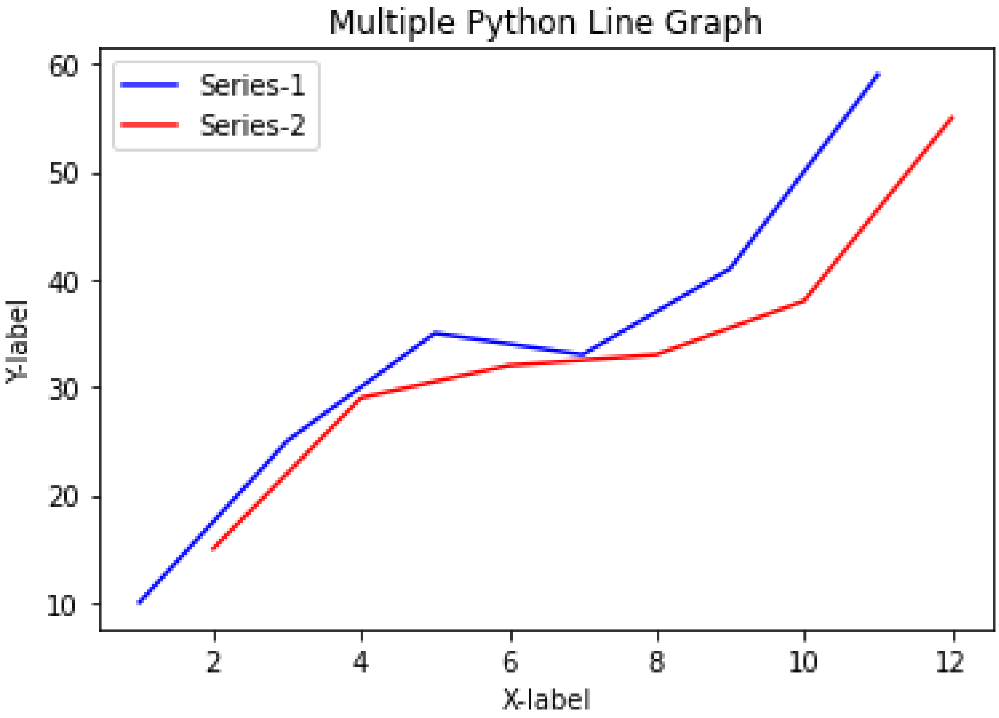
[1]:

```
# data visualization

# add the essential library matplotlib
import matplotlib.pyplot as plt
import numpy as np

# create the data
a = np.linspace(0, 20)

# draw the plot
plt.plot(a, a + 0.5, label='linear')

# display the chart
plt.show()
```



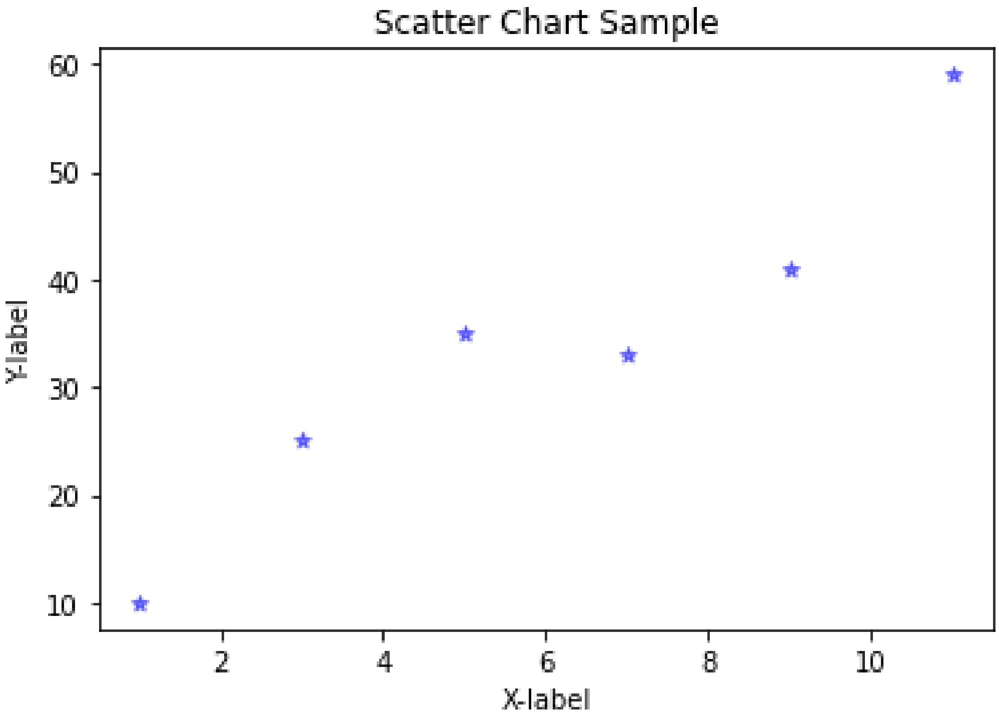
[2]:

```python
# create the data
x = [1, 3, 5, 7, 9, 11]
y = [10, 25, 35, 33, 41, 59]

# let's plot the data
plt.plot(x, y, label='Series-1', color='blue')

# create the data
x = [2, 4, 6, 8, 10, 12]
y = [15, 29, 32, 33, 38, 55]

# plot the data
plt.plot(x, y, label='Series-2', color='red')

# add X label on X-axis
plt.xlabel("X-label")

# add Y label on Y-axis
plt.ylabel("Y-label")

# append the title to graph
plt.title("Multiple Python Line Graph")

# add legend to graph
plt.legend()

# display the plot
plt.show()
```



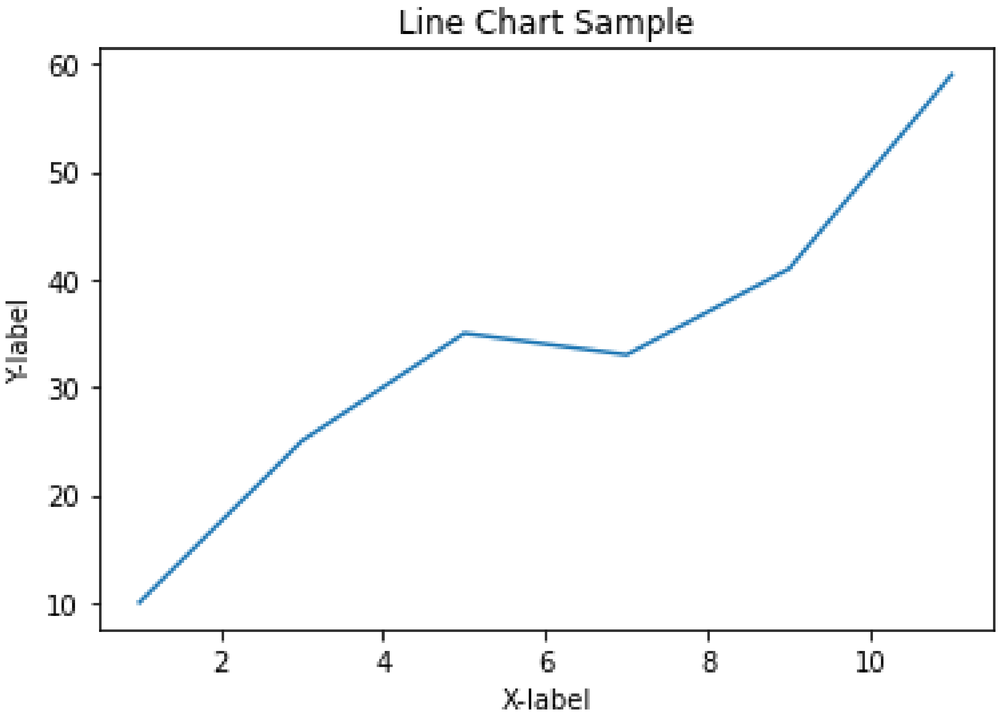
[5]:

```python
# scatter plot
# add the essential library matplotlib
import matplotlib.pyplot as plt

# create the data
x = [1,3,5,7,9,11]
y = [10,25,35,33,41,59]

# draw the scatter chart
plt.scatter(x,y,c='blue', marker='*', alpha=0.5)

# append the label on X-axis
plt.xlabel("X-label")

# append the label on Y-axis
plt.ylabel("Y-label")

# add the title to graph
plt.title("Scatter Chart Sample")

# display the chart
plt.show()
```

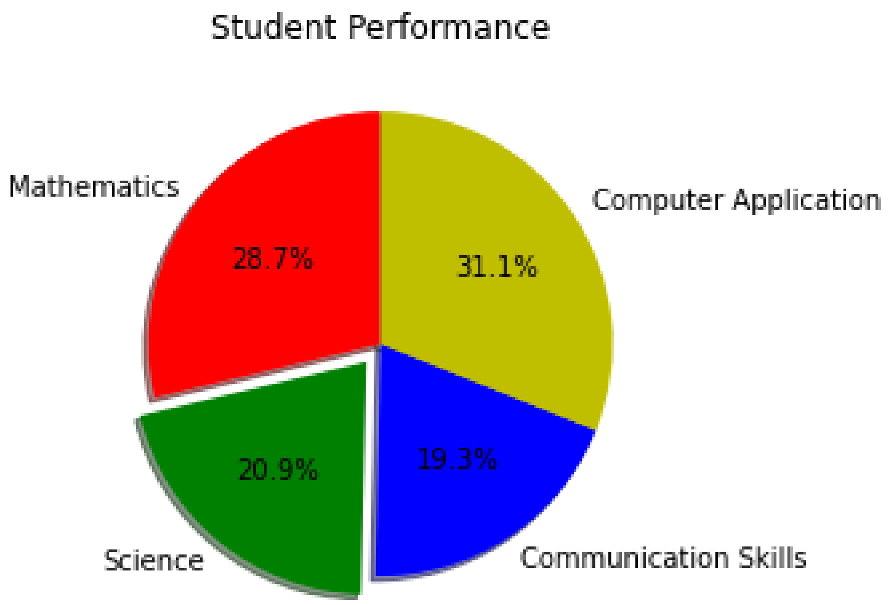


[6]:

```
# line plot

# create the data
x = [1,3,5,7,9,11]
y = [10,25,35,33,41,59]

# draw the line chart
plt.plot(x,y)

# append the label on X-axis
plt.xlabel("X-label")

# append the label on Y-axis
plt.ylabel("Y-label")

# append the title to chart
plt.title("Line Chart Sample")

# display the chart
plt.show()
```

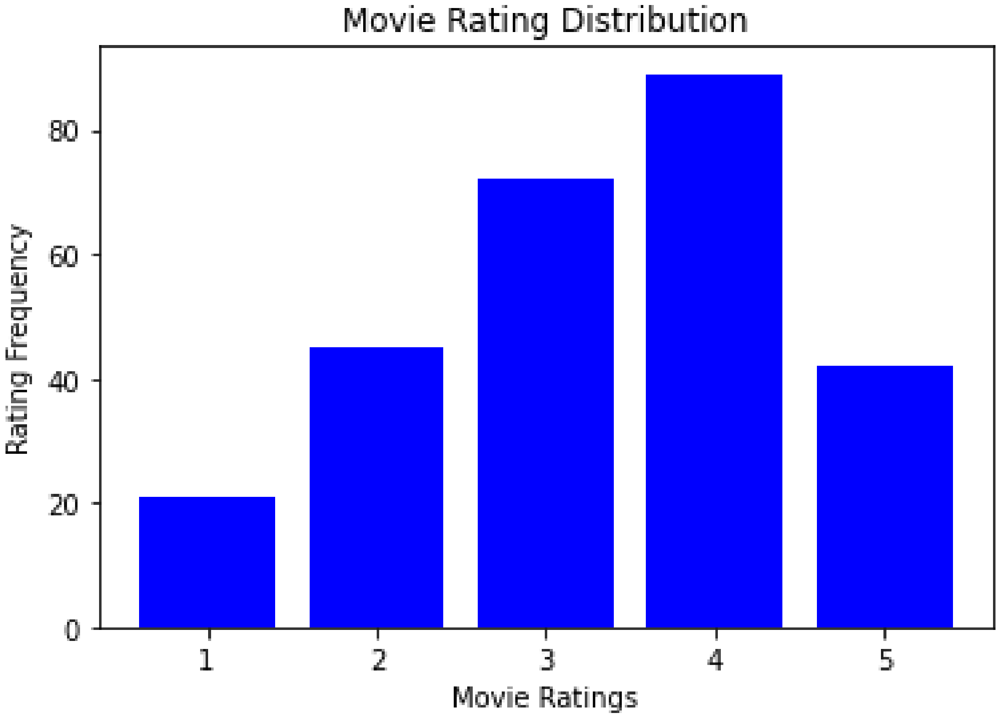


[9]:
```
# pie plot

# create the data
subjects = ["Mathematics", "Science", "Communication Skills", "Computer␣
 ↪Application"]
scores = [85,62,57,92]

# plot the pie plot
plt.pie(scores,
       labels=subjects,
       colors=['r', 'g', 'b', 'y'],
       startangle=90,
       shadow=True,
       explode=(0, 0.1, 0,0),
       autopct='%1.1f%%')
# add title to graph
plt.title("Student Performance")

# draw the chart
plt.show()
```

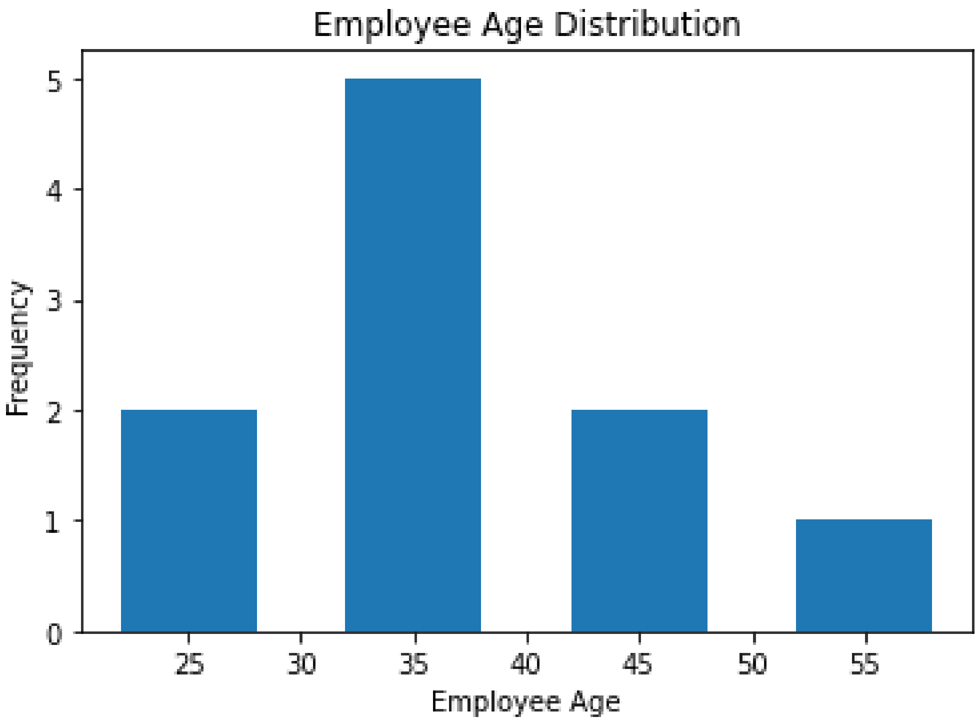


[10]:

```
# bar plot

# create data
movie_ratings = [1,2,3,4,5]
rating_counts = [21, 45, 72, 89, 42]

# plot the data
plt.bar(movie_ratings, rating_counts, color='blue')

# add X label on X-axis
plt.xlabel("Movie Ratings")

# add Y label on Y-axis
plt.ylabel("Rating Frequency")

# add a title to graph
plt.title("Movie Rating Distribution")

# show the plot
plt.show()
```



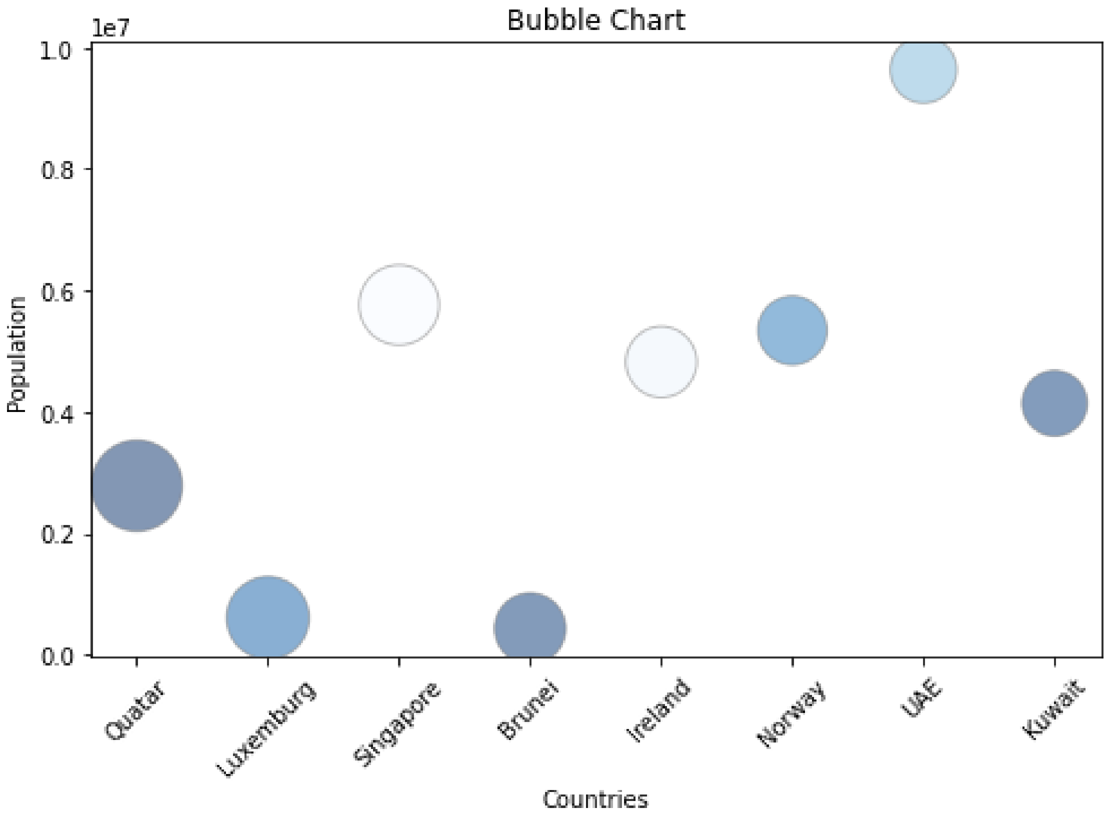
[11]:
```
# histogram plot

# create the data
employee_age = [21, 28, 32, 34, 35, 35, 37, 42, 47, 55]

# create bins for histogram
bins = [20, 30, 40, 50, 60]

# plot the histogram
plt.hist(employee_age, bins, rwidth=0.6)

# add X label on X-axis
plt.xlabel("Employee Age")

# add Y label on Y-axis
plt.ylabel("Frequency")

# add title to graph
plt.title("Employee Age Distribution")

# show the plot
plt.show()
```



[13]:

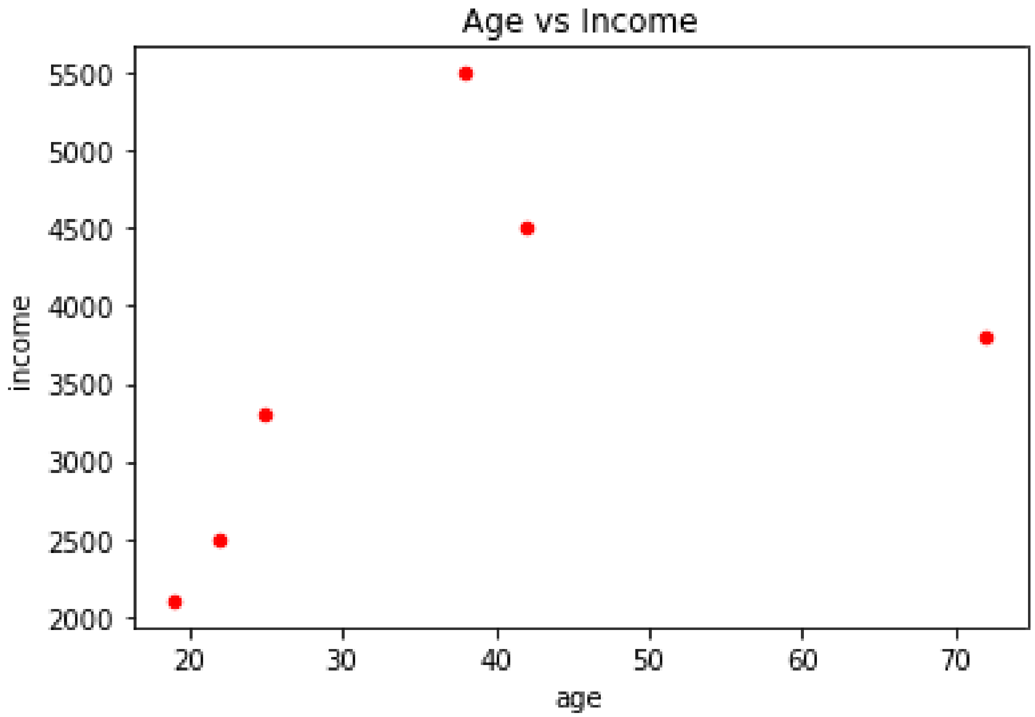
```
# bubble plot
import numpy as np

# set figure size
plt.figure(figsize=(8,5))

# create the data
countries = ['Quatar', 'Luxemburg', 'Singapore', 'Brunei', 'Ireland', 'Norway',␣
 ↪'UAE', 'Kuwait']

populations = [2781682, 604245, 5757499, 428963, 4818690, 5337962, 9630959,␣
 ↪4137312]

gpd_per_capita = [130475, 106705, 100345, 79530, 78785, 74356, 69382, 67000]

# scale GDP per capita income to shoot the bubbles in the graph

scaled_gpd_per_capita = np.divide(gpd_per_capita, 80)


colors = np.random.rand(8)

# draw the scatter diagram
```

```
plt.scatter(countries, populations, s=scaled_gpd_per_capita, c=colors,␣
 ↪cmap="Blues", edgecolors="grey", alpha=0.5)

# add X Label on X-axis
plt.xlabel("Countries")

# add Y Label on Y-axis
plt.ylabel("Population")

# add title to graph
plt.title("Bubble Chart")

# rotate x label for clear visualization
plt.xticks(rotation=45)

# show the plot
plt.show()
```



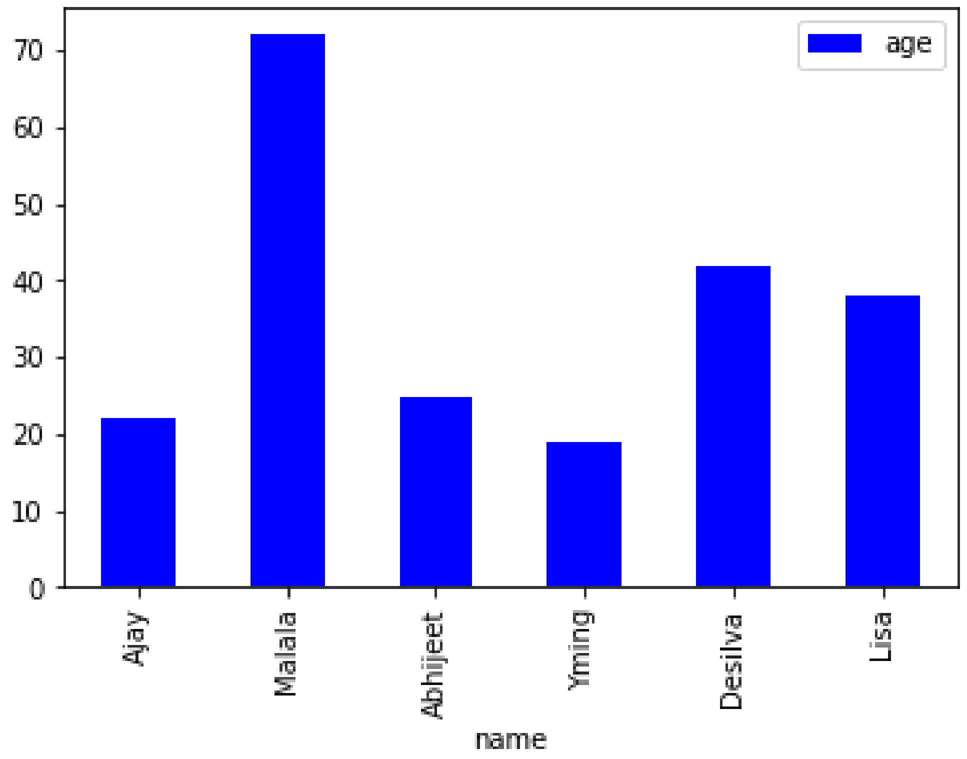
[16]:
```
# pandas plotting

# import required libraries
```

```
import pandas as pd

# let's create a Dataframe
df = pd.DataFrame({
    'name': ['Ajay', 'Malala', 'Abhijeet', 'Yming', 'Desilva', 'Lisa'],
    'age' : [22, 72, 25, 19, 42, 38],
    'gender': ['M', 'F', 'M', 'M', 'M', 'F'],
    'country':['India', 'Pakistan', 'Bangladesh', 'China', 'Srilanka', 'UK'],
    'income': [2500, 3800, 3300, 2100, 4500, 5500]
})

# create a scatter plot
df.plot(kind = 'scatter', x='age', y='income', color='red', title='Age vs␣
 ↪Income')

# show figure
plt.show()
```

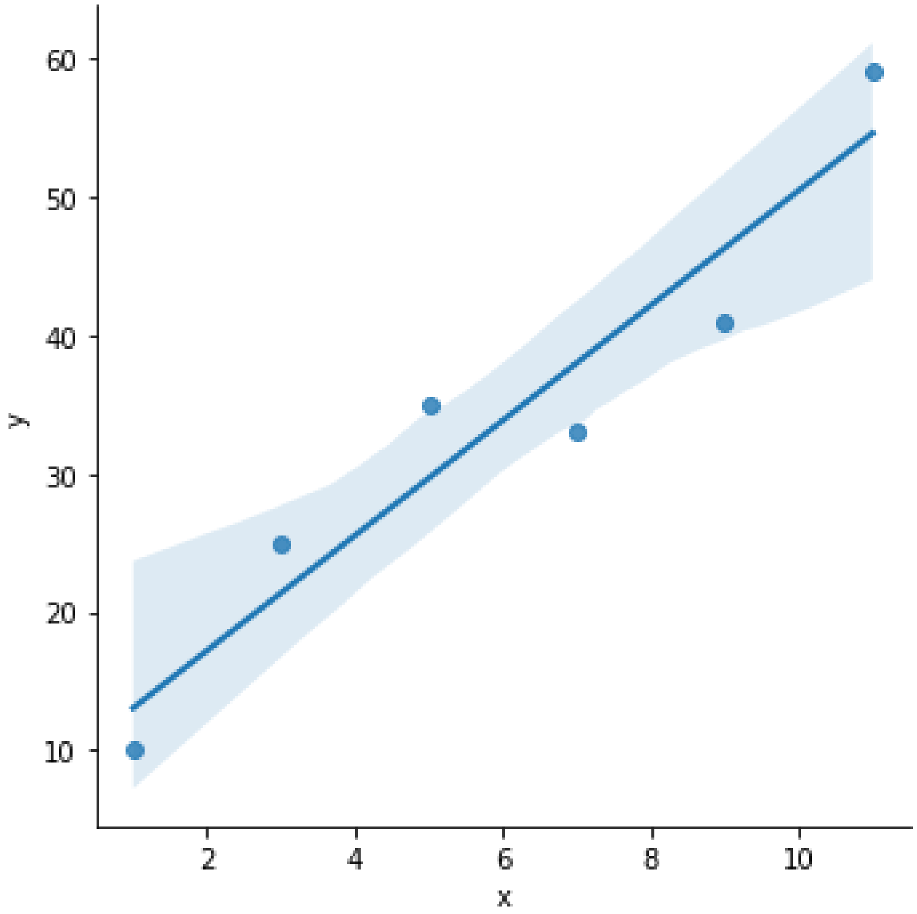


[17]:

```
# create bar plot
df.plot(kind='bar', x='name', y='age', color='blue')

# show figure
plt.show()
```

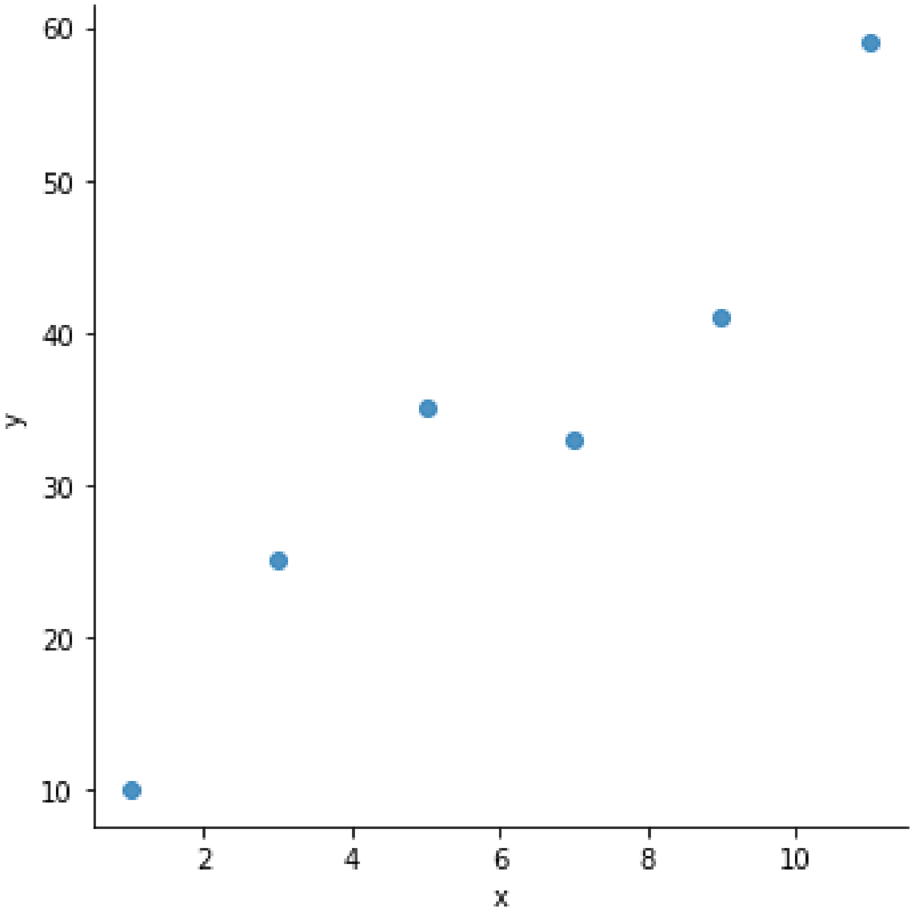


```
[19]: # seaborn package

      # lm plots

      # import required libs
      import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt

      # create DataFrame
      df = pd.DataFrame({'x':[1, 3, 5, 7, 9, 11], 'y': [10,25,35,33,41,59]})

      # create lmplot
      sns.lmplot(x='x', y = 'y', data=df)

      # show figure
      plt.show()
```

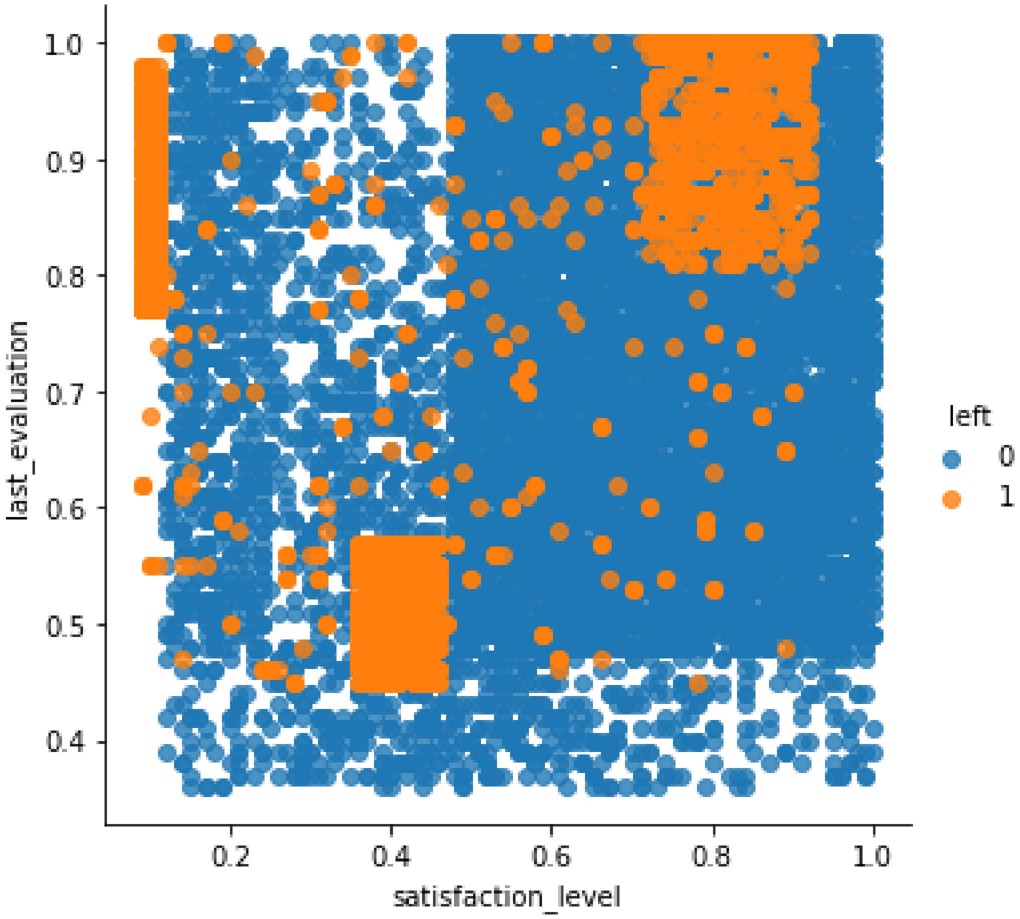


[20]:

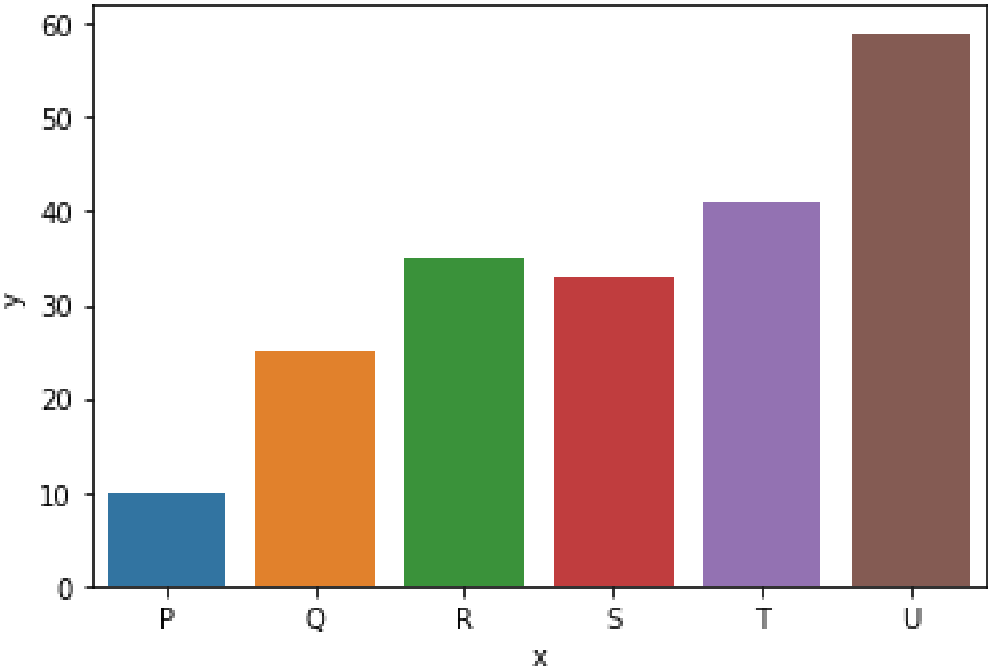
```
# create lmplot
sns.lmplot(x = 'x', y = 'y', data=df, fit_reg=False)

# show figure
plt.show()
```



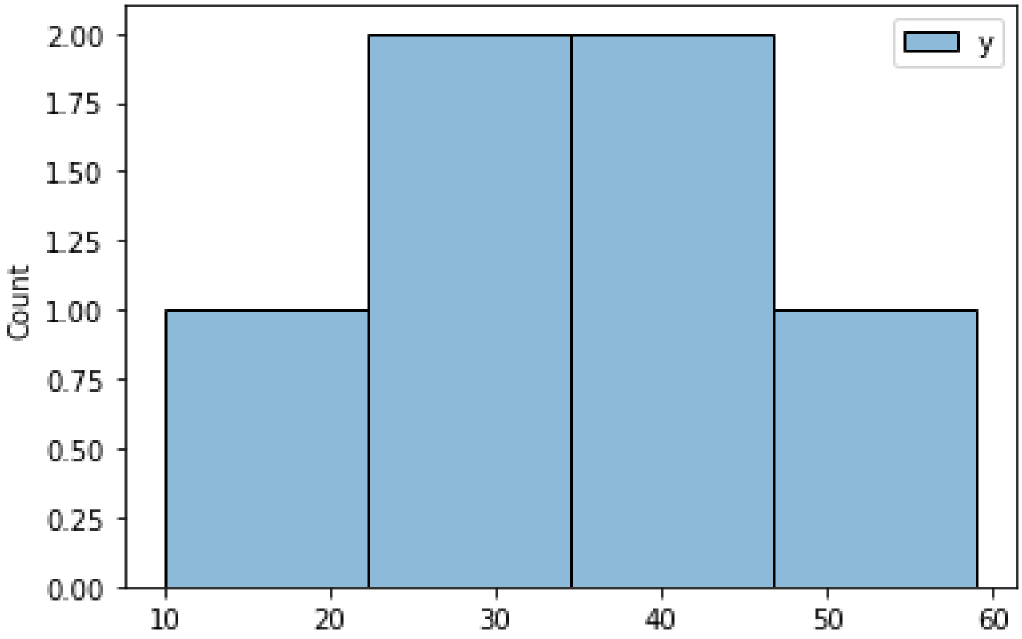
[22]:
```
# load the dataset
df = pd.read_csv("HR_comma_sep.csv")

# create lmplot
sns.lmplot(x='satisfaction_level', y='last_evaluation', data=df, fit_reg=False,␣
  ↪hue='left')

# show figure
plt.show()
```



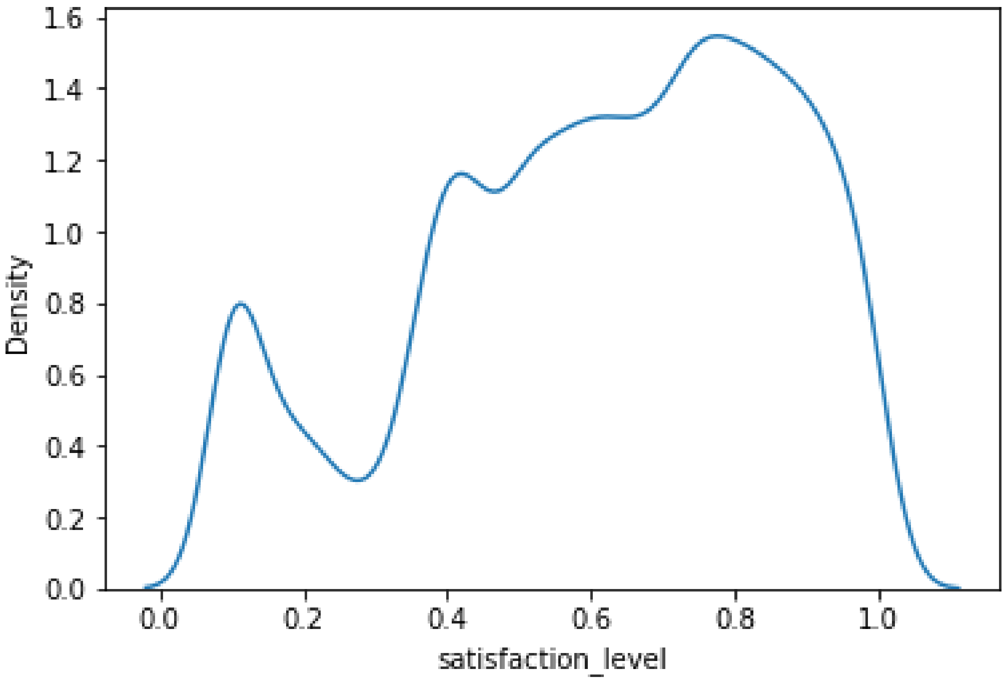
[23]:
```
# bar plots

# create DataFrame
df = pd.DataFrame({'x':['P', 'Q', 'R', 'S', 'T', 'U'], 'y':[10,25,35,33,41,59]})

# create lmplot
sns.barplot(x='x', y='y', data=df)

# show figure
plt.show()
```



[29]:
```
# distribution plots

# create a distribution plot (Histogram)
# sns.displot(df.satisfaction_level - failed)
sns.histplot(df)

# show figure
plt.show()
```

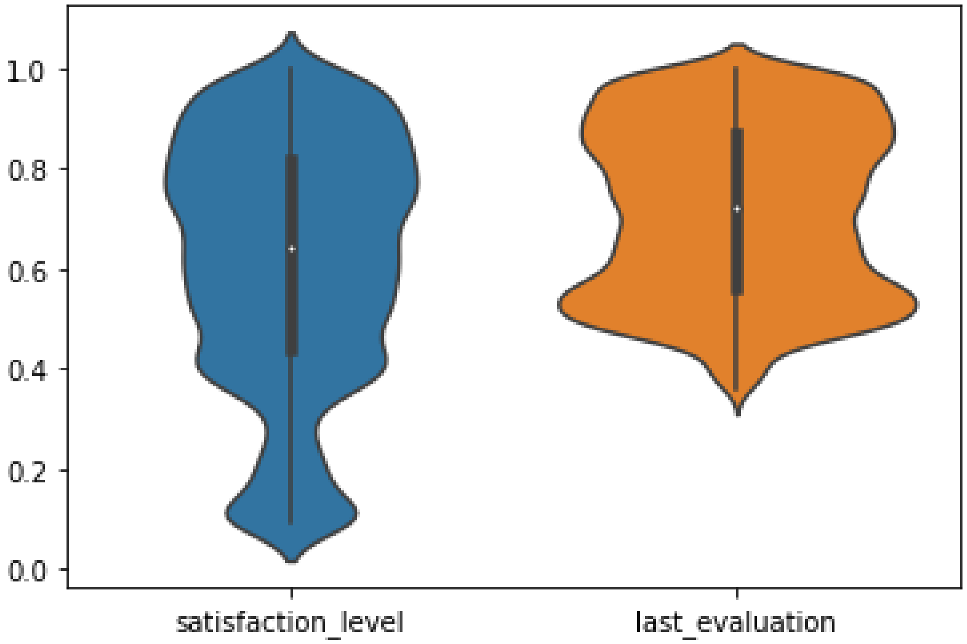


[39]:
```
# KDE plots

# create density plot
sns.kdeplot(df.satisfaction_level)

# show figure
plt.show()
```

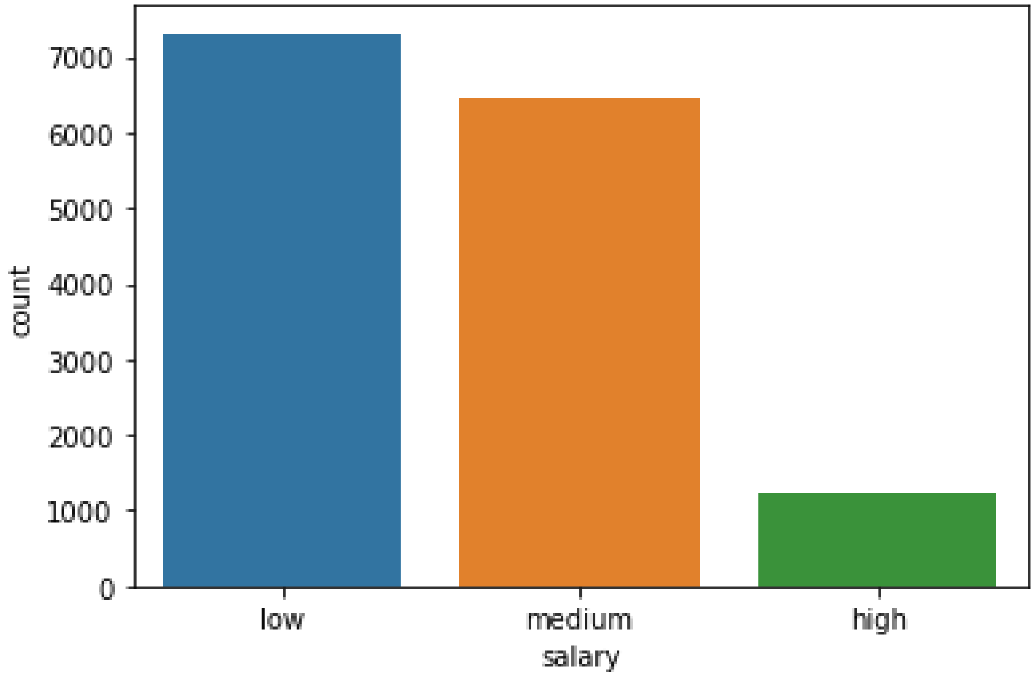


[41]:

```
# violin plots

# create violin plot
sns.violinplot(data=df[['satisfaction_level','last_evaluation']])

# show figure
plt.show()
```

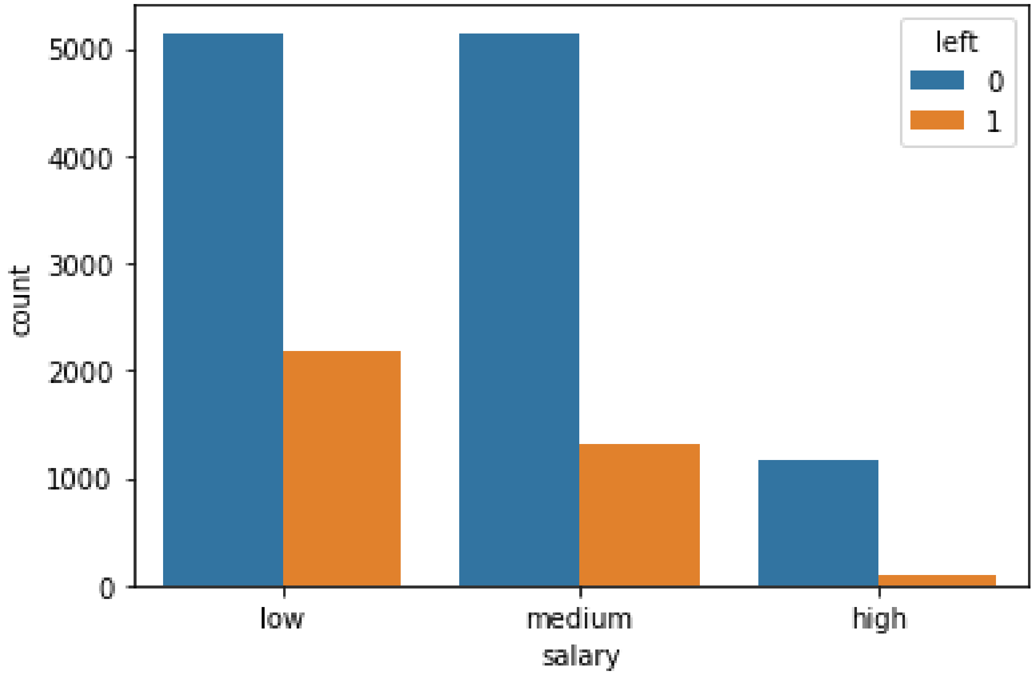


[42]:
```
# count plots

# create count plot (Histogram)

sns.countplot(x='salary', data=df)

# show figure
plt.show()
```



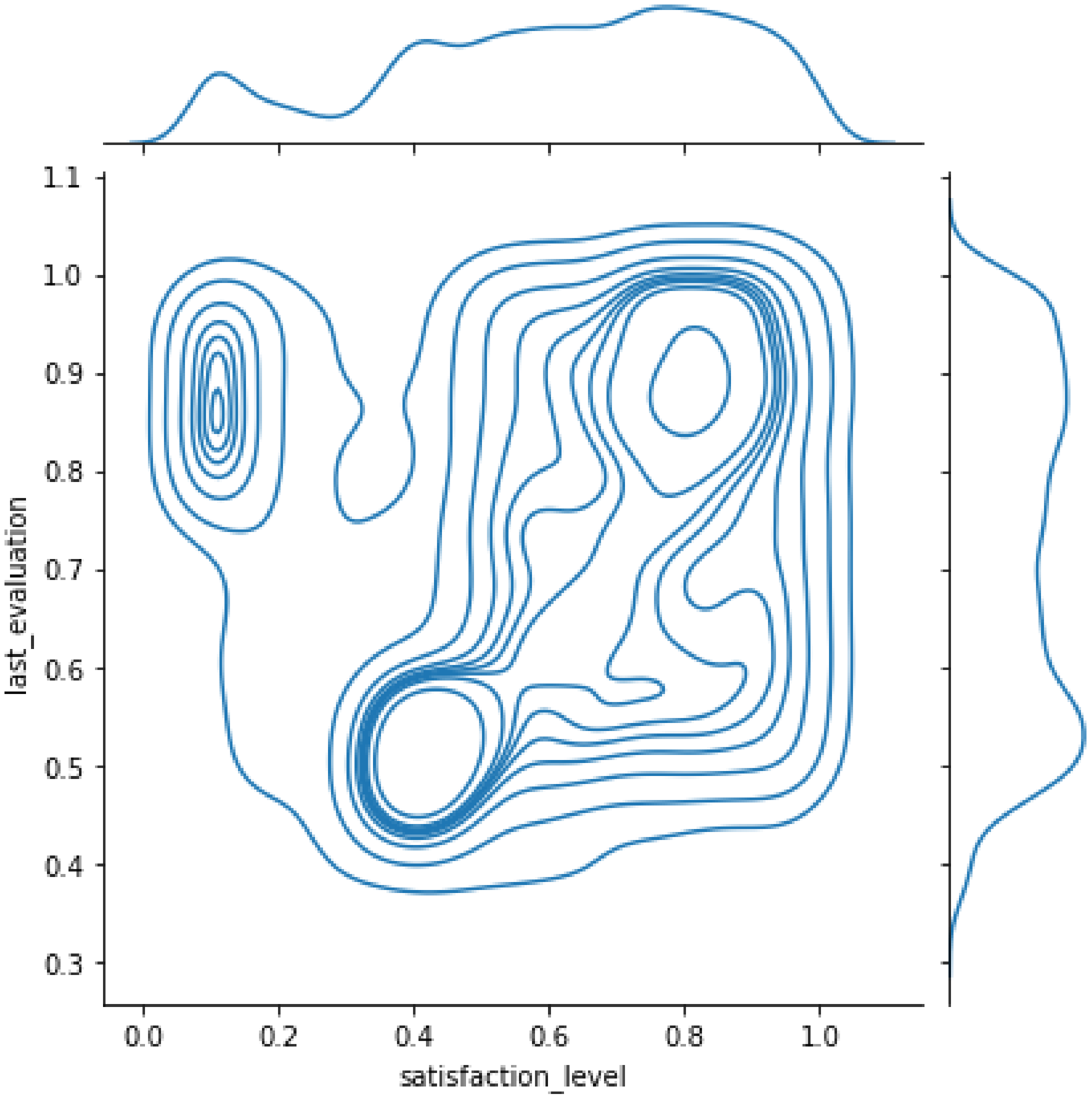
[43]:

```
# create count plot (Histogram)
sns.countplot(x='salary', data=df, hue='left')

# show figure
plt.show()
```



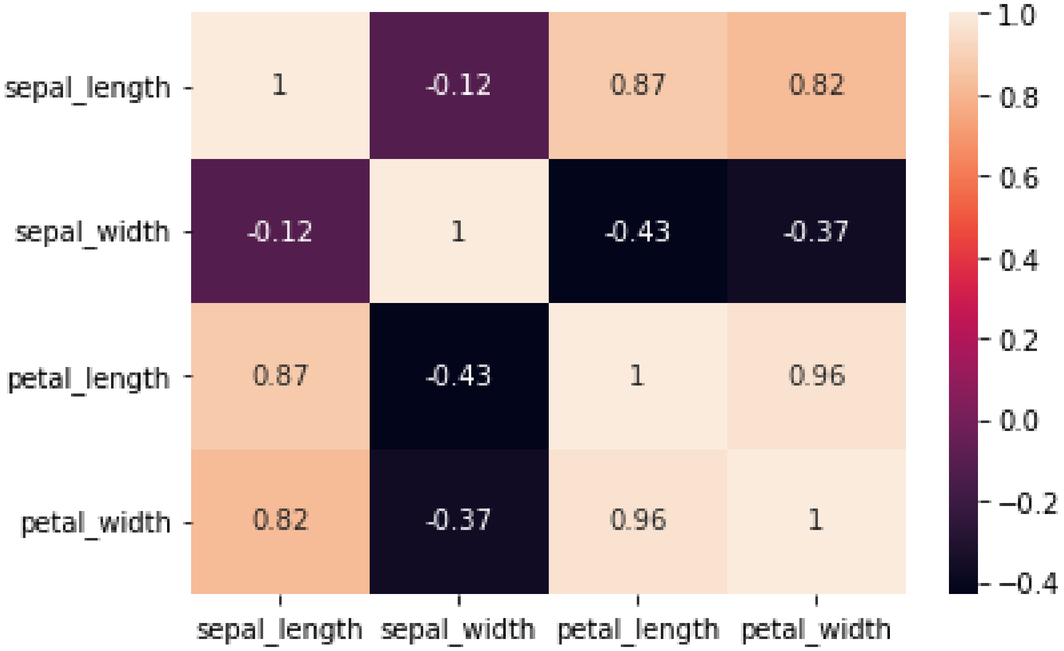
[44]:

```
# join plots

# create joint plot using kernel density estimation(kde)
sns.jointplot(x='satisfaction_level', y='last_evaluation', data=df, kind="kde")

# show figure
plt.show()
```



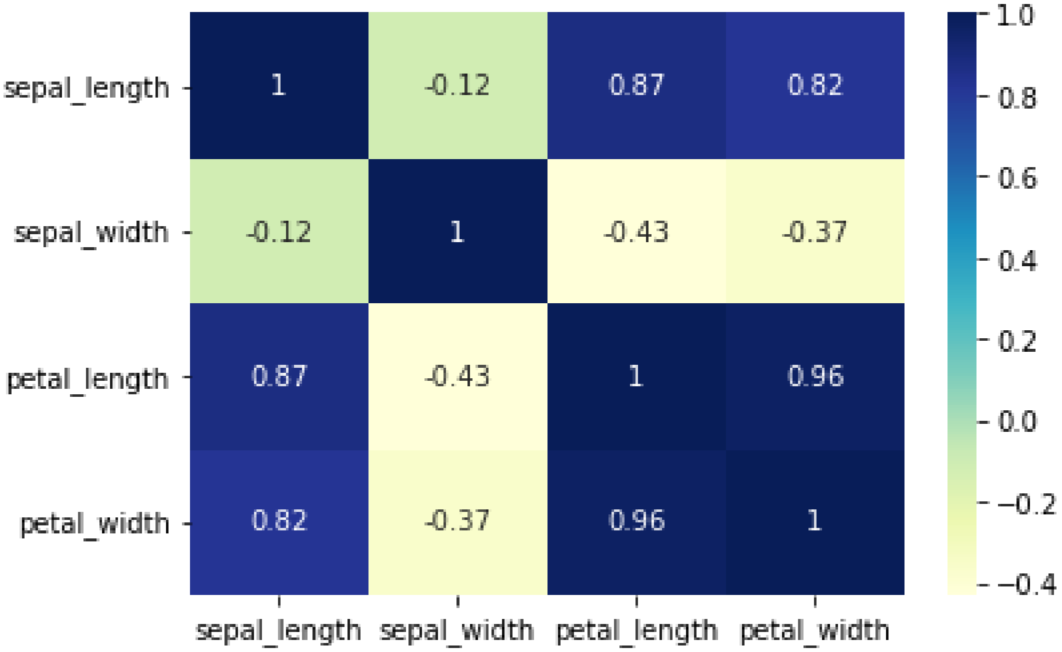
[45]:

```
# heatmaps
# import required library
import seaborn as sns

# read iris data using laod_dataset() func
data = sns.load_dataset("iris")

# find correlation
cor_matrix = data.corr()

# create heatmap
sns.heatmap(cor_matrix, annot=True)

# show figure
plt.show()
```



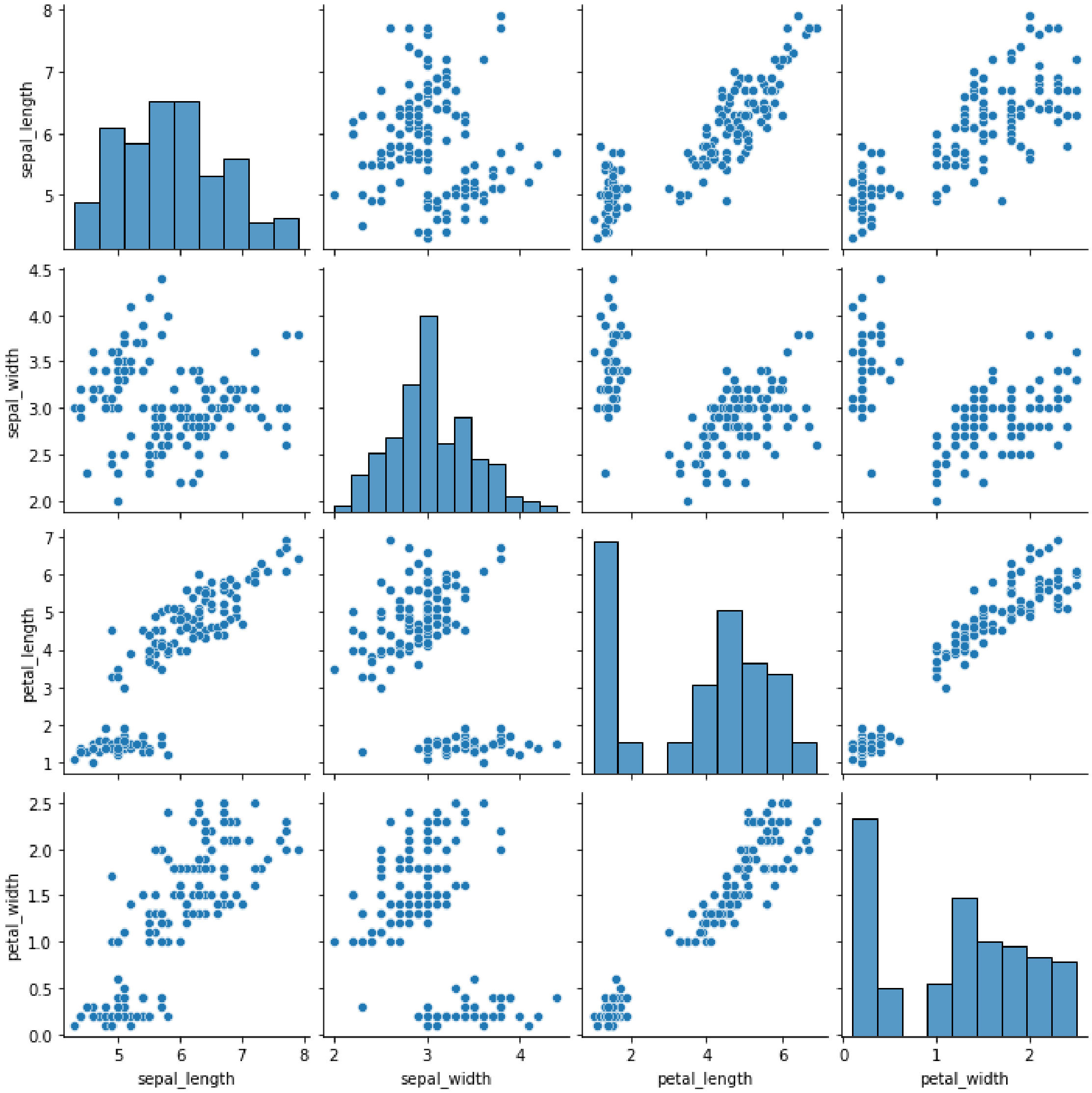
[46]:

```
# create heatmap
sns.heatmap(cor_matrix, annot=True, cmap="YlGnBu")

# show figure
plt.show()
```



[47]:

```
# pair plots

# load iris data using load_dataset() func
data = sns.load_dataset("iris")

# create a pair plot
sns.pairplot(data)

# show figure
plt.show()
```



[49]:

```
# plotting a simple graph

# import the required modules
from bokeh.plotting import figure
from bokeh.plotting import output_notebook
from bokeh.plotting import show

# create the data
x = [1, 3, 5, 7, 9, 11]
y = [10, 25, 35, 33, 41, 59]

# output to notebook
output_notebook()
```

```python
# instantiate a figure
fig = figure(plot_width = 500, plot_height = 350)

# create scatter circle marker plot by rendering the circles
fig.circle(x, y, size = 10, color = "red", alpha = 0.7)

# show the plot
show(fig)
```

[50]:

```python
# glyphs

# create the data
x_values = [1, 3, 5, 7, 9, 11]
y_values = [10, 25, 35, 33, 41, 59]

# output to notebook
output_notebook()

# instantiate a figure
p = figure(plot_width = 500, plot_height = 350)

# create a line plot
p.line(x_values, y_values, line_width = 1, color = "blue")

# show the plot
show(p)
```

[51]:

```python
# layouts

# import layouts lib
from bokeh.layouts import row, column

# import iris flower dataset as pandas DataFrame
from bokeh.sampledata.iris import flowers as df

# output to notebook
output_notebook()

# instantiate a figure
fig1 = figure(plot_width = 300, plot_height = 300)
fig2 = figure(plot_width = 300, plot_height = 300)
fig3 = figure(plot_width = 300, plot_height = 300)

# create scatter marker plot by render the circles
fig1.circle(df['petal_length'], df['sepal_length'], size = 8,
            color = "green", alpha = 0.5)
```

```
fig2.circle(df['petal_length'], df['sepal_length'], size = 8,
            color = "blue", alpha = 0.5)
fig3.circle(df['petal_length'], df['sepal_length'], size = 8,
            color = "red", alpha = 0.5)

# create row layout
row_layout = row(fig1, fig2, fig3)

# show the plot
show(row_layout)
```

[52]:

```
# create column layout
col_layout = column(fig1, fig2, fig3)

# show the plot
show(col_layout)
```

[53]:

```
# nested layout using row and column layouts

# create nested layout
nasted_layout = row(fig1, column(fig2, fig3))

# show the plot
show(nasted_layout)
```

[58]:

```
# interactions
# import missing libs
from bokeh.models import CategoricalColorMapper

# instantiate a figure object
fig = figure(plot_width = 500, plot_height = 350,
             title = "Petal length Vs. Petal Width", x_axis_label =␣
 ↪'petal_length',
             y_axis_label='petal_width')

# create scatter marker plot by render the circles
for specie, color in zip(['setosa', 'virginica', 'versicolor'],
                         ['blue', 'green', 'red']):
    data = df[df.species==specie]
    fig.circle('petal_length', 'petal_width', size=8, color=color, alpha = 0.7,
              legend_label=specie, source = data)

# set the legend location and click policy
fig.legend.location = 'top_left'
fig.legend.click_policy = "hide"

# show the plot
```

```
show(fig)
```

[60]:

```
# mute click policy

# create scatter marker plot by render the circles
for specie, color in zip(['setosa', 'virginica', 'versicolor'],
                         ['blue', 'green', 'red']):
    data = df[df.species==specie]
    fig.circle('petal_length', 'petal_width', size=8, color=color,
               alpha = 0.7, legend_label=specie, source = data,
               muted_color=color, muted_alpha = 0.2)
```

[61]:

```
# set the legend location and click policy
fig.legend.location = 'top_left'
fig.legend.click_policy = "mute"

# show the plot
show(fig)
```

[63]:

```
# hover tool
from bokeh.models import HoverTool

# output to notebook
output_notebook()

# create color mapper for categorical column
mapper = CategoricalColorMapper(factors=['setosa', 'virginica', 'versicolor'],
                               palette=['blue', 'green', 'red'])

color_dict = {'field': 'species', 'transform':mapper}

# create hovertool and specify the hovering information
hover = HoverTool(tooltips=[('Species type', '@species'),
                  ('IRIS Petal Length', '@petal_length'),
                  ('IRIS Petal Width', '@petal_width')])

# instantiate a figure object
p = figure(plot_width = 500, plot_height = 350, title =
           "Petal length Vs. Petal Width",
          x_axis_label = 'petal_length',
          y_axis_label = 'petal_width',
          tools=[hover, 'pan', 'wheel_zoom'])

# create scatter marker plot by render the circles
p.circle('petal_length', 'petal_width', size=8, color = color_dict,
        alpha=0.5, legend_group='species', source=df)
```

```
# set the legend location
p.legend.location = 'top_left'

# show the plot
show(p)
```

[69]:

```
# widgets

# import missing libs
from bokeh.plotting import figure
from bokeh.plotting import output_notebook
from bokeh.plotting import show
from bokeh.models.widgets import Tabs
from bokeh.models.widgets import Panel

# import iris flower dataset as pandas DataFrame
from bokeh.sampledata.iris import flowers as df

# output to notebook
output_notebook()

# instantiate a figure
fig1 = figure(plot_width = 300, plot_height = 300)
fig2 = figure(plot_width = 300, plot_height = 300)

# create scatter marker plot by render the circles
fig1.circle(df['petal_length'], df['sepal_length'], size=8,
           color="green", alpha=0.5)
fig2.circle(df['petal_length'], df['sepal_length'], size=8,
           color="blue", alpha=0.5)

# create panels
tab1 = Panel(child=fig1, title='tab1')
tab2 = Panel(child=fig2, title='tab2')

# create tab by putting panels into it
tab_layout = Tabs(tabs=[tab1, tab2])

# show the plot
show(tab_layout)
```

[70]:

```
# slider

# import the required modules
from bokeh.plotting import Figure
from bokeh.plotting import output_notebook
from bokeh.plotting import show
```

```python
from bokeh.models import CustomJS
from bokeh.models import ColumnDataSource
from bokeh.models import Slider
from bokeh.layouts import column

# show output in notebook
output_notebook()

# create list of data
x = [x for x in range(0, 100)]
y = x

# create a DataFrame
df = ColumnDataSource(data={"x_values":x, "y_values":y})

# instantiate the Figure object
fig = Figure(plot_width = 350, plot_height = 350)

# create a line plot
fig.line('x_values', 'y_values', source=df, line_width=2.5, line_alpha=0.8)

# create a callback using CustomJS
callback = CustomJS(args=dict(source=df), code="""
    var data = source.data;
    var f = cb_obj.value
    var x_values = data['x_values']
    var y_values = data['y_values']
    for(var i = 0; i < x_values.length; i++) {
        y_values[i] = Math.pow(x_values[i], f)
    }
    source.change.emit();
""")

slider_widget = Slider(start=0.0, end=10, value=1, step=.1,
                       title="Display power of x")
slider_widget.js_on_change('value', callback)

# create layout
slider_widget_layout = column(fig, slider_widget)

# display the layout
show(slider_widget_layout)
```

[ ]: