Importing and Visualizing data in Google Colab.

❖ LOADING DIRECTLY:

```
import pandas as pd
mydata=pd.read_csv('/content/HR.csv')
```

1. **import pandas as pd**: This line imports the Pandas library and gives it an alias **pd**. This alias is a common convention in the Pandas community to make it easier to reference the library in the code.
2. **mydata = pd.read_csv('/content/HR.csv')**: This line reads data from a CSV (Comma-Separated Values) file located at the path **/content/HR.csv** and stores it in a Pandas DataFrame called **mydata**.
   - **pd.read_csv()** is a Pandas function that reads data from a CSV file and creates a DataFrame from it.
   - **/content/HR.csv** is the path to the CSV file you want to read. Make sure the file exists at this location.
   - The result of **pd.read_csv()** is assigned to the variable **mydata**, so you can work with the data in the DataFrame using this variable.

```
mydata.head(20)
```

The code **mydata.head(20)** is used to display the first 20 rows of the DataFrame **mydata**. Specifically, it will show the top 20 rows of your dataset. This is a common operation in data analysis to quickly inspect the structure and content of the data

❖ LOAD DATA FROM DRIVE:

```
from google.colab import drive
drive.mount('/content/GEO') #Create a virtual drive
```

1. **from google.colab import drive**: This line imports the **drive** module from the **google.colab** library. Google Colab provides this module to allow you to connect and interact with your Google Drive from within the Colab environment.
2. **drive.mount('/content/GEO')**: This line initiates the process of mounting your Google Drive. Here's what each part of this line does:
   - **drive.mount()**: This is a function provided by the **drive** module, and it is used to mount your Google Drive. It will prompt you to authorize access to your Google Drive account.
   - **'/content/GEO'**: This is the path where you want to mount your Google Drive. In this case, you are specifying the mount point as '/content/GEO'. This means that once the code is executed, you will be able to access the contents of your Google Drive under the '/content/GEO' directory in the Colab environment.

```
mydata=pd.read_csv('/content/GEO/MyDrive/sales.csv')
N/B mydata is just a normal variable.
```

- **mydata**: This is the variable where the data from the CSV file will be stored. It's a Pandas DataFrame.
- **pd.read_csv()**: This is a Pandas function used to read data from a CSV file and create a DataFrame from it.
- **'/content/GEO/MyDrive/sales.csv'**: This is the file path to the CSV file you want to read. In this case, you've provided an absolute path within your Google Drive. It starts with '/content/GEO/MyDrive/', which is a common path used to access your Google Drive in Google Colab, and then specifies 'sales.csv' as the file name.

mydata.head(20)   TO READ THE DATA.

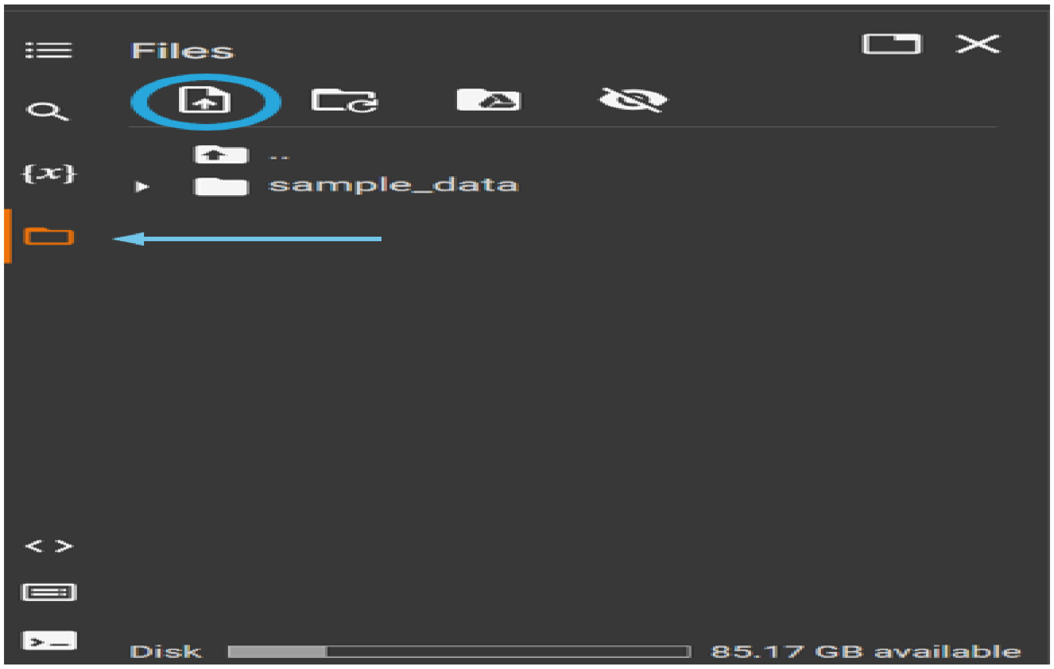
You can use:
pd.read_csv('/content/GEO/MyDrive/sales.csv').head

1. **pd.read_csv('/content/GEO/MyDrive/sales.csv')**: This part of the code reads data from the CSV file 'sales.csv' located in your Google Drive, as specified by the file path '/content/GEO/MyDrive/sales.csv'. It creates a Pandas DataFrame containing the data.
2. **.head()**: This is a Pandas method used to display the first few rows of the DataFrame. By default, it shows the first 5 rows, but you can specify the number of rows you want to display by passing a number within the parentheses. In this case, with empty parentheses, it will display the first 5 rows of the DataFrame.

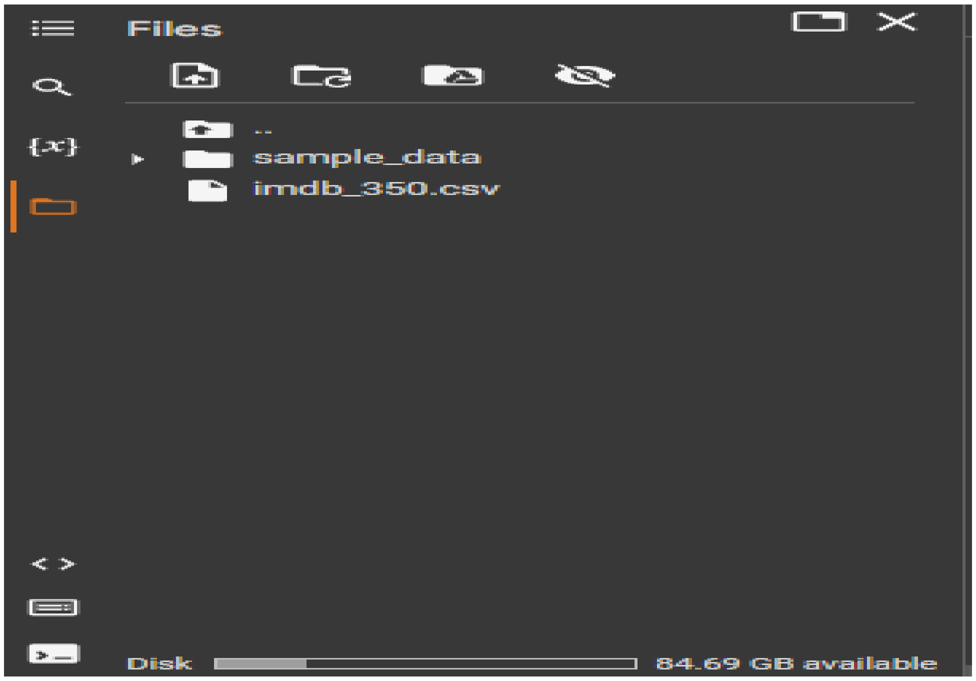
To read all data use mydata.head(20)

For ease of use, follow the steps below to upload the data file to the Colab environment.

1. **Open the FILES tab** on the left side of the window and **select the circled file upload button**.



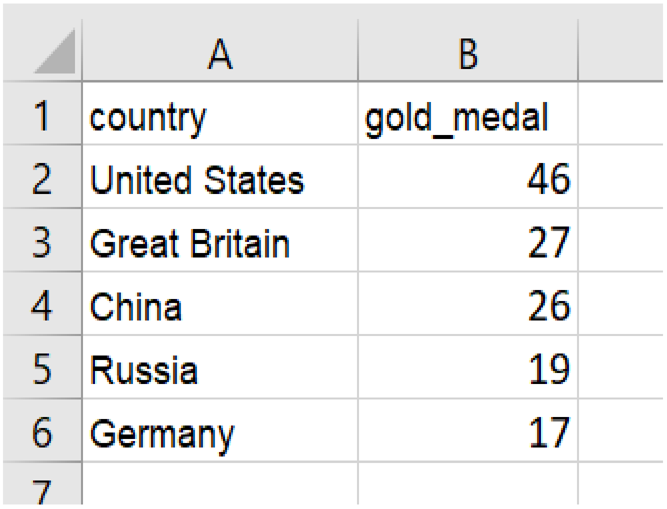
2. Select the data file from its **location on your system**.



3. You can refer to this file using a **relative path ("./imdb_350.csv")** in the environment.

EXAMPLE:

Create the dataset below and name in medal.csv



	A	B
1	country	gold_medal
2	United States	46
3	Great Britain	27
4	China	26
5	Russia	19
6	Germany	17
7		

Load and read the data:

```
mydata=pd.read_csv('/content/GEO/MyDrive/medal.csv')
```

```
mydata.head(20)
```

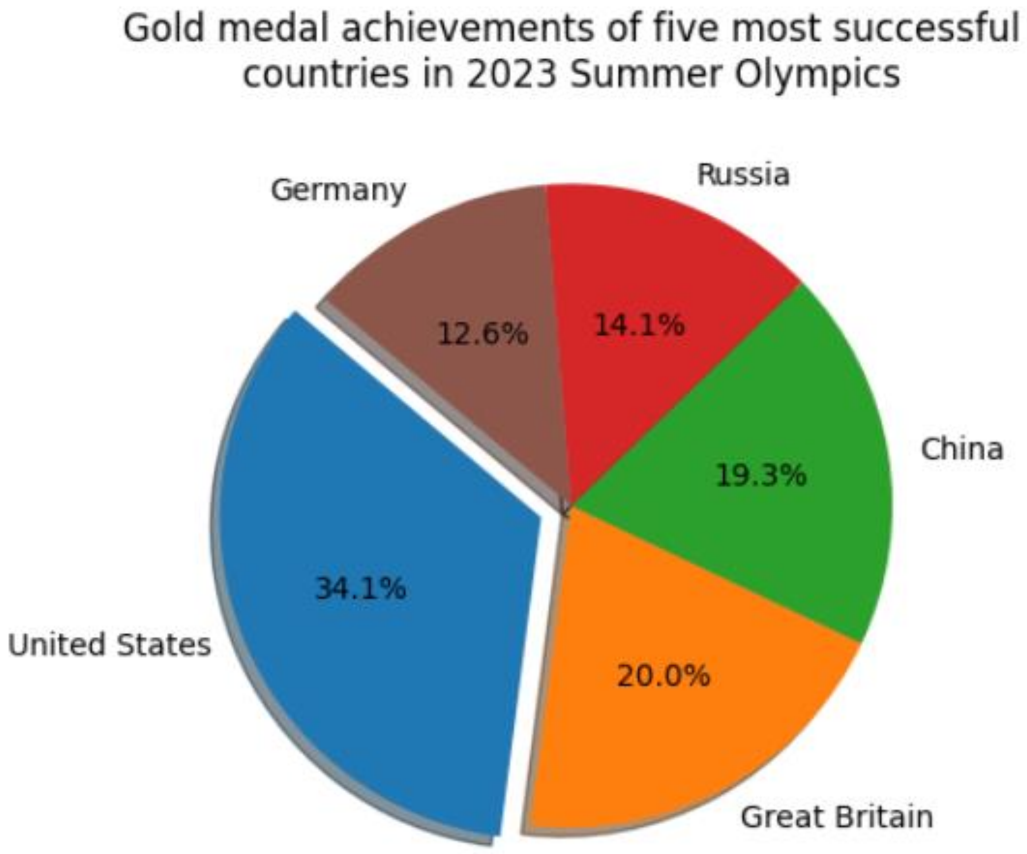
Visualization Code:

```
import matplotlib.pyplot as plt
import pandas as pd
wainaina = pd.read_csv('/content/GEO/MyDrive/medal.csv')
country_data = wainaina["country"]
medal_data = wainaina["gold_medal"]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#8c564b"]
explode = (0.1, 0, 0, 0, 0)
plt.pie(medal_data, labels=country_data, explode=explode, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("Gold medal achievements of five most successful\n"+"countries in 2023 Summer
Olympics")
plt.show()
```

Code Explanation:

1. **Importing Libraries**:
   - The code begins by importing the necessary Python libraries, including Pandas for data handling and Matplotlib.pyplot for creating visualizations. Pandas allows you to read data from various file formats, including CSV, Excel, SQL databases, and more. You can use functions like pd.read_csv() and pd.read_excel() to import data into Pandas DataFrames.
2. **Reading Data**:
   - It reads data from a CSV file named 'medal.csv' located in a specific directory. This data is loaded into a Pandas DataFrame.
3. **Data Extraction**:
   - Two specific columns are extracted from the DataFrame: "country" and "gold_medal." "country_data" contains the country names, and "medal_data" contains the number of gold medals won by each country.
4. **Color and Explode Parameters**:
   - A list of colors is defined to distinguish the sections (slices) of the pie chart.
   - The "explode" parameter determines how much each slice should be separated from the center. In this case, the first slice is slightly separated (exploded) for emphasis.
5. **Creating the Pie Chart**:
   - The pie chart is created using the extracted "medal_data" as the data values and "country_data" as the labels for each slice.
   - The "explode" and "colors" parameters are used to customize the appearance of the slices.
   - The "autopct" parameter formats and displays the percentage values on each slice.
   - "shadow=True" adds a shadow effect to the chart for a 3D appearance.
   - "startangle" specifies the angle at which the first slice of the chart begins.
6. **Setting the Title**:
   - A title is set for the pie chart to provide context and information about what the chart represents.
7. **Displaying the Chart**:
   - Finally, the code displays the pie chart using **plt.show()**

OUTPUT:



BAR CHARTS:

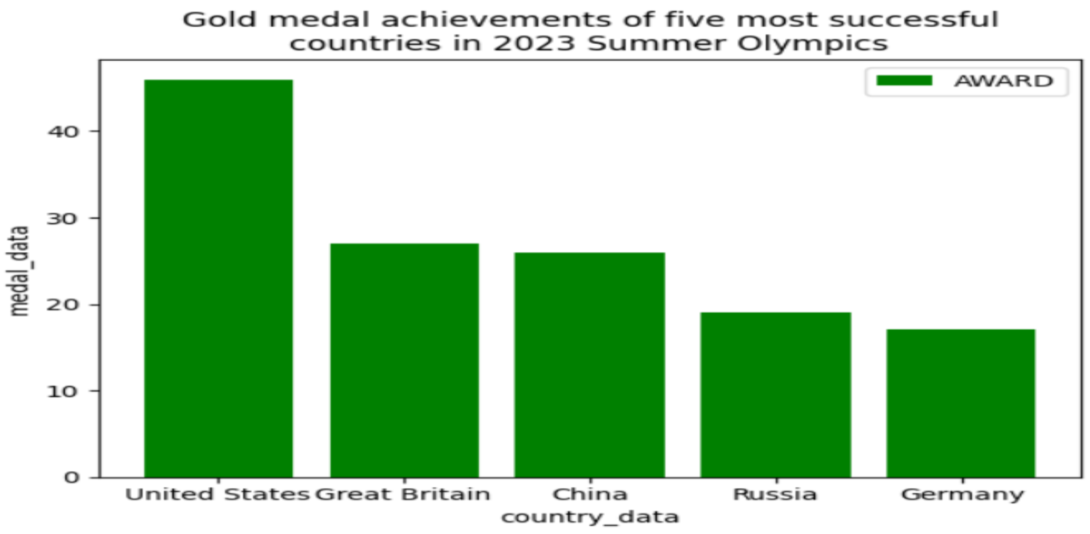
Visualization Code:

```
import matplotlib.pyplot as plt
import pandas as pd
wainaina = pd.read_csv('/content/ZU/MyDrive/zetech.csv')
country_data = wainaina["country"]
medal_data = wainaina["gold_medal"]
plt.bar(country_data, medal_data, label="AWARD", color='g')
plt.plot()
plt.xlabel("country_data")
plt.ylabel("medal_data")
plt.title("Gold medal achievements of five most successful\n"+"countries in 2023 Summer
Olympics")
plt.legend()
plt.show()
```

Code Explanation:

1. **import matplotlib.pyplot as plt**: This line imports the **matplotlib.pyplot** library and assigns it the alias **plt**. This library is commonly used for creating various types of plots and charts in Python.
2. **import pandas as pd**: This line imports the **pandas** library and assigns it the alias **pd**. Pandas is used for data manipulation and analysis, and it's often used to work with structured data like CSV files.
3. **wainaina = pd.read_csv('/content/ZU/MyDrive/zetech.csv')**: This line reads a CSV file located at the specified path and assigns the data to a DataFrame called **wainaina**. The DataFrame is a two-dimensional data structure commonly used in pandas for tabular data.
4. **country_data = wainaina["country"]**: This line creates a Series called **country_data** by extracting the "country" column from the **wainaina** DataFrame. This Series will contain the country names.
5. **medal_data = wainaina["gold_medal"]**: This line creates another Series called **medal_data** by extracting the "gold_medal" column from the **wainaina** DataFrame. This Series will contain the number of gold medals won by each country.
6. **plt.bar(country_data, medal_data, label="AWARD", color='g')**: This line creates a bar chart using **plt.bar()**. It uses **country_data** as the x-axis values (countries) and **medal_data** as the y-axis values (number of gold medals). The **label** parameter sets the label for the data, and **color** parameter sets the color of the bars to green ('g').
7. **plt.plot()**: This line is not needed in this context. It is an empty **plt.plot()** call and doesn't affect the bar chart.
8. **plt.xlabel("country_data")**: Sets the x-axis label to "country_data."
9. **plt.ylabel("medal_data")**: Sets the y-axis label to "medal_data."
10. **plt.title("Gold medal achievements of five most successful\n"+"countries in 2023 Summer Olympics")**: Sets the title of the chart to "Gold medal achievements of five most successful countries in 2023 Summer Olympics." The "\n" is used to create a line break in the title for better formatting.
11. **plt.legend()**: Adds a legend to the chart, which indicates the meaning of the data represented by the bars.
12. **plt.show()**: Finally, this line displays the plot on the screen. It's necessary to visualize the chart.

OUTPUT:



INVERTED BAR GRAPH:

Visualization Code:

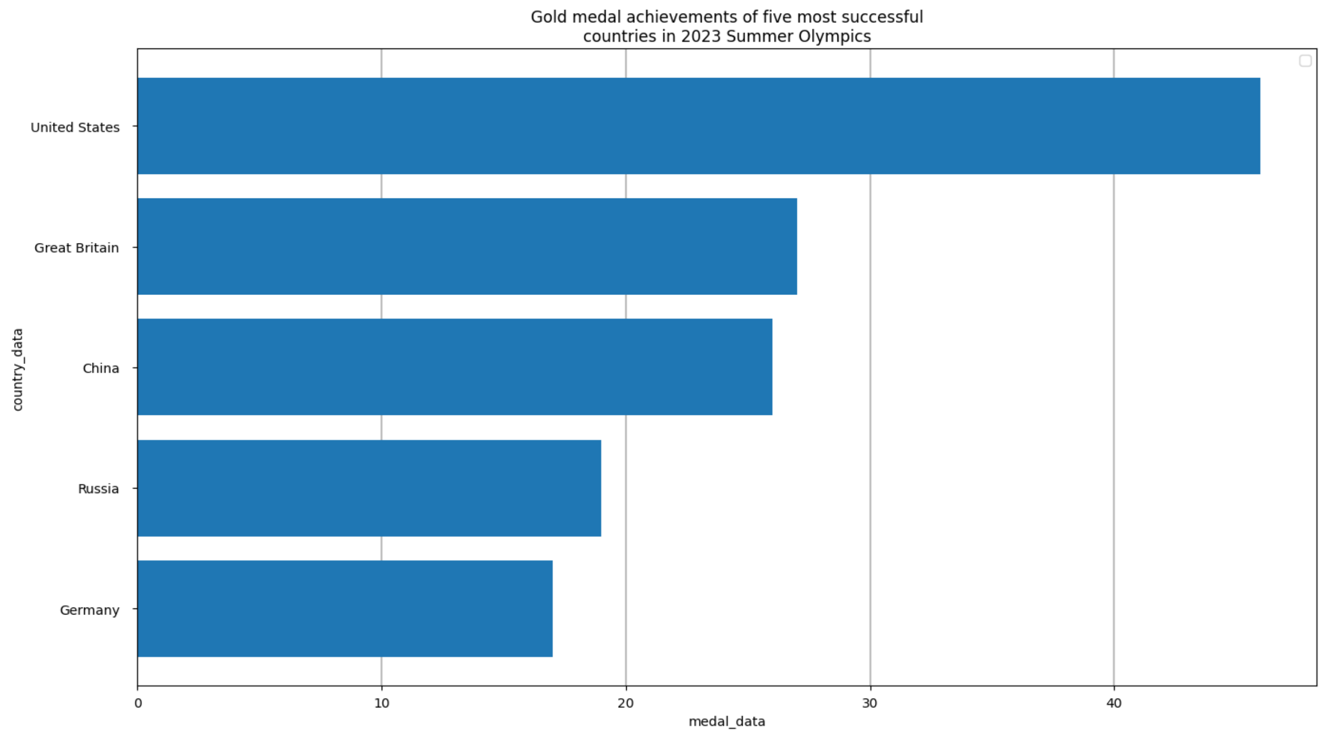
```
import matplotlib.pyplot as plt
wainaina = pd.read_csv('/content/ZETECH/MyDrive/se.csv')
country_data = wainaina["county"]
medal_data = wainaina["gold_medal"]
plt.bar(country_data, medal_data, label="AWARD", color='g')
fig, ax = plt.subplots(figsize = (16, 9))
ax.barh(country_data, medal_data)
ax.xaxis.set_tick_params(pad = 5)
ax.yaxis.set_tick_params(pad = 10)
ax.xaxis.grid(True, color ='grey',
        linestyle ='-', linewidth = 1.5,
        alpha = 0.5)
ax.set_axisbelow(True)
ax.invert_yaxis()
plt.plot()
plt.ylabel("country_data")
plt.xlabel("medal_data")
plt.title("Gold medal achievements of five most successful\n"+"countries in 2023 Summer
Olympics")
plt.legend()
plt.show()
```

Code Explanation:

1. **import matplotlib.pyplot as plt**: This line imports the **matplotlib.pyplot** module and gives it the alias **plt**. This module is used for creating various types of plots and charts.
2. **wainaina = pd.read_csv('/content/ZETECH/MyDrive/se.csv')**: This line reads a CSV file located at the given path and stores its contents in a DataFrame named **wainaina**. It appears that the Pandas library (**pd**) is used to handle data.
3. **country_data = wainaina["county"]**: This line extracts the data in the "county" column from the **wainaina** DataFrame and stores it in the **country_data** variable.
4. **medal_data = wainaina["gold_medal"]**: This line extracts the data in the "gold_medal" column from the **wainaina** DataFrame and stores it in the **medal_data** variable.
5. **plt.bar(country_data, medal_data, label="AWARD", color='g')**: This line creates a bar chart using **plt.bar()**. It plots **country_data** on the x-axis and **medal_data** on the y-axis, using green color ('g') for the bars. The **label** parameter is set to "AWARD" for the legend.
6. **fig, ax = plt.subplots(figsize = (16, 9))**: This line creates a figure and axis for the plot with a specified size of 16x9 inches. **fig** is the figure object, and **ax** is the axis object.
7. **ax.barh(country_data, medal_data)**: This line creates a horizontal bar chart using **ax.barh()**. It uses the same **country_data** and **medal_data** as before.
8. **ax.xaxis.set_tick_params(pad = 5)**: This adjusts the tick label padding for the x-axis.
9. **ax.yaxis.set_tick_params(pad = 10)**: This adjusts the tick label padding for the y-axis.
10. **ax.xaxis.grid(True, color ='grey', linestyle ='-', linewidth = 1.5, alpha = 0.5)**: This adds horizontal gridlines to the plot's x-axis for better readability.
11. **ax.set_axisbelow(True)**: This ensures that the bars are drawn below the gridlines.

12. **ax.invert_yaxis()**: This inverts the y-axis to display the countries from top to bottom.
13. **plt.plot()**: This is not necessary and can be removed as it doesn't affect the plot.
14. **plt.ylabel("country_data")**: Sets the y-axis label as "country_data."
15. **plt.xlabel("medal_data")**: Sets the x-axis label as "medal_data."
16. **plt.title("Gold medal achievements of five most successful\n"+"countries in 2023 Summer Olympics")**: Sets the title of the plot.
17. **plt.legend()**: Adds a legend to the plot.
18. **plt.show()**: Finally, this line displays the plot on the screen.

<u>OUTPUT:</u>



<u>LINE GRAPHS:</u>

<u>Visualization Code:</u>

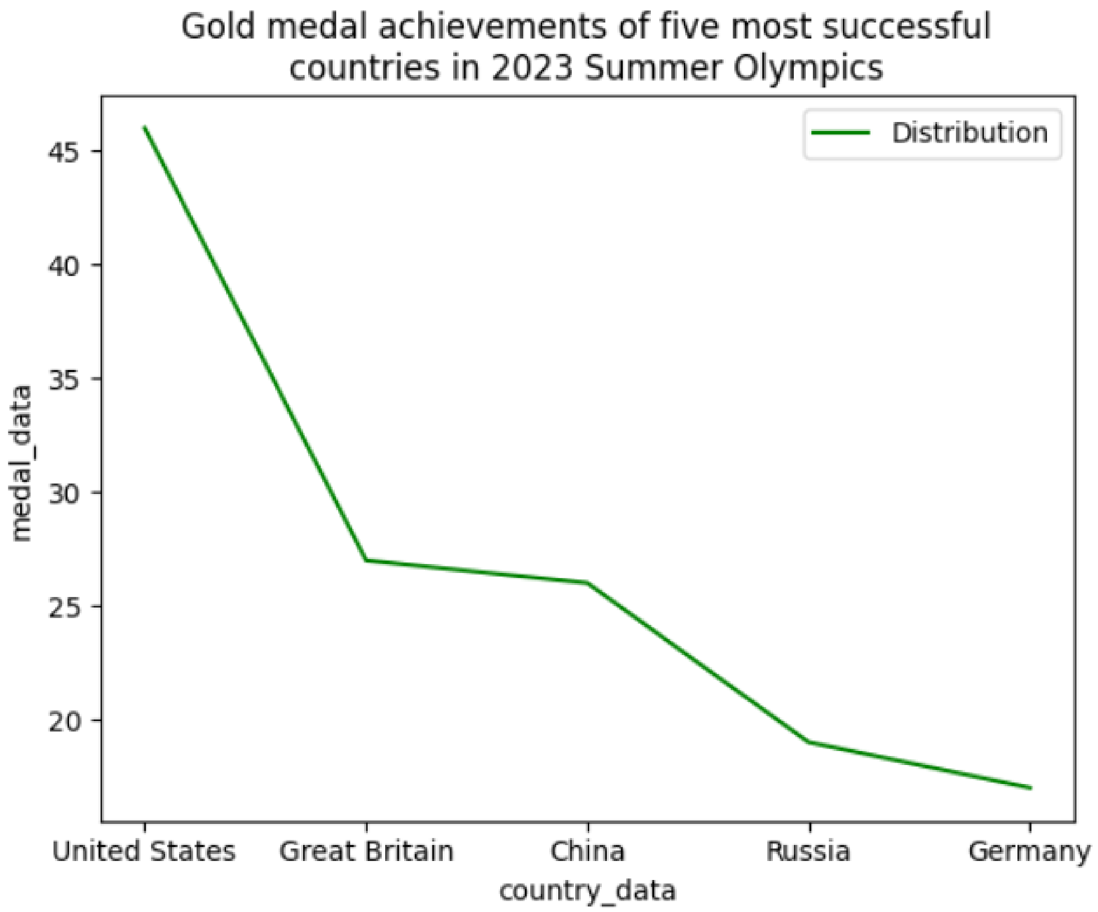
```
import matplotlib.pyplot as plt
import pandas as pd
wainaina = pd.read_csv('/content/GEO/MyDrive/medal.csv')
country_data = wainaina["country"]
medal_data = wainaina["gold_medal"]
plt.plot(country_data, medal_data, label="Distribution",color='g')
plt.plot()
plt.xlabel("country_data")
plt.ylabel("medal_data")
plt.title("Gold medal achievements of five most successful\n"+"countries in 2023 Summer
Olympics")
```

```
plt.legend()
plt.show()
```

Code Explanation:

1. **Importing Matplotlib and Pandas Libraries**:
   - The code begins by importing two Python libraries: **matplotlib.pyplot** for creating plots and charts and **pandas** for data manipulation.
2. **Reading Data from CSV**:
   - The next line reads data from a CSV file located at '/content/GEO/MyDrive/medal.csv' using Pandas. The data is loaded into a Pandas DataFrame named **wainaina**.
3. **Data Extraction**:
   - Two specific columns are extracted from the DataFrame:
     - **country_data**: This variable holds the data from the "country" column of the DataFrame, which likely contains the names of different countries.
     - **medal_data**: This variable holds the data from the "gold_medal" column of the DataFrame, which likely contains the number of gold medals won by each country.
4. **Creating a Line Plot**:
   - **plt.plot()**: This function is used to create a line plot. It takes several parameters, including:
     - **country_data**: The data for the x-axis, which represents the countries.
     - **medal_data**: The data for the y-axis, which represents the number of gold medals won.
     - **label**: A label for the plot, in this case, "Distribution."
     - **color**: The color of the line in the plot, which is set to green ('g').
5. **Customizing Plot Parameters**:
   - **plt.plot()**: This line creates the line plot but doesn't display it immediately. It prepares the plot for further customization.
6. **Setting the Axes Labels**:
   - **plt.xlabel()**: This sets the label for the x-axis, which is set to "country_data."
   - **plt.ylabel()**: This sets the label for the y-axis, which is set to "medal_data."
7. **Setting the Chart Title**:
   - **plt.title()**: This line sets the title of the line plot, providing information about what the chart represents. The title mentions "Gold medal achievements of five most successful countries in 2023 Summer Olympics."
8. **Adding a Legend**:
   - **plt.legend()**: This function adds a legend to the plot, which explains the meaning of the line(s) on the chart. In this case, the legend includes a label, "Distribution."
9. **Displaying the Plot**:
   - **plt.show()**: Finally, this line displays the line plot in your Python environment.

OUTPUT:



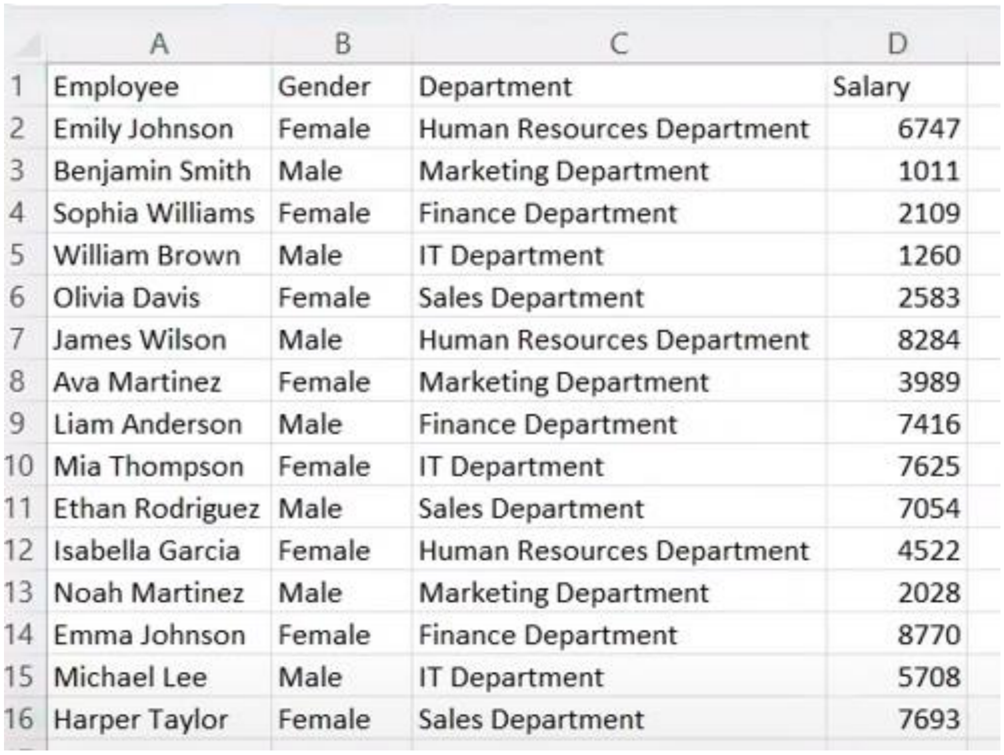
EXERCISE1:

Load the data below and generate a line graph for all the data sets given:

```
Name, Age, City, Salary
John, 30, New York, 75000
Alice, 25, Los Angeles, 68000
Bob, 35, Chicago, 80000
Sara, 28, San Francisco, 90000
Mike, 32, Boston, 82000
```

EXERCISE2:

Load the data below and generate a line graph, Bar chart and Line Graph for all the data sets given:



	A	B	C	D
1	Employee	Gender	Department	Salary
2	Emily Johnson	Female	Human Resources Department	6747
3	Benjamin Smith	Male	Marketing Department	1011
4	Sophia Williams	Female	Finance Department	2109
5	William Brown	Male	IT Department	1260
6	Olivia Davis	Female	Sales Department	2583
7	James Wilson	Male	Human Resources Department	8284
8	Ava Martinez	Female	Marketing Department	3989
9	Liam Anderson	Male	Finance Department	7416
10	Mia Thompson	Female	IT Department	7625
11	Ethan Rodriguez	Male	Sales Department	7054
12	Isabella Garcia	Female	Human Resources Department	4522
13	Noah Martinez	Male	Marketing Department	2028
14	Emma Johnson	Female	Finance Department	8770
15	Michael Lee	Male	IT Department	5708
16	Harper Taylor	Female	Sales Department	7693