To create a Series, first import the Pandas library into your notebook.

```
import pandas as pd
```

✓ 0.3s

Now, you can use the Pandas. To create a Series, you need a collection of data. Here is a list of the company's employees.

```
employees = ['Georgia','Edward','William','Jack','Juliet','Daisy','Frank']
```

✓ 0.0s

Creating a Series is super easy. To create a Series named **ds**, just use a name of a data collection.

```
ds = pd.Series(data=employees)
```

To display the contents of a Series, simply use its name.

```
ds
```

✓ 0.0s

```
0    Georgia
1     Edward
2    William
3       Jack
4     Juliet
5      Daisy
6      Frank
dtype: object
```

You can also display the contents of a Series using the print function.

```
print(ds)
```

✓ 0.0s

```
0    Georgia
1     Edward
2    William
3       Jack
4     Juliet
5      Daisy
6      Frank
dtype: object
```

Try to add an additional two employees to the list created earlier. Then, create and display Series once again.

```
employees.append('Adam')
employees.append('Ewa')
ds = pd.Series(data=employees)
ds
```

✓ 0.0s

```
0    Georgia
1     Edward
2    William
3       Jack
4     Juliet
5      Daisy
6      Frank
7       Adam
8        Ewa
dtype: object
```

Create your first Series called **components** containing the names of the five components of your computer. Then, display the contents of the Series.

```
components = ['Myszka', 'Klawiatura', 'Monitor', 'Karta graficzna', 'Głośnik']
ds = pd.Series(data=components)
ds
```

✓ 0.0s

```
0            Myszka
1        Klawiatura
2           Monitor
3    Karta graficzna
4           Głośnik
dtype: object
```

To create a Series, as a collection of data you can also use a dictionary, which consists of key and value pairs of information.

```
tools_in_stock = {'hammer':15, 'drill':4, 'screwdriver':23, 'saw':7, 'knife':9}
tools = pd.Series(data=tools_in_stock)
tools
```

✓ 0.0s

```
hammer         15
drill           4
screwdriver    23
saw             7
knife           9
dtype: int64
```

## Tasks

Modify the data collection by adding two additional tools. Then, create and display Series once again.

```
tools_in_stock.update({'hoe':23, 'axe':56})
tools = pd.Series(data=tools_in_stock)
tools
```

✓ 0.0s

```
hammer         15
drill           4
screwdriver    23
saw             7
knife           9
hoe            23
axe            56
dtype: int64
```

Pandas includes a number of functions to calculate various statistics. You can, for example, calculate the sum of tools in stock using sum(), or their arithmetic mean using mean().

```
tools.sum()
```
✓ 0.0s
```
137
```

You can also generate descriptive statistics (tools.describe()) that quantitatively describes or summarizes features from a collection of information.

Calculate the average number of tools in stock.

```
tools.mean()
```
✓ 0.0s
```
19.571428571428573
```

Generate descriptive statistics for the tools in stock.

```
tools.describe()
```
✓ 0.0s
```
count     7.000000
mean     19.571429
std      17.718701
min       4.000000
25%       8.000000
50%      15.000000
75%      23.000000
max      56.000000
dtype: float64
```

You can specify a single value in Series.

```
tools['drill']
```
✓ 0.0s
```
4
```

You can also specify a subset of values [from:to].

```
tools['drill':'saw']
```
✓ 0.0s
```
drill          4
screwdriver    23
saw             7
dtype: int64
```

For the selected values you can then calculate various statistics.

```
tools['drill':'saw'].sum()
```
✓ 0.0s
```
34
```

```
restaurant = {'Monday':23, 'Tuesday':8, 'Wednesday':20, 'Thursday':38, 'Friday':19, 'Saturday':53, 'Sunday':1}
ds = pd.Series(data=restaurant)
ds
```
✓ 0.0s
```
Monday       23
Tuesday       8
Wednesday    20
Thursday     38
Friday       19
Saturday     53
Sunday        1
dtype: int64
```

Create a Series called **restaurant** containing the number of restaurant customers on each of the seven days of the week.

With the created data series, follow the instructions below.

Display the contents of the Series.

Display the total number of restaurant customers.

```
ds.sum()
```
✓ 0.0s
```
162
```

Display the median number of restaurant customers.

```
ds.median()
```
✓ 0.0s
```
20.0
```

Display descriptive statistics.

```
ds.describe()
```
✓ 0.0s
```
count     7.000000
mean     23.142857
std      17.601407
min       1.000000
25%      13.500000
50%      20.000000
75%      30.500000
max      53.000000
dtype: float64
```

Display the list of restaurant customers on business days.

```
ds['Monday':'Friday']
```
✓ 0.0s
```
Monday       23
Tuesday       8
Wednesday    20
Thursday     38
Friday       19
dtype: int64
```

Display the average number of customers visiting the restaurant during working days.

```
ds['Monday':'Friday'].mean()
```
✓ 0.0s
```
21.6
```

Display the list of restaurant customers on the weekend.

```
ds['Saturday':'Sunday']
```
✓ 0.0s
```
Saturday    53
Sunday       1
dtype: int64
```

Display the total number of customers visiting the restaurant over the weekend.

```
ds['Saturday':'Sunday'].sum()
```
✓ 0.0s
```
54
```

create a Series containing information about the population of Krakow in the years 1985 to 2023. Calculate and display:

the contents of the Series along with the descriptive statistics population in Krakow in 2005 the average pupulation in Krakow between 1985 and 2023 descriptive statistics

```python
import pandas as pd
population_of_krakow = {'1985':740120, '1995':744987, '2005':756629, '2015':76250
population = pd.Series(data=population_of_krakow)
population
```

✓ 0.0s Python

```
1985    740120
1995    744987
2005    756629
2015    762508
2019    774839
2023    804237
dtype: int64
```

```python
population.describe()
```

✓ 0.0s Python

```
count         6.000000
mean     763886.666667
std       23344.075468
min      740120.000000
25%      747897.500000
50%      759568.500000
75%      771756.250000
max      804237.000000
dtype: float64
```

```python
population['2005']
```

✓ 0.0s Python

```
756629
```

```python
population['1985':'2023'].mean()
```

✓ 0.0s

```
763886.6666666666
```

You can create a DataFrame by adding columns in the data structure. Each column is created from a collection of data represented as a list.

```python
import pandas as pd
```

✓ 0.0s

```python
#creating an empty DataFrame
company = pd.DataFrame()

# creating data collections as lists
company_months = ['January','February','March','April','May','June']
company_income = [23500,19700,31150,27365,67394,47392]

# adding columns to DataFrame
company['Month'] = company_months
company['Income'] = company_income

# displaying DataFrame contents
company
```

✓ 0.0s

| | Month | Income |
|---|---|---|
| 0 | January | 23500 |
| 1 | February | 19700 |
| 2 | March | 31150 |
| 3 | April | 27365 |
| 4 | May | 67394 |
| 5 | June | 47392 |

Display descriptive statistics for income earned.

```
company['Income'].describe()
```

✓ 0.0s

```
count        6.000000
mean     36083.500000
std      18085.047921
min      19700.000000
25%      24466.250000
50%      29257.500000
75%      43331.500000
max      67394.000000
Name: Income, dtype: float64
```

Display company income for the months of the second quarter.

```
company.iloc[3:6]
```

✓ 0.0s

| | Month | Income |
|---|---|---|
| 3 | April | 27365 |
| 4 | May | 67394 |
| 5 | June | 47392 |

Display descriptive statistics for the income earned in the months of the second quarter.

```
company.iloc[3:6].describe()
```

✓ 0.0s

| | Income |
|---|---|
| count | 3.000000 |
| mean | 47383.666667 |
| std | 20014.501301 |
| min | 27365.000000 |
| 25% | 37378.500000 |

Instead of adding each column separately, you can create a DateFrame based on a two-dimensional (2D) list. Note that you will then need to add names to the columns you create.

```
# creating data collection as 2D list
company_data = [
    ['January',23500],
    ['February',19700],
    ['March',31150]
    ]

# creating DataFrame with column names
company = pd.DataFrame(data=company_data, columns=['Month','Income'])

# displaying DataFrame contents
company
```

✓ 0.0s

| | Month | Income |
|---|---|---|
| 0 | January | 23500 |
| 1 | February | 19700 |
| 2 | March | 31150 |

The table below lists the university's students.

| StudentID | Name | Surname | Age | Program |
|---|---|---|---|---|
| 902311 | Peter | Red | 21 | Accounting |
| 915027 | Sofia | White | 19 | Computer Science |
| 900004 | Jack | Grey | 24 | Accounting |
| 994031 | Mark | Brown | 22 | Engineering |

Create a DataFrame using a 2D list. Then, display the contents of the DataFrame.

```
students_data = [
    ['902311','Peter','Red',21,'Accounting'],
    ['915027','Sofia','White',19,'Computer Science'],
    ['900004','Jack','Grey',24,'Accounting'],
    ['994031','Mark','Brown',22,'Engineering']
]

students = pd.DataFrame(data=students_data, columns=['Student ID','Name','Surname','Age','Program'])
students
```

✓ 0.0s

| | Student ID | Name | Surname | Age | Program |
|---|---|---|---|---|---|
| 0 | 902311 | Peter | Red | 21 | Accounting |
| 1 | 915027 | Sofia | White | 19 | Computer Science |
| 2 | 900004 | Jack | Grey | 24 | Accounting |
| 3 | 994031 | Mark | Brown | 22 | Engineering |

Calculate and display the average age of students.

```
students['Age'].mean()
```

✓ 0.0s

```
21.5
```

As you know, a dictionary contains data consisting of key and value pairs of information, separated by a colon. Each pair of information represents one column in the DataFrame. The key is the name of the column and the value is the data collection (list). Below is an example of creating a DataFrame based on a dictionary.

```
# creating data collection as a dictionary
company_data = {
    'Month':['January','February','March'],
    'Income':[23500,19700,31150],
    'Tax':[1200, 2350, 995]
    }

#creating DataFrame
company = pd.DataFrame(data=company_data)

#displaying DataFrame contents
company
```

✓ 0.0s

Python

| | Month | Income | Tax |
|---|---|---|---|
| 0 | January | 23500 | 1200 |
| 1 | February | 19700 | 2350 |
| 2 | March | 31150 | 995 |

Display descriptive statistics for the company income and tax.

```
company[['Income','Tax']].describe()
```

✓ 0.0s

| | Income | Tax |
|---|---|---|
| count | 3.000000 | 3.000000 |
| mean | 24783.333333 | 1515.000000 |
| std | 5831.880772 | 730.359501 |
| min | 19700.000000 | 995.000000 |
| 25% | 21600.000000 | 1097.500000 |
| 50% | 23500.000000 | 1200.000000 |
| 75% | 27325.000000 | 1775.000000 |
| max | 31150.000000 | 2350.000000 |

Creating a DataFrame based on the data contained in a CSV file is incredibly simple. All you need to do is use the read_csv() function.

```
sales = pd.read_csv('product_sales.csv')
sales
```

✓ 0.0s

| | SaleRep | Region | Orders | TotalSales |
|---|---|---|---|---|
| 0 | Felice Lunck | West | 218 | 44489 |
| 1 | Doralynn Pesak | West | 233 | 61035 |
| 2 | Madelle Martland | East | 264 | 62603 |
| 3 | Yasmin Myhan | South | 110 | 59377 |
| 4 | Marmaduke Webbe | East | 188 | 78771 |
| 5 | Christiano Vero | East | 265 | 68506 |
| 6 | Cecelia Jealous | West | 93 | 53634 |
| 7 | Isaak Housiaux | East | 189 | 62455 |
| 8 | Derril Howland | East | 385 | 73460 |
| 9 | Judon Allom | West | 230 | 51067 |

## Tasks

For sales data, calculate and display the average number of orders.

```
sales['Orders'].mean()
```

✓ 0.0s

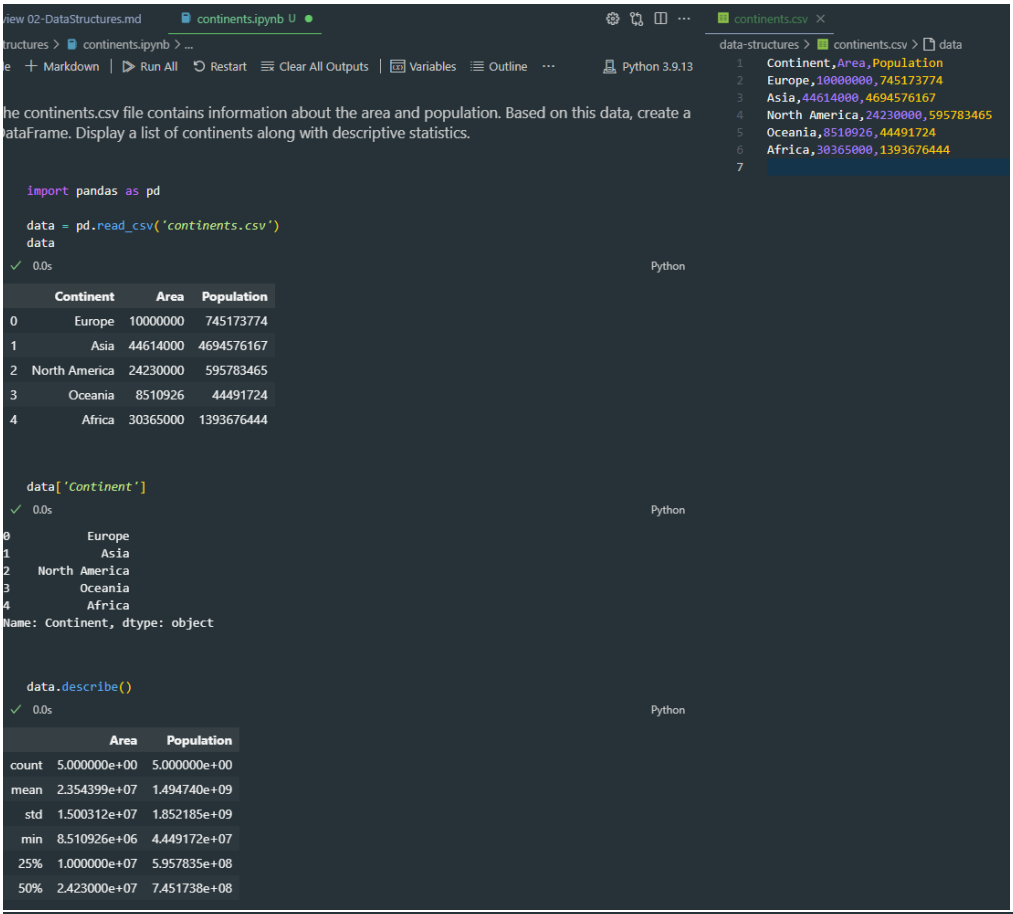
```
217.5
```

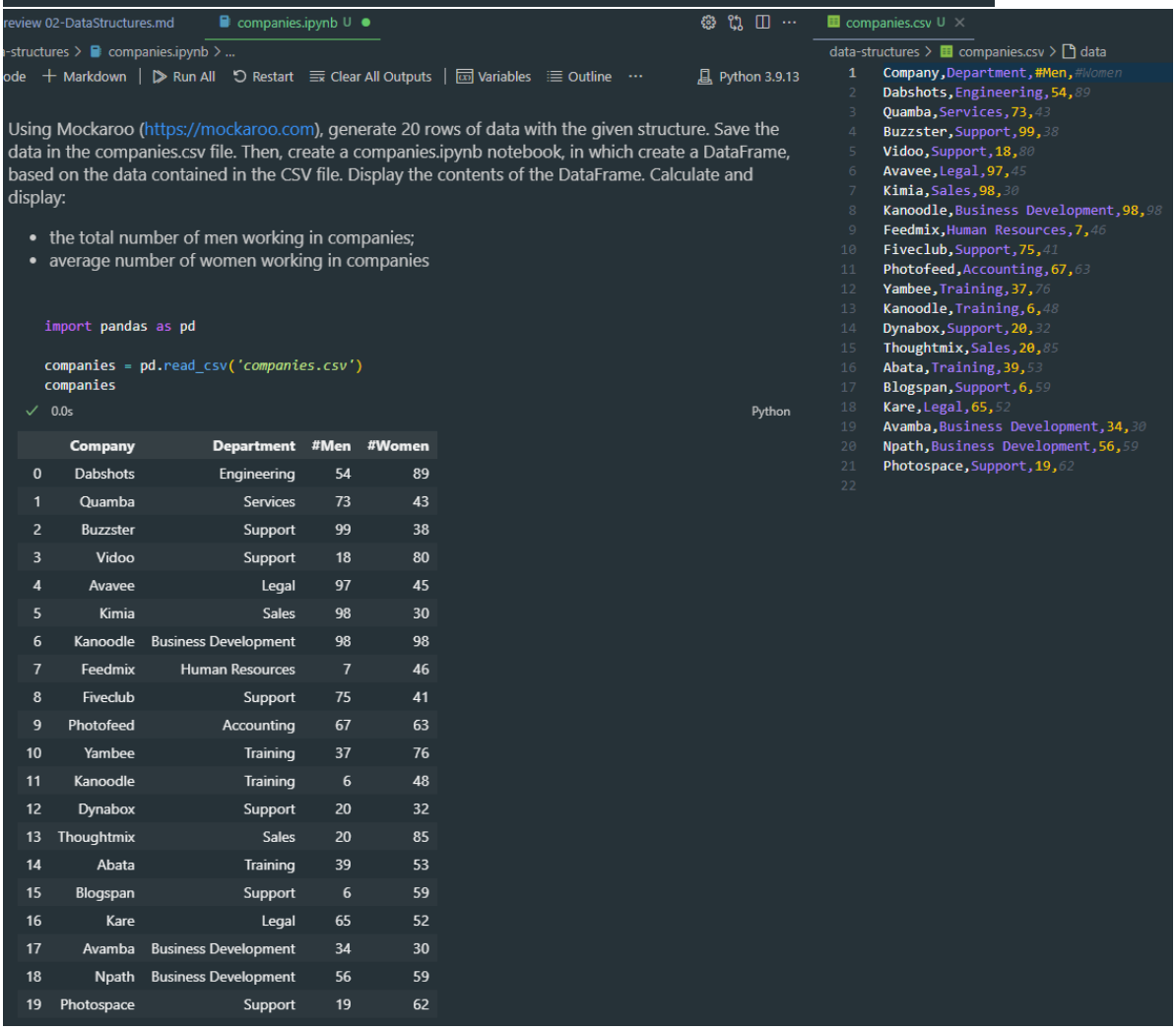
For sales data, calculate and display the total sales value.

```
sales['TotalSales'].sum()
```

✓ 0.0s

```
615397
```


iew 02-DataStructures.md
continents.ipynb U
tructures > continents.ipynb > ...
+ Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | Python 3.9.13
he continents.csv file contains information about the area and population. Based on this data, create a ataFrame. Display a list of continents along with descriptive statistics.
import pandas as pd
data = pd.read_csv('continents.csv')
data
0.0s
Python
Continent Area Population
0 Europe 10000000 745173774
1 Asia 44614000 4694576167
2 North America 24230000 595783465
3 Oceania 8510926 44491724
4 Africa 30365000 1393676444
data['Continent']
0.0s
Python
0 Europe
1 Asia
2 North America
3 Oceania
4 Africa
Name: Continent, dtype: object
data.describe()
0.0s
Python
Area Population
count 5.000000e+00 5.000000e+00
mean 2.354399e+07 1.494740e+09
std 1.500312e+07 1.852185e+09
min 8.510926e+06 4.449172e+07
25% 1.000000e+07 5.957835e+08
50% 2.423000e+07 7.451738e+08
continents.csv
data-structures > continents.csv > data
Continent,Area,Population
Europe,10000000,745173774
Asia,44614000,4694576167
North America,24230000,595783465
Oceania,8510926,44491724
Africa,30365000,1393676444


review 02-DataStructures.md
companies.ipynb U
-structures > companies.ipynb > ...
ode + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | Python 3.9.13
Using Mockaroo (https://mockaroo.com), generate 20 rows of data with the given structure. Save the data in the companies.csv file. Then, create a companies.ipynb notebook, in which create a DataFrame, based on the data contained in the CSV file. Display the contents of the DataFrame. Calculate and display:
the total number of men working in companies;
average number of women working in companies
import pandas as pd
companies = pd.read_csv('companies.csv')
companies
0.0s
Python
Company Department #Men #Women
0 Dabshots Engineering 54 89
1 Quamba Services 73 43
2 Buzzster Support 99 38
3 Vidoo Support 18 80
4 Avavee Legal 97 45
5 Kimia Sales 98 30
6 Kanoodle Business Development 98 98
7 Feedmix Human Resources 7 46
8 Fiveclub Support 75 41
9 Photofeed Accounting 67 63
10 Yambee Training 37 76
11 Kanoodle Training 6 48
12 Dynabox Support 20 32
13 Thoughtmix Sales 20 85
14 Abata Training 39 53
15 Blogspan Support 6 59
16 Kare Legal 65 52
17 Avamba Business Development 34 30
18 Npath Business Development 56 59
19 Photospace Support 19 62
companies.csv U
data-structures > companies.csv > data
Company,Department,#Men,#Women
Dabshots,Engineering,54,89
Quamba,Services,73,43
Buzzster,Support,99,38
Vidoo,Support,18,80
Avavee,Legal,97,45
Kimia,Sales,98,30
Kanoodle,Business Development,98,98
Feedmix,Human Resources,7,46
Fiveclub,Support,75,41
Photofeed,Accounting,67,63
Yambee,Training,37,76
Kanoodle,Training,6,48
Dynabox,Support,20,32
Thoughtmix,Sales,20,85
Abata,Training,39,53
Blogspan,Support,6,59
Kare,Legal,65,52
Avamba,Business Development,34,30
Npath,Business Development,56,59
Photospace,Support,19,62

```
companies['#Men'].sum()
```
✓ 0.0s Python

```
988
```

```
companies['#Women'].mean()
```
✓ 0.0s Python

```
56.45
```

You can indicate rows and/or columns that will then be further processed. A DataFrame contains the loc[] and iloc[] methods that allow access to specified rows and columns.

a. loc[row,column] – selection based on row/column names
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html

b. iloc[row,column] – selection based on row/column numbers
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.iloc.html

To display the name of actor Leonardo DiCaprio, enter the row and column number. Remember that numbering starts from 0.

```
oscars.iloc[15,2]
```
✓ 0.0s

```
'Leonardo DiCaprio'
```

Instead of row and column numbers, you can enter row and column names.

```
oscars.loc[15,'Name']
```
✓ 0.0s

```
'Leonardo DiCaprio'
```

Now try displaying the movie title 'Three Billboards Outside Ebbing Missouri'. Use two methods, specifying:

- row and column numbers
- row and column names

```
oscars.iloc[10,3]
```
✓ 0.0s

```
'Three Billboards Outside Ebbing Missouri'
```

```
oscars.loc[10,'Movie']
```
✓ 0.0s

```
'Three Billboards Outside Ebbing Missouri'
```

To select any rows and columns, you can list their names/numbers or specify a range of names/numbers separated by a colon. For example:

```
oscars.iloc[3:7]
```
✓ 0.0s

| | Year | Category | Name | Movie |
|---|---|---|---|---|
| 3 | 2022 | Best Actor | Will Smith | King Richard |
| 4 | 2021 | Best Actress | Frances McDormand | Nomadland |
| 5 | 2021 | Best Actor | Anthony Hopkins | The Father |
| 6 | 2020 | Best Actress | Renée Zellweger | Judy |

```
oscars.iloc[5:10,2:4]
```
✓ 0.0s

| | Name | Movie |
|---|---|---|
| 5 | Anthony Hopkins | The Father |
| 6 | Renée Zellweger | Judy |
| 7 | Joaquin Phoenix | Joker |
| 8 | Olivia Colman | The Favourite |
| 9 | Rami Malek | Bohemian Rhapsody |

```
oscars.iloc[[10,3,17],[2,3]]
```
✓ 0.0s

| | Name | Movie |
|---|---|---|
| 10 | Frances McDormand | Three Billboards Outside Ebbing Missouri |
| 3 | Will Smith | King Richard |
| 17 | Eddie Redmayne | The Theory of Everything |

```
oscars.loc[:,['Movie','Name','Year']]
```
✓ 0.0s

| | Movie | Name | Year |
|---|---|---|---|
| 0 | Everything Everywhere All at Once | Michelle Yeoh | 2023 |
| 1 | The Whale | Brendan Fraser | 2023 |
| 2 | The Eyes of Tammy Faye | Jessica Chastain | 2022 |
| 3 | King Richard | Will Smith | 2022 |
| 4 | Nomadland | Frances McDormand | 2021 |
| 5 | The Father | Anthony Hopkins | 2021 |
| 6 | Judy | Renée Zellweger | 2020 |
| 7 | Joker | Joaquin Phoenix | 2020 |
| 8 | The Favourite | Olivia Colman | 2019 |
| 9 | Bohemian Rhapsody | Rami Malek | 2019 |
| 10 | Three Billboards Outside Ebbing Missouri | Frances McDormand | 2018 |
| 11 | Darkest Hour | Gary Oldman | 2018 |
| 12 | La La Land | Emma Stone | 2017 |
| 13 | Manchester by the Sea | Casey Affleck | 2017 |
| 14 | Room | Brie Larson | 2016 |
| 15 | The Revenant | Leonardo DiCaprio | 2016 |
| 16 | Still Alice | Julianne Moore | 2015 |
| 17 | The Theory of Everything | Eddie Redmayne | 2015 |
| 18 | Blue Jasmine | Cate Blanchett | 2014 |
| 19 | Dallas Buyers Club | Matthew McConaughey | 2014 |

Display the first two rows of data.

```
oscars.iloc[0:2]
```

Display the first three columns of data.

```
oscars.iloc[:,0:3]
```

Display data without a Name column.

```
oscars.iloc[:,[0,1,3]]
```

In addition, you can display the first or last few rows of data. To do this, use the head() or tail() methods. You can even specify how many rows to display.

```
oscars.head()
```
✓ 0.0s

| | Year | Category | Name | Movie |
|---|---|---|---|---|
| 0 | 2023 | Best Actress | Michelle Yeoh | Everything Everywhere All at Once |
| 1 | 2023 | Best Actor | Brendan Fraser | The Whale |
| 2 | 2022 | Best Actress | Jessica Chastain | The Eyes of Tammy Faye |
| 3 | 2022 | Best Actor | Will Smith | King Richard |
| 4 | 2021 | Best Actress | Frances McDormand | Nomadland |

```
oscars.tail(3)
```
✓ 0.0s

| | Year | Category | Name | Movie |
|---|---|---|---|---|
| 17 | 2015 | Best Actor | Eddie Redmayne | The Theory of Everything |
| 18 | 2014 | Best Actress | Cate Blanchett | Blue Jasmine |
| 19 | 2014 | Best Actor | Matthew McConaughey | Dallas Buyers Club |

Display the first 10 rows.

```
oscars.head(10)
```

| | Year | Category | Name | Movie |
|---|---|---|---|---|
| 0 | 2023 | Best Actress | Michelle Yeoh | Everything Everywhere All at Once |
| 1 | 2023 | Best Actor | Brendan Fraser | The Whale |
| 2 | 2022 | Best Actress | Jessica Chastain | The Eyes of Tammy Faye |
| 3 | 2022 | Best Actor | Will Smith | King Richard |
| 4 | 2021 | Best Actress | Frances McDormand | Nomadland |
| 5 | 2021 | Best Actor | Anthony Hopkins | The Father |
| 6 | 2020 | Best Actress | Renée Zellweger | Judy |
| 7 | 2020 | Best Actor | Joaquin Phoenix | Joker |
| 8 | 2019 | Best Actress | Olivia Colman | The Favourite |
| 9 | 2019 | Best Actor | Rami Malek | Bohemian Rhapsody |

Display the last 5 rows of data with the columns: Movie, Year.

```
oscars.iloc[:,[0,3]].tail(5)
```

| | Year | Movie |
|---|---|---|
| 15 | 2016 | The Revenant |
| 16 | 2015 | Still Alice |
| 17 | 2015 | The Theory of Everything |
| 18 | 2014 | Blue Jasmine |
| 19 | 2014 | Dallas Buyers Club |

Display the first and last rows of data containing only the first and last columns.

```
oscars.iloc[[0,19],[0,3]]
# lub oscars.iloc[[0,-1],[0,-1]]
```

| | Year | Movie |
|---|---|---|
| 0 | 2023 | Everything Everywhere All at Once |
| 19 | 2014 | Dallas Buyers Club |

- display the list of universities in Warsaw along with the city and the number of students
- display only the list of universities in Warsaw
- display only the last 3 universities in Krakow

```
data[data['City'] == 'Warsaw']
```

✓ 0.0s

| | City | Name | Students |
|---|---|---|---|
| 21 | Warsaw | University of Warsaw | 56000 |
| 22 | Warsaw | Warsaw University of Technology | 34000 |
| 23 | Warsaw | School of Economics in Warsaw | 22000 |
| 24 | Warsaw | SGH Warsaw School of Economics | 13000 |
| 25 | Warsaw | Medical University of Warsaw | 12000 |
| 26 | Warsaw | National Defence University of Warsaw | 5000 |
| 27 | Warsaw | Cardinal Stefan Wyszyński University in Warsaw | 12000 |
| 28 | Warsaw | University of Life Sciences in Warsaw | 14000 |
| 29 | Warsaw | Warsaw University of Life Sciences (SGGW) | 14000 |
| 30 | Warsaw | Fryderyk Chopin University of Music | 500 |
| 31 | Warsaw | Academy of Fine Arts in Warsaw | 2000 |

```
data.loc[data['City'] == 'Warsaw','Name']
```

✓ 0.0s

```
21                          University of Warsaw
22                 Warsaw University of Technology
23                   School of Economics in Warsaw
24                 SGH Warsaw School of Economics
25                   Medical University of Warsaw
26            National Defence University of Warsaw
27    Cardinal Stefan Wyszyński University in Warsaw
28            University of Life Sciences in Warsaw
29       Warsaw University of Life Sciences (SGGW)
30             Fryderyk Chopin University of Music
31                   Academy of Fine Arts in Warsaw
Name: Name, dtype: object
```

```
data[data['City'] == 'Krakow'].tail(3)
```

✓ 0.0s

| | City | Name | Students |
|---|---|---|---|
| 18 | Krakow | Wyższa Szkoła Informatyki i Zarządzania (WSIiZ) | 800 |
| 19 | Krakow | Władysław Szafer Academy of Agriculture | 700 |
| 20 | Krakow | Krakow Teacher Training College | 600 |

To save a DataFrame to a file, use one of the available methods. Try to save data from all rows but selected columns to the file.

```
import pandas as pd
oscars = pd.read_csv('oscars.csv')
```

✓ 0.3s

```
movies = oscars.loc[:,['Movie','Year']]
movies.to_csv('movies.csv', index=False)
```

✓ 0.0s

# Methods of Writing Data to File

To save a DataFrame to a file, use one of the available methods. Try to save data from all rows but selected columns to the file.

```
import pandas as pd
oscars = pd.read_csv('oscars.csv')
```

✓ 0.0s

```
movies = oscars.loc[:,['Movie','Year']]
movies.to_csv('movies.csv', index=False)
```

✓ 0.0s

## Tasks

Display the contents of the created CSV file in the editor and:

- Verify that the file contains two columns of data.
- Note that if you use index=False when writing data to the file, the file will not contain the names/numbers of the data rows.

Now try to save the first 6 rows to a JSON file. Then, display the contents of the created file in the editor.

```
movies = pd.read_csv('movies.csv')

six_films = movies.head(6)
six_films.to_json('six_films.json', index=False)
```

✓ 0.0s

Try to save the last 8 rows with the columns Movie and Year to an HTML file. Then, display the contents of the created file in the editor. Finally, display the created HTML document in any web browser.

```
eight_films = oscars.loc[:,['Movie','Year']].tail(8)
eight_films.to_html('eight_films.html', index=False)
```

- display the data content
- export data to an html document
- display the html document in a web browser
- export the first and the last column to a csv file
- display the csv document in a text editor
- export data related to the west region to an html file
- display the content of the html file in a web browser

```
import pandas as pd

sales = pd.read_csv('product_sales.csv')
sales
```

✓ 0.0s

| | SaleRep | Region | Orders | TotalSales |
|---|---|---|---|---|
| 0 | Felice Lunck | West | 218 | 44489 |
| 1 | Doralynn Pesak | West | 233 | 61035 |
| 2 | Madelle Martland | East | 264 | 62603 |
| 3 | Yasmin Myhan | South | 110 | 59377 |
| 4 | Marmaduke Webbe | East | 188 | 78771 |
| 5 | Christiano Vero | East | 265 | 68506 |
| 6 | Cecelia Jealous | West | 93 | 53634 |
| 7 | Isaak Housiaux | East | 189 | 62455 |
| 8 | Derril Howland | East | 385 | 73460 |
| 9 | Judon Allom | West | 230 | 51067 |

```
sales.to_html('sales.html', index=False)
```

✓ 0.0s

```
sales.iloc[:,[0,-1]].to_csv('first_last_sales.csv', index=False)
```

✓ 0.0s

```
sales.loc[sales['Region'] == 'West',:].to_html('sales_west.html', index=False)
```

✓ 0.0s

You can extend an existing DataFrame with columns whose values depend on the values of already existing columns.

```
import pandas as pd
```

✓ 0.3s

Imagine that the data contains the results of two tests of several students.

```
test_results = {'Test1':[84,37,55], 'Test2':[91,62,74]}
tests = pd.DataFrame(test_results)
tests.index = ['John Brown','Sarah Green','Peter Black']
tests
```

✓ 0.0s

| | Test1 | Test2 |
|---|---|---|
| John Brown | 84 | 91 |
| Sarah Green | 37 | 62 |
| Peter Black | 55 | 74 |

You want to add a column that contains the sum of the points obtained.

```
tests['Total'] = tests['Test1'] + tests['Test2']
tests
```

✓ 0.0s

| | Test1 | Test2 | Total |
|---|---|---|---|
| John Brown | 84 | 91 | 175 |
| Sarah Green | 37 | 62 | 99 |
| Peter Black | 55 | 74 | 129 |

To the test results, add a column containing the arithmetic mean of Test1 and Test2.

```
tests['Average'] = tests['Total'] / 2
tests
```

✓ 0.0s

| | Test1 | Test2 | Total | Average |
|---|---|---|---|---|
| John Brown | 84 | 91 | 175 | 87.5 |
| Sarah Green | 37 | 62 | 99 | 49.5 |
| Peter Black | 55 | 74 | 129 | 64.5 |

1. Create a DataFrame containing the speed camera data.
2. Display the contents of the DataFrame
3. Add a 'Limit' column containing the permitted vehicle speed, i.e. 50 km/h
4. Display the contents of the DataFrame
5. Add a column in which calculate how many km/h each vehicle exceeded the speed limit
6. Display the contents of the DataFrame

```
data = pd.DataFrame({'vehicle number':['BW3941','GM2309','WX1515','BB0099'], 'KMH':[58,76,47,50]})
data
```

✓ 0.0s

| | vehicle number | KMH |
|---|---|---|
| 0 | BW3941 | 58 |
| 1 | GM2309 | 76 |
| 2 | WX1515 | 47 |
| 3 | BB0099 | 50 |

```
data['Limit'] = [50,60,20,30]
data
```

✓ 0.0s

| | vehicle number | KMH | Limit |
|---|---|---|---|
| 0 | BW3941 | 58 | 50 |
| 1 | GM2309 | 76 | 60 |
| 2 | WX1515 | 47 | 20 |
| 3 | BB0099 | 50 | 30 |

```
data['Amount exceeded'] = data['KMH'] - data['Limit']
data
```

✓ 0.0s

| | vehicle number | KMH | Limit | Amount exceeded |
|---|---|---|---|---|
| 0 | BW3941 | 58 | 50 | 8 |
| 1 | GM2309 | 76 | 60 | 16 |
| 2 | WX1515 | 47 | 20 | 27 |

The continents.csv file contains data on the area and number of inhabitants of some continents. Create a notebook in which create a DataFrame with the data contained in the CSV file. Display the contents of the DataFrame. Then, add a column in which calculate the population density, i.e. the number of inhabitants per 1 km2. View the modified DataFrame.

```
import pandas as pd

data = pd.read_csv('continents.csv')
data
```

✓ 0.0s Python

| | Continent | Area | Population |
|---|---|---|---|
| 0 | Europe | 10000000 | 745173774 |
| 1 | Asia | 44614000 | 4694576167 |
| 2 | North America | 24230000 | 595783465 |
| 3 | Oceania | 8510926 | 44491724 |
| 4 | Africa | 30365000 | 1393676444 |

```
data['Density'] = data['Population'] / data['Area']
data
```

✓ 0.0s Python

| | Continent | Area | Population | Density |
|---|---|---|---|---|
| 0 | Europe | 10000000 | 745173774 | 74.517377 |
| 1 | Asia | 44614000 | 4694576167 | 105.226525 |
| 2 | North America | 24230000 | 595783465 | 24.588670 |
| 3 | Oceania | 8510926 | 44491724 | 5.227601 |
| 4 | Africa | 30365000 | 1393676444 | 45.897462 |

In order to sort a DataFrame, you can use the function sort_values(). Data can be sorted in Ascending or Descending order.

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html

The data in the cars.csv file contains basic information about several luxury cars.

```
import pandas as pd
cars = pd.read_csv('cars.csv')
cars
```

✓ 0.3s

| | Name | PriceUSD | MotorCapacityL |
|---|---|---|---|
| 0 | Rolls-Royce Phantom | 535500 | 6.75 |
| 1 | Bentley Continental GT | 185000 | 4.0 |
| 2 | Mercedes-Maybach S-Class | 173100 | 4.0 |
| 3 | Ferrari 812 Superfast | 335250 | 6.5 |
| 4 | Lamborghini Aventador | 417800 | 6.5 |
| 5 | Aston Martin DB11 | 208100 | 4.0 |
| 6 | Porsche Panamera | 86300 | 3.0 |
| 7 | Maserati Quattroporte | 106000 | 3.0 |
| 8 | Lexus LS | 80500 | 3.5 |
| 9 | Tesla Model S | 82990 | Dual Electric Motors |

To sort the list of cars alphabetically, use the sort function and enter the name of the column by which the data should be sorted.

```
cars.sort_values(by='Name')
```

You can also arrange the data in reverse order, e.g. starting with the most expensive car.

```
cars.sort_values(by='PriceUSD', ascending=False)
```

1. Based on the data in the table, create the content of the DataFrame.

```
results = pd.DataFrame({'Name':['Olivier','Naomi','Olivia','Nolan','Dylan'],'Total':[90,42,71,100,39],'Result':['passed','failed','passed','passed','failed']})
results
```

✓ 0.0s

| | Name | Total | Result |
|---|---|---|---|
| 0 | Olivier | 90 | passed |
| 1 | Naomi | 42 | failed |
| 2 | Olivia | 71 | passed |
| 3 | Nolan | 100 | passed |
| 4 | Dylan | 39 | failed |

Sort the list of people alphabetically.

```
results.sort_values(by='Name')
```

✓ 0.0s

| | Name | Total | Result |
|---|---|---|---|
| 4 | Dylan | 39 | failed |
| 1 | Naomi | 42 | failed |
| 3 | Nolan | 100 | passed |
| 2 | Olivia | 71 | passed |
| 0 | Olivier | 90 | passed |

1. Sort the list of people according to their points, starting with the highest value.

```
results.sort_values(by='Total', ascending=False)
```

✓ 0.0s

| | Name | Total | Result |
|---|---|---|---|
| 3 | Nolan | 100 | passed |
| 0 | Olivier | 90 | passed |
| 2 | Olivia | 71 | passed |

1. Sort the list of people by their score, starting with the people who passed the exam.

```
results.sort_values(by=['Result','Total'], ascending=False)
```

✓ 0.0s

| | Name | Total | Result |
|---|---|---|---|
| 3 | Nolan | 100 | passed |
| 0 | Olivier | 90 | passed |
| 2 | Olivia | 71 | passed |
| 1 | Naomi | 42 | failed |
| 4 | Dylan | 39 | failed |

1. Sort the list of people by their score, starting with those who passed the exam and then by those names alphabetically. Hint. Use the link above to see how to sort data by more than one column.

```
results.sort_values(by=['Result','Total','Name'], ascending=[False, False, True])
```

✓ 0.0s

| | Name | Total | Result |
|---|---|---|---|
| 3 | Nolan | 100 | passed |
| 0 | Olivier | 90 | passed |
| 2 | Olivia | 71 | passed |
| 1 | Naomi | 42 | failed |
| 4 | Dylan | 39 | failed |

Add the 'University' column to your exam DataFrame. Assign the first two people the name 'UEK' and the remaining three people the name 'AGH'. Display updated DataFrame.

```
results['University'] = ['UEK','UEK','AGH','AGH','AGH']
results
```

✓ 0.0s

| | Name | Total | Result | University |
|---|---|---|---|---|
| 0 | Olivier | 90 | passed | UEK |
| 1 | Naomi | 42 | failed | UEK |
| 2 | Olivia | 71 | passed | AGH |

With the modified DataFrame:

1. Sort the list by the univeristy, and then by the name. Display the sorted data.
2. Display the people names alphabetically (only one column of data).
3. Display two columns: Total and Name, in the given order. Sort the data by the number of points obtained, descending.

```
results.sort_values(by=['University','Name'])
```

✓ 0.0s

| | Name | Total | Result | University |
|---|---|---|---|---|
| 4 | Dylan | 39 | failed | AGH |
| 3 | Nolan | 100 | passed | AGH |
| 2 | Olivia | 71 | passed | AGH |
| 1 | Naomi | 42 | failed | UEK |
| 0 | Olivier | 90 | passed | UEK |

```
results.sort_values(by='Name').loc[:,'Name']
```

✓ 0.0s

```
4      Dylan
1      Naomi
3      Nolan
2     Olivia
0    Olivier
Name: Name, dtype: object
```

```
results.sort_values(by='Total', ascending=False).loc[:,['Total','Name']]
```

✓ 0.0s

| | Total | Name |
|---|---|---|
| 3 | 100 | Nolan |
| 0 | 90 | Olivier |
| 2 | 71 | Olivia |
| 1 | 42 | Naomi |
| 4 | 39 | Dylan |

Create a new notebook. Based on the file european-countries.csv, create a DataFrame. Then, do the following tasks:

- display currency names alphabetically (only one column)

```
import pandas as pd

data = pd.read_csv('european-countries.csv')
data.sort_values(by='Currency').loc[:,'Currency']
```

✓ 0.0s Pyth

- display data organised by the population, descending

```
data.sort_values(by='Population', ascending=False)
```

- display data organised by the region and then by the country

```
data.sort_values(by=['Region','Name'])
```

- display data organised by the population density (first, add an extra column)

```
data['Density'] = data['Population'] / data['Area']
data.sort_values(by='Density')
```

- display the name of the capital and currency (only two columns), arrange the data by currency and then by the name of the capital city

```
data.sort_values(by=['Currency','Capital']).loc[:,['Capital','Currency']]
```

Create a query that selects only universities from Krakow.

```
universities.query("City=='Krakow'")
```

You can create a query with more complex criteria. For example, to get a list of universities in Krakow with at least 10,000 students:

```
universities.query("City=='Krakow' and Students >= 10000")
```

Display a list of universities in Warsaw with at least 8,000 students. Sort the list by number of students in descending order.

```
universities.query("City=='Warsaw' and Students >= 8000").sort_values(by='Students', ascending=False)
```

```
# Load your data into a DataFrame
data = pd.read_csv('your_file.csv', delimiter=';')

# Set the option to display all rows
pd.set_option('display.max_rows', None)
```

```
import pandas as pd
visits_data = {
    'Day':['Monday','Monday','Monday','Friday','Friday','Sunday'],
    'Browser':['Edge','Chrome','Safari','Edge','Safari','Chrome'],
    'Visits':[15,17,11,10,23,34],
    'Purchase':[3,1,2,2,7,12]
    }
visits = pd.DataFrame(visits_data)
visits
```

✓ 0.3s

| | Day | Browser | Visits | Purchase |
|---|---|---|---|---|
| 0 | Monday | Edge | 15 | 3 |
| 1 | Monday | Chrome | 17 | 1 |
| 2 | Monday | Safari | 11 | 2 |
| 3 | Friday | Edge | 10 | 2 |
| 4 | Friday | Safari | 23 | 7 |
| 5 | Sunday | Chrome | 34 | 12 |

To calculate and display the total number of visits on individual days of the week, specify the result columns, the columns by which the data are grouped and the aggregation function:

```
visits.loc[:,['Day','Visits']].groupby(['Day']).sum()
```

✓ 0.0s

| Day | Visits |
|---|---|
| Friday | 33 |
| Monday | 43 |
| Sunday | 34 |

Calculate and display the number of web browsers used on specific days of the week. Sort the results in descending order.

```
visits.loc[:,['Day','Browser']].groupby(['Day']).count().sort_values(by='Browser', ascending=False)
```

✓ 0.0s

| Day | Browser |
|---|---|
| Monday | 3 |
| Friday | 2 |
| Sunday | 1 |

1. Minimum age of women and men by country. Sort the results in descending order.
2. Median salary of women and men by country.
3. Median salary of women and men by country. Save the results to a csv file.

```
data.loc[:,['sex','country','age']].groupby(['country']).min('age').sort_values(by="age",ascending=False)
```

✓ 0.0s

| country | age |
|---|---|
| United Kingdom | 37 |
| Czech Republic | 20 |
| Poland | 19 |

```
data.loc[:,['sex','country','salary']].groupby(['sex','country']).median('salary')
```

✓ 0.0s

| sex | country | salary |
|---|---|---|
| Female | Czech Republic | 44973.700 |
| | Poland | 107833.315 |
| Male | Czech Republic | 57259.410 |
| | Poland | 72688.560 |
| | United Kingdom | 148308.960 |

```
data.loc[:,['sex','country','salary']].groupby(['sex','country']).median('salary').to_csv('median.csv')
```

Calculate and display the total number of students studying in Warsaw and Krakow. The data is available in the universitites.csv file. Do your calculations in a separate notebook.

```
import pandas as pd

data = pd.read_csv('universities.csv')
data.loc[:,['Students','City']].groupby(['City']).sum()
```

✓ 0.0s

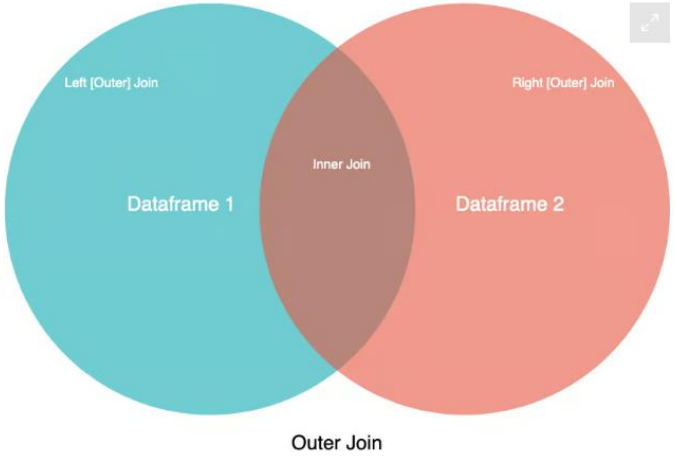
| City | Students |
|---|---|
| Krakow | 136300 |
| Warsaw | 184500 |

- the total number of inhabitants in each of the regions in the 'opolskie' province; sort the values in descending order

```
data.query("Province == 'opolskie'").loc[:,['Population','Region','Province']].groupby(['Region']).sum('Population').sort_values(by="Population", ascending=False)
```

✓ 0.0s

| Region | Population |
|---|---|
| Opole[a] | 119574 |
| nyski | 73571 |
| kędzierzyńsko-kozielski | 62840 |
| brzeski | 51431 |
| krapkowicki | 35279 |
| kluczborski | 34055 |
| strzelecki | 33640 |
| prudnicki | 29874 |
| oleski | 23715 |
| głubczycki | 22087 |
| opolski | 18426 |
| namysłowski | 16126 |




Outer Join

1. `merge()` for combining data on common columns or indices
2. `.join()` for combining data on a key column or an index
3. `concat()` for combining DataFrames across rows or columns

First, you need to create data collections corresponding to the contents of the tables.

```python
orders_data = {
    'OrderNo':['295','295','312','312','314'],
    'Date':['2024-02-09','2024-02-09','2024-02-17','2024-02-17','2024-02-18'],
    'Product':['chair','lamp','desk','lamp','desk'],
    'Quantity':[6,6,2,1,4]}
price_list = {'Product':['desk','chair','lamp'], 'Price':[450.00, 275.00, 79.00]}
```

Python

Now, based on the data collections, create DataFrames.

```python
import pandas as pd
orders = pd.DataFrame(orders_data)
prices = pd.DataFrame(price_list)
```

Python

Finally, you join both DataFrames and display their common content. Note the use of the 'merge' function. You must provide the names of both DataFrames and the name of the column that contains the common data.

```python
orders_with_prices = pd.merge(orders,prices,on='Product')
orders_with_prices
```

Python

You can also complete the final DataFrame by adding a new column containing the amount to be paid for the ordered products.

```python
orders_with_prices['Total'] = orders_with_prices['Quantity'] * orders_with_prices['Price']
orders_with_prices
```

It turns out that the store offers a discount on selected products. Currently, the discount on desks is 20%, while the discount on lamps is 30%. Create another DataFrame containing a list of discounted products. Then, join the discounted data with the previous data collections. Calculate and display the amounts to pay, after taking into account the discount.

```python
discount_data = {'Product':['desk','lamp','chair'], 'Discount':[20,30,0]}
discount = pd.DataFrame(discount_data)
discount
```

✓ 0.0s

Python

| | Product | Discount |
|---|---|---|
| 0 | desk | 20 |
| 1 | lamp | 30 |
| 2 | chair | 0 |

```python
data = pd.merge(orders_with_prices,discount,on='Product')
data['Price after discount'] = data['Total'] - (data['Total'] * data['Discount'] / 100)
data
```

✓ 0.0s

Python

| | OrderNo | Date | Product | Quantity | Price | Total | Discount | Price after discount |
|---|---|---|---|---|---|---|---|---|
| 0 | 295 | 2024-02-09 | chair | 6 | 275.0 | 1650.0 | 0 | 1650.0 |
| 1 | 295 | 2024-02-09 | lamp | 6 | 79.0 | 474.0 | 30 | 331.8 |
| 2 | 312 | 2024-02-17 | desk | 2 | 450.0 | 900.0 | 20 | 720.0 |
| 3 | 312 | 2024-02-17 | lamp | 1 | 79.0 | 79.0 | 30 | 55.3 |
| 4 | 314 | 2024-02-18 | desk | 4 | 450.0 | 1800.0 | 20 | 1440.0 |

```python
df = pd.DataFrame({
    'from': [1, 2, 3],  # 'from' is a Python keyword
    'value': [10, 20, 30]
})

# Using backticks to avoid keyword conflict
result = df.query('`from` > 1')
print(result)
```

The files krk-airlines.csv, krk-flights.csv and krk-passengers.csv contain data about flights from Krakow Airport. In a separate notebook, calculate and display:

- list including flight number and destination (two columns)

```python
import pandas as pd

krk_airlines = pd.read_csv('krk-airlines.csv')
krk_passengers = pd.read_csv('krk-passengers.csv')
krk_flight = pd.read_csv('krk-flights.csv')

flights_and_airlines = pd.merge(krk_flight,krk_airlines,on='airlineid')
data = pd.merge(flights_and_airlines,krk_passengers,on='flight')

data.to_csv('data_planes.csv')

data.loc[:,['flight','to']].groupby(['flight','to']).sum()
```

✓ 0.0s

| flight | to |
|---|---|
| BE321 | Berlin |
| LN222 | London |
| NY777 | New York |
| PA006 | Paris |

- flight list with the full name of the airline and the name of the aircraft

```
data.loc[:,['airline','airplane','flight']].groupby(['flight','airline','airplane']).sum()
```

✓ 0.0s

| flight | airline | airplane |
|---|---|---|
| BE321 | SunHoliday | Airbus A319 |
| LN222 | PanEurope | Boeing 787 |
| NY777 | BlueSky | Airbus A330 |
| PA006 | SunHoliday | Airbus A319 |

- a list of passengers on a flight to London sorted by surname

```
data.query("to == 'London'").loc[:,['name','surname','gender']].groupby(['name','surname','gender']).sum().sort_values(by="surname")
```

✓ 0.0s

| name | surname | gender |
|---|---|---|
| Claudie | Braid | Female |
| Nehemiah | Budcock | Male |
| Ingmar | Cockman | Male |
| Willyt | Matlock | Female |
| Rennie | McComiskey | Female |
| Ferguson | Osban | Male |
| Nolana | Pattie | Female |
| Mareah | Peplay | Female |
| Araldo | Permain | Male |
| Stormy | Quare | Female |
| Edik | Smieton | Male |
| Lindi | Snoad | Female |
| Emmanuel | Tickel | Male |

- number of passengers on the flight to Berlin

```
data.query("to == 'Berlin'").loc[:,['to','name','surname']].groupby(['name','surname']).count().sum()
```

✓ 0.0s

```
to    17
dtype: int64
```

- number of men flying from Krakow

```
data.query("`from` == 'Krakow' and gender == 'Male'").loc[:,['to','name','surname']].groupby(['name','surname']).count().sum()
```

✓ 0.0s

```
to    54
dtype: int64
```

- number of passengers for each flight

```
data.loc[:,['to','name','surname']].groupby(['name','surname']).count().sum()
```

✓ 0.0s

```
to    100
dtype: int64
```

To create a chart, first you have to import the necessary librares:

```
import pandas as pd
import matplotlib.pyplot as plt
```

✓ 1.6s

Now, you need data to plot a chart. The data below shows the number of people who watched the latest movie in the cinema, from Monday to Friday.

```
data = [410,395,515,457,490]
ds = pd.Series(data)
ds
```

✓ 0.0s

```
0    410
1    395
2    515
3    457
4    490
dtype: int64
```

To create a plot, use the plot() function:
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html

```
ds.plot()
```

```
ds.index = ['Monday','Tuesday','Wednesday','Thursday','Friday']
ds
```

✓ 0.0s

```
Monday       410
Tuesday      395
Wednesday    515
Thursday     457
Friday       490
dtype: int64
```

Each chart should have a title. To add it, use the plot() function attribute.

```
ds.plot(title="People Watched a Movie")
```

To add a description of the X-axis of the plot, use the appropriate attribute of the plot() function.

```
ds.plot(title="People Watched a Movie", xlabel='Days of Week')
```

Now try to add the correct description of the Y axis.

```
ds.plot(title="People Watched a Movie", xlabel='Days of Week', ylabel='how many times')
```

By default, a line chart is created. To change the type of chart to a bar, use the kind attribute.

```
ds.plot(kind='bar')
```

If you would like to create a pie chart, apply the value 'pie' to the kind attribute.

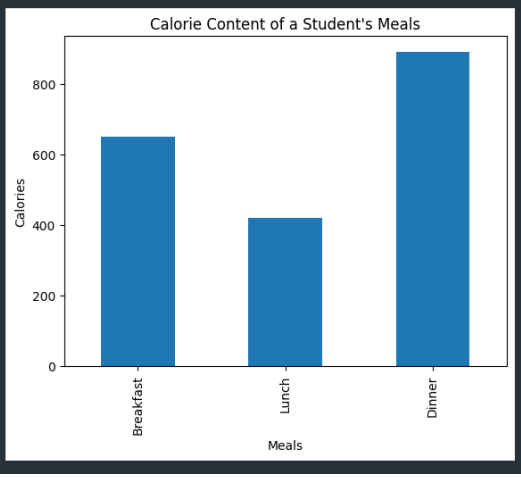
```
ds.plot(kind='pie', autopct='%1.0f%%')
```

The calorie content of a student's meals is as follows: breakfast 650 calories, lunch 420 calories and dinner 890 calories. Present the data in a bar chart. Add a chart title, axis descriptions, and correct x-axis labels.

```
calories = [650,420,890]
chart = pd.Series(calories)
chart.index = ['Breakfast','Lunch','Dinner']
chart.plot(kind='bar', title = "Calorie Content of a Student's Meals", xlabel='Meals', ylabel='Calories')
```

✓ 0.1s

```
<AxesSubplot: title={'center': "Calorie Content of a Student's Meals"}, xlabel='Meals', ylabel='Calories'>
```

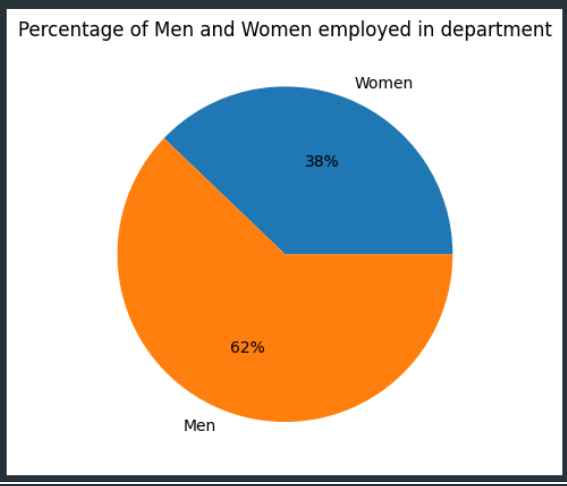


The company's department employs 37 people, 14 of which are women. Create a pie chart showing the percentage of men and women employed in this department.

```
employees = [14,23]
company_data = pd.Series(employees)
company_data.index = ['Women','Men']
company_data.plot(kind='pie', autopct='%1.0f%%', title="Percentage of Men and Women employed in department")
```

✓ 0.0s

```
<AxesSubplot: title={'center': 'Percentage of Men and Women employed in department'}>
```




Player Titles

Novak Djokovic 24

Rafael Nadal 22

Roger Federer 20

Pete Sampras 14

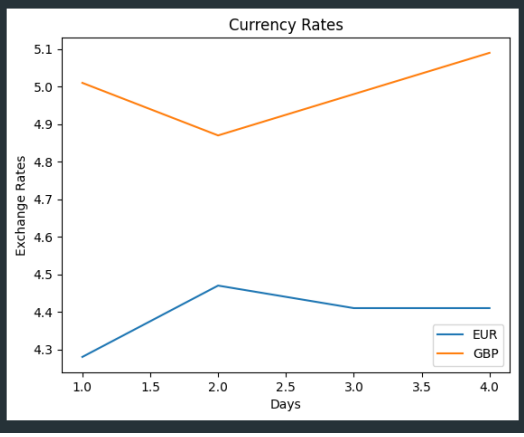
```
import pandas as pd
import matplotlib.pyplot as plt

data = [24,22,20,14]
chart = pd.Series(data)
chart.index = ['Novak Djokovic','Rafael Nadal','Roger Federer','Pete Sampras']
chart.plot(kind='bar', title="Tennis Players who have won the most Grand Slam tournaments", xlabel='Players', ylabel='Tournaments Won')
```

The exchange rates in the last 4 days for Pound Sterling (GBP) were: 5.01, 4.87, 4.98, 5.09. Add the values of the new currency to your dataset. Create a chart with EUR and GBP (excluding USD) exchange rates.

```
currency['GBP'] = [5.01, 4.87, 4.98, 5.09]
df = pd.DataFrame(data=currency)
df.index = [1,2,3,4]
df.loc[:,['EUR','GBP']].plot(title='Currency Rates', xlabel='Days', ylabel='Exchange Rates')
```

✓ 0.1s

```
<AxesSubplot: title={'center': 'Currency Rates'}, xlabel='Days', ylabel='Exchange Rates'>
```

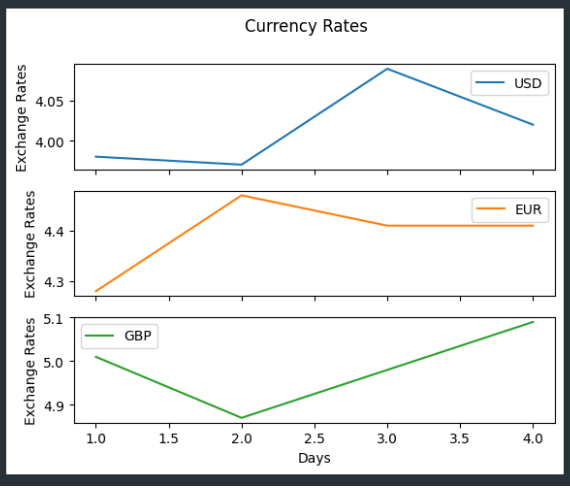


Each series in a DataFrame can be also plotted on a different axis with the using of subplots. The 'layout' parameter specifies the number of rows and columns in which the charts will be placed.

```
df.plot(subplots=True, title='Currency Rates', xlabel='Days', ylabel='Exchange Rates', layout=(3,1))
```

✓ 0.3s

```
array([[<AxesSubplot: xlabel='Days', ylabel='Exchange Rates'>],
       [<AxesSubplot: xlabel='Days', ylabel='Exchange Rates'>],
       [<AxesSubplot: xlabel='Days', ylabel='Exchange Rates'>]],
      dtype=object)
```




```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))  # Create two subplots

# First subplot
ax1.plot(x, y, marker='o')
ax1.grid(True, color='gray', linestyle='--', linewidth=0.5)

# Second subplot
ax2.plot(y, x, marker='o')
ax2.grid(True, color='gray', linestyle='-.', linewidth=0.5)

plt.show()
```

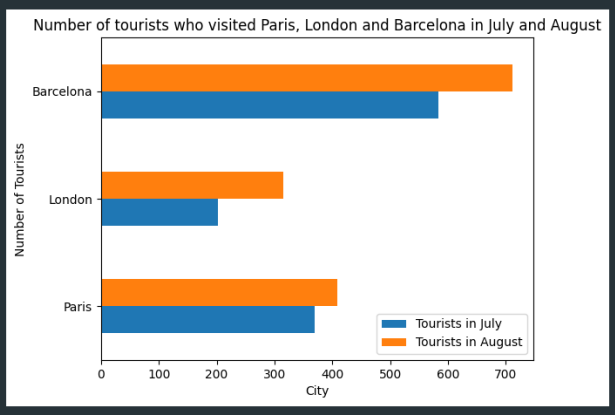
The travel agency organizes holiday trips to London, Paris and Barcelona. The table below shows the number of tourists who visited the given cities in July and August. Plot the data on a horizontal(!) bar chart. Do not forget to add a chart title and axis descriptions.

| Destination | Toursits in July | Tourists in August |
|---|---|---|
| Paris | 370 | 409 |
| London | 202 | 315 |
| Barcelona | 584 | 712 |

```
data = {'Destination':['Paris','London','Barcelona'], 'Tourists in July': [370, 202, 584], 'Tourists in August': [409, 315, 712]}
df = pd.DataFrame(data=data)
df.index = data['Destination']
df.plot(kind='barh', title="Number of tourists who visited Paris, London and Barcelona in July and August", xlabel='City', ylabel='Number of Tourists')
```

✓ 0.1s Python

```
<AxesSubplot: title={'center': 'Number of tourists who visited Paris, London and Barcelona in July and August'}, xlabel='City', ylabel='Number of Tourists'>
```



Using the continents.csv file, create separate bar charts to present information about the population and area of the continents.

```
continents = pd.read_csv('continents.csv')
continents
```

✓ 0.0s

| | Continent | Area | Population |
|---|---|---|---|
| 0 | Europe | 10000000 | 745173774 |
| 1 | Asia | 44614000 | 4694576167 |
| 2 | North America | 24230000 | 595783465 |
| 3 | Oceania | 8510926 | 44491724 |
| 4 | Africa | 30365000 | 1393676444 |

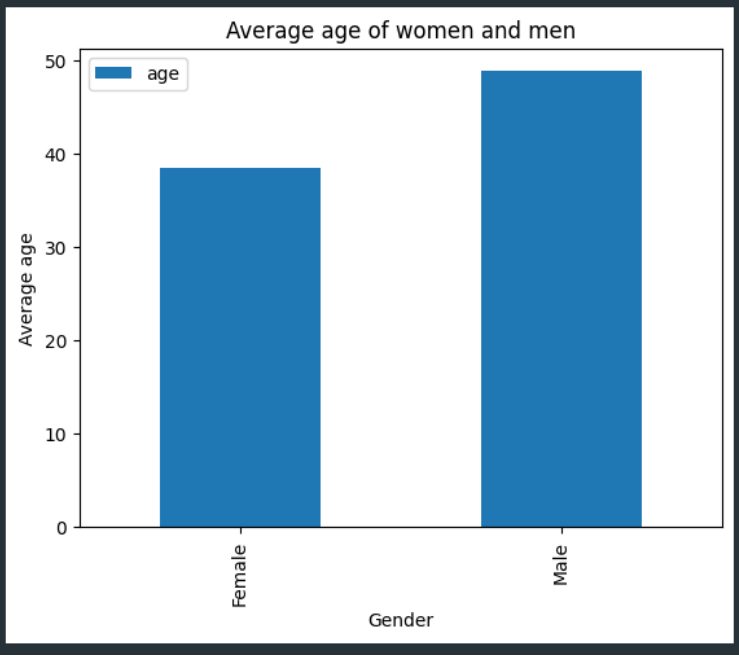
```
continents.index = continents['Continent']
continents.loc[:,['Continent','Area']].plot(kind='bar',title='Area of continents', xlabel='Continents', ylabel='Area')
continents.loc[:,['Continent','Population']].plot(kind='bar',title='Population of continents', xlabel='Continents', ylabel='Population')
```

based on the employees.csv file, create a bar chart presenting the average age of women and men (two bars).

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('employees.csv')
df = data.loc[:,['sex','age']].groupby(['sex']).mean('age')
df.index = ['Female','Male']
df.plot(kind='bar', title="Average age of women and men", xlabel='Gender', ylabel='Average age')
```

✓ 0.1s

```
<AxesSubplot: title={'center': 'Average age of women and men'}, xlabel='Gender', ylabel='Average age'>
```



Assuming now you have two queries loaded into Power Query (let's call them `Data1` and `Data2`), you can merge them based on a common column.

1. **Open Power Query:** Go to `Data` > `Get Data` > `Launch Power Query Editor`.
2. **Select the First Query:** In the Queries pane on the left, click on `Data1`.
3. **Merge Queries:** Go to the `Home` tab in the Ribbon, and click on `Merge Queries`.
4. **Configure Merge:**
   - In the merge window, select `Data2` from the dropdown to merge with `Data1`.
   - Click on the column in `Data1` that you want to match with a column in `Data2`.
   - Click on the corresponding column in `Data2`.
   - Choose the type of join you need (e.g., Inner join, Left outer join, etc.). An `Inner join` will combine rows from both datasets where there is a match; a `Left outer join` will include all rows from `Data1` and the matched rows from `Data2`.
5. **Complete Merge:** Click `OK`. Power Query will now create a new query with the merged data.

calculate the number of women in Slovakia: =COUNTIFS(G2:G101; "Slovakia"; C2:C101;"Female")

display a list of people aged 18-25: =FILTER(A:G; (D:D >= 18)*(D:D <= 25); "No data")