Pandas Library

One of the core libraries for data analysis in Python.

Library website: https://pandas.pydata.org/

Documentation: https://pandas.pydata.org/docs/

Installation: `pip install pandas`

Import Pandas library

In [1]:
```
import pandas as pd
pd.__version__
```

Out[1]:
```
'2.2.2'
```

Basic data structures of the Pandas library

The Pandas library provides two basic data structures:

- pd.Series - stores data in the form of a vector
- pd.DataFrame - stores data in the form of a set of vectors

pd.Series

The `pd.Series` is a one-dimensional sequential data structure that is capable of handling any type of data, such as strings, numeric, date/time (datetime), Python language lists, and dictionaries with labels and indexes.

Creating the `pd.Series` element

In [2]:
```
series = pd.Series(data = [1,3,5,8,9])
print(series)
```

```
0    1
1    3
2    5
3    8
4    9
dtype: int64
```

In [3]:
```
series = pd.Series(data = [1.,3,5,8,9])
print(series)
```

```
0    1.0
1    3.0
2    5.0
3    8.0
4    9.0
dtype: float64
```

In [4]:
```
series = pd.Series(data = [True, False, True, True])
print(series)
```

```
0     True
1    False
2     True
3     True
dtype: bool
```

In [5]:
```
series = pd.Series(data = ['Mazda', 'Opel', 'Mercedes', 'BMW', 'Audi'])
print(series)
```

```
0       Mazda
1        Opel
2    Mercedes
3         BMW
4        Audi
dtype: object
```

In [6]:
```
import numpy as np

series = pd.Series(data = np.arange(start=1, stop=21))
print(series)
```

```
0      1
1      2
2      3
3      4
4      5
5      6
6      7
7      8
8      9
9     10
10    11
11    12
12    13
13    14
14    15
15    16
16    17
17    18
18    19
19    20
dtype: int64
```

Defining an index

In [7]:
```
series = pd.Series(data = [1,3,5,8,9], index=['a','b','c','d','e'])
print(series)
```

```
a    1
b    3
c    5
d    8
e    9
dtype: int64
```

In [8]:
```
series = pd.Series(data = [1,3,5,8,9], index=pd.date_range(start='20240101', periods=5))
print(series)
```

```
2024-01-01    1
2024-01-02    3
2024-01-03    5
2024-01-04    8
2024-01-05    9
Freq: D, dtype: int64
```

Naming the data series

In [9]:
```
series = pd.Series(data = [1,3,5,8,9], index=pd.date_range(start='20240101', periods=5), name='data')
print(series)
```

```
2024-01-01    1
2024-01-02    3
2024-01-03    5
2024-01-04    8
2024-01-05    9
Freq: D, Name: data, dtype: int64
```

Representation of missing values

A constant `np.nan` from the Numpy library is used to represent missing data values. By default, functions that operate on data omit null values.

In [10]:
```
series = pd.Series(data = [2,np.nan,5,8,np.nan], index=pd.date_range(start='20220101', periods=5), name='data')
print(series)
print(series.mean())
```

```
2022-01-01    2.0
2022-01-02    NaN
2022-01-03    5.0
2022-01-04    8.0
2022-01-05    NaN
Freq: D, Name: data, dtype: float64
5.0
```

Attributes and functions of the `Series` object

`index` - index of the `Series` object

In [11]:
```
series = pd.Series(data = [1.,3,5,8,9])
print(series.index)
```

```
RangeIndex(start=0, stop=5, step=1)
```

In [12]:
```
series = pd.Series(data = [1,3,5,8,9], index=['a','b','c','d', 'e'])
print(series.index)
list(series.index)
```

```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

Out[12]:
```
['a', 'b', 'c', 'd', 'e']
```

In [13]:
```
series = pd.Series(data = [1,np.nan,5,8,np.nan], index=pd.date_range(start='20220101', periods=5), name='data')
print(series.index)
```

```
list(series.index)
```

```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05'],
              dtype='datetime64[ns]', freq='D')
```

Out[13]:
```
[Timestamp('2022-01-01 00:00:00'),
 Timestamp('2022-01-02 00:00:00'),
 Timestamp('2022-01-03 00:00:00'),
 Timestamp('2022-01-04 00:00:00'),
 Timestamp('2022-01-05 00:00:00')]
```

`values` - values of the `Series` object

In [14]:
```
series = pd.Series(data = [1.,3,5,8,9])
print(series.values)
```

```
[1. 3. 5. 8. 9.]
```

In [15]:
```
series = pd.Series(data = [1,np.nan,5,8,np.nan], index=pd.date_range(start='20220101', periods=5), name='dane')
print(series.values)
```

```
[ 1. nan  5.  8. nan]
```

`dtypes` - the value type of the `Series` object

In [16]:
```
series = pd.Series(data = [1.,3,5,8,9])
print(series.dtypes)
```

```
float64
```

In [17]:
```
series = pd.Series(data = [1,3,5,8,9], index=pd.date_range(start='20220101', periods=5))
print(series.dtype)
```

```
int64
```

`shape` - data shape of the `Series` object

In [18]:
```
series = pd.Series(data = [1,3,5,8,9], index=pd.date_range(start='20220101', periods=5))
print(series.shape)
```

```
(5,)
```

`count()` - the function returns a number of values

In [19]:
```
series = pd.Series(data = [1,3,5,8,9], index=pd.date_range(start='20220101', periods=5))
print(series.count())
```

```
5
```

`values_count()` - the function returns the number of occurrences of each value

In [20]:
```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
print(series.value_counts())
```

```
1    3
3    2
5    2
8    1
9    1
Name: count, dtype: int64
```

`sum()` - returns the sum of the values of series

`mean()` - returns the average value of series

`min()` - returns the minimum value of series

`max()` - returns the maximum value of series

`std()` - returns the standard deviation value of series

In [21]:
```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
print(series.sum())
print(series.mean())
print(series.min())
print(series.max())
print(series.std())
```

```
36
4.0
1
9
3.0
```

`describe()` - returns basic information about the data (count, sum, maximum, minimum, quantiles of the distribution, standard deviation)

In [22]:
```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
print(series.describe())
```

```
count    9.0
mean     4.0
std      3.0
min      1.0
25%      1.0
50%      3.0
75%      5.0
max      9.0
dtype: float64
```

`nlargest(n)` - returns the n largest values

In [23]:
```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
print(series.nlargest(3))
print(series.nlargest(3,'all'))
```

```
2022-01-05    9
2022-01-04    8
2022-01-03    5
Freq: -1D, dtype: int64
2022-01-05    9
2022-01-04    8
2022-01-03    5
2022-01-08    5
dtype: int64
```

`nsmallest(n)` - returns the n smallest values

In [24]:
```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
print(series.nsmallest(4))
print(series.nsmallest(4,'all'))
```

```
2022-01-01    1
2022-01-07    1
2022-01-09    1
2022-01-02    3
dtype: int64
2022-01-01    1
2022-01-07    1
2022-01-09    1
2022-01-02    3
2022-01-06    3
dtype: int64
```

`sort_values()` - returns sorted values

In [25]:
```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
series.sort_values()
```

Out[25]:

	0
2022-01-01	1
2022-01-07	1
2022-01-09	1
2022-01-02	3
2022-01-06	3
2022-01-03	5
2022-01-08	5
2022-01-04	8
2022-01-05	9

dtype: int64

In [26]:
```
series.sort_values(ascending=False)
```

Out[26]:

	0
2022-01-05	9
2022-01-04	8
2022-01-03	5
2022-01-08	5
2022-01-02	3
2022-01-06	3
2022-01-01	1
2022-01-07	1
2022-01-09	1

dtype: int64

`apply()` **- performs the specified function on the data**

In [27]:

```
series = pd.Series(data = [1,3,5,8,9,3,1,5,1], index=pd.date_range(start='20220101', periods=9))
print(series.apply(lambda x : x**2))
print(series)
new_series = series.apply(lambda x : x**2)
print(new_series)
```

```
2022-01-01     1
2022-01-02     9
2022-01-03    25
2022-01-04    64
2022-01-05    81
2022-01-06     9
2022-01-07     1
2022-01-08    25
2022-01-09     1
Freq: D, dtype: int64
2022-01-01    1
2022-01-02    3
2022-01-03    5
2022-01-04    8
2022-01-05    9
2022-01-06    3
2022-01-07    1
2022-01-08    5
2022-01-09    1
Freq: D, dtype: int64
2022-01-01     1
2022-01-02     9
2022-01-03    25
2022-01-04    64
2022-01-05    81
2022-01-06     9
2022-01-07     1
2022-01-08    25
2022-01-09     1
Freq: D, dtype: int64
```

pd.DataFrame

The `pd.DataFrame` is a two-dimensional data structure consisting of elements of type pd.Series.

Creating a `pd.DataFrame` element

In [28]:
```
dataf = pd.DataFrame(data=[10, 20, 30, 40])
dataf
```

Out[28]:

	0
0	10
1	20
2	30
3	40

In [29]:
```
dataf = pd.DataFrame(data=[10, 20, 30, 40], index=['a', 'b', 'c', 'd'], columns=['col_1'])
dataf
```

Out[29]:

	col_1
a	10
b	20
c	30
d	40

In [30]:
```
dataf = pd.DataFrame(data=[[10, 20, 30], [40, 50, 60]], index=['a', 'b'], columns=['col_1', 'col_2', 'col_3'])
dataf
```

Out[30]:

	col_1	col_2	col_3
a	10	20	30
b	40	50	60

In [31]:
```
dataf = pd.DataFrame(data={'Brand':['Audi', 'BMW', 'Mercedes', 'Lexus'],
                           'Model':['A6', 'X7', 'E300', 'LS500']})

dataf
```

Out[31]:

	Brand	Model
0	Audi	A6
1	BMW	X7
2	Mercedes	E300
3	Lexus	LS500

Information on frames and data

Retrieving a list of columns - `columns` attribute

In [32]:
```
dataf = pd.DataFrame(data=[[10, 20, 30], [40, 50, 60]], index=['a', 'b'], columns=['col_1', 'col_2', 'col_3'])
dataf.columns
```

Out[32]:
```
Index(['col_1', 'col_2', 'col_3'], dtype='object')
```

Retrieving the index list - `index` attribute

In [33]:
```
dataf = pd.DataFrame(data=[[10, 20, 30], [40, 50, 60]], index=['a', 'b'], columns=['col_1', 'col_2', 'col_3'])
dataf.index
```

Out[33]:
```
Index(['a', 'b'], dtype='object')
```

Retrieving data frame information - function `info()`.

In [34]:
```
dataf = pd.DataFrame(data=[[10, 20, 30], [40, 50, 60]], index=['a', 'b'], columns=['col_1', 'col_2', 'col_3'])
dataf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2 entries, a to b
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col_1   2 non-null      int64
 1   col_2   2 non-null      int64
 2   col_3   2 non-null      int64
dtypes: int64(3)
memory usage: 64.0+ bytes
```

Retrieving basic information about the data - function `describe()`.

In [35]:
```
dataf = pd.DataFrame(data=[[10, 20, 30], [40, 50, 60]], index=['a', 'b'], columns=['col_1', 'col_2', 'col_3'])
dataf.describe()
dataf.describe().T
```

Out[35]:

	count	mean	std	min	25%	50%	75%	max
col_1	2.0	25.0	21.213203	10.0	17.5	25.0	32.5	40.0
col_2	2.0	35.0	21.213203	20.0	27.5	35.0	42.5	50.0
col_3	2.0	45.0	21.213203	30.0	37.5	45.0	52.5	60.0

Retrieving first or last data records - `head` and `tail` functions

In [36]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 5 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   country                       193 non-null    object
 1   beer_servings                 193 non-null    int64
 2   spirit_servings               193 non-null    int64
 3   wine_servings                 193 non-null    int64
 4   total_litres_of_pure_alcohol  193 non-null    float64
dtypes: float64(1), int64(3), object(1)
memory usage: 7.7+ KB
```

In [37]:
```
drinks.head()
```

Out[37]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9

In [38]:
```
drinks.head(10)
```

Out[38]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7

In [39]:
```
drinks.tail()
```

Out[39]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
188	Venezuela	333	100	3	7.7
189	Vietnam	111	2	1	2.0
190	Yemen	6	0	0	0.1
191	Zambia	32	19	4	2.5
192	Zimbabwe	64	18	4	4.7

In [40]:
```
drinks.tail(20)
```

Out[40]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
173	Tonga	36	21	5	1.1
174	Trinidad & Tobago	197	156	7	6.4
175	Tunisia	51	3	20	1.3
176	Turkey	51	22	7	1.4
177	Turkmenistan	19	71	32	2.2
178	Tuvalu	6	41	9	1.0
179	Uganda	45	9	0	8.3
180	Ukraine	206	237	45	8.9
181	United Arab Emirates	16	135	5	2.8
182	United Kingdom	219	126	195	10.4
183	Tanzania	36	6	1	5.7
184	USA	249	158	84	8.7
185	Uruguay	115	35	220	6.6
186	Uzbekistan	25	101	8	2.4
187	Vanuatu	21	18	11	0.9
188	Venezuela	333	100	3	7.7
189	Vietnam	111	2	1	2.0
190	Yemen	6	0	0	0.1
191	Zambia	32	19	4	2.5
192	Zimbabwe	64	18	4	4.7

Getting and setting the column index - `columns` attribute

In [41]:
```
drinks.columns
```

Out[41]:
```
Index(['country', 'beer_servings', 'spirit_servings', 'wine_servings',
       'total_litres_of_pure_alcohol'],
      dtype='object')
```

In [42]:
```
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
drinks.head()
```

Out[42]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9

In [43]:
```
drinks.describe().T
```

Out[43]:

	count	mean	std	min	25%	50%	75%	max
beer	193.0	106.160622	101.143103	0.0	20.0	76.0	188.0	376.0
spirit	193.0	80.994819	88.284312	0.0	4.0	56.0	128.0	438.0
wine	193.0	49.450777	79.697598	0.0	1.0	8.0	59.0	370.0
total	193.0	4.717098	3.773298	0.0	1.3	4.2	7.2	14.4

Data selection

Columns selection

In [44]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
drinks = drinks.head(10)
drinks
```

Out[44]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7

Using the column name attribute - an object of type `Series` is obtained as the result.

(this method does not work for columns containing characters that cannot be in Python variable names).

In [45]:
```
print(drinks.country)
type(drinks.country)
```

```
0          Afghanistan
1              Albania
2              Algeria
3              Andorra
4               Angola
5    Antigua & Barbuda
6            Argentina
7              Armenia
8            Australia
9              Austria
Name: country, dtype: object
```

Out[45]:

pandas.core.series.Series

def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None, fastpath: bool | lib.NoDefault=lib.no_default) -> None

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical

Using the column name in square brackets - as a result we get an object of type `Series`.

In [46]:
```
print(drinks['country'])
type(drinks['country'])
```

```
0          Afghanistan
1              Albania
2              Algeria
3              Andorra
4               Angola
5    Antigua & Barbuda
6            Argentina
7              Armenia
8            Australia
9              Austria
Name: country, dtype: object
```

Out[46]:

pandas.core.series.Series

def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None, fastpath: bool | lib.NoDefault=lib.no_default) -> None

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical

Using a list of column names in square brackets - the result is an object of type `DataFrame`.

In [47]:
```
drinks[['country']]
```

Out[47]:

	country
0	Afghanistan
1	Albania
2	Algeria
3	Andorra
4	Angola
5	Antigua & Barbuda
6	Argentina
7	Armenia
8	Australia
9	Austria

In [48]:
```
type(drinks[['country']])
```

Out[48]:

pandas.core.frame.DataFrame

def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype | None=None, copy: bool | None=None) -> None

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary

In [49]:
```
drinks[['country','total']]
```

Out[49]:

	country	total
0	Afghanistan	0.0
1	Albania	4.9
2	Algeria	0.7
3	Andorra	12.4
4	Angola	5.9
5	Antigua & Barbuda	4.9
6	Argentina	8.3
7	Armenia	3.8
8	Australia	10.4
9	Austria	9.7

Using the index (index list) of the column using the `iloc` attribute - the result is a DataFrame type object.

The `:` sign indicates all records/columns.

In [50]:
```
drinks.iloc[:, 0:3]
```

Out[50]:

	country	beer	spirit
0	Afghanistan	0	0
1	Albania	89	132
2	Algeria	25	0
3	Andorra	245	138
4	Angola	217	57
5	Antigua & Barbuda	102	128
6	Argentina	193	25
7	Armenia	21	179
8	Australia	261	72
9	Austria	279	75

In [51]:
```
drinks.iloc[:, [0,-2]]
```

Out[51]:

	country	wine
0	Afghanistan	0
1	Albania	54
2	Algeria	14
3	Andorra	312
4	Angola	45
5	Antigua & Barbuda	45
6	Argentina	221
7	Armenia	11
8	Australia	212
9	Austria	191

Row selection

In [52]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
drinks.index = drinks['country']
print(drinks.index)
drinks.head(10)
```

```
Index(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua & Barbuda', 'Argentina', 'Armenia', 'Australia', 'Austria',
       ...
       'Tanzania', 'USA', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela',
       'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype='object', name='country', length=193)
```

Out[52]:

country	country	beer	spirit	wine	total
Afghanistan	Afghanistan	0	0	0	0.0
Albania	Albania	89	132	54	4.9
Algeria	Algeria	25	0	14	0.7
Andorra	Andorra	245	138	312	12.4
Angola	Angola	217	57	45	5.9
Antigua & Barbuda	Antigua & Barbuda	102	128	45	4.9
Argentina	Argentina	193	25	221	8.3
Armenia	Armenia	21	179	11	3.8
Australia	Australia	261	72	212	10.4
Austria	Austria	279	75	191	9.7

The `loc` method - selecting by name or by a list of index names (integers or labels).

In [53]:
```
print(drinks.loc['Poland'])
type(drinks.loc['Poland'])
```

```
country    Poland
beer          343
spirit        215
wine           56
total        10.9
Name: Poland, dtype: object
```

Out[53]:

pandas.core.series.Series

def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None, fastpath: bool | lib.NoDefault=lib.no_default) -> None

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical

In [54]:
```
drinks.loc[['Brazil']]
```

Out[54]:

	country	beer	spirit	wine	total
country					
Brazil	Brazil	245	145	16	7.2

In [55]:
```
type(drinks.loc[['Poland']])
```

Out[55]:

pandas.core.frame.DataFrame

def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype | None=None, copy: bool | None=None) -> None

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary

In [56]:
```
drinks.loc[['Poland','Portugal', 'Brazil']]
```

Out[56]:

country	country	beer	spirit	wine	total
Poland	Poland	343	215	56	10.9
Portugal	Portugal	194	67	339	11.0
Brazil	Brazil	245	145	16	7.2

In [57]:
```
drinks.loc[['Poland','Portugal', 'Brazil'],['total']]
```

Out[57]:

country	total
Poland	10.9
Portugal	11.0
Brazil	7.2

In [58]:
```
drinks.loc[['Poland','Portugal', 'Brazil'],['beer', 'wine']]
```

Out[58]:

country	beer	wine
Poland	343	56
Portugal	194	339
Brazil	245	16

In [59]:
```
drinks.loc[:,['beer', 'wine']]
```

Out[59]:

country	beer	wine
Afghanistan	0	0
Albania	89	54
Algeria	25	14
Andorra	245	312
Angola	217	45
...	...	...
Venezuela	333	3
Vietnam	111	1
Yemen	6	0
Zambia	32	4
Zimbabwe	64	4

193 rows × 2 columns

The `iloc` method - selecting by number, range or index list.

In [60]:
```
drinks.iloc[1]
```

Out[60]:

	Albania
country	Albania
beer	89
spirit	132
wine	54
total	4.9

dtype: object

In [61]:
```
drinks.iloc[0:5]
```

Out[61]:

country	country	beer	spirit	wine	total
Afghanistan	Afghanistan	0	0	0	0.0
Albania	Albania	89	132	54	4.9
Algeria	Algeria	25	0	14	0.7
Andorra	Andorra	245	138	312	12.4
Angola	Angola	217	57	45	5.9

In [62]:
```
drinks.iloc[0:5,1:3]
```

Out[62]:

country	beer	spirit
Afghanistan	0	0
Albania	89	132
Algeria	25	0
Andorra	245	138
Angola	217	57

In [63]:
```
drinks.iloc[[1,2,3,5,10,20]]
```

Out[63]:

	country	beer	spirit	wine	total
country					
Albania	Albania	89	132	54	4.9
Algeria	Algeria	25	0	14	0.7
Andorra	Andorra	245	138	312	12.4
Antigua & Barbuda	Antigua & Barbuda	102	128	45	4.9
Azerbaijan	Azerbaijan	21	46	5	1.3
Bolivia	Bolivia	167	41	8	3.8

Data filtering

In [64]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
```

Selection masks

In [65]:
```
drinks.total > 10
```

Out[65]:

	total
0	False
1	False
2	False
3	True
4	False
...	...
188	False
189	False
190	False
191	False
192	False

193 rows × 1 columns

dtype: bool

In [66]:

```
drinks[drinks.total > 10]
```

Out[66]:

	country	beer	spirit	wine	total
3	Andorra	245	138	312	12.4
8	Australia	261	72	212	10.4
15	Belarus	142	373	42	14.4
16	Belgium	295	84	212	10.5
25	Bulgaria	231	252	94	10.3
42	Croatia	230	87	254	10.2
45	Czech Republic	361	170	134	11.8
48	Denmark	224	81	278	10.4
61	France	127	151	370	11.8
65	Germany	346	117	175	11.3
68	Grenada	199	438	28	11.9
75	Hungary	234	215	185	11.3
81	Ireland	313	118	165	11.4
93	Latvia	281	216	62	10.5
98	Lithuania	343	244	56	12.9
99	Luxembourg	236	133	271	11.4
135	Poland	343	215	56	10.9
136	Portugal	194	67	339	11.0
140	Romania	297	122	167	10.4
141	Russian Federation	247	326	73	11.5
144	St. Lucia	171	315	71	10.1
155	Slovakia	196	293	116	11.4
156	Slovenia	270	51	276	10.6
166	Switzerland	185	100	280	10.2
182	United Kingdom	219	126	195	10.4

In [67]:
```
drinks.beer < drinks.wine
```

Out[67]:

	0
0	False
1	False
2	False
3	True
4	False
...	...
188	False
189	False
190	False
191	False
192	False

193 rows × 1 columns

dtype: bool

In [68]:
```
wine_above_beer = drinks[drinks.beer < drinks.wine]
wine_above_beer
```

Out[68]:

	country	beer	spirit	wine	total
3	Andorra	245	138	312	12.4
6	Argentina	193	25	221	8.3
35	Chile	130	124	172	7.6
40	Cook Islands	0	254	74	5.9
42	Croatia	230	87	254	10.2
48	Denmark	224	81	278	10.4
55	Equatorial Guinea	92	0	233	5.8
61	France	127	151	370	11.8
64	Georgia	52	100	149	5.4
67	Greece	133	112	218	8.3
83	Italy	85	42	237	6.5
92	Laos	62	0	123	6.2
94	Lebanon	20	55	31	1.9
99	Luxembourg	236	133	271	11.4
113	Montenegro	31	114	128	4.9
136	Portugal	194	67	339	11.0
137	Qatar	1	42	7	0.9
148	Sao Tome & Principe	56	38	140	4.2
156	Slovenia	270	51	276	10.6
165	Sweden	152	60	186	7.2
166	Switzerland	185	100	280	10.2
167	Syria	5	35	16	1.0
171	Timor-Leste	1	1	4	0.1
177	Turkmenistan	19	71	32	2.2
178	Tuvalu	6	41	9	1.0

	country	beer	spirit	wine	total
185	Uruguay	115	35	220	6.6

In [69]:
```
(drinks.beer == drinks.wine) | (drinks.beer == drinks.spirit) | (drinks.spirit == drinks.wine)
```

Out[69]:

	0
0	True
1	False
2	False
3	False
4	False
...	...
188	False
189	False
190	True
191	False
192	False

193 rows × 1 columns

dtype: bool

In [70]:
```
drinks[(drinks.beer == drinks.wine) | (drinks.beer == drinks.spirit) | (drinks.spirit == drinks.wine)]
```

Out[70]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
13	Bangladesh	0	0	0	0.0
19	Bhutan	23	0	0	0.4
22	Botswana	173	35	35	5.4
26	Burkina Faso	25	7	7	4.3
27	Burundi	88	0	0	6.3
34	Chad	15	1	1	0.4
38	Comoros	1	3	1	0.1
46	North Korea	0	0	0	0.0
56	Eritrea	18	0	0	0.5
73	Haiti	1	326	1	5.9
79	Iran	0	0	0	0.0
90	Kuwait	0	0	0	0.0
97	Libya	0	0	0	0.0
103	Maldives	0	0	0	0.0
104	Mali	5	1	1	0.6
106	Marshall Islands	0	0	0	0.0
107	Mauritania	0	0	0	0.0
111	Monaco	0	0	0	0.0
128	Pakistan	0	0	0	0.0
147	San Marino	0	0	0	0.0
149	Saudi Arabia	0	5	0	0.1
158	Somalia	0	0	0	0.0
164	Swaziland	90	2	2	4.7
171	Timor-Leste	1	1	4	0.1

	country	beer	spirit	wine	total
190	Yemen	6	0	0	0.1

In [71]:
```
drinks[(drinks.beer == drinks.wine) & (drinks.beer == drinks.spirit) & (drinks.spirit == drinks.wine)]
```

Out[71]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
13	Bangladesh	0	0	0	0.0
46	North Korea	0	0	0	0.0
79	Iran	0	0	0	0.0
90	Kuwait	0	0	0	0.0
97	Libya	0	0	0	0.0
103	Maldives	0	0	0	0.0
106	Marshall Islands	0	0	0	0.0
107	Mauritania	0	0	0	0.0
111	Monaco	0	0	0	0.0
128	Pakistan	0	0	0	0.0
147	San Marino	0	0	0	0.0
158	Somalia	0	0	0	0.0

In [72]:
```
drinks[drinks.country.str.startswith('A')]
```

Out[72]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7
10	Azerbaijan	21	46	5	1.3

Data operations

Data linking

Data appending is done using the `concat` function.

In [73]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
drinks.head(10)
```

Out[73]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7

In [74]:

```
country_A = drinks[drinks.country.str.startswith('A')]
country_A
```

Out[74]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7
10	Azerbaijan	21	46	5	1.3

In [75]:
```
country_B = drinks[drinks.country.str.startswith('B')]
country_B
```

Out[75]:

	country	beer	spirit	wine	total
11	Bahamas	122	176	51	6.3
12	Bahrain	42	63	7	2.0
13	Bangladesh	0	0	0	0.0
14	Barbados	143	173	36	6.3
15	Belarus	142	373	42	14.4
16	Belgium	295	84	212	10.5
17	Belize	263	114	8	6.8
18	Benin	34	4	13	1.1
19	Bhutan	23	0	0	0.4
20	Bolivia	167	41	8	3.8
21	Bosnia-Herzegovina	76	173	8	4.6
22	Botswana	173	35	35	5.4
23	Brazil	245	145	16	7.2
24	Brunei	31	2	1	0.6
25	Bulgaria	231	252	94	10.3
26	Burkina Faso	25	7	7	4.3
27	Burundi	88	0	0	6.3

In [76]:

```
country_AB = pd.concat([country_A, country_B], ignore_index=True)
country_AB
```

Out[76]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7
10	Azerbaijan	21	46	5	1.3
11	Bahamas	122	176	51	6.3
12	Bahrain	42	63	7	2.0
13	Bangladesh	0	0	0	0.0
14	Barbados	143	173	36	6.3
15	Belarus	142	373	42	14.4
16	Belgium	295	84	212	10.5
17	Belize	263	114	8	6.8
18	Benin	34	4	13	1.1
19	Bhutan	23	0	0	0.4
20	Bolivia	167	41	8	3.8
21	Bosnia-Herzegovina	76	173	8	4.6
22	Botswana	173	35	35	5.4
23	Brazil	245	145	16	7.2
24	Brunei	31	2	1	0.6

	country	beer	spirit	wine	total
25	Bulgaria	231	252	94	10.3
26	Burkina Faso	25	7	7	4.3
27	Burundi	88	0	0	6.3

Joining tables

In [77]:
```
capitals = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/capital_cities.xlsx')
capitals
```

Out[77]:

	Country	Capital City
0	Afghanistan	Kabul
1	Albania	Tirana
2	Algeria	Algiers
3	Andorra	Andorra la Vella
4	Angola	Luanda
...	...	...
198	Yemen	Sana'a[26]
199	Zambia	Lusaka
200	Zimbabwe	Harare
201	Niue	Alofi
202	Cook Islands	Avarua

203 rows × 2 columns

In [78]:
```
capitals.columns = ['country', 'capital_city']
```

In [79]:
```
kon_alk = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
kon_alk.columns = ['country', 'beer', 'spirit', 'wine', 'total']
```

In [80]:
```
result = pd.merge(kon_alk, capitals, left_on='country', right_on='country')
#result = pd.merge(kon_alk, capitals, on='country')
result
```

Out[80]:

	country	beer	spirit	wine	total	capital_city
0	Afghanistan	0	0	0	0.0	Kabul
1	Albania	89	132	54	4.9	Tirana
2	Algeria	25	0	14	0.7	Algiers
3	Andorra	245	138	312	12.4	Andorra la Vella
4	Angola	217	57	45	5.9	Luanda
...	...	...	...	...	...	...
188	Venezuela	333	100	3	7.7	Caracas
189	Vietnam	111	2	1	2.0	Hanoi
190	Yemen	6	0	0	0.1	Sana'a[26]
191	Zambia	32	19	4	2.5	Lusaka
192	Zimbabwe	64	18	4	4.7	Harare

193 rows × 6 columns

Data sorting

The sorting function is `sort_values`, which allows the sorting attributes to be indicated.

In [81]:
```
drinks_sorted = drinks.sort_values(by='total', ascending=False)
drinks_sorted.head(20)
```

Out[81]:

	country	beer	spirit	wine	total
15	Belarus	142	373	42	14.4
98	Lithuania	343	244	56	12.9
3	Andorra	245	138	312	12.4
68	Grenada	199	438	28	11.9
45	Czech Republic	361	170	134	11.8
61	France	127	151	370	11.8
141	Russian Federation	247	326	73	11.5
81	Ireland	313	118	165	11.4
155	Slovakia	196	293	116	11.4
99	Luxembourg	236	133	271	11.4
65	Germany	346	117	175	11.3
75	Hungary	234	215	185	11.3
136	Portugal	194	67	339	11.0
135	Poland	343	215	56	10.9
156	Slovenia	270	51	276	10.6
93	Latvia	281	216	62	10.5
16	Belgium	295	84	212	10.5
48	Denmark	224	81	278	10.4
182	United Kingdom	219	126	195	10.4
140	Romania	297	122	167	10.4

The `inplace` attribute allows the result to be sorted and stored in the current data frame.

In [82]:
```
drinks.sort_values(by='total', ascending=False, inplace=True)
drinks.head(20)
```

Out[82]:

	country	beer	spirit	wine	total
15	Belarus	142	373	42	14.4
98	Lithuania	343	244	56	12.9
3	Andorra	245	138	312	12.4
68	Grenada	199	438	28	11.9
45	Czech Republic	361	170	134	11.8
61	France	127	151	370	11.8
141	Russian Federation	247	326	73	11.5
81	Ireland	313	118	165	11.4
155	Slovakia	196	293	116	11.4
99	Luxembourg	236	133	271	11.4
65	Germany	346	117	175	11.3
75	Hungary	234	215	185	11.3
136	Portugal	194	67	339	11.0
135	Poland	343	215	56	10.9
156	Slovenia	270	51	276	10.6
93	Latvia	281	216	62	10.5
16	Belgium	295	84	212	10.5
48	Denmark	224	81	278	10.4
182	United Kingdom	219	126	195	10.4
140	Romania	297	122	167	10.4

Data grouping

The `groupby` function is used to group data.

In [83]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
```

```
drinks['start_letter'] = drinks.country.str[0]
drinks.head(20)
```

Out[83]:

	country	beer	spirit	wine	total	start_letter
0	Afghanistan	0	0	0	0.0	A
1	Albania	89	132	54	4.9	A
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	A
4	Angola	217	57	45	5.9	A
5	Antigua & Barbuda	102	128	45	4.9	A
6	Argentina	193	25	221	8.3	A
7	Armenia	21	179	11	3.8	A
8	Australia	261	72	212	10.4	A
9	Austria	279	75	191	9.7	A
10	Azerbaijan	21	46	5	1.3	A
11	Bahamas	122	176	51	6.3	B
12	Bahrain	42	63	7	2.0	B
13	Bangladesh	0	0	0	0.0	B
14	Barbados	143	173	36	6.3	B
15	Belarus	142	373	42	14.4	B
16	Belgium	295	84	212	10.5	B
17	Belize	263	114	8	6.8	B
18	Benin	34	4	13	1.1	B
19	Bhutan	23	0	0	0.4	B

In [84]:

```
country_group = drinks.groupby(by='start_letter')
country_group.total.sum().sort_values(ascending=False)
```

Out[84]:

start_letter	total
S	137.0
B	90.3
C	89.7
A	62.3
G	62.0
P	55.5
N	53.1
L	48.8
U	48.1
M	42.1
I	29.5
R	28.7
T	27.2
D	26.6
F	23.8
E	23.1
H	20.2
K	12.0
J	10.9
V	10.6
Z	7.2
Q	0.9
O	0.7
Y	0.1

dtype: float64

Operations on columns

Adding columns

In [85]:
```
drinks = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
drinks.columns = ['country', 'beer', 'spirit', 'wine', 'total']
drinks.head(10)
```

Out[85]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
5	Antigua & Barbuda	102	128	45	4.9
6	Argentina	193	25	221	8.3
7	Armenia	21	179	11	3.8
8	Australia	261	72	212	10.4
9	Austria	279	75	191	9.7

In [86]:
```
drinks['sum_value'] = drinks.beer + drinks.spirit + drinks.wine
drinks.head(10)
```

Out[86]:

	country	beer	spirit	wine	total	sum_value
0	Afghanistan	0	0	0	0.0	0
1	Albania	89	132	54	4.9	275
2	Algeria	25	0	14	0.7	39
3	Andorra	245	138	312	12.4	695
4	Angola	217	57	45	5.9	319
5	Antigua & Barbuda	102	128	45	4.9	275
6	Argentina	193	25	221	8.3	439
7	Armenia	21	179	11	3.8	211
8	Australia	261	72	212	10.4	545
9	Austria	279	75	191	9.7	545

In [87]:
```
drinks['over_median'] = drinks.sum_value > np.median(drinks.sum_value)
drinks.head(10)
```

Out[87]:

	country	beer	spirit	wine	total	sum_value	over_median
0	Afghanistan	0	0	0	0.0	0	False
1	Albania	89	132	54	4.9	275	True
2	Algeria	25	0	14	0.7	39	False
3	Andorra	245	138	312	12.4	695	True
4	Angola	217	57	45	5.9	319	True
5	Antigua & Barbuda	102	128	45	4.9	275	True
6	Argentina	193	25	221	8.3	439	True
7	Armenia	21	179	11	3.8	211	True
8	Australia	261	72	212	10.4	545	True
9	Austria	279	75	191	9.7	545	True

Deletion of columns

Operator `del`

In [88]:
```
drinks['sum'] = drinks.beer + drinks.spirit + drinks.wine
drinks.head(10)
```

Out[88]:

	country	beer	spirit	wine	total	sum_value	over_median	sum
0	Afghanistan	0	0	0	0.0	0	False	0
1	Albania	89	132	54	4.9	275	True	275
2	Algeria	25	0	14	0.7	39	False	39
3	Andorra	245	138	312	12.4	695	True	695
4	Angola	217	57	45	5.9	319	True	319
5	Antigua & Barbuda	102	128	45	4.9	275	True	275
6	Argentina	193	25	221	8.3	439	True	439
7	Armenia	21	179	11	3.8	211	True	211
8	Australia	261	72	212	10.4	545	True	545
9	Austria	279	75	191	9.7	545	True	545

In [89]:
```
del drinks['sum']
drinks.head(10)
```

Out[89]:

	country	beer	spirit	wine	total	sum_value	over_median
0	Afghanistan	0	0	0	0.0	0	False
1	Albania	89	132	54	4.9	275	True
2	Algeria	25	0	14	0.7	39	False
3	Andorra	245	138	312	12.4	695	True
4	Angola	217	57	45	5.9	319	True
5	Antigua & Barbuda	102	128	45	4.9	275	True
6	Argentina	193	25	221	8.3	439	True
7	Armenia	21	179	11	3.8	211	True
8	Australia	261	72	212	10.4	545	True
9	Austria	279	75	191	9.7	545	True

Downloading and storing external data

The Pandas library provides a number of methods for retrieving and recording data from external sources, and a list of these can be found at:

https://pandas.pydata.org/docs/user_guide/io.html

HTML pages

The `read_html` and `to_html` functions allow you to retrieve and write data from/to tables in html pages.

In [90]:
```
download_methods = pd.read_html('https://pandas.pydata.org/docs/user_guide/io.html')
for a in download_methods:
  print(a)
```

	Format Type	Data Description	\
0	text	CSV	
1	text	Fixed-Width Text File	
2	text	JSON	
3	text	HTML	
4	text	LaTeX	
5	text	XML	
6	text	Local clipboard	
7	binary	MS Excel	
8	binary	OpenDocument	
9	binary	HDF5 Format	
10	binary	Feather Format	
11	binary	Parquet Format	
12	binary	ORC Format	
13	binary	Stata	
14	binary	SAS	
15	binary	SPSS	
16	binary	Python Pickle Format	
17	SQL	SQL	
18	SQL	Google BigQuery;:ref:read_gbq<io.bigquery>;:re...	

	Reader	Writer
0	read_csv	to_csv
1	read_fwf	NaN
2	read_json	to_json
3	read_html	to_html
4	Styler.to_latex	NaN
5	read_xml	to_xml
6	read_clipboard	to_clipboard
7	read_excel	to_excel
8	read_excel	NaN
9	read_hdf	to_hdf
10	read_feather	to_feather
11	read_parquet	to_parquet
12	read_orc	to_orc
13	read_stata	to_stata
14	read_sas	NaN
15	read_spss	NaN
16	read_pickle	to_pickle
17	read_sql	to_sql
18	NaN	NaN

	0	1
0	split	dict like {index -> [index]; columns -> [colum...
1	records	list like [{column -> value}; …]
2	index	dict like {index -> {column -> value}}
3	columns	dict like {column -> {index -> value}}

```
4   values                               just the values array
5    table                   adhering to the JSON Table Schema
```

	0	1
0	split	dict like {index -> [index]; columns -> [colum...
1	records	list like [{column -> value} …]
2	index	dict like {index -> {column -> value}}
3	columns	dict like {column -> {index -> value}}
4	values	just the values array
5	table	adhering to the JSON Table Schema

	pandas type	Table Schema type
0	int64	integer
1	float64	number
2	bool	boolean
3	datetime64[ns]	datetime
4	timedelta64[ns]	duration
5	categorical	any
6	object	str

	Type	Represents missing values
0	floating : float64, float32, float16	np.nan
1	integer : int64, int32, int8, uint64,uint32, u...	NaN
2	boolean	NaN
3	datetime64[ns]	NaT
4	timedelta64[ns]	NaT
5	categorical : see the section below	NaN
6	object : strings	np.nan

	0 \
0	read_sql_table(table_name, con[, schema, ...])
1	read_sql_query(sql, con[, index_col, ...])
2	read_sql(sql, con[, index_col, ...])
3	DataFrame.to_sql(name, con, *[, schema, ...])

	1
0	Read SQL database table into a DataFrame.
1	Read SQL query into a DataFrame.
2	Read SQL query or database table into a DataFr...
3	Write records stored in a DataFrame to a SQL d...

	id	Date	Col_1	Col_2	Col_3
0	26	2012-10-18	X	25.70	True
1	42	2012-10-19	Y	-12.40	False
2	63	2012-10-20	Z	5.73	True

	numpy/pandas	arrow	postgres	sqlite
0	int16/Int16	int16	SMALLINT	INTEGER
1	int32/Int32	int32	INTEGER	INTEGER
2	int64/Int64	int64	BIGINT	INTEGER
3	float32	float32	REAL	REAL
4	float64	float64	DOUBLE PRECISION	REAL

```
5              object                  string              TEXT     TEXT
6                bool                    bool_           BOOLEAN      NaN
7      datetime64[ns]            timestamp(us)         TIMESTAMP      NaN
8   datetime64[ns,tz]         timestamp(us,tz)       TIMESTAMPTZ      NaN
9                 NaN                  date32              DATE      NaN
10                NaN  month_day_nano_interval          INTERVAL      NaN
11                NaN                  binary            BINARY     BLOB
12                NaN           decimal128        DECIMAL [1]      NaN
13                NaN                    list         ARRAY [1]      NaN
14                NaN                  struct  COMPOSITE TYPE[1]      NaN
     Database                    SQL Datetime Types Timezone Support
0      SQLite                                 TEXT               No
1       MySQL                TIMESTAMP or DATETIME               No
2  PostgreSQL  TIMESTAMP or TIMESTAMP WITH TIME ZONE              Yes
```

In [91]:

```
includes = "Data Description"
download_methods = pd.read_html('https://pandas.pydata.org/docs/user_guide/io.html', match=includes)
for a in download_methods:
  print(a)
```

	Format Type	Data Description	Reader	Writer
0	text	CSV	read_csv	to_csv
1	text	Fixed-Width Text File	read_fwf	NaN
2	text	JSON	read_json	to_json
3	text	HTML	read_html	to_html
4	text	LaTeX	Styler.to_latex	NaN
5	text	XML	read_xml	to_xml
6	text	Local clipboard	read_clipboard	to_clipboard
7	binary	MS Excel	read_excel	to_excel
8	binary	OpenDocument	read_excel	NaN
9	binary	HDF5 Format	read_hdf	to_hdf
10	binary	Feather Format	read_feather	to_feather
11	binary	Parquet Format	read_parquet	to_parquet
12	binary	ORC Format	read_orc	to_orc
13	binary	Stata	read_stata	to_stata
14	binary	SAS	read_sas	NaN
15	binary	SPSS	read_spss	NaN
16	binary	Python Pickle Format	read_pickle	to_pickle
17	SQL	SQL	read_sql	to_sql
18	SQL	Google BigQuery;:ref:read_gbq<io.bigquery>;:re...	NaN	NaN

In [92]:
```
mp = download_methods[0]
mp
```

Out[92]:

	Format Type	Data Description	Reader	Writer
0	text	CSV	read_csv	to_csv
1	text	Fixed-Width Text File	read_fwf	NaN
2	text	JSON	read_json	to_json
3	text	HTML	read_html	to_html
4	text	LaTeX	Styler.to_latex	NaN
5	text	XML	read_xml	to_xml
6	text	Local clipboard	read_clipboard	to_clipboard
7	binary	MS Excel	read_excel	to_excel
8	binary	OpenDocument	read_excel	NaN
9	binary	HDF5 Format	read_hdf	to_hdf
10	binary	Feather Format	read_feather	to_feather
11	binary	Parquet Format	read_parquet	to_parquet
12	binary	ORC Format	read_orc	to_orc
13	binary	Stata	read_stata	to_stata
14	binary	SAS	read_sas	NaN
15	binary	SPSS	read_spss	NaN
16	binary	Python Pickle Format	read_pickle	to_pickle
17	SQL	SQL	read_sql	to_sql
18	SQL	Google BigQuery;:ref:read_gbq<io.bigquery>;:re...	NaN	NaN

In [93]:
```
mp.to_html('formats.html')
```

Spreadsheets

MS Excel

In [94]:
```
kon_alk = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
kon_alk.columns = ['country', 'beer', 'spirit', 'wine', 'total']
kon_alk.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   country  193 non-null    object 
 1   beer     193 non-null    int64  
 2   spirit   193 non-null    int64  
 3   wine     193 non-null    int64  
 4   total    193 non-null    float64
dtypes: float64(1), int64(3), object(1)
memory usage: 7.7+ KB
```

In [95]:
```
kon_alk.to_excel('drinks_serving.xlsx')
```

Libre Office Calc

In [96]:
```
!pip install odfpy

kon_alk = pd.DataFrame(pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.ods', engine='odf'))
kon_alk.columns = pd.Index(['country', 'beer', 'spirit', 'wine', 'total'])
kon_alk
```

```
Collecting odfpy
  Downloading odfpy-1.4.1.tar.gz (717 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/717.0 kB ? eta -:--:--
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 122.9/717.0 kB 3.5 MB/s eta 0:00:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 716.8/717.0 kB 10.2 MB/s eta 0:00:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 717.0/717.0 kB 8.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from odfpy) (0.7.1)
Building wheels for collected packages: odfpy
  Building wheel for odfpy (setup.py) ... done
  Created wheel for odfpy: filename=odfpy-1.4.1-py2.py3-none-any.whl size=160672 sha256=0adf1dc741733b07b6a692c809c651798104542e8988b602645160a47780a164
  Stored in directory: /root/.cache/pip/wheels/c8/2e/95/90d94fe33903786937f3b8c33dd88807f792359c6424b40469
Successfully built odfpy
Installing collected packages: odfpy
Successfully installed odfpy-1.4.1
```

Out[96]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
...	...	...	...	...	...
188	Venezuela	333	100	3	7.7
189	Vietnam	111	2	1	2.0
190	Yemen	6	0	0	0.1
191	Zambia	32	19	4	2.5
192	Zimbabwe	64	18	4	4.7

193 rows × 5 columns

In [97]:
```
kon_alk.to_excel('drinks_serving.ods', engine='odf')
```

CSV files

In [98]:
```
kon_alk = pd.DataFrame(pd.read_csv('http://bartoszj.prz-rzeszow.pl/dane/drinks.csv', sep=';'))
kon_alk.columns = pd.Index(['country', 'beer', 'spirit', 'wine', 'total'])
kon_alk
```

Out[98]:

	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
...	...	...	...	...	...
188	Venezuela	333	100	3	7.7
189	Vietnam	111	2	1	2.0
190	Yemen	6	0	0	0.1
191	Zambia	32	19	4	2.5
192	Zimbabwe	64	18	4	4.7

193 rows × 5 columns

In [99]:
```
kon_alk[kon_alk.beer > kon_alk.spirit].to_csv('beer_gt_wine.csv', sep = ',')
```

Databases

SQLite

In [100…
```
kon_alk = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
kon_alk.columns = ['country', 'beer', 'spirit', 'wine', 'total']
```

In [101…
```
from sqlite3 import connect
conn = connect('plik.db')
kon_alk.to_sql('drinks_serving_data', conn, if_exists='replace')
```

Out[101… 193

In [102…
```
kon_alk_1 = pd.read_sql_query("SELECT * FROM drinks_serving_data where total = 0", conn)
kon_alk_1
```

Out[102…

	index	country	beer	spirit	wine	total
0	0	Afghanistan	0	0	0	0.0
1	13	Bangladesh	0	0	0	0.0
2	46	North Korea	0	0	0	0.0
3	79	Iran	0	0	0	0.0
4	90	Kuwait	0	0	0	0.0
5	97	Libya	0	0	0	0.0
6	103	Maldives	0	0	0	0.0
7	106	Marshall Islands	0	0	0	0.0
8	107	Mauritania	0	0	0	0.0
9	111	Monaco	0	0	0	0.0
10	128	Pakistan	0	0	0	0.0
11	147	San Marino	0	0	0	0.0
12	158	Somalia	0	0	0	0.0

In [103…

```
kon_alk_1 = pd.read_sql_query("SELECT * FROM drinks_serving_data where total = 0", conn, index_col='index')
kon_alk_1
```

Out[103…

index	country	beer	spirit	wine	total
0	Afghanistan	0	0	0	0.0
13	Bangladesh	0	0	0	0.0
46	North Korea	0	0	0	0.0
79	Iran	0	0	0	0.0
90	Kuwait	0	0	0	0.0
97	Libya	0	0	0	0.0
103	Maldives	0	0	0	0.0
106	Marshall Islands	0	0	0	0.0
107	Mauritania	0	0	0	0.0
111	Monaco	0	0	0	0.0
128	Pakistan	0	0	0	0.0
147	San Marino	0	0	0	0.0
158	Somalia	0	0	0	0.0

PostgreSQL

In [104…

```
!sudo apt-get -y -qq update
!sudo apt-get -y -qq install postgresql
!sudo service postgresql start

!sudo -u postgres psql -U postgres -c "ALTER USER postgres PASSWORD 'postgres';"

!sudo -u postgres psql -U postgres -c 'DROP DATABASE IF EXISTS baza_demo;'
!sudo -u postgres psql -U postgres -c 'CREATE DATABASE baza_demo;'
```

```
W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not s
eem to provide it (sources.list entry misspelt?)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dial
og.pm line 78, <> line 13.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package logrotate.
(Reading database ... 123632 files and directories currently installed.)
Preparing to unpack .../00-logrotate_3.19.0-1ubuntu1.1_amd64.deb ...
Unpacking logrotate (3.19.0-1ubuntu1.1) ...
Selecting previously unselected package netbase.
Preparing to unpack .../01-netbase_6.3_all.deb ...
Unpacking netbase (6.3) ...
Selecting previously unselected package libcommon-sense-perl:amd64.
Preparing to unpack .../02-libcommon-sense-perl_3.75-2build1_amd64.deb ...
Unpacking libcommon-sense-perl:amd64 (3.75-2build1) ...
Selecting previously unselected package libjson-perl.
Preparing to unpack .../03-libjson-perl_4.04000-1_all.deb ...
Unpacking libjson-perl (4.04000-1) ...
Selecting previously unselected package libtypes-serialiser-perl.
Preparing to unpack .../04-libtypes-serialiser-perl_1.01-1_all.deb ...
Unpacking libtypes-serialiser-perl (1.01-1) ...
Selecting previously unselected package libjson-xs-perl.
Preparing to unpack .../05-libjson-xs-perl_4.030-1build3_amd64.deb ...
Unpacking libjson-xs-perl (4.030-1build3) ...
Selecting previously unselected package postgresql-client-common.
Preparing to unpack .../06-postgresql-client-common_238_all.deb ...
Unpacking postgresql-client-common (238) ...
Selecting previously unselected package postgresql-client-14.
Preparing to unpack .../07-postgresql-client-14_14.15-0ubuntu0.22.04.1_amd64.deb ...
Unpacking postgresql-client-14 (14.15-0ubuntu0.22.04.1) ...
Selecting previously unselected package ssl-cert.
Preparing to unpack .../08-ssl-cert_1.1.2_all.deb ...
Unpacking ssl-cert (1.1.2) ...
Selecting previously unselected package postgresql-common.
Preparing to unpack .../09-postgresql-common_238_all.deb ...
Adding 'diversion of /usr/bin/pg_config to /usr/bin/pg_config.libpq-dev by postgresql-common'
Unpacking postgresql-common (238) ...
Selecting previously unselected package postgresql-14.
Preparing to unpack .../10-postgresql-14_14.15-0ubuntu0.22.04.1_amd64.deb ...
Unpacking postgresql-14 (14.15-0ubuntu0.22.04.1) ...
Selecting previously unselected package postgresql.
```

```
Preparing to unpack .../11-postgresql_14+238_all.deb ...
Unpacking postgresql (14+238) ...
Selecting previously unselected package sysstat.
Preparing to unpack .../12-sysstat_12.5.2-2ubuntu0.2_amd64.deb ...
Unpacking sysstat (12.5.2-2ubuntu0.2) ...
Setting up logrotate (3.19.0-1ubuntu1.1) ...
Created symlink /etc/systemd/system/timers.target.wants/logrotate.timer → /lib/systemd/system/logrotate.timer.
Setting up libcommon-sense-perl:amd64 (3.75-2build1) ...
Setting up ssl-cert (1.1.2) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dial
og.pm line 78.)
debconf: falling back to frontend: Readline
Setting up libtypes-serialiser-perl (1.01-1) ...
Setting up libjson-perl (4.04000-1) ...
Setting up netbase (6.3) ...
Setting up sysstat (12.5.2-2ubuntu0.2) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dial
og.pm line 78.)
debconf: falling back to frontend: Readline

Creating config file /etc/default/sysstat with new version
update-alternatives: using /usr/bin/sar.sysstat to provide /usr/bin/sar (sar) in auto mode
Created symlink /etc/systemd/system/sysstat.service.wants/sysstat-collect.timer → /lib/systemd/system/sysstat-collect.timer.
Created symlink /etc/systemd/system/sysstat.service.wants/sysstat-summary.timer → /lib/systemd/system/sysstat-summary.timer.
Created symlink /etc/systemd/system/multi-user.target.wants/sysstat.service → /lib/systemd/system/sysstat.service.
Setting up postgresql-client-common (238) ...
Setting up libjson-xs-perl (4.030-1build3) ...
Setting up postgresql-client-14 (14.15-0ubuntu0.22.04.1) ...
update-alternatives: using /usr/share/postgresql/14/man/man1/psql.1.gz to provide /usr/share/man/man1/psql.1.gz (psql.1.gz) in auto mode
Setting up postgresql-common (238) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dial
og.pm line 78.)
debconf: falling back to frontend: Readline
Adding user postgres to group ssl-cert

Creating config file /etc/postgresql-common/createcluster.conf with new version
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Removing obsolete dictionary files:
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service → /lib/systemd/system/postgresql.service.
Setting up postgresql-14 (14.15-0ubuntu0.22.04.1) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dial
og.pm line 78.)
```

```
debconf: falling back to frontend: Readline
Creating new PostgreSQL cluster 14/main ...
/usr/lib/postgresql/14/bin/initdb -D /var/lib/postgresql/14/main --auth-local peer --auth-host scram-sha-256 --no-instructions
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/14/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
update-alternatives: using /usr/share/postgresql/14/man/man1/postmaster.1.gz to provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.g
z) in auto mode
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Setting up postgresql (14+238) ...
Processing triggers for man-db (2.10.2-1) ...
 * Starting PostgreSQL 14 database server
   ...done.
ALTER ROLE
NOTICE:  database "baza_demo" does not exist, skipping
DROP DATABASE
CREATE DATABASE
```

In [105…

```
kon_alk = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
kon_alk.columns = ['country', 'beer', 'spirit', 'wine', 'total']
```

In [106…

```
import psycopg2
from sqlalchemy import create_engine
engine  = create_engine('postgresql+psycopg2://postgres:postgres@localhost/baza_demo');
dbconn = engine.connect();
kon_alk.to_sql('drinks_serving_data', dbconn, if_exists='replace')
kon_from_postgresql = pd.read_sql('drinks_serving_data', dbconn)
kon_from_postgresql
```

Out[106…

	index	country	beer	spirit	wine	total
0	0	Afghanistan	0	0	0	0.0
1	1	Albania	89	132	54	4.9
2	2	Algeria	25	0	14	0.7
3	3	Andorra	245	138	312	12.4
4	4	Angola	217	57	45	5.9
...	...	...	...	...	...	...
188	188	Venezuela	333	100	3	7.7
189	189	Vietnam	111	2	1	2.0
190	190	Yemen	6	0	0	0.1
191	191	Zambia	32	19	4	2.5
192	192	Zimbabwe	64	18	4	4.7

193 rows × 6 columns

In [107…

```
kon_from_postgresql = pd.read_sql_query('SELECT * FROM drinks_serving_data where total = 0', dbconn)
kon_from_postgresql
```

Out[107…

	index	country	beer	spirit	wine	total
0	0	Afghanistan	0	0	0	0.0
1	13	Bangladesh	0	0	0	0.0
2	46	North Korea	0	0	0	0.0
3	79	Iran	0	0	0	0.0
4	90	Kuwait	0	0	0	0.0
5	97	Libya	0	0	0	0.0
6	103	Maldives	0	0	0	0.0
7	106	Marshall Islands	0	0	0	0.0
8	107	Mauritania	0	0	0	0.0
9	111	Monaco	0	0	0	0.0
10	128	Pakistan	0	0	0	0.0
11	147	San Marino	0	0	0	0.0
12	158	Somalia	0	0	0	0.0

Data visualisation

The Pandas library offers the `plot()` method as a wrapper for the Matplotlib library.

The `plot()` method allows graphs to be created directly on DataFrames.

The following parameters of the `plot()` method, among others, are used to create graphs:

- `x` , `y` : determine the data for the x and y axes,
- `kind` : a string specifying the chart type, e.g. `line` , `bar` , `barh` , `hist` , `box` , `KDE` , `pie` , `area` lub `scatter` ,
- `figsize` : specifies the size of the chart using a tuple (width, height) in inches,
- `title` : a string specifying the title of the chart,
- `grid` : indicates whether grid lines are to be displayed,
- `xticks` : determines the pitch sequence of the x-axis,
- `yticks` : determines the y-axis pitch sequence,

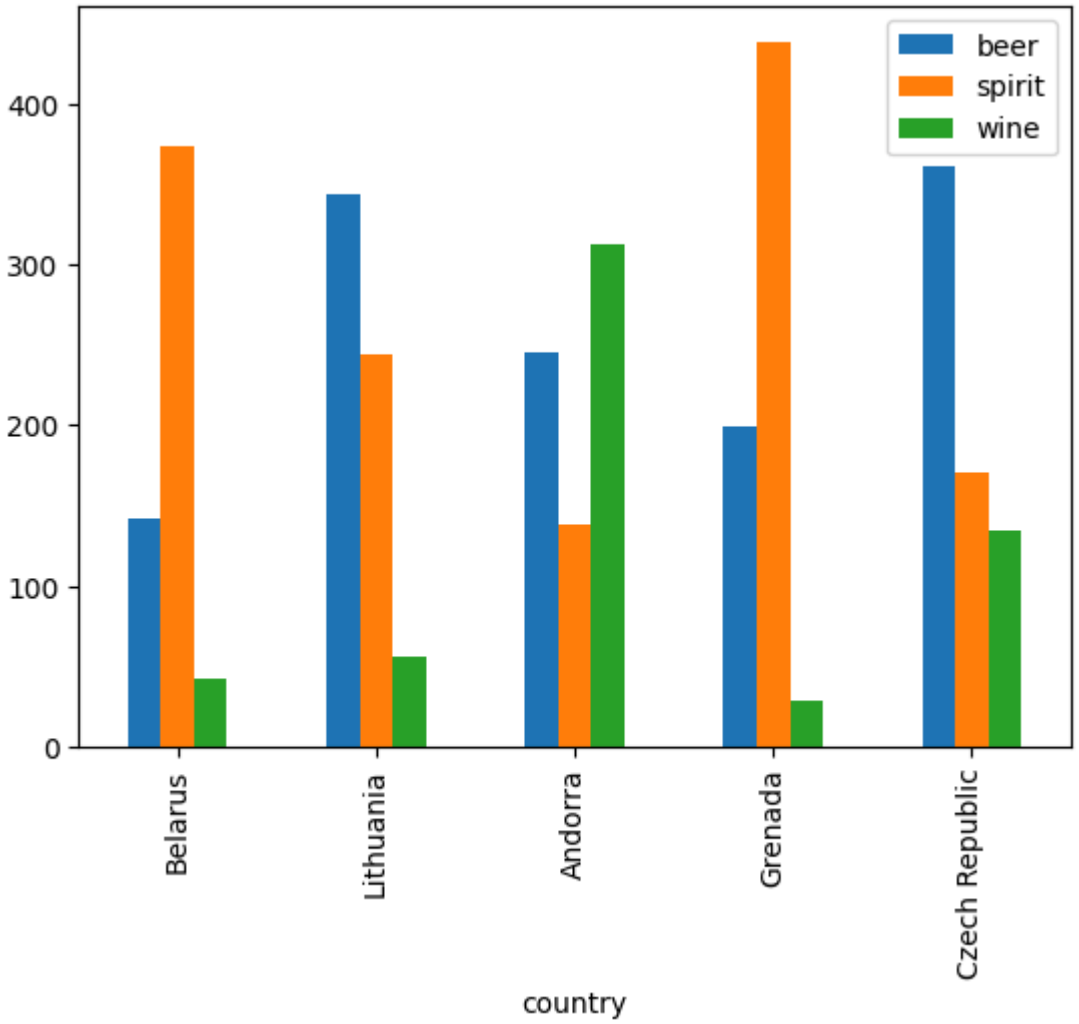
- `subplots` : determines whether subcharts are to be drawn.

In [108…
```
kon_alk = pd.read_excel('http://bartoszj.prz-rzeszow.pl/dane/drinks.xlsx')
kon_alk.columns = ['country', 'beer', 'spirit', 'wine', 'total']
kon_alk.sort_values(by='total', ascending=False, inplace=True)
kon_alk.iloc[:5]
```

Out[108…

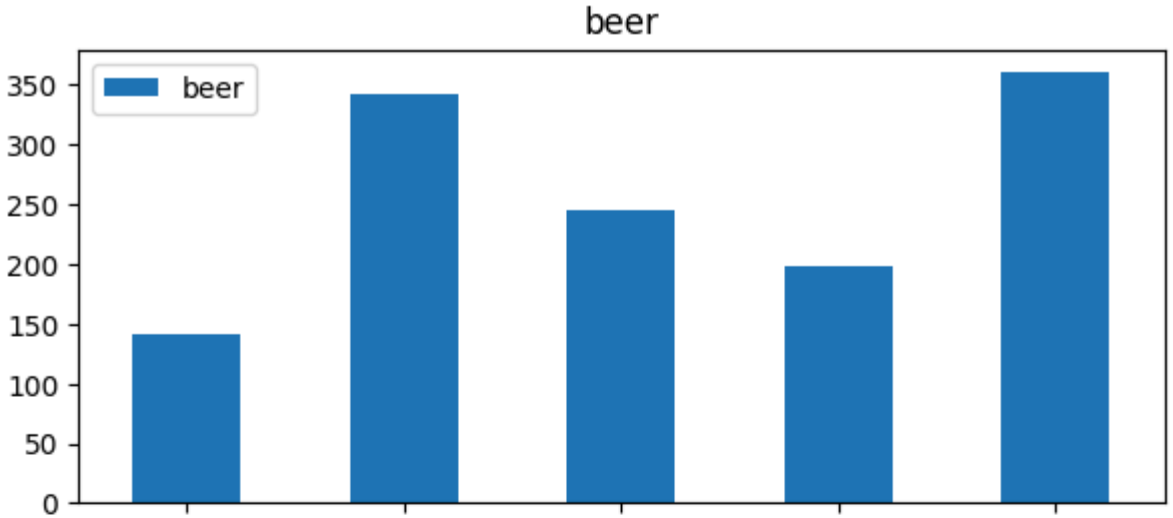
	country	beer	spirit	wine	total
15	Belarus	142	373	42	14.4
98	Lithuania	343	244	56	12.9
3	Andorra	245	138	312	12.4
68	Grenada	199	438	28	11.9
45	Czech Republic	361	170	134	11.8

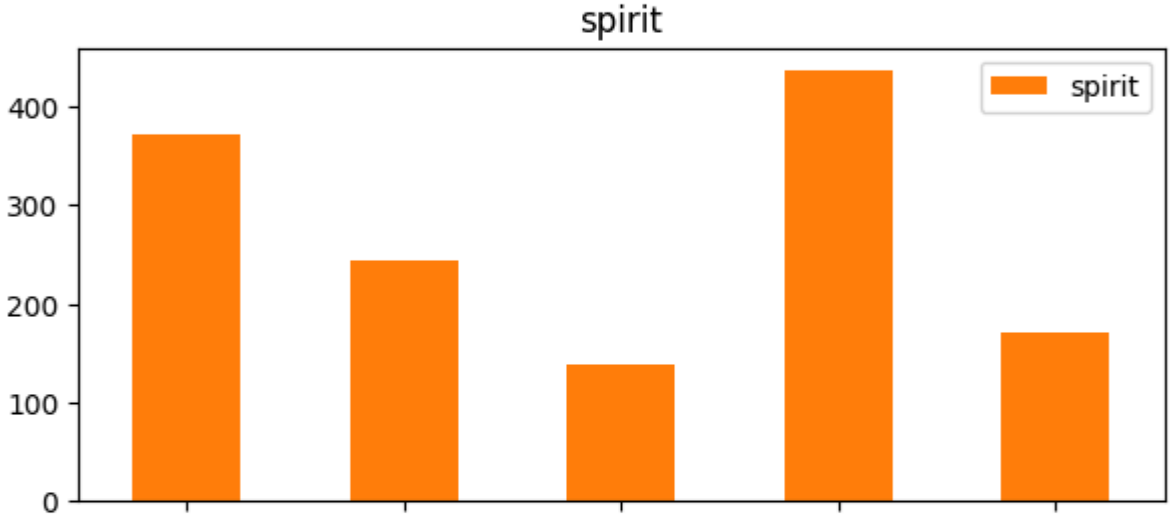
In [109…
```
_ = kon_alk.iloc[:5].plot(kind='bar', x='country', y=['beer', 'spirit', 'wine'])
```

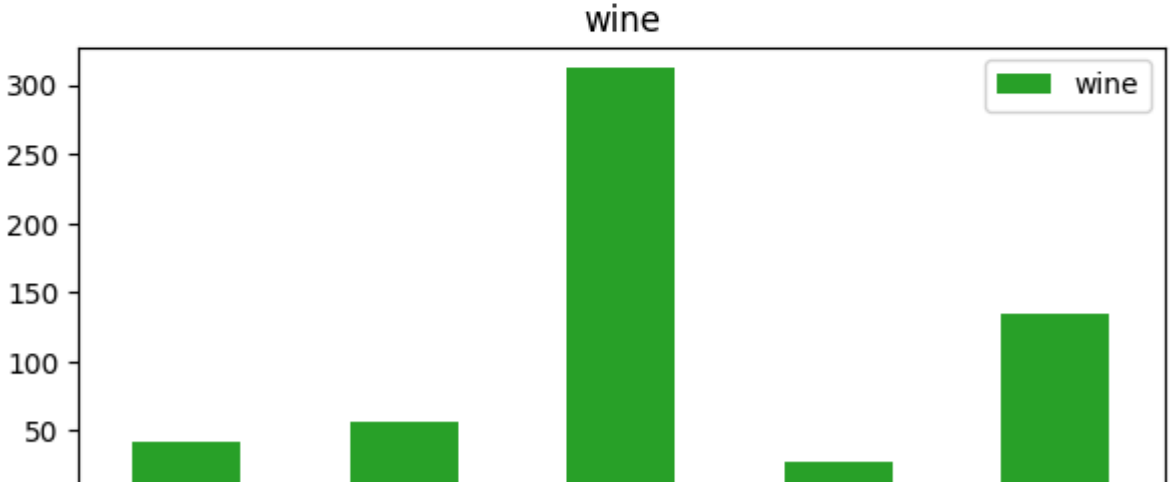


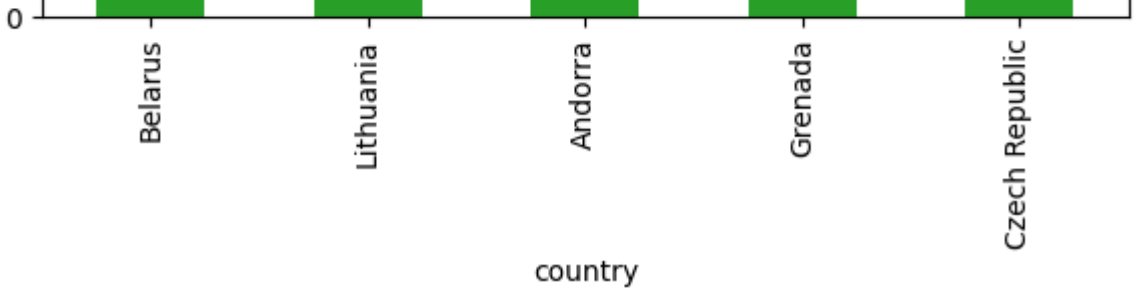
In [110…

```
_ = kon_alk.iloc[:5].plot(kind='bar', x='country', y=['beer', 'spirit', 'wine'], subplots=True, figsize=(7,10))
```


beer
beer
350
300
250
200
150
100
50
0


spirit
spirit
400
300
200
100
0


wine
wine
300
250
200
150
100
50


0
Belarus
Lithuania
Andorra
Grenada
Czech Republic
country