In [1]:
```
import pandas as pd
import numpy as np
```

Series

Она представляет из себя объект, похожий на одномерный массив, но отличительной чертой является наличие индексов. Индекс находится слева, а сам элемент справа.

Синтаксис создания:

```
pandas.Series(input_data, index, data_type)
```

- input_data: ввод в виде списка, константы, массива NumPy, Dict и т. д.
- index: значения индексов.
- data_type (опционально): тип данных.

In [2]:
```
a = pd.Series([4, 7, 6, 3, 9],
              index=['one', 'two', 'three', 'four', 'five'])
a
```

Out[2]:
```
one      4
two      7
three    6
four     3
five     9
dtype: int64
```

In [3]:
```
a = pd.Series([4, 7, 6, 3, 9])
a
```

Out[3]:
```
0    4
1    7
2    6
3    3
4    9
dtype: int64
```

In [4]:
```
a.index
```

Out[4]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [5]:
```
a.values
```

Out[5]:
```
array([4, 7, 6, 3, 9])
```

In [6]:
```
a[0]
```

Out[6]:
```
4
```

In [7]:
```
a[1]
```

Out[7]:
```
7
```

DataFrame

Объект DataFrame является табличной структурой данных. В любой таблице всегда присутствуют строки и столбцы. При этом в столбцах можно хранить данные разных типов данных. Столбцами в объекте DataFrame выступают объекты Series, строки которых являются их элементами.

Синтаксис создания:

```
pandas.DataFrame(input_data, index)
```

- input_data: ввод в виде Dict, 2D массива NumPy, Series и т. д.
- index: значения индексов.

In [8]:
```
df = pd.DataFrame({
    'Age': [46, 37, 44, 42, 42],
    'Country': ['Spain', 'Spain', 'Germany', 'Germany', 'France'],
    'Gender': ['Female', 'Female', 'Male', 'Male', 'Male']
})

df
```

Out[8]:

	Age	Country	Gender
0	46	Spain	Female
1	37	Spain	Female
2	44	Germany	Male
3	42	Germany	Male
4	42	France	Male

In [9]:
```
df['Age']
```

Out[9]:
```
0    46
1    37
2    44
3    42
4    42
Name: Age, dtype: int64
```

In [10]:
```
df.Country
```

Out[10]:
```
0      Spain
1      Spain
2    Germany
3    Germany
4     France
Name: Country, dtype: object
```

In [11]:
```
df[['Country', 'Age']]
```

Out[11]:

	Country	Age
0	Spain	46
1	Spain	37
2	Germany	44
3	Germany	42
4	France	42

In [12]:
```
df.columns
```

Out[12]:
```
Index(['Age', 'Country', 'Gender'], dtype='object')
```

In [13]:
```
df.index
```

Out[13]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [14]:
```
df = pd.DataFrame({
    'Age': [46, 37, 44, 42, 42],
    'Country': ['Spain', 'Spain', 'Germany', 'Germany', 'France'],
    'Gender': ['Female', 'Female', 'Male', 'Male', 'Male']
}, index=[5, 4, 6, 3, 2])

df
```

Out[14]:

	Age	Country	Gender
5	46	Spain	Female
4	37	Spain	Female
6	44	Germany	Male
3	42	Germany	Male
2	42	France	Male

In [15]:
```
df.index = [101, 102, 103, 104, 105]
df
```

Out[15]:

	Age	Country	Gender
101	46	Spain	Female
102	37	Spain	Female
103	44	Germany	Male
104	42	Germany	Male
105	42	France	Male

Считывание данных

В целом, pandas поддерживает все самые популярные форматы хранения данных: csv, excel, sql, html и многое другое, но чаще всего приходится работать именно с csv файлами (comma separated values).

Будем работать с датасетом по оттоку клиентов из банка https://www.kaggle.com/datasets/shubh0799/churn-modelling.

Характеристики каждого клиента:

1. RowNumber - Номер строки
2. CustomerId - Уникальный идентификатор клиента
3. Surname - Фамилия клиента
4. CreditScore - Кредитная оценка клиента
5. Geography - Из какой страны клиент
6. Gender - Пол клиента
7. Age - Возраст клиента
8. Tenure - Сколько лет человек является клиентом банка
9. Balance - Баланс счета
10. NumOfProducts - Количество открытых продуктов
11. HasCrCard - Есть ли у клиента кредитная карта
12. IsActiveMember - Является ли клиент активные участником
13. EstimatedSalary - Предположительная зарплата клиента
14. Exited - Уйдет ли человек в отток

In [16]:
```
df = pd.read_csv('./Churn_Modelling.csv')
df
```

Out[16]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0
1	2	15647311	Hill	608	Spain	Female	41	1	83807
2	3	15619304	Onio	502	France	Female	42	8	159660
3	4	15701354	Boni	699	France	Female	39	1	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369
9997	9998	15584532	Liu	709	France	Female	36	7	0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075
9999	10000	15628319	Walker	792	France	Female	28	4	130142

10000 rows × 14 columns

In [17]:
```
pd.read_csv('./Churn_Modelling.csv', header=1)
```

Out[17]:

	1	15634602	Hargrave	619	France	Female	42	2	0	1.1	1.2	1.3	1013
0	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	1125
1	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	1139
2	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	938
3	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	790
4	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	1497
...	...	...	...	...	...	...	...	...	...	...	...	...	
9994	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	962
9995	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	1016
9996	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	420
9997	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	928
9998	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	381

9999 rows × 14 columns

In [18]:
```
pd.read_csv('./Churn_Modelling.csv', sep=';')
```

Out[18]:

	RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProd
0	
1	
2	
3	
4	
...	
9995	
9996	
9997	
9998	
9999	

10000 rows × 1 columns

In [19]:
```
df
```

Out[19]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	0
1	2	15647311	Hill	608	Spain	Female	41	1	83807
2	3	15619304	Onio	502	France	Female	42	8	159660
3	4	15701354	Boni	699	France	Female	39	1	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369
9997	9998	15584532	Liu	709	France	Female	36	7	0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075
9999	10000	15628319	Walker	792	France	Female	28	4	130142

10000 rows × 14 columns

In [20]:
```
df.head()
```

Out[20]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82

In [21]:
```
df.tail()
```

Out[21]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369
9997	9998	15584532	Liu	709	France	Female	36	7	0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075
9999	10000	15628319	Walker	792	France	Female	28	4	130142

In [22]:
```
df.sample()
```

Out[22]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
2057	2058	15679550	Chukwualuka	743	France	Male	32	9	

In [23]:
```
df.sample(frac=1)
```

Out[23]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
4506	4507	15635177	Williamson	597	Spain	Female	66	3	
6087	6088	15730759	Chukwudi	561	France	Female	27	9	1356
7529	7530	15575430	Robson	579	France	Female	33	1	1183
6273	6274	15576935	Ampt	743	Spain	Male	43	2	1618
838	839	15585888	Nwokezuike	553	Spain	Female	48	3	
...	...	...	...	...	...	...	...	...	
439	440	15690134	Hughes	464	Germany	Female	42	3	856
2314	2315	15756056	Ku	561	Spain	Female	28	3	
8129	8130	15729246	Hardacre	847	Spain	Male	31	5	
5459	5460	15617507	Wilson	530	Spain	Female	36	7	
1677	1678	15801767	Yin	784	Spain	Female	40	8	

10000 rows × 14 columns

In [26]:
```
df.sample(frac=0.5)
```

Out[26]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
4510	4511	15657747	Zito	611	Germany	Female	43	9	127216.
7244	7245	15670029	Marcelo	445	France	Female	33	7	0.
4463	4464	15778975	Nnonso	850	Germany	Female	70	1	96947.
8997	8998	15631063	Trentino	710	France	Female	33	2	0.
9465	9466	15815259	Fang	835	France	Female	56	2	0.
...	...	...	...	...	...	...	...	...	
2943	2944	15639277	Lin	678	France	Female	41	9	0.
3359	3360	15747878	Aiken	739	Spain	Male	60	4	0.
29	30	15656300	Lucciano	411	France	Male	29	0	59697.
659	660	15603065	Grubb	751	France	Female	30	6	0.
1948	1949	15569187	Fleming	680	Spain	Male	35	9	0.

5000 rows × 14 columns

In [24]:
```
df.shape
```

Out[24]:
```
(10000, 14)
```

Первичный анализ данных

Типы данных:

- int: целочисленные значения. Пример: 9, 56, 30

- float: вещественные значения (с плавающей точкой). Пример: 7.3, 9.0, 45.334
- object/str: строковые значения. Пример: 'hello, world', '50 000'

In [80]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Выводятся значения:

- Count - количество непропущенных объектов (там, где нет nan значений)
- mean - арифметическое среднее
- std - стандартное отклонение
- min - минимальное значение
- 25% - квантиль 25 процентов
- 50% - квантиль 50 процентов или же медиана
- 75% - квантиль 75 процентов
- max - максимальное значение

In [81]:
```
df.describe()
```

Out[81]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfP
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4

In [82]:
```
df['Age'].min()
```

Out[82]:
```
18
```

In [83]:
```
df['Balance'].max()
```

Out[83]:
```
250898.09
```

In [84]:
```
df[['CreditScore', 'Age', 'Tenure']].mean()
```

Out[84]:
```
CreditScore    650.5288
Age             38.9218
Tenure           5.0128
dtype: float64
```

Получаем 4 значения:

- count - количество непропущенных объектов
- unique - количество уникальных значений
- top - самое частотное значение (мода)
- freq - частота появления самого частотного значения

In [92]:
```
df.describe(include=['object'])
```

Out[92]:

	Surname	Geography	Gender
count	10000	10000	10000
unique	2932	3	2
top	Smith	France	Male
freq	32	5014	5457

In [85]:
```
df.dtypes
```

Out[85]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [86]:
```
df['Age'].dtype
```

Out[86]:
```
dtype('int64')
```

In [87]:
```
df['HasCrCard'].astype('bool')
```

Out[87]:
```
0        True
1       False
2        True
3       False
4        True
        ...
9995     True
9996     True
9997    False
9998     True
9999     True
Name: HasCrCard, Length: 10000, dtype: bool
```

In [88]:
```
df['HasCrCard'].dtype
```

Out[88]:
```
dtype('int64')
```

In [89]:
```
df['HasCrCard'] = df['HasCrCard'].astype('bool')
```

In [90]:
```
df['HasCrCard'].dtype
```

Out[90]:
```
dtype('bool')
```

In [93]:
```
df['Geography'].unique()
```

Out[93]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

In [94]:
```
df['Geography'].nunique()
```

Out[94]:
```
3
```

In [95]:
```
df['Geography'].value_counts()
```

Out[95]:
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

In [96]:
```
df['Geography'].value_counts(normalize=True)
```

Out[96]:
```
France     0.5014
Germany    0.2509
Spain      0.2477
Name: Geography, dtype: float64
```

Фильтрация

Фильтрация в pandas основывается на булевых масках.

> **Булевая маска** — бинарные данные, которые используются для выбора определенных объектов из структуры данных.

In [98]:
```
df['Gender'] == 'Male'
```

Out[98]:
```
0       False
1       False
2       False
3       False
4       False
        ...
9995     True
9996     True
9997    False
9998     True
9999    False
Name: Gender, Length: 10000, dtype: bool
```

In [100]:
```
male = df[df['Gender'] == 'Male']
male
```

Out[100]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	B
5	6	15574012	Chu	645	Spain	Male	44	8	113
6	7	15592531	Bartlett	822	France	Male	50	7	
8	9	15792365	He	501	France	Male	44	4	142
9	10	15592389	H?	684	France	Male	27	2	134
10	11	15767821	Bearce	528	France	Male	31	6	102
...	...	...	...	...	...	...	...	...	
9992	9993	15657105	Chukwualuka	726	Spain	Male	36	2	
9993	9994	15569266	Rahman	644	France	Male	28	7	155
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75

5457 rows × 14 columns

Логические И

При операторе & нужно, чтобы выполнялось два условия одновременно:

In [101]:
```
df[(df['Gender'] == 'Female') & (df['NumOfProducts'] >= 3)]
```

Out[101]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
2	3	15619304	Onio	502	France	Female	42	8
7	8	15656148	Obinna	376	Germany	Female	29	4
30	31	15589475	Azikiwe	591	Spain	Female	39	3
88	89	15622897	Sharpe	646	France	Female	46	4
90	91	15757535	Heap	647	Spain	Female	44	5
...	...	...	...	...	...	...	...	...
9565	9566	15752294	Long	582	France	Female	38	9
9747	9748	15775761	Iweobiegbunam	610	Germany	Female	69	5
9800	9801	15640507	Li	762	Spain	Female	35	3
9877	9878	15572182	Onwuamaeze	505	Germany	Female	33	3
9895	9896	15796764	Bruno	684	Germany	Female	56	3

187 rows × 14 columns

Логические ИЛИ

При операторе | нужно, чтобы выполнялось хотя бы одно условие:

In [103]:
```
df[(df['HasCrCard']) | (df['NumOfProducts'] >= 3)]
```

Out[103]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	0
2	3	15619304	Onio	502	France	Female	42	8	159660
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510
5	6	15574012	Chu	645	Spain	Male	44	8	113755
6	7	15592531	Bartlett	822	France	Male	50	7	0
...	...	...	...	...	...	...	...	...	
9993	9994	15569266	Rahman	644	France	Male	28	7	155060
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075
9999	10000	15628319	Walker	792	France	Female	28	4	130142

7150 rows × 14 columns

Логические НЕ

При операторе ~ булевая маска обращается: True меняется на False и наоборот:

In [104]:
```
df[~(df['Geography'] == 'Spain')]
```

Out[104]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0
2	3	15619304	Onio	502	France	Female	42	8	159660
3	4	15701354	Boni	699	France	Female	39	1	0
6	7	15592531	Bartlett	822	France	Male	50	7	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115046
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369
9997	9998	15584532	Liu	709	France	Female	36	7	0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075
9999	10000	15628319	Walker	792	France	Female	28	4	130142

7523 rows × 14 columns

In [106]:
```
df[df['Geography'].isin(['France', 'Germany'])]
```

Out[106]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0
2	3	15619304	Onio	502	France	Female	42	8	159660
3	4	15701354	Boni	699	France	Female	39	1	0
6	7	15592531	Bartlett	822	France	Male	50	7	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115046
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369
9997	9998	15584532	Liu	709	France	Female	36	7	0
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075
9999	10000	15628319	Walker	792	France	Female	28	4	130142

7523 rows × 14 columns

Индексация

In [112]:
```
df_small = df[(df['Geography'] == 'Spain')][['Geography', 'Gender', 'Age']]
df_small.head()
```

Out[112]:

	Geography	Gender	Age
1	Spain	Female	41
4	Spain	Female	43
5	Spain	Male	44
11	Spain	Male	24
14	Spain	Female	35

loc

In [113]:
```
df_small.loc[1]
```

Out[113]:
```
Geography     Spain
Gender       Female
Age              41
Name: 1, dtype: object
```

In [114]:
```
df_small.loc[3]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexes/base.py in
get_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get
_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get
_item()

KeyError: 3

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-114-826e5ec6c72e> in <module>
----> 1 df_small.loc[3]

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexing.py in __g
etitem__(self, key)
   1422
   1423             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1424             return self._getitem_axis(maybe_callable, axis=axis)
   1425
   1426     def _is_scalar_access(self, key: Tuple):

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexing.py in _ge
titem_axis(self, key, axis)
   1848         # fall thru to straight lookup
   1849         self._validate_key(key, axis)
-> 1850         return self._get_label(key, axis=axis)
   1851
   1852

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexing.py in _ge
t_label(self, label, axis)
    158             raise IndexingError("no slices here, handle elsewhere")
    159
--> 160         return self.obj._xs(label, axis=axis)
    161
    162     def _get_loc(self, key: int, axis: int):

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/generic.py in xs(s
elf, key, axis, level, drop_level)
   3735             loc, new_index = self.index.get_loc_level(key, drop_level=drop_l
evel)
   3736         else:
-> 3737             loc = self.index.get_loc(key)
   3738
   3739             if isinstance(loc, np.ndarray):

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexes/base.py in
get_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
```

```
   2900         indexer = self.get_indexer([key], method=method, tolerance=toleranc
e)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get
_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get
_item()

KeyError: 3
```

In [119]:
```
df_small.loc[[1, 4, 5], ['Gender', 'Age']]
```

Out[119]:

	Gender	Age
1	Female	41
4	Female	43
5	Male	44

iloc

In [121]:
```
df_small.head()
```

Out[121]:

	Geography	Gender	Age
1	Spain	Female	41
4	Spain	Female	43
5	Spain	Male	44
11	Spain	Male	24
14	Spain	Female	35

In [123]:
```
df_small.iloc[[0, 1, 2]]
```

Out[123]:

	Geography	Gender	Age
1	Spain	Female	41
4	Spain	Female	43
5	Spain	Male	44

In [125]:
```
df_small.iloc[2500]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-125-fdbc3008acb9> in <module>
----> 1 df_small.iloc[2500]

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexing.py in __getitem__(self, key)
   1422
   1423             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1424             return self._getitem_axis(maybe_callable, axis=axis)
   1425
   1426     def _is_scalar_access(self, key: Tuple):

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexing.py in _getitem_axis(self, key, axis)
   2155
   2156             # validate the location
-> 2157             self._validate_integer(key, axis)
   2158
   2159             return self._get_loc(key, axis=axis)

~/teacher_geekbrains/venv/lib/python3.8/site-packages/pandas/core/indexing.py in _validate_integer(self, key, axis)
   2086         len_axis = len(self.obj._get_axis(axis))
   2087         if key >= len_axis or key < -len_axis:
-> 2088             raise IndexError("single positional indexer is out-of-bounds")
   2089
   2090     def _getitem_tuple(self, tup):

IndexError: single positional indexer is out-of-bounds
```

In [127]:
```
df_small.iloc[0, [0, 2]]
```

Out[127]:
```
Geography    Spain
Age             41
Name: 1, dtype: object
```

Сортировки

In [128]:
```
df.sort_values('Age')
```

Out[128]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
3512	3513	15657779	Boylan	806	Spain	Male	18	3	
1678	1679	15569178	Kharlamov	570	France	Female	18	4	82
3517	3518	15757821	Burgess	771	Spain	Male	18	1	
9520	9521	15673180	Onyekaozulu	727	Germany	Female	18	2	93
2021	2022	15795519	Vasiliev	716	Germany	Female	18	3	128
...	...	...	...	...	...	...	...	...	
3387	3388	15798024	Lori	537	Germany	Male	84	8	92
3033	3034	15578006	Yao	787	France	Female	85	10	
2458	2459	15813303	Rearick	513	Spain	Male	88	10	
6759	6760	15660878	T'ien	705	France	Male	92	1	126
6443	6444	15764927	Rogova	753	France	Male	92	3	121

10000 rows × 14 columns

In [129]:
```
df.sort_values('Age', ascending=False)
```

Out[129]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
6443	6444	15764927	Rogova	753	France	Male	92	3	121
6759	6760	15660878	T'ien	705	France	Male	92	1	126
2458	2459	15813303	Rearick	513	Spain	Male	88	10	
3033	3034	15578006	Yao	787	France	Female	85	10	
3387	3388	15798024	Lori	537	Germany	Male	84	8	92
...	...	...	...	...	...	...	...	...	
9782	9783	15728829	Weigel	509	France	Male	18	7	102
2141	2142	15758372	Wallace	674	France	Male	18	7	
9501	9502	15634146	Hou	835	Germany	Male	18	2	142
9520	9521	15673180	Onyekaozulu	727	Germany	Female	18	2	93
1619	1620	15770309	McDonald	656	France	Male	18	10	151

10000 rows × 14 columns

In [130]:
```
df.sort_values(['Age', 'CreditScore'])
```

Out[130]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
9782	9783	15728829	Weigel	509	France	Male	18	7
1678	1679	15569178	Kharlamov	570	France	Female	18	4
9029	9030	15722701	Bruno	594	Germany	Male	18	1
7334	7335	15759133	Vaguine	616	France	Male	18	6
9526	9527	15665521	Chiazagomekpele	642	Germany	Male	18	5
...	...	...	...	...	...	...	...	...
3387	3388	15798024	Lori	537	Germany	Male	84	8
3033	3034	15578006	Yao	787	France	Female	85	10
2458	2459	15813303	Rearick	513	Spain	Male	88	10
6759	6760	15660878	T'ien	705	France	Male	92	1
6443	6444	15764927	Rogova	753	France	Male	92	3

10000 rows × 14 columns