## 1) Лекция 8. Анализ данных с библиотекой Pandas

## 2) Краткая цель лекции:

a) Узнать, как анализировать табличные данные

b) Научиться считать статистики датафрейма

c) Более детально изучить фильтрацию данных

d) Разобраться с сортировками

## 3) Дополнительные материалы к лекции

(ссылки, видео, файлы и т.п.)

## 4) Краткая выжимка, о чём говорилось в предыдущей лекции (если она была), 2-3 предложения

На предыдущей лекции узнали, как сделать работу с jupyter notebook эффективной. Вспомнили устройство словарей и функций. А также изучили генераторы, модель random и list comprehensions.

## 5) Термины лекции

**Объект** — это набор данных (переменных) и методов (функций), которые с этими данными взаимодействуют.

**Булевая маска** — бинарные данные, которые используются для выбора определенныхобъектов изструктуры данных.

## 6) Краткий план лекции

(Может собираться на основе подзаголовков из текста лекции, как содержание)

1. Библиотека Pandas
2. Объекты Series и DataFrame
3. Считывание данных
4. Отображение данных в pandas
5. Первичный анализ данных
6. Фильтрация
   1. Логическое И
   2. Логическое ИЛИ
   3. Логическое НЕ
7. Индексация
   1. loc
   2. iloc
8. Сортировки

## 7) Подробный текст лекции

Сегодня будем анализировать табличные данные с помощью библиотеки Pandas.

### Библиотека Pandas

Для начала разберемся, а что такое библиотека Pandas и каким функционалом она обладает. Pandas - наиболее продвинутая библиотека для обработки и анализа данных. Это некий аналог Microsoft Excel или Google Sheets для языка программирования Python. Но в Excel и Sheets есть ограничения на количество строк/количество ячеек, поэтому большие датасеты там не проанализируешь, благо есть Python, на котором можно анализировать очень большие массивы данных, ограничение одно - оперативная память машины, на которой работаете.

|  | Google Sheets | Microsoft Excel | Pandas |
| --- | --- | --- | --- |
| Ограничение по данным | 4 989 894 ячеек | 1 048 576 строк и 16 384 столбцов | Нет ограничений |
| Стоимость | Бесплатно | 8.25$ в месяц | Бесплатно |

Глядя на таблицу выше, делаем вывод, что Pandas подходит лучше для задач аналитика, т.к. нет ограничений по данным.

Чтобы начать работу с функциями и классами из библиотеки:

* В командной строке нужно написать *pip install pandas* (если работаете не через анаконду)
* В командной строке нужно написать *conda install pandas* (если работаете через анаконду)
* Импортировать себе в ноутбук библиотеку через import pandas as pd:



### Объекты Series и DataFrame

***Объект*** — это набор данных (переменных) и методов (функций), которые с этими данными взаимодействуют.

Чтобы эффективно работать с pandas, необходимо освоить самые главные структуры данных библиотеки: DataFrame и Series.

Series в pandas можно сравнить с одной характеристикой клиента/предмета/объекта. К примеру, у нас задача по анализу рекламной кампании и возраст людей, которые участвовали в ней - это и есть одна характеристика, которую можно представить через Series.

DataFrame - это уже полноценная таблица. Это все сведения, которые мы собирали на протяжении всей рекламной кампании: возраст клиентов, время показа рекламы, рекламный заголовок и многие другие.

#### Series

Теперь давайте обсудим, а какими объектами оперирует библиотека Pandas.

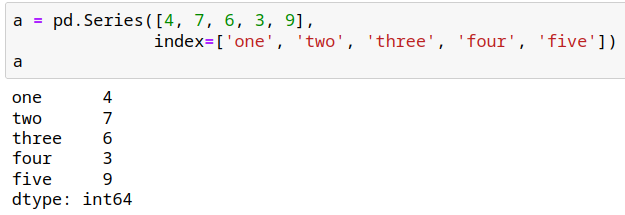
Первый - это pandas Series. Она представляет из себя объект, похожий на одномерный массив, но отличительной чертой является наличие индексов. Индекс находится слева, а сам элемент справа.

**Синтаксис создания:**

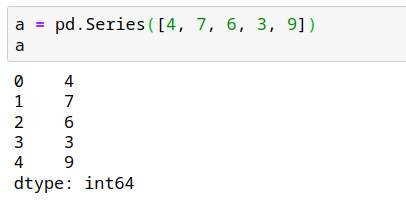
pandas.Series(input\_data, index, data\_type)

* input\_data: ввод в виде списка, константы, массива NumPy, Dict и т. д.
* index: значения индексов.
* data\_type (опционально): тип данных.

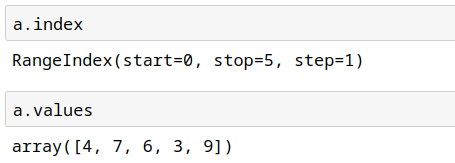
Создадим небольшую Series по синтаксису выше:



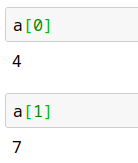
Если индекс явно не задан, то pandas автоматически создаёт индексы от 0 до N-1, где N - общее количество элементов:



У объекта Series есть атрибуты, через которые можно получить список элементов и индексы, это *values* и *index* соответственно:



Получить элемент из Series можно по его индексу:



#### 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:**

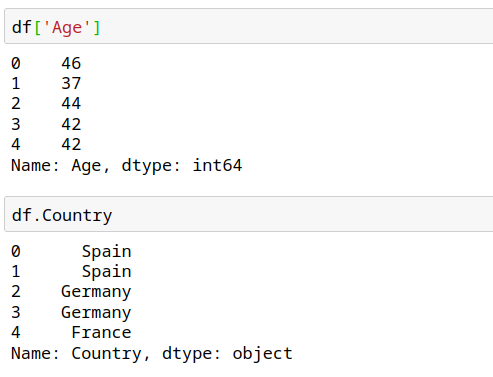
pandas.DataFrame(input\_data, index)

* input\_data: ввод в виде Dict, 2D массива NumPy, Series и т. д.
* index: значения индексов.

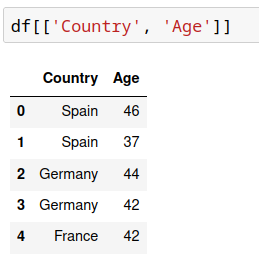
DataFrame проще всего создать, используя питоновский словарь:



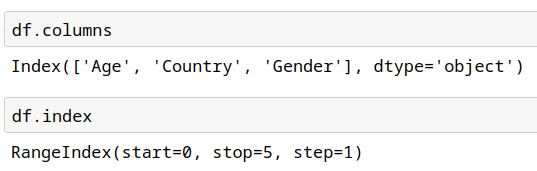
Чтобы убедиться, что столбец в DataFrame это Series, можем извлечь любой столбец, используя квадратные скобки или точку:



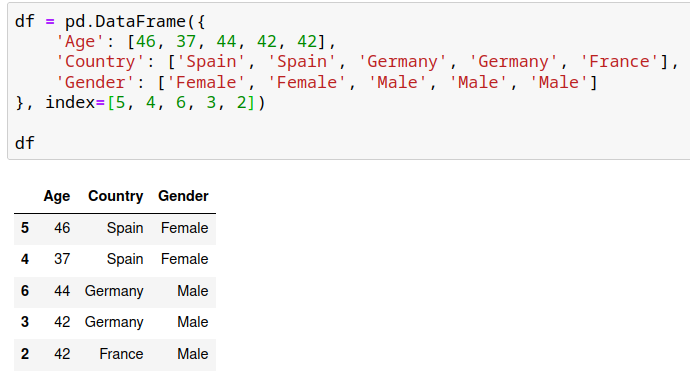
А если захотим извлечь больше, чем один признак, то получим снова DataFrame:



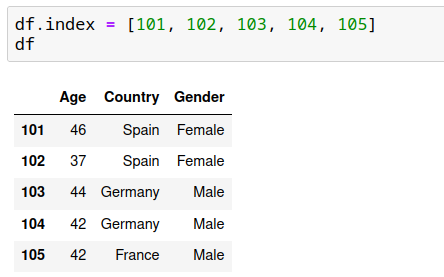
У объекта DataFrame есть названия столбцов и названия индексов, чтобы их получить нужно воспользоваться атрибутами columns и index:



Индекс по строкам можно задать разными способами, например, при формировании самого объекта DataFrame:



Или же поменять атрибут index уже при работе с датафреймом:



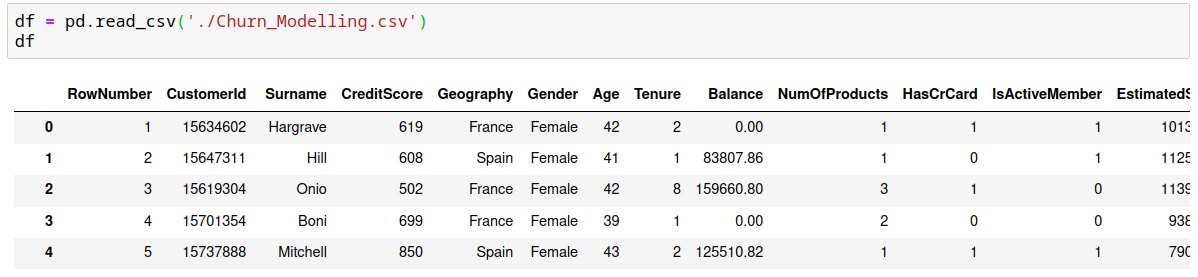
### Считывание данных

С базовыми объектами, которыми оперирует pandas разобрались, теперь давайте возьмем датасет и проведем его анализ.

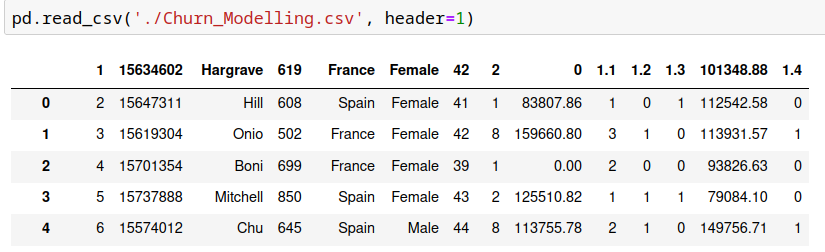
В целом, pandas поддерживает все самые популярные форматы хранения данных: csv, excel, sql, html и многое другое, но чаще всего приходится работать именно с csv файлами (comma separated values).

Будем работать с набором данных по оттоку клиентов из банка <https://www.kaggle.com/datasets/shubh0799/churn-modelling>. Нам нужно провести аналитику и понять, почему люди решают покинуть наш банк.

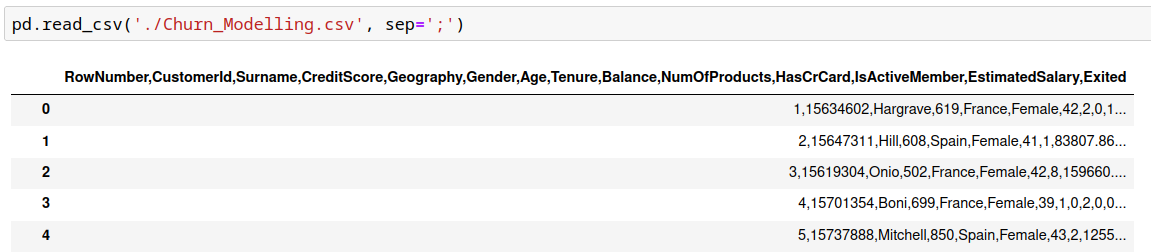
Считать данные из csv-файла и превратить в DataFrame можно функцией read\_csv:



Аргумент header указывает названия столбцов датафрейма, по умолчанию header=0, значит значения шапки таблицы формируются из нулевой строки файла, а если, к примеру, указать header=1, то значения шапки таблицы будут формироваться из первой строки файла:

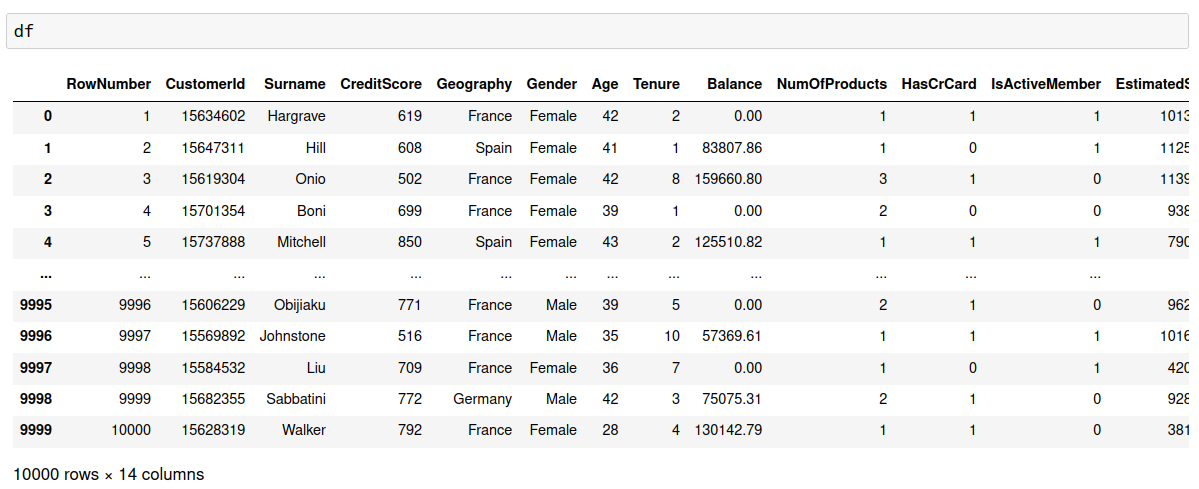


Аргумент *sep* указывает разделитесь столбцов, поставим, к примеру sep=’;’, в этом случае pandas будет искать символ **;**, чтобы разбить столбцы, но ничего не найдет, поэтому все значения сольются воедино:

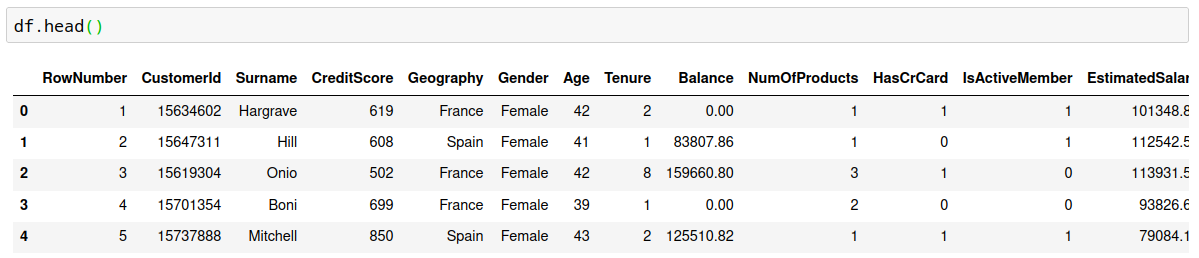


### Отображение данных в pandas

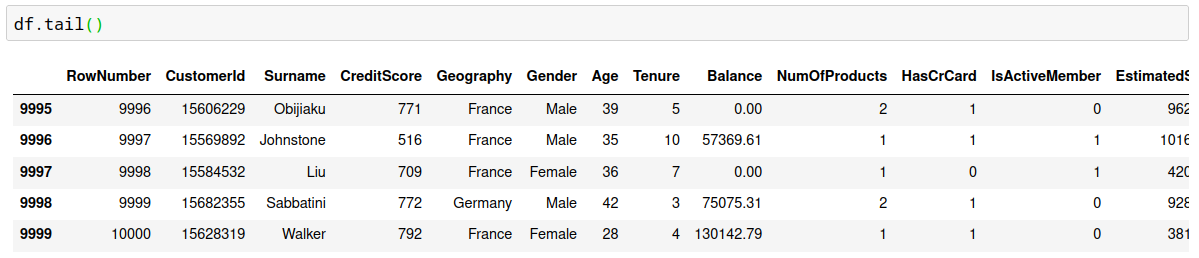
Чтобы отобразить весь датафрейм на экран, можно написать название переменной df, тогда вся таблица отобразится:



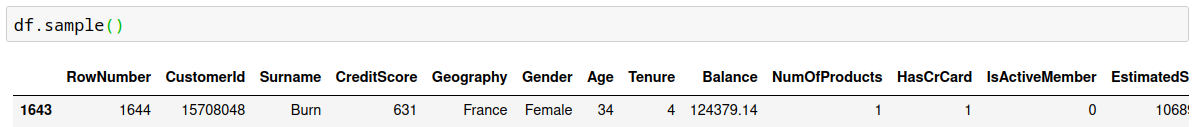
Можно воспользоваться методом head(), чтобы отобразить ограниченное количество строк с начала таблицы (по дефолту 5):



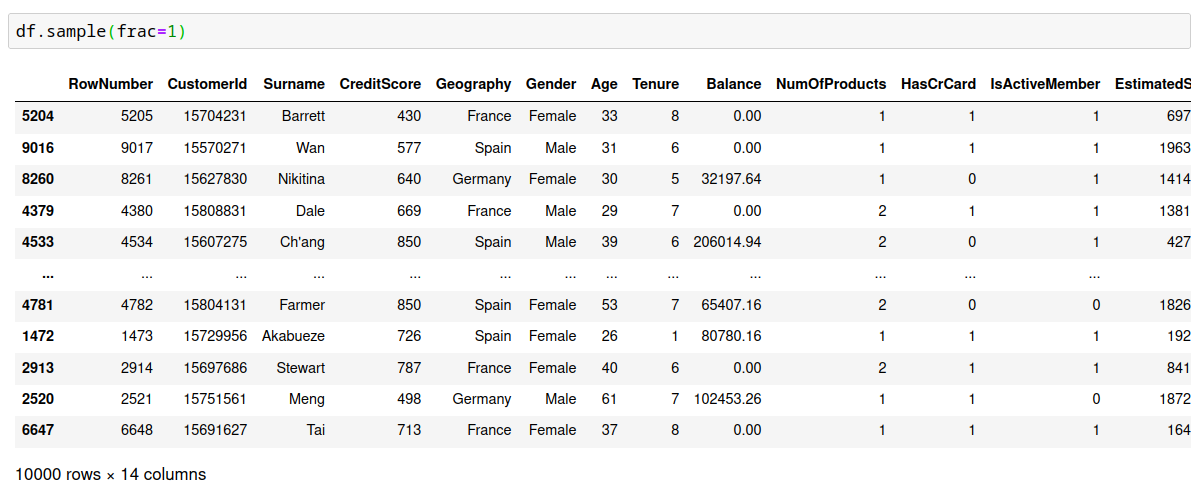
А метод tail() возвращает по умолчанию 5 строк с конца таблицы:



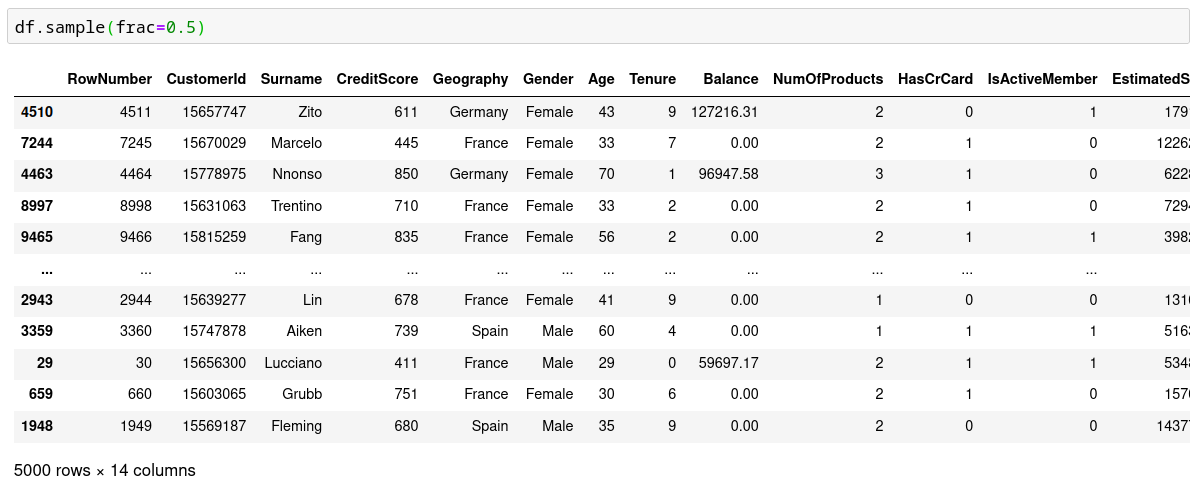
Метод sample() вернет один случайный объект:



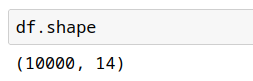
А если в этот метод передать атрибут frac, то можно возвращать долю объектов в случайном порядке. Атрибут frac может принимать значения от 0 до 1. К примеру, можно поставить frac=1, тогда вернутся все объекты, но в случайном порядке:



А если, к примеру, указать значение frac=0.5, то вернется половина значений (5000 записей):



Чтобы узнать, сколько есть строк и столбцов можно вызвать атрибут shape - это будет кортеж из двух значений, первое - количество строк, второе - количество столбцов:

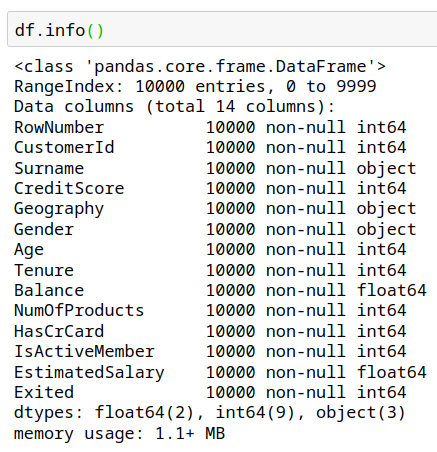


### Первичный анализ данных

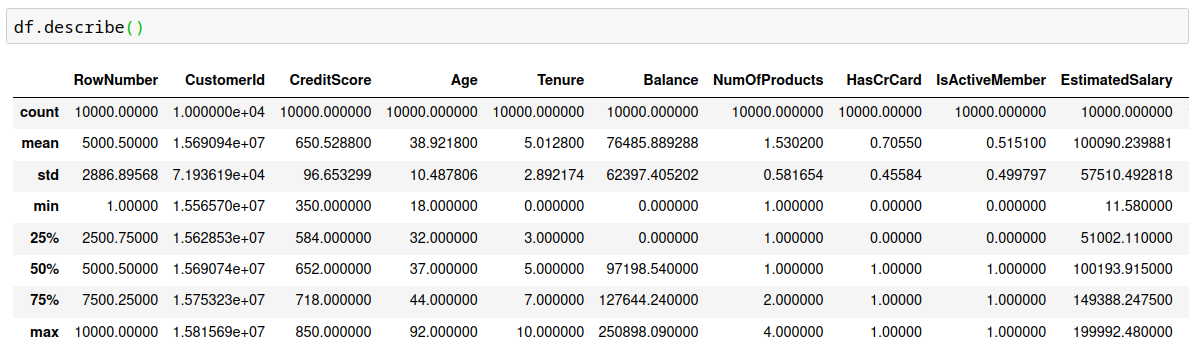
Переходим к аналитике, первым делом можно вывести сводную таблицу по типам данных, по количеству непропущенных объектов и по потреблению памяти в таблице через метод info().

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’



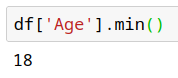
Основные статистики можно получить через метод describe():



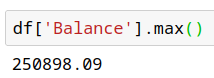
Выводятся значения:

1. Count - количество непропущенных объектов (там, где нет nan значений)
2. mean - арифметическое среднее
3. std - стандартное отклонение
4. min - минимальное значение
5. 25% - квантиль 25 процентов
6. 50% - квантиль 50 процентов или же медиана
7. 75% - квантиль 75 процентов
8. max - максимальное значение

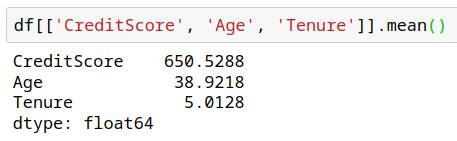
При этом данные статистики можно выводить и по отдельности, к примеру возьмем минимальный возраст клиента:



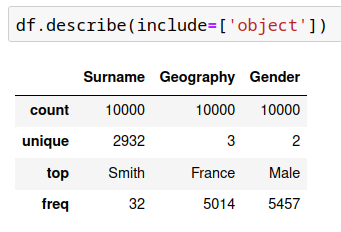
Или же максимальный баланс на счете:



А можем подсчитать статистику сразу на нескольких признаках:



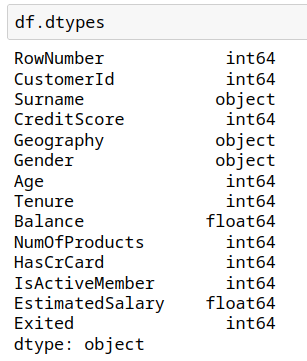
Особо внимательные могли заметить, что в статистики describe не было некоторых признаков, а все так вышло, потому что describe() по умолчанию считает статистики только для вещественных признаков, а строковые игнорирует, чтобы describe посчитал на них статистики, нужно явно это указать:



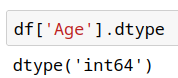
Получаем 4 значения:

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

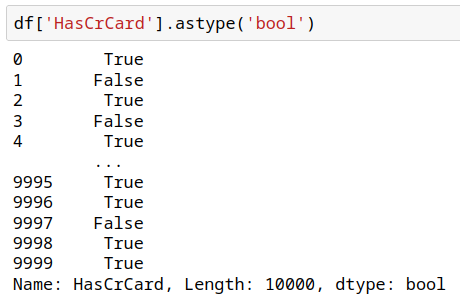
Если интересно поизучать типы данных, то воспользуйтесь атрибутом dtypes, который вызывается у датафрейма. Изучение типов данных помогает понять, с какими характеристиками имеем дело, есть ли здесь строковые характеристики или же мы работам только с численными показателями:



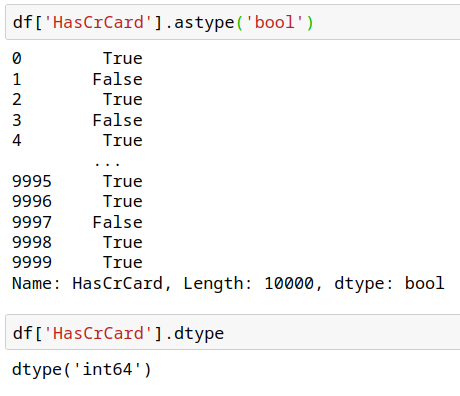
Или же, если интересует тип только одного столбца, то смотрите атрибут dtype:



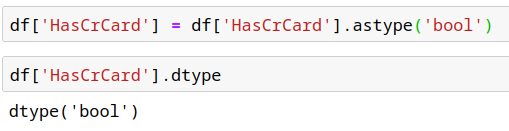
Типы данных можно менять через атрибут astype():



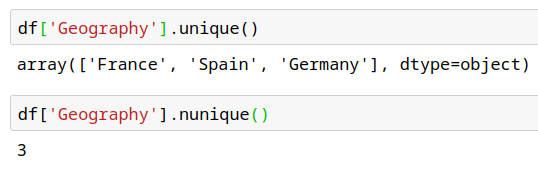
Но если посмотрим на тип данных, то он не поменялся, остался int:



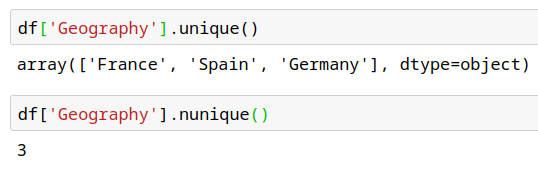
Чтобы изменения вступили в силу нужно переопределить признак:



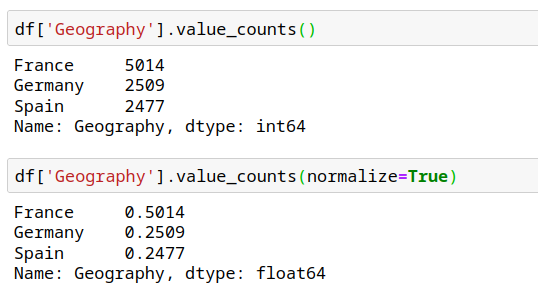
Чтобы найти уникальные значения в признаке можно вызвать метод unique():



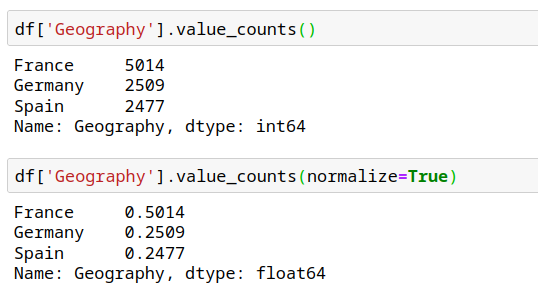
А найти их количество через nunique():



value\_counts() выведет уникальные значения и их частоту появления:



А если вызвать value\_counts() с атрибутом normalize=True, то частотность будет нормированная:



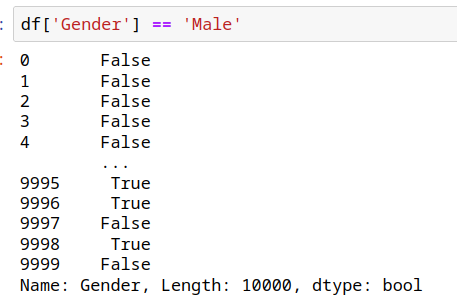
### Фильтрация

Переходим к фильтрациям, чтобы с помощью них можно было бы изучать отдельные группы клиентов.

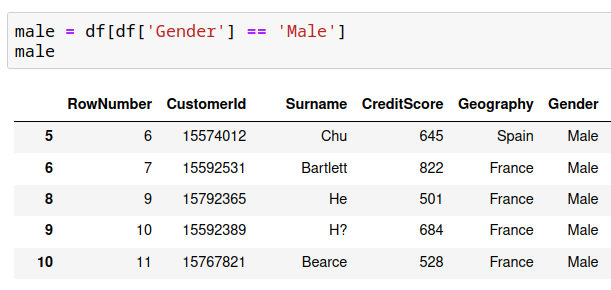
Фильтрация в pandas основывается на булевых масках.

**Булевая маска** — бинарные данные, которые используются для выбора определенныхобъектов изструктуры данных.

Ниже видим пример булевой маски, если стоит значение False, то значение в этой строке не ‘Male’, а если стоит значение True, то объект принимает значение ‘Male’:



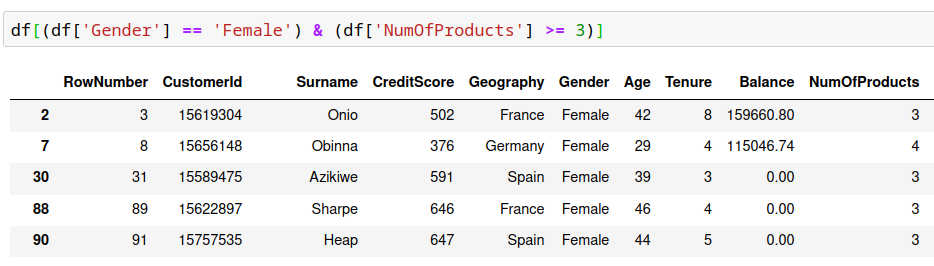
Данную маску можно передать в датафрейм и на выходе мы получим только людей, у которых признак Gender == ‘Male’:



При этом условия можно объединять с помощью логических операторов.

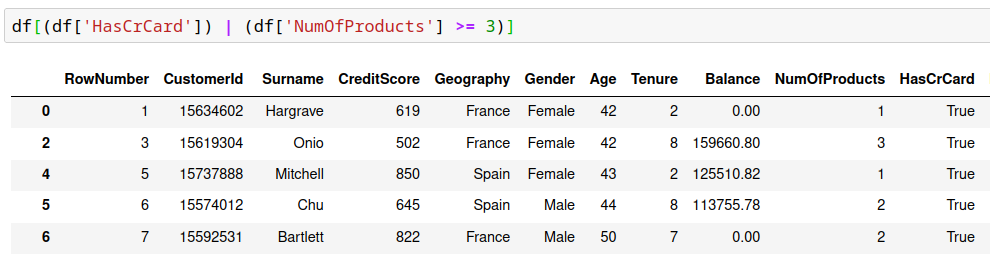
#### Логическое И

При операторе & нужно, чтобы выполнялось два условия одновременно:



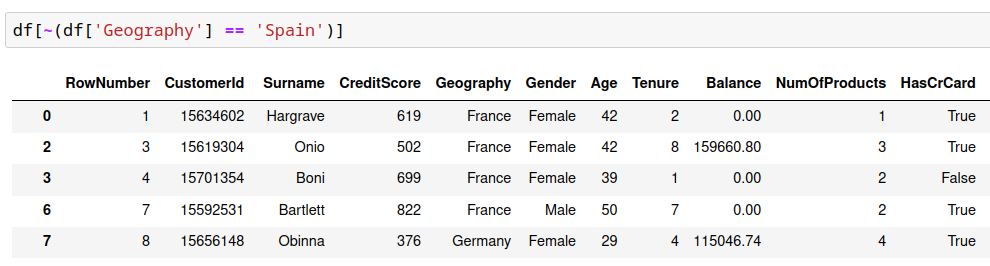
#### Логическое ИЛИ

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

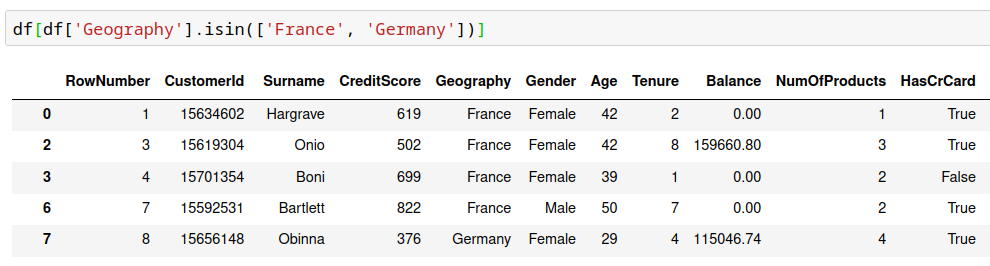


#### Логическое НЕ

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



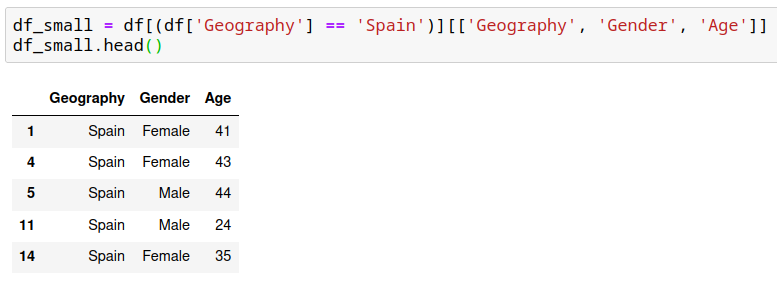
Данное условие могли переписать через метод isin():



### Индексация

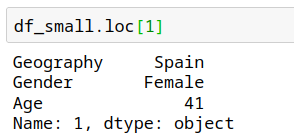
Порой бывает полезно получать объект по его индексу, чтобы проверить правильность измененных данных, чтобы смотреть не на всю большую таблицу, а на один объект из таблицы и в pandas это можно сделать двумя способами.

Для наглядности возьмем датасет поменьше:

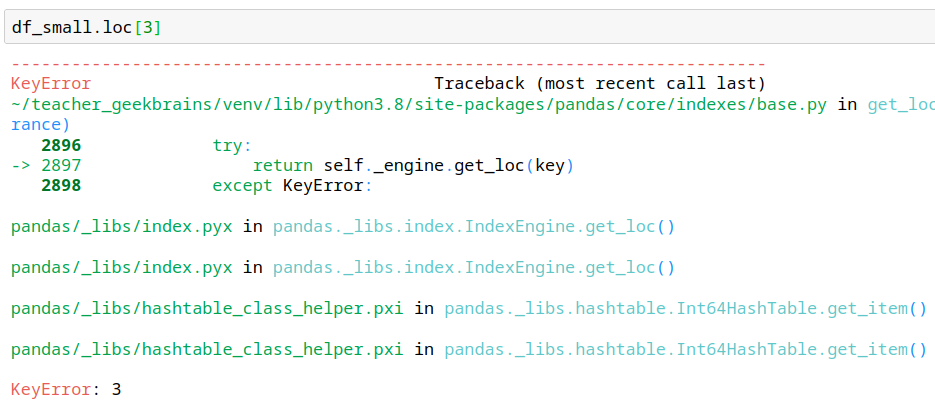


#### loc

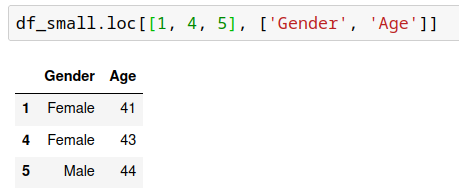
Данный метод позволяет взять объект по конкретному ключу строки или столбца:



Если такого ключа нет в данных, то будет ошибка KeyError:

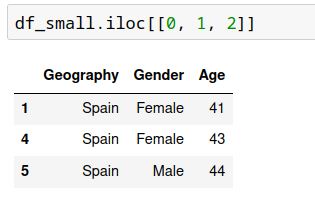


Помимо ключей строки, можно передавать и ключи столбцов:

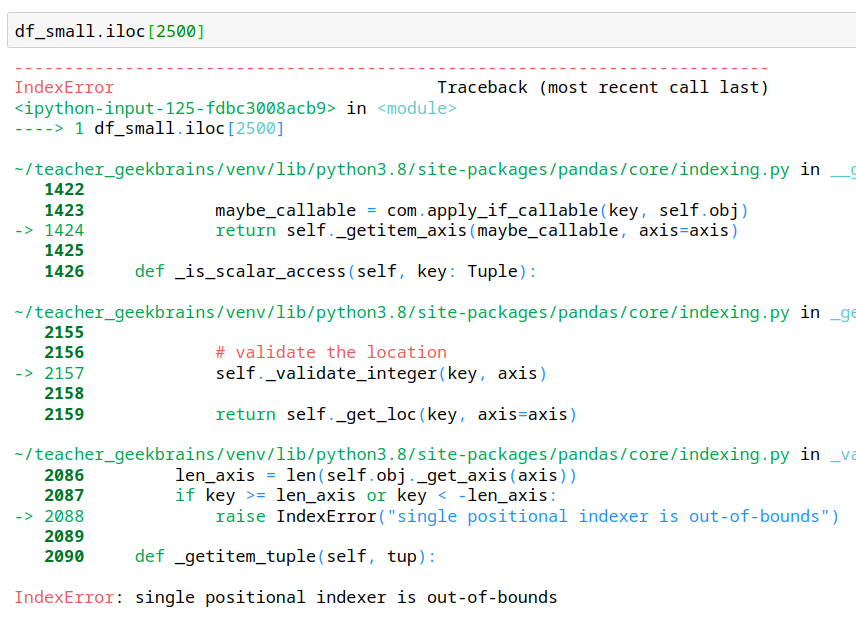


#### iloc

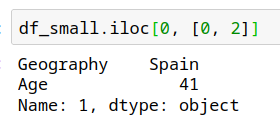
Данный метод позволяет взять объект по порядковому ключу строки или столбца. И без разницы, какие на самом деле лежат значения ключей в этих строках. В нашем примере есть ключи по индексам 1, 4, 5, но через iloc можем их достать по порядку: 0,1,2, не вдаваясь в подробности, какие именно индексы у этих объектов:



Если такого порядкового индекса нет в данных, то будет ошибка IndexError (по аналогии с выходом за массив в списках):

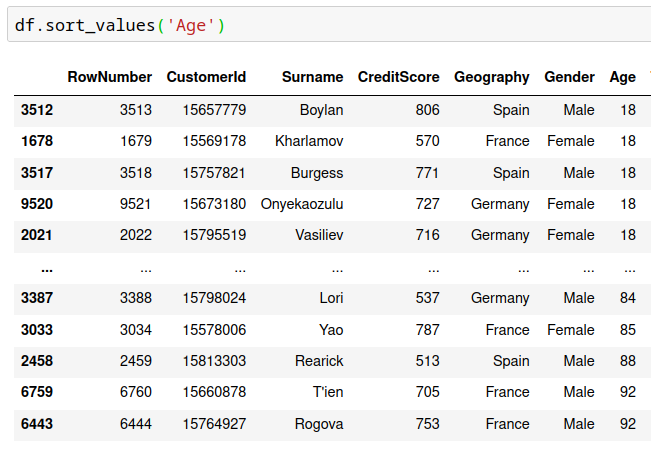


Помимо порядковых индексов строки, можно передавать и порядковые индексы столбцов:

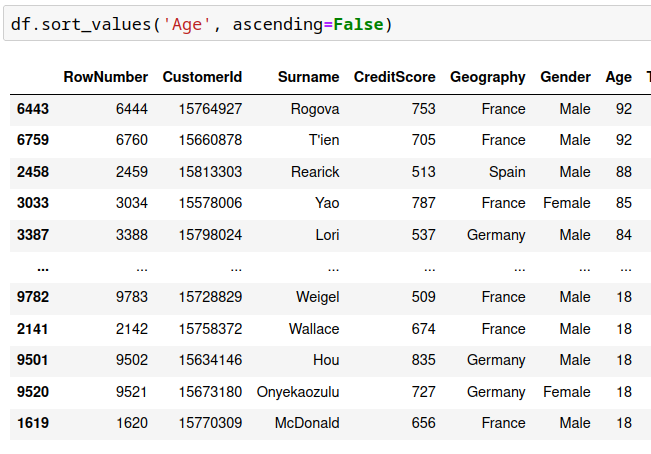


### Сортировки

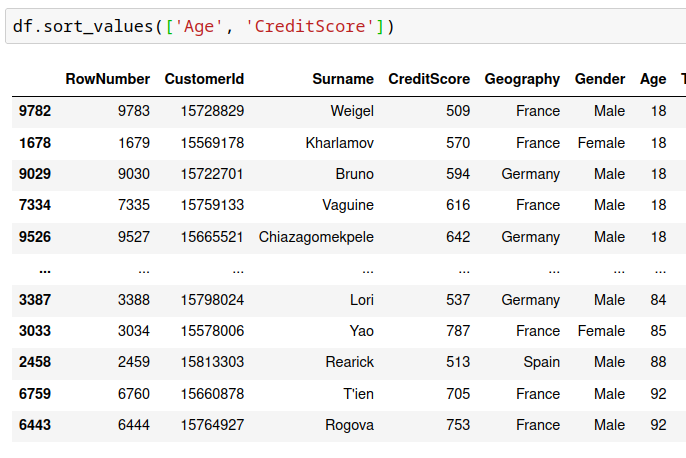
Если хочется вывести данные в определенном порядке, то можно пользоваться методом sort\_values(), который сортирует таблицу по признаку. Сортировка будет выполнена от меньшего к большему:



Если хочется применить сортировку наоборот от большего к меньшему, то воспользуйтесь атрибутом ascending=False:



А еще можно сортироваться по двум и более признакам. Сначала сортировка пройдет в первом признаке, а если встречаются одинаковые значения, то сортировка коснется и второго признака:



## 8) Краткий анонс следующей лекции, 2-3 предложения

На следующей лекции будем продолжать наше знакомство с анализом данных с помощью библиотеки Pandas. Обсудим создание новых характеристик данных, группировки, объединения и встроенную визуализацию.

## 9) Рекомендуемая дополнительная литература или материалы

1. Типы данных<https://youtu.be/c4Cg3TUIH0E>
2. Введение в pandas https://youtu.be/gJKN8zyG5c0
3. Документация pandas <https://pandas.pydata.org/docs/>
4. Аналитикам: большая шпаргалка по Pandas https://smysl.io/blog/pandas/

## 10) Используемая литература или материалы

1. Введение в pandas: анализ данных на Python <https://khashtamov.com/ru/pandas-introduction/>

## 11) Домашнее задание(-я) или задание (-я) для закрепления

(если есть необходимость)

Обращаем внимание, что домашние задания, которые даются на лекции, не являются обязательными для выполнения. Вы можете дать задания для самостоятельной работы на лекции, в этом разделе опишите задание и его решение.