## **Лекция 9. Модификация таблиц с Pandas**

## Краткая цель лекции:

a) Научиться создавать, изменять и удалять признаки

b) Изучить группировку данных и объединение таблиц

c) Познакомиться со встроенными визуализациями

На предыдущей лекции знакомились с анализом данных с помощью библиотеки Pandas. Она позволяет быстро считать статистики, фильтроваться по данным и сортировать их.

## **Термины лекции**

Признак (англ.feature) — это индивидуальное свойство или характеристика объекта.

Агрегация, или агрегирование — процесс объединения элементов в одно значение.

## **Краткий план лекции**

1. Работа с признаками
   1. Создание признака
   2. Удаление признака
   3. Изменение признака
      1. .loc
      2. .replace
2. Методы агрегации
3. Методы объединения
   1. Метод merge
   2. Метод join
   3. Атрибут how
4. Методы группировки
   1. groupby
   2. pivot\_table
   3. crosstab
5. Встроенные визуализации

## 

**В этом уроке продолжаем знакомиться с анализом данных с помощью библиотеки Pandas.**

### Работа с признаками

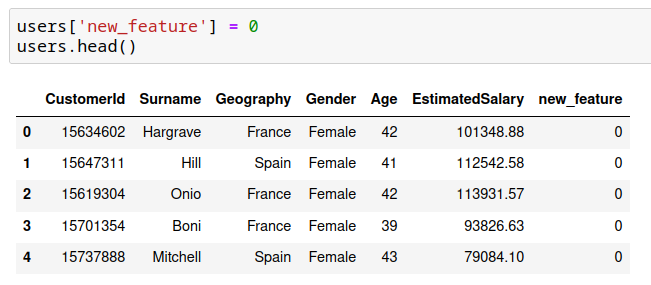
Признак (англ.feature) — это индивидуальное свойство или характеристика объекта.

Признак - это любая характеристика в данных. Возраст всех наших клиентов - это признаке. Их пол - это тоже признак.

#### Создание признака

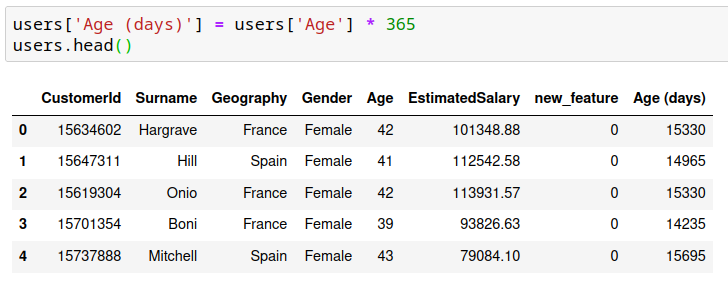
И начнем наше занятие с создания новых признаков. Создавать новые признаки можно очень большим количеством подходов, мы рассмотрим некоторые из них.

Самый простой способ - заполнить новый признак одинаковыми числами:

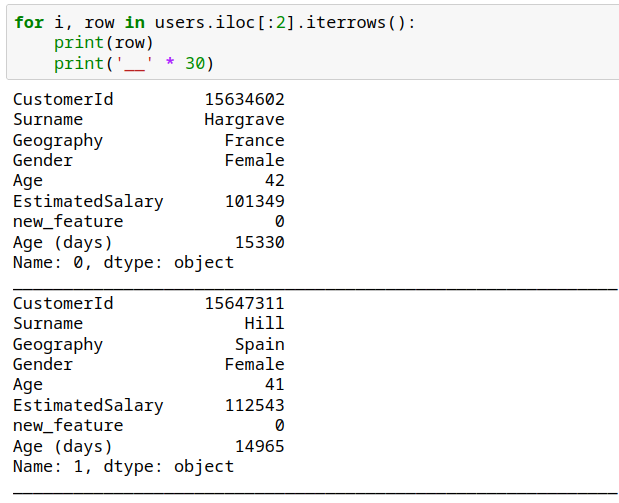


Можно создать новый признак, основываясь на уже имеющихся. Для этого:

* Берем признак через квадратные скобки (users[‘Age’])
* Делаем любое преобразование (у нас это умножение на 365)
* Присваиваем измененные данных в новый признак (users[‘Age (days)’] = )



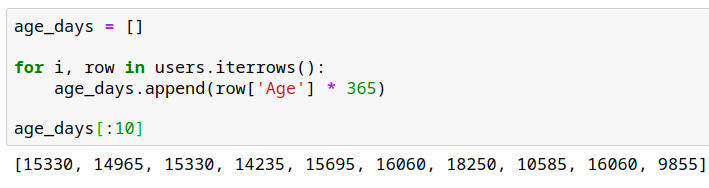
Или же можно пройтись по циклы, используя метод iterrows(), в этом случае мы итерируемся по строкам и получаем все значения одного объекта за раз:



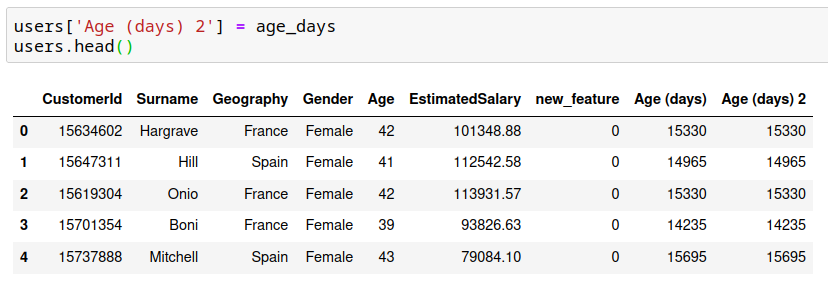
Можем проводить разные математические вычисления со значениями в признаке, к примеру:

1. Сложение +
2. Вычитание -
3. Деление /
4. Умножение \*
5. И так далее

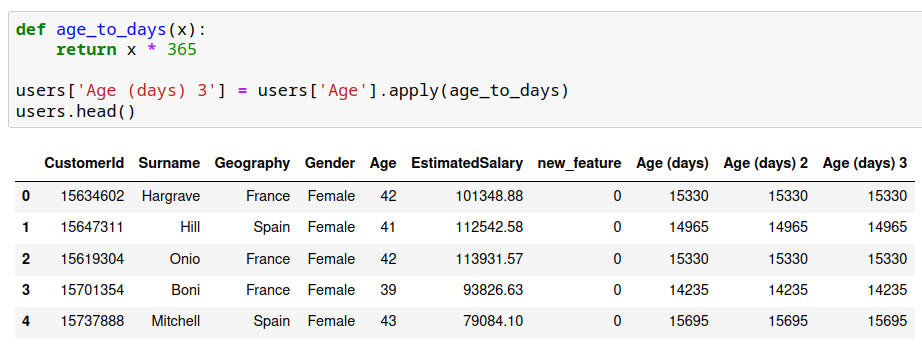
А результат работы можем записать в заранее подготовленный массив:



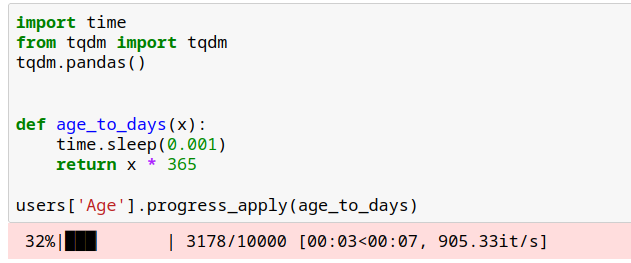
И затем добавить этот массив в датафрейм:



Но одна проблема с перебором всех строк датафрейма в цикле - это очень долго. Поэтому можем пользоваться методом apply, который ускоряет обход всех строк. Для этого нужно реализовать функцию (или же пользоваться анонимными функциями) и передать её в метод apply:

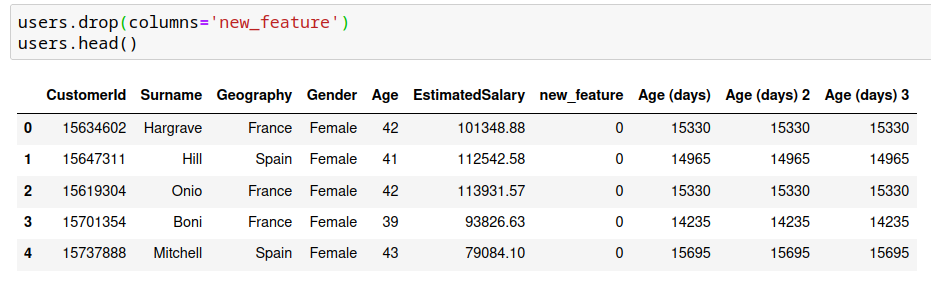


Если в вашем наборе данных очень много строк, то и выполнение apply может затянуться, в этом случае можно визуализировать процесс обхода датафрейма через библиотеку tqdm:

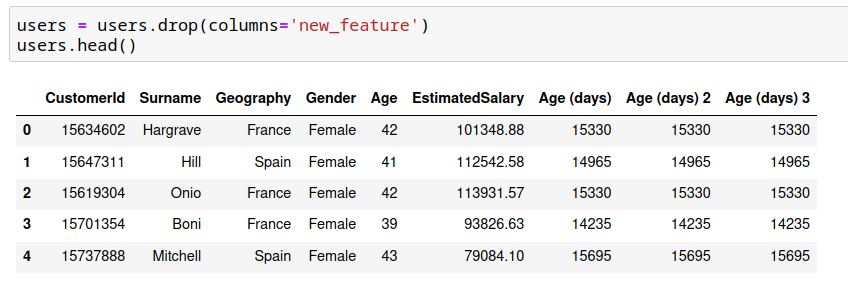


#### Удаление признаков

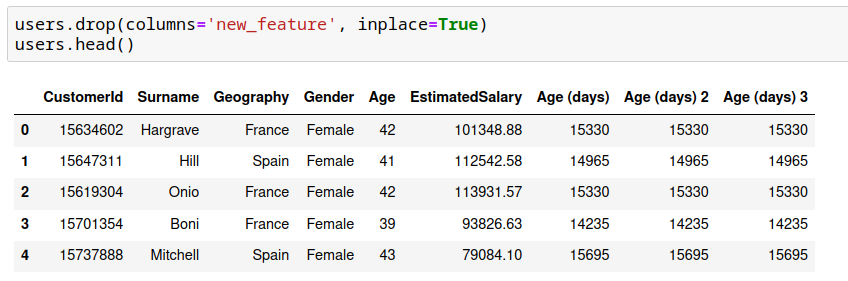
Чтобы удалить столбец из таблицы можем пользоваться методом drop():



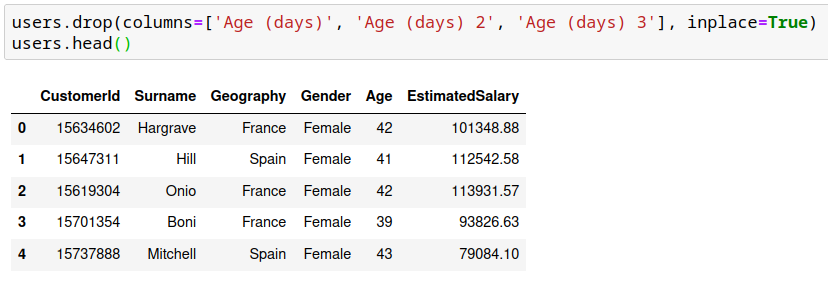
Но замечаем такую особенность, что признак всё равно на месте, он не удалился. Чтобы он на самом деле исчез нужно либо явно переопределить датафрейм:



Либо указать атрибут inplace=True:



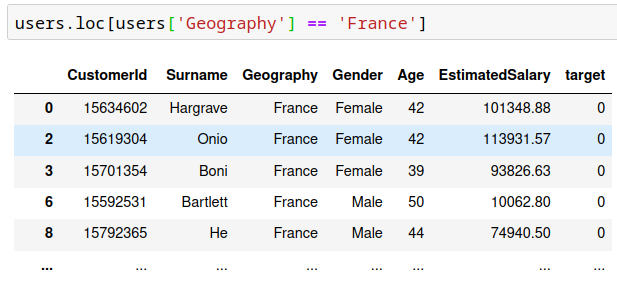
При этом можно удалять не только один признак за раз, а целый список признаков:



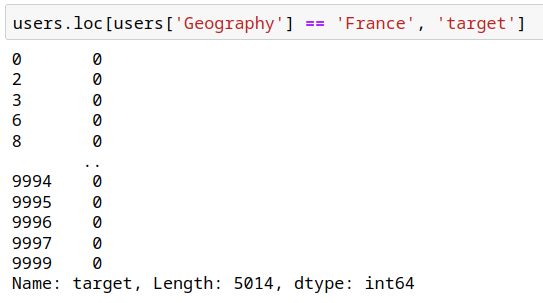
#### Изменение признаков

##### .loc

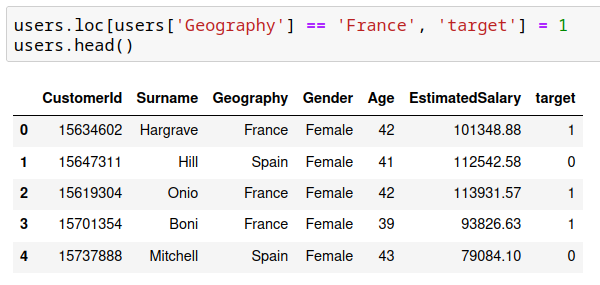
Чтобы изменить существующий признак пользуйтесь фильтрацией с помощью .loc, чтобы выделить нужные строки:



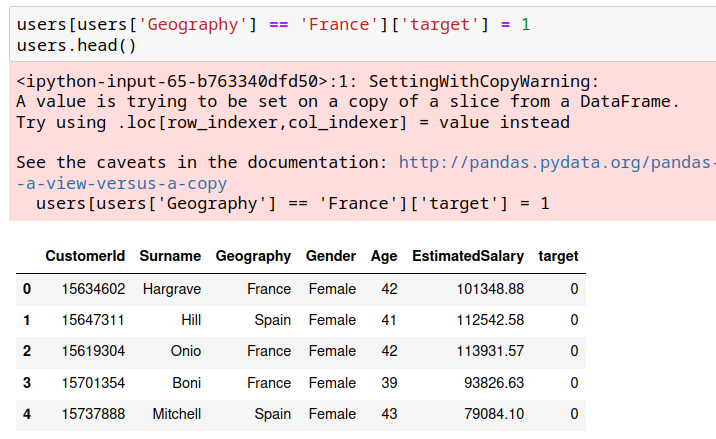
Затем указываете, какую колонку хотите изменить:



И на какой значение:



Если будете менять существующие признаки без .loc, то будете наблюдать подобное предупреждение и признак не будет меняться. Всё из-за того, что меняются значения не в исходном датафрейме, а в его копии:

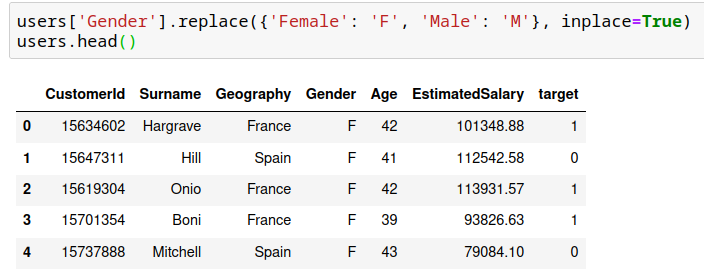


##### .replace

Будем работать с pd.Series. pd.Series представляет из себя объект, похожий на одномерный массив, но отличительной чертой является наличие индексов. Индекс находится слева, а сам элемент справа.

Метод replace() можно вызвать у этого объекта. Вызвать метод значит обратиться к нему через точку у pandas серии (users[‘Gender’].replace()).

Затем в этот метод можем передать словарь, состоящий из старого значения и нового. При этом, чтобы изменения вступили в силу, нужно либо переопределить признак, либо воспользоваться атрибутом inplace=True:



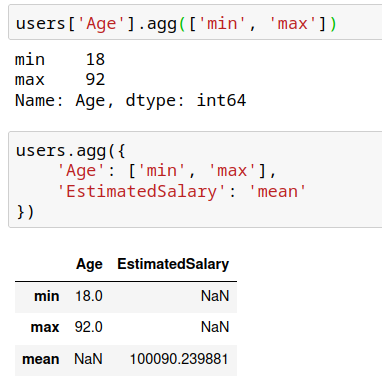
### Методы агрегации

Агрегация, или агрегирование — процесс объединения элементов в одно значение.

Агрегация данных нужна для того, чтобы подсчитать статистики в таблицы, например, для того, чтобы узнать средний возраст клиентов или же их максимальный чек.

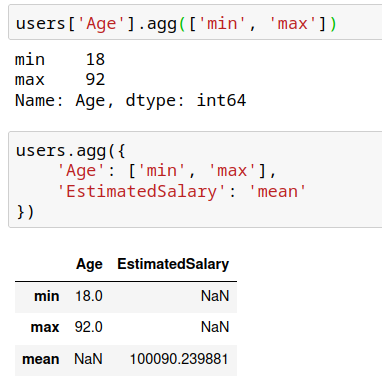
Дальше обсудим, как можно агрегировать значения в таблицах.

Первый способ - вызвать метод agg() у pd.Series и передать в него список желаемых агрегаций:

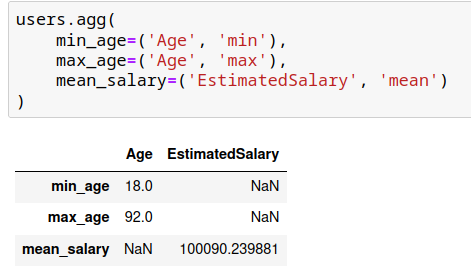


Второй способ - вызвать метод agg() у датафрейма. Это значит, что после переменной с датафреймом нужно через точку указать название метода .replace. В вызов этого метода можно передать словарь, где

* ключ - название признака
* значение - список желаемых агрегаций



И третий способ - вызвать метод agg() у датафрейма и передать аргументы (которые будут названиями строк) со значениями кортежа:

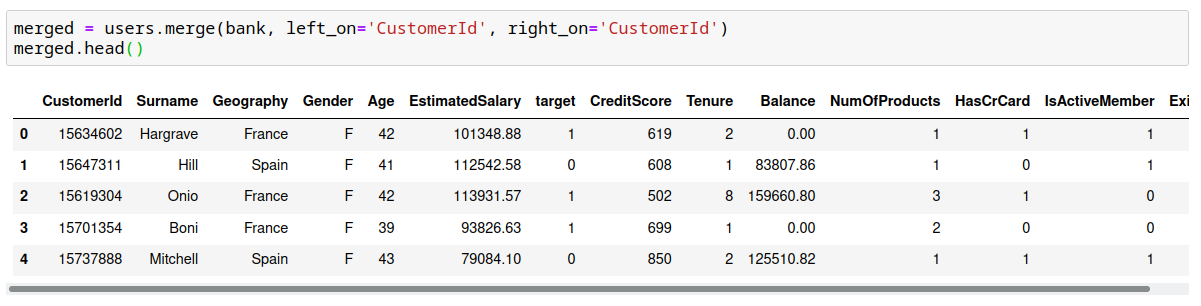


### Методы объединения

#### Метод merge

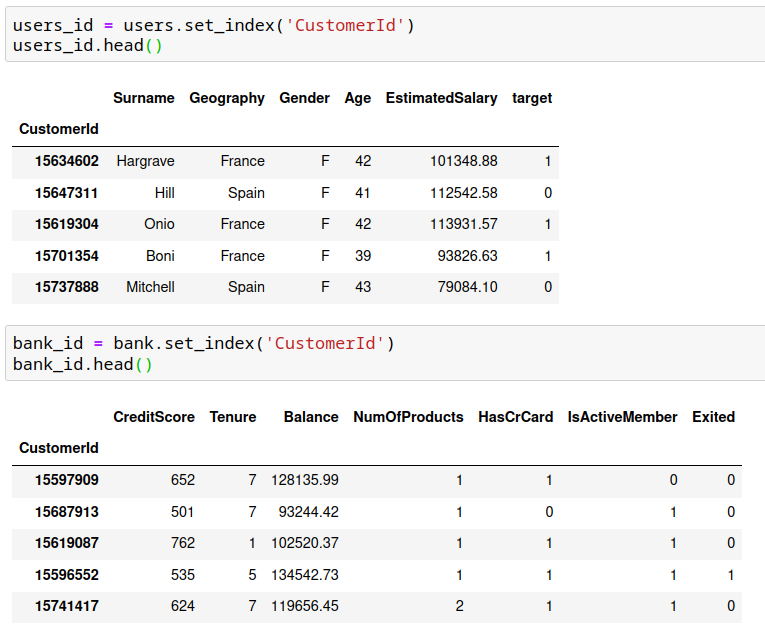
С помощью метода merge() можно объединить две таблицы по столбцам. У первого датафрейма вызываем метод merge() и в него передаем:

* второй датафрейм
* столбец с левого датафрейма (он же первый), по которому нужно объединение (аргумент left\_on)
* столбец с правого датафрейма (он же второй), по которому нужно объединение (аргумент right\_on)

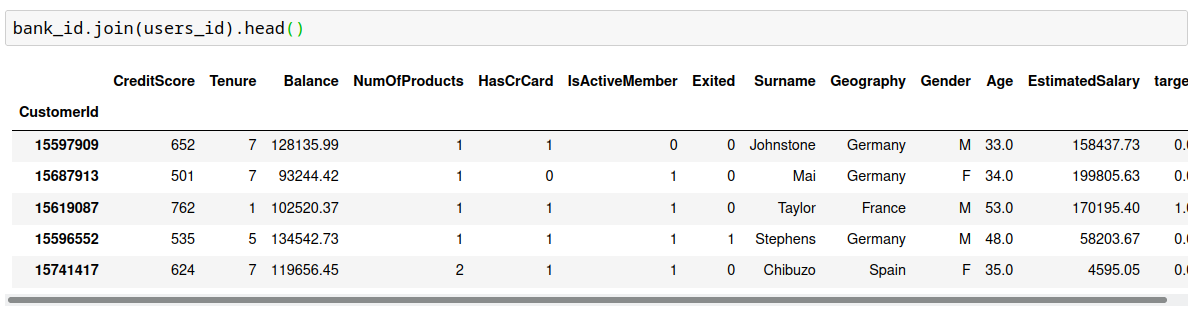


#### Метод join

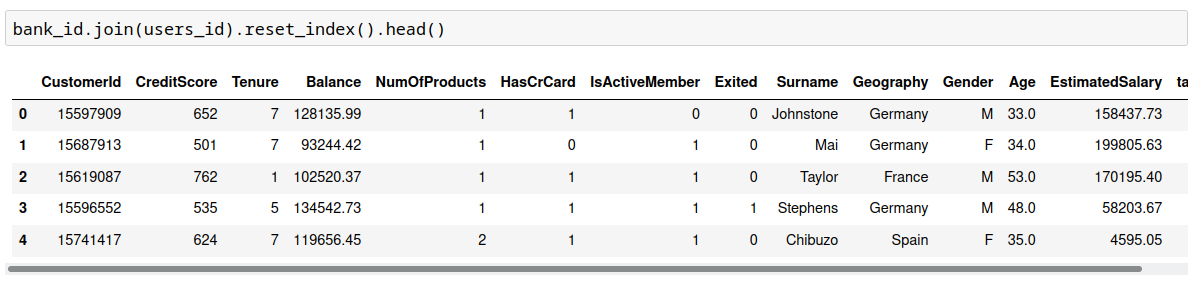
Данный метод объединяет две таблицы по индексам, поэтому для успешного объединения нужно указать индексы:



И затем вызвать метод join у одного из датафреймов:

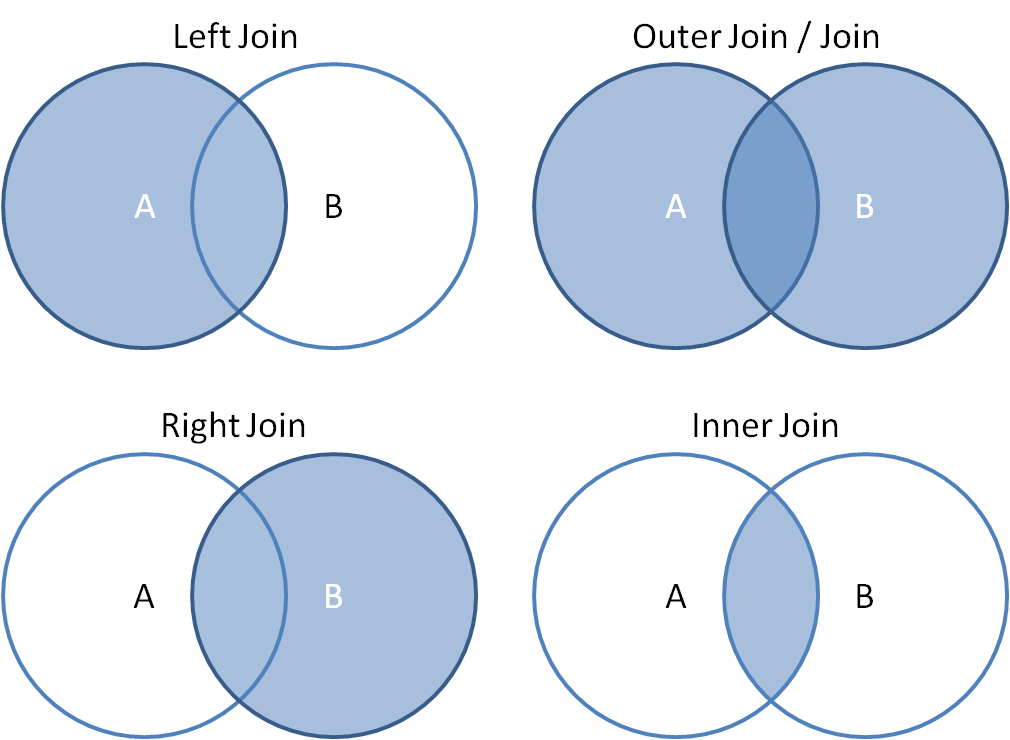


Чтобы вытащить уникальные идентификаторы пользователей из индексов можно вызвать атрибут reset\_index():



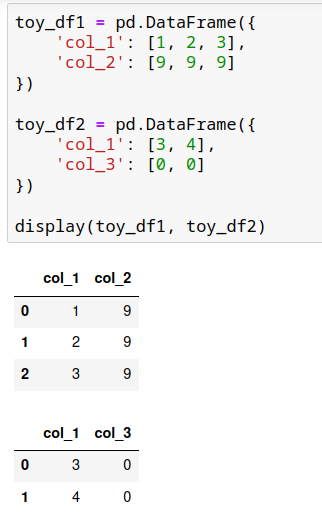
#### Атрибут how

В методах merge и join есть атрибут how, который позволяет указать способ объединения таблиц.

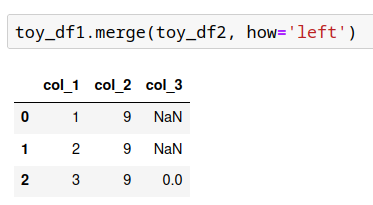


* left - остаются все объекты с левого датафрейма и ищутся совпадения из правого
* right - остаются все объекты с правого датафрейма и ищутся совпадения из левого
* inner - остаются объекты, которые есть и в левом датафрейме, и в правом
* outer - остаются все объекты из двуз датафреймов

Возьмем небольшие два датафрейма для наглядности:

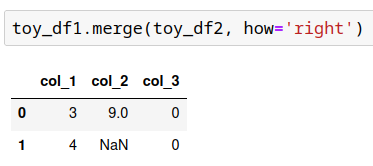


А теперь объединим их с помощью merge с атрибутом how=’left’:



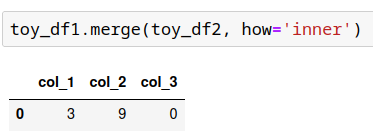
Все строки с левого датафрейма сохранились, а вот с правого одна потерялась, потому что в левом датафрейме не было значения в col\_1 равное 4, поэтому эта строка не появилась в результирующем датафрейме.

Теперь объединим их с помощью how=’right’:



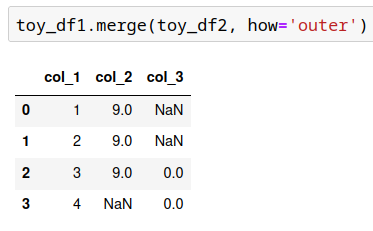
Остались все строки с правого датафрейма, а с левого две не оказались в результирующем датафрейме.

Следующий пример - атрибут how=’inner’:



Осталось только их пересечение, а остальные строки не остались.

И заключительный пример с how=’outer’:

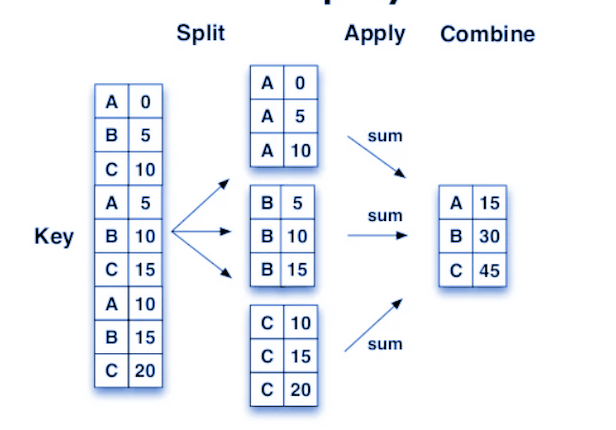


А здесь вернулись все строки и с левого, и с правого датафрейма.

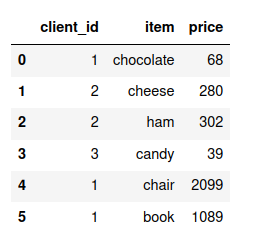
### Методы группировки

#### groupby

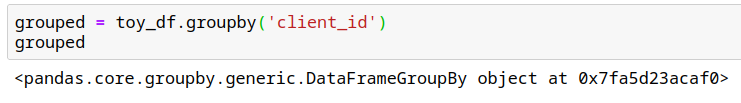
В данном методе вначале происходит разбиение на группы, а затем можно сделать агрегацию по любой агрегирующей функции:



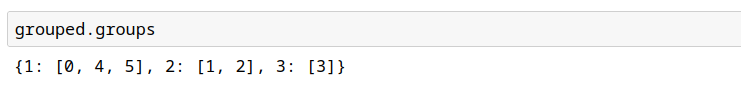
Рассмотрим принцип работы на небольшом наборе данных:



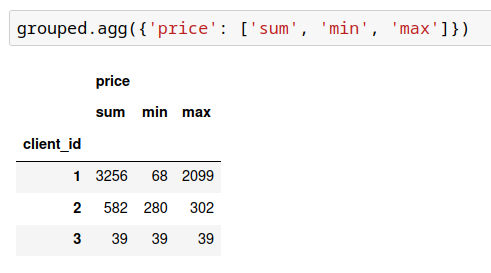
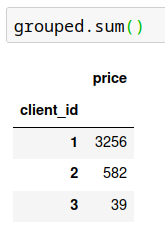
Сделаем группировку по идентификатору клиента, на выходе видим результат группировки:



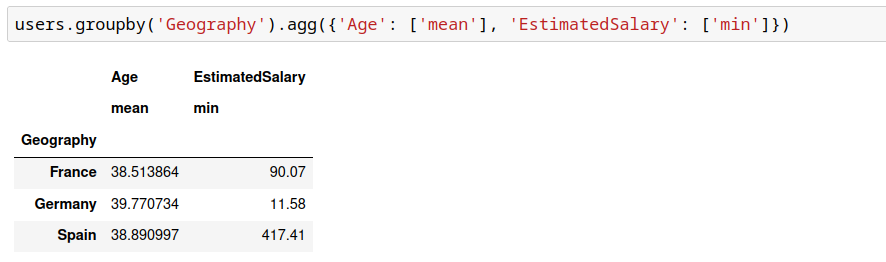
По атрибуту groups можем получить 3 группы наших клиентов и их индексы:



И теперь можем делать любые агрегации, которые захотим, при этом их может быть несколько:



И посмотрим группировки на данных про клиентов банка, для этого сгруппируем их по признаку Geography и найдем средний возраст и минимальную заработную плату по странам:



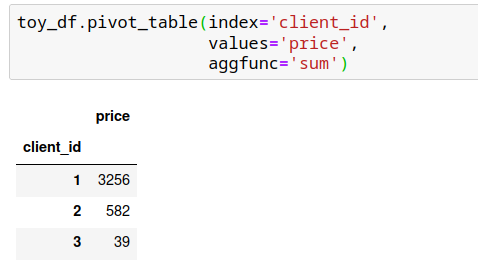
#### pivot\_table

pivot\_table (сводная таблица) - это мощный инструмент для обобщения и представления данных.

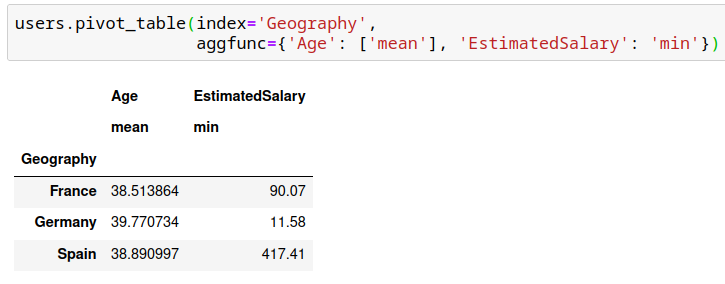
Параметры pivot\_table():

* index – столбец, который будет использован для строк
* columns – столбец, который будет использован для столбцов
* values – столбец обрабатываемых значений
* aggfunc – функция, применяемая к values
* fill\_value – значение по умолчанию
* margins – если True, то добавляется столбец All (Итого). По умолчанию: False

Вызовем pivot\_table для игрушечного примера. Посчитаем сумму для столбца price для клиентов:



При этом в атрибут aggfunc можно передавать словарь и тем самым мы получим точно такую же группировку, как и при использовании groupby:



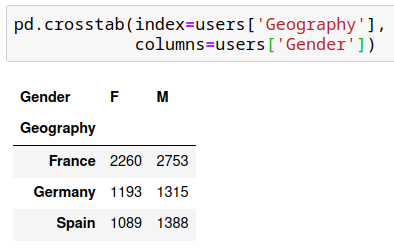
#### crosstab

Функция crosstab создает таблицу кросс-табуляции, которая по умолчанию может показать частоту, с которой появляются определенные группы данных.

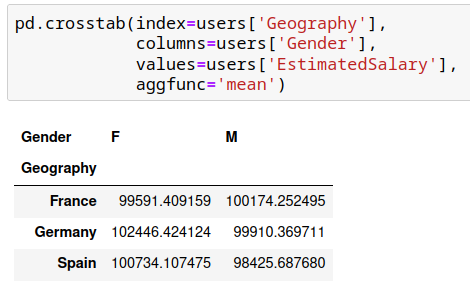
Параметры crosstab():

* index - значения для группировки по строкам
* columns - значения для группировки по столбцам
* values - агрегируемый столбец (или столбцы)
* aggfunc - функция, которая будет применена к каждой группе значений values, сгруппированным по значениям index и columns. Значения этой функции и есть значения сводной таблицы
* margins - добавляет результирующий столбец/строку
* normalize: boolean, {'all', 'index', 'columns'} - нормировка всей таблицы (или только по строкам/столбцам).

По умолчанию crosstab подсчитывает количество значений в каждой выделенной категории, таким образом мы можем посчитать, сколько у нас есть людей разных полов в зависимости от страны. При этом обратите внимание, что в атрибуты нужно передавать всю колонку, а не только название признака:

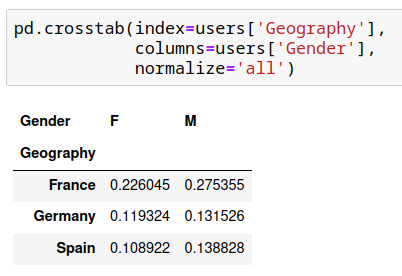


Можем добавить и колонку, которую надо агрегировать, теперь подсчитываем, какая средняя зарплата в зависимости от пола и страны человека:

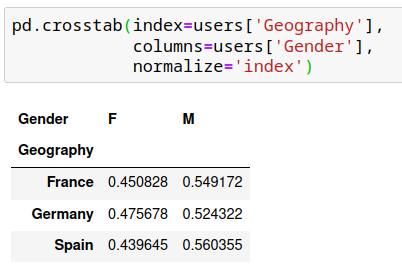


В функции crosstab есть нормировка по значениям:

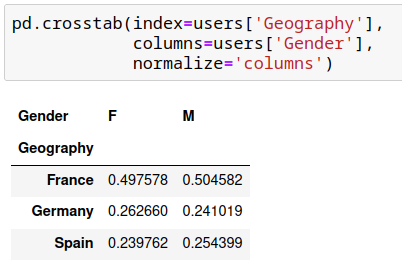
* атрибут normalize=’all’ - нормировка по всем значениям



* атрибут normalize=’index’ - нормировка по строкам



* атрибут normalize=’columns’ - нормировка по столбцам



### Встроенные визуализации

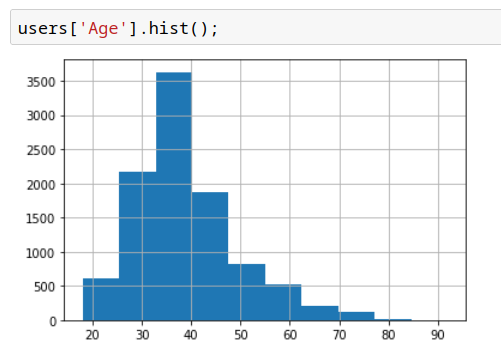
В связи с особенностями человеческого мозга в обрабатывании информации, проще использовать диаграммы или графики для визуализации больших объемов сложных данных, чем разбираться с таблицами или отчетами.

Хорошая визуализация выделяет полезную информацию. Визуализация данных позволяет определить области, которые требуют внимания или улучшения, выяснить, какие характеристики влияют на поведение клиентов, а также понять и объяснить сложные процессы.

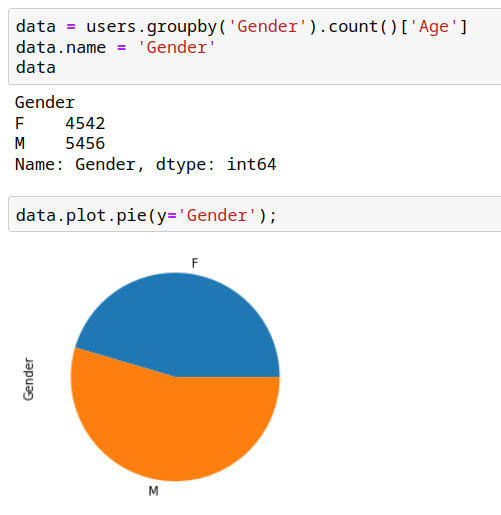
Визуализация делает данные более запоминающимися.

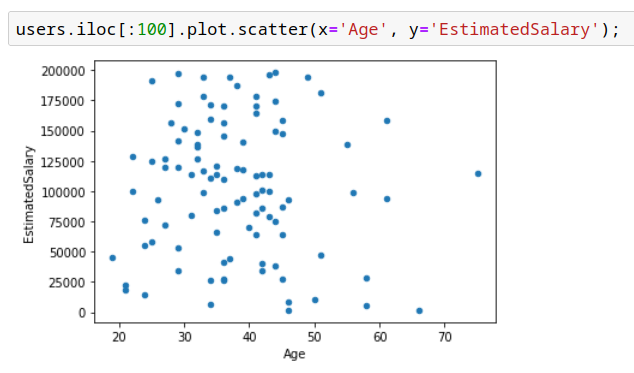
Посмотрим на некоторые встроенные визуализации, которые доступны в самой библиотеке pandas, при этом разбираться в самих графиках и их интерпретациях будем на следующем занятии, а сейчас только посмотрим на возможности библиотеки pandas:

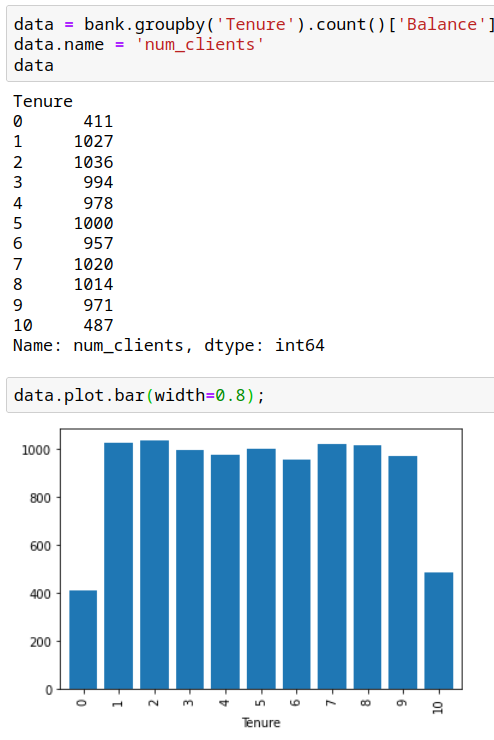
1. hist() - гистограмма. С помощью неё можем изучить распределение возраста наших клиентов:



1. pie() - круговая диаграмма, с помощью неё поизучаем долю мужчин и долю женщин среди наших клиентов:



1. scatter() - точечный график, он показывает взаимное распределение признаков. Изучим, а есть ли зависимость между возрастом клиента и его заработной платой:  
   
2. bar() - столбчатая диаграмма, показывает количество объектов в каждой категории. Посмотрим, сколько уже лет люди являются клиентами нашего банка:



Это далеко не все графики, которые можно строить с помощью pandas, это лишь некоторые из них.

На следующей лекции будем знакомиться с визуальным анализом данных с помощью библиотек Matplotlib и Seaborn. Узнаем, какие бывают графики, как их отрисовывать и как их интерпретировать.

## Рекомендуемая дополнительная литература или материалы

1. pandas.DataFrame.merge <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html>
2. pandas.DataFrame.plot <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html>

## Используемая литература или материалы

1. Агрегации и группировки <https://pythonru.com/uroki/osnovy-pandas-2-agregacija-i-gruppirovka>