# Базовые методы обработки данных с использованием Python

*Лекция 2. Работа с модулем Pandas в Python. Группировка и агрегация данных.*

Киреев Василий Сергеевич

к.т.н., доцент

**Москва, 2024**

# Пакет Pandas

Pandas - это библиотека с открытым исходным кодом на языке Python, созданная в основном для простой и интуитивно понятной работы с реляционными или маркированными данными. Она предоставляет различные структуры данных и операции для манипулирования числовыми данными и временными рядами.

Её название происходит от «panel data» («панельные данные»). Панельными данными называют информацию, полученную в результате исследований и структурированную в виде таблиц. Эта библиотека построена на основе библиотеки NumPy языка Python. Изначально библиотека Pandas была разработана Уэсом Маккинни в 2008 году во время его работы в компании AQR Capital Management.

# Пакет Pandas. Предназначение

Очистка данных

Предварительная обработка данных

Исследование данных

Генерация признаков (feature engineering)

Анализ временных рядов

Наука о данных

# Типы данных Pandas

| Pandas dtype | Python type | NumPy type | Применение |
|---|---|---|---|
| object | str or mixed | string_, unicode_, mixed types | Текстовые или смешанные числовые и нечисловые значения |
| int64 | int | int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64 | Целые числа |
| float64 | float | float_, float16, float32, float64 | Числа с плавающей точкой |
| bool | bool | bool_ | Истинные/ложные значения |
| datetime64 | NA | datetime64[ns] | Значения даты и времени |
| timedelta[ns] | NA | NA | Разница между двумя датами |
| category | NA | NA | Конечный список текстовых значений |

# Основные объекты Pandas

Датафрейм (DataFrame)

Серия (Series)

Индекс (Index)

# Pandas Series (Серия)

Pandas Series (Серия) - это одномерный маркированный массив, способный хранить данные любого типа (целые числа, строки, float, объекты python и т.д.). Метки осей в совокупности называются индексами. Метки не обязательно должны быть уникальными, но должны быть хэшируемого типа. Объект поддерживает как целочисленную, так и основанную на метках индексацию и предоставляет множество методов для выполнения операций с индексом.

# Методы Pandas Серии

| Функция | Описание |
|---|---|
| **Series()** | Серия pandas может быть создана с помощью метода-конструктора Series(). Этот метод-конструктор принимает различные входные данные |
| **combine_first()** | Метод используется для объединения двух серий в одну |
| **count()** | Возвращает количество не-NA/null наблюдений в серии |
| **size()** | Возвращает количество элементов в исходных данных |
| **name()** | Метод позволяет присвоить имя объекту Series, т.е. столбцу |
| **is_unique()** | Метод возвращает булево значение, если значения в объекте уникальны |
| **idxmax()** | Метод извлечения позиций индексов наибольших значений в серии |
| **idxmin()** | Метод извлечения индексов наименьших значений в серии |
| **sort_values()** | Метод вызывается на серии для сортировки значений по возрастанию или убыванию |
| **sort_index()** | Метод вызывается на pandas Series для сортировки по индексу, а не по значениям |

# Методы Pandas Серии

| Функция | Описание |
|---|---|
| **head()** | Метод используется для возврата заданного количества строк из начала Серии. Метод возвращает совершенно новую серию |
| **tail()** | Метод используется для возврата заданного количества строк из конца серии. Метод возвращает совершенно новую серию |
| **le()** | Используется для сравнения каждого элемента вызываемой серии с передаваемой серией. Возвращает True для каждого элемента, который меньше или равен элементу передаваемой серии. |
| **ne()** | Используется для сравнения каждого элемента вызывающей серии с переданной серией. Возвращает True для каждого элемента, который не равен элементу переданной серии |
| **ge()** | Используется для сравнения каждого элемента вызывающей серии с передаваемой серией. Возвращает True для каждого элемента, который больше или равен элементу переданной серии |
| **eq()** | Используется для сравнения каждого элемента вызывающей серии с передаваемой серией. Возвращает True для каждого элемента, который равен элементу переданной серии |
| **gt()** | Используется для сравнения двух серий и возвращает булево значение для каждого соответствующего элемента |
| **lt()** | Используется для сравнения двух серий и возвращает булево значение для каждого соответствующего элемента |

# Методы Pandas Серии

| Функция | Описание |
|---|---|
| **clip()** | Используется для обрезки значений ниже и выше переданного наименьшего и максимального значения |
| **clip_lower()** | Используется для фиксации значений ниже переданного наименьшего значения |
| **clip_upper()** | Используется для фиксации значений выше переданного максимального значения |
| **astype()** | Метод используется для изменения типа данных ряда |
| **tolist()** | Метод используется для преобразования серии в список |
| **get()** | Метод вызывается на серии для извлечения значений из серии. Это альтернативный синтаксис по сравнению с традиционным синтаксисом скобок |
| **unique()** | Pandas unique() используется для просмотра уникальных значений в конкретном столбце |
| **nunique()** | Pandas nunique() используется для получения количества уникальных значений |

# Методы Pandas Серии

| Функция | Описание |
|---|---|
| **value_counts()** | Метод подсчета количества повторений каждого уникального значения в серии |
| **factorize()** | Метод позволяет получить числовое представление массива путем выделения отдельных значений |
| **map()** | Метод, позволяющий связать значения одного объекта с другим |
| **between()** | Метод Pandas between() используется в сериях для проверки того, какие значения лежат между первым и вторым аргументами |
| **apply()** | Метод вызывается и передает функцию Python в качестве аргумента для использования функции на каждом значении серии. Этот метод полезен для выполнения пользовательских операций, которые не включены в pandas или numpy |

# Создание Серии. Пример

В реальном мире серия Pandas создается путем загрузки наборов данных из существующего хранилища, в качестве которого может выступать база данных SQL, файл CSV, файл Excel. Серии Pandas могут быть созданы из списков, словаря, скалярного значения и т.д.

```
import pandas as pd
import numpy as np

data = np.array(['g','e','e','n','a','s'])
ser = pd.Series(data)
print(ser)
```

```
0    g
1    e
2    e
3    n
4    a
5    s
dtype: object
```

# Pandas DataFrame

Pandas DataFrame - это двумерная, изменяемая по размеру, потенциально неоднородная табличная структура данных с маркированными осями (строками и столбцами). Фрейм данных - это двумерная структура данных, т.е. данные выстраиваются в табличной форме в виде строк и столбцов. Pandas DataFrame состоит из трех основных компонентов - данных, строк и столбцов.

# Методы Pandas Датафрейма

| Функция | Описание |
|---|---|
| **index()** | Метод возвращает индекс (метки строк) датафрейм |
| **insert()** | Метод вставляет столбец в датафрейм |
| **add()** | Метод возвращает сложение dataframe и other по элементам (бинарный оператор add) |
| **sub()** | Метод возвращает вычитание dataframe и других, поэлементно (бинарный оператор sub) |
| **mul()** | Метод возвращает умножение датафрейма и других элементов (двоичный оператор mul) |
| **div()** | Метод возвращает плавающее деление датафрейма и других, поэлементно (бинарный оператор truediv) |
| **unique()** | Метод извлекает уникальные значения в датафрейме |
| **nunique()** | Метод возвращает счетчик уникальных значений в датафрейме |

# Методы Pandas Датафрейма

| Функция | Описание |
|---|---|
| axes() | Метод возвращает список, представляющий оси кадра данных |
| isnull() | Метод создает булеву серию для извлечения строк с нулевыми значениями |
| notnull() | Метод создает булеву серию для извлечения строк с ненулевыми значениями |
| between() | Метод извлекает строки, в которых значение столбца попадает в заданный диапазон |
| isin() | Метод извлекает строки из DataFrame, в которых значение столбца находится в предопределенной коллекции |
| dtypes() | Метод возвращает серию с типом данных каждого столбца. Индексом результата являются столбцы исходного DataFrame |
| astype() | Метод преобразует типы данных в серии |
| values() | Метод возвращает Numpy-представление DataFrame, т.е. возвращаются только значения в DataFrame, метки осей удаляются |
| sort_values()- Set1, Set2 | Метод сортирует кадр данных в порядке возрастания или убывания по переданному столбцу |
| sort_index() | Метод сортирует значения в DataFrame на основе их индексных позиций или меток, а не значений, но иногда фрейм данных состоит из двух или более фреймов данных, поэтому впоследствии индекс может быть изменен с помощью этого метода |

# Методы Pandas Датафрейма

| Функция | Описание |
|---|---|
| **loc[]** | Метод извлекает строки на основе метки индекса |
| **iloc[]** | Метод извлекает строки на основе позиции индекса |
| **ix[]** | Метод извлекает строки DataFrame на основе либо метки индекса, либо позиции индекса. Этот метод сочетает в себе лучшие свойства методов .loc[] и .iloc[]. |
| **rename()** | Метод вызывается на DataFrame для изменения имен индексных меток или имен столбцов |
| **columns()** | Метод является альтернативным атрибутом для изменения имени столбца |
| **drop()** | Метод используется для удаления строк или столбцов из DataFrame |
| **pop()** | Метод используется для удаления строк или столбцов из кадра DataFrame |
| **sample()** | Метод вытягивает случайную выборку строк или столбцов из DataFrame |
| **nsmallest()** | Метод извлекает строки с наименьшими значениями в столбце |
| **nlargest()** | Метод извлекает строки с наибольшими значениями в столбце |

# Методы Pandas Датафрейма

| Функция | Описание |
|---|---|
| **shape()** | Метод возвращает кортеж, представляющий размерность фрейма DataFrame |
| **ndim()** | Метод возвращает значение 'int', представляющее количество осей / размерность массива. Возвращает 1, если Series, иначе возвращает 2, если DataFrame |
| **dropna()** | Метод позволяет пользователю анализировать и отбрасывать строки/столбцы с нулевыми значениями различными способами |
| **fillna()** | Метод позволяет пользователю заменять значения NaN собственным значением |
| **rank()** | С помощью этого метода значения в серии могут быть упорядочены по порядку |
| **query()** | Метод представляет собой альтернативный строковый синтаксис для извлечения подмножества из DataFrame |
| **copy()** | Метод создает независимую копию объекта pandas |
| **duplicated()** | Метод создает булевую серию и использует ее для извлечения строк с дублирующимися значениями |

# Методы Pandas Датафрейма

| Функция | Описание |
|---|---|
| **reset_index()** | Метод сбрасывает индекс фрейма данных. В качестве индекса метод задает список целых чисел от 0 до длины данных |
| **where()** | Метод используется для проверки фрейма данных на соответствие одному или нескольким условиям и возвращает соответствующий результат. По умолчанию строки, не удовлетворяющие условию, заполняются значением NaN |
| **drop_duplicates()** | Метод является альтернативным вариантом выявления дублирующихся строк и их удаления с помощью фильтрации |
| **set_index()** | Метод устанавливает индекс DataFrame (метки строк), используя один или несколько существующих столбцов |
| **value_counts()** | Метод подсчитывает, сколько раз каждое уникальное значение встречается в серии |
| **columns()** | Метод возвращает метки столбцов датафрейма |

# Объекты Pandas. Pandas DataFrame

## Pandas DataFrame – двухмерные таблицы

```
1 D=pd.DataFrame({'пол':np.random.choice(['муж','жен'],size=10),
2                 'возраст':np.random.randint(18,45,size=10)})
3 D.head(4)
```

| | пол | возраст |
|---|---|---|
| 0 | жен | 36 |
| 1 | жен | 21 |
| 2 | жен | 25 |
| 3 | жен | 43 |

# Считывание данных в Pandas

```
1 df=pd.read_csv(path,sep='\t')
2 df.head(3)
```

| | <TICKER> | <PER> | <DATE> | <TIME> | <OPEN> | <HIGH> | <LOW> | <CLOSE> | <VOL> |
|---|---|---|---|---|---|---|---|---|---|
| **0** | GAZP | 1 | 20210802 | 100100 | 287.88 | 288.75 | 287.80 | 288.51 | 530780 |
| **1** | GAZP | 1 | 20210802 | 100200 | 288.49 | 288.66 | 288.28 | 288.35 | 138600 |
| **2** | GAZP | 1 | 20210802 | 100300 | 288.35 | 288.35 | 287.65 | 287.92 | 249950 |

```
[ ]  1 df.dtypes
```

```
<TICKER>     object
<PER>         int64
<DATE>        int64
```

# Преобразование типов в Pandas

```
1 pd.to_datetime(df['DATE'].astype(str)+'|'+df.TIME.astype(str),format='%Y%m%d|%H%M%S')
```

```
0        2021-08-02 10:01:00
1        2021-08-02 10:02:00
2        2021-08-02 10:03:00
3        2021-08-02 10:04:00
4        2021-08-02 10:05:00
                 ...
23510    2021-09-09 23:46:00
23511    2021-09-09 23:47:00
23512    2021-09-09 23:48:00
23513    2021-09-09 23:49:00
23514    2021-09-09 23:50:00
Length: 23515, dtype: datetime64[ns]
```

# Pandas Index

Объект Index можно рассматривать либо как неизменяемый массив, либо как упорядоченное множество (технически - мультимножество, поскольку объекты Index могут содержать повторяющиеся значения). Эти представления приводят к интересным последствиям в операциях, доступных для объектов Index.

# Заполнение пропущенных значений в Pandas

```
[ ]  1 low_ts.loc[low_ts.index=='2021-09-04 10:01:00',]
```

| | LOW |
|---|---|
| **2021-09-04 10:01:00** | NaN |

```
[ ]  1 low_ts_b=low_ts.fillna(method='bfill')
     2 low_ts_b.loc[low_ts_b.index=='2021-09-04 10:01:00',]
```

| | LOW |
|---|---|
| **2021-09-04 10:01:00** | 318.6 |

```
     1 low_ts_f=low_ts.fillna(method='ffill')
     2 low_ts_f.loc[low_ts_f.index=='2021-09-04 10:01:00',]
```

| | LOW |
|---|---|
| **2021-09-04 10:01:00** | 317.68 |

# Методы датафрейма Pandas

```
[38]  1 D.index
```

```
RangeIndex(start=0, stop=10, step=1)
```

```
[39]  1 D.columns
```

```
Index(['пол', 'возраст', 'образование'], dtype='object')
```

```
  1 D.shape
```

```
(10, 3)
```

```
  1 D.values
```

```
array([['муж', 32, 'высшее'],
       ['муж', 42, 'высшее'],
       ['жен', 25, 'среднее'],
       ['жен', 26, 'среднее']
```

# Apply  и Applymap в Pandas

```
1 %%time
2 df[['level','level']].applymap(lambda x: x*2)
```

```
CPU times: user 4.08 ms, sys: 71 µs, total: 4.15 ms
Wall time: 5.46 ms
```

| | level | level |
|---|---|---|
| **0** | 198 | 198 |
| **1** | 2 | 2 |
| **2** | 6 | 6 |
| **3** | 198 | 198 |

```
1 %%time
2 df[['level','level']].apply(lambda x: x*2)
```

```
CPU times: user 4.32 ms, sys: 6 µs, total: 4.32 ms
Wall time: 4.24 ms
```

| | level | level |
|---|---|---|
| **0** | 198 | 198 |
| **1** | 2 | 2 |
| **2** | 6 | 6 |
| **3** | 198 | 198 |

# Создание колонки «на лету» в Pandas

```
df2=df1.assign(employer_name=df1.employer.apply(lambda x: x['name'])).
drop(columns=['employer'])
df2.head(3)
```

| | id | name | salary | | employer_name |
|---|---|---|---|---|---|
| 0 | 48225041 | Менеджер | {'from': 160000, 'to': None, 'currency': 'RUR'... | {'requirement': | ММС-Клиника |
| 1 | 47955705 | Менеджер | {'from': 83496, 'to': 200000, 'currency': 'RUR... | {'requirement | FitClub |
| 2 | 48308835 | Менеджер | {'from': 70000, 'to': 150000, 'currency': 'RUR... | {'requirement': ' | МенИКварти.Ру |

```
df.filter(like='5',axis=0)[5:10]
```

| | 2col1 | col2 |
|---|---|---|
| 50 | ngzdr | 5.720504 |
| 51 | jfcif | 74.637850 |
| 52 | ucuxu | -37.912361 |
| 53 | ealft | 26.944114 |
| 54 | aodqx | -70.004290 |

# Описательная статистика в Pandas

```
1 df.describe()
```

| | ENERGY CONSUMPTION |
|---|---|
| **count** | 1160.000000 |
| **mean** | 663.550235 |
| **std** | 475.362310 |
| **min** | -96.523940 |
| **25%** | 344.777565 |
| **50%** | 639.384268 |
| **75%** | 937.010406 |
| **max** | 6038.420889 |

# Срезы в Pandas

```
1 df.loc[(df.col2>80),['2col1']]
```

| | 2col1 |
|---|---|
| **1** | dxxkb |
| **7** | kragg |
| **21** | jpygy |
| **28** | vnzwy |
| **35** | ogvyn |
| **...** | ... |
| **974** | dmtbv |

```
1 df.iloc[10:15,1:2]
```

| | col2 |
|---|---|
| **10** | -40.518751 |
| **11** | 34.160978 |
| **12** | 51.151810 |
| **13** | -33.207958 |
| **14** | 57.808095 |

# Фильтрация в Pandas

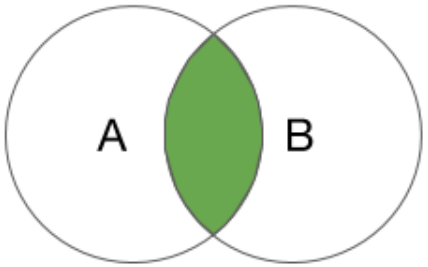
```
1 df.filter(regex='2$',axis=1)[5:10]
```

| | col2 |
|---|---|
| **5** | 40.277527 |
| **6** | -53.900701 |
| **7** | 99.224680 |
| **8** | 26.004142 |
| **9** | 9.267271 |

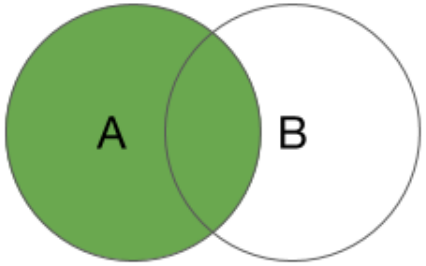
```
1 df.filter(like='5',axis=0)[5:10]
```

| | 2col1 | col2 |
|---|---|---|
| **50** | ngzdr | 5.720504 |
| **51** | jfcif | 74.637850 |
| **52** | ucuxu | -37.912361 |
| **53** | ealft | 26.944114 |
| **54** | aodqx | -70.004290 |

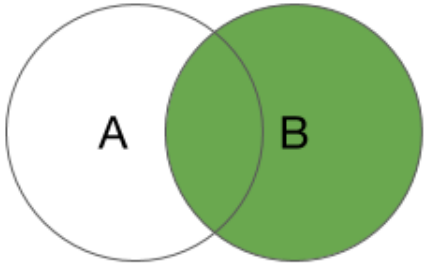
# Виды слияний датафреймов



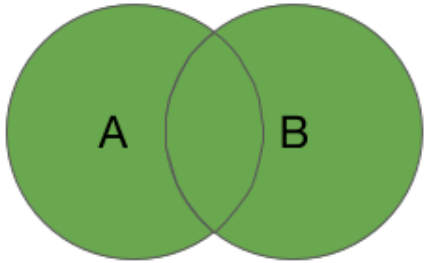

INNER JOIN



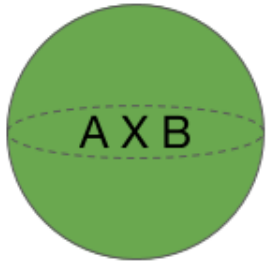

LEFT OUTER JOIN




RIGHT OUTER JOIN




FULL OUTER JOIN




CARTESIAN (CROSS) JOIN

# Сравнение Pandas и SQL. WHERE и TOP

| SQL | Pandas |
| --- | --- |
| `select * from airports where iso_region = 'US-CA' and type = 'seaplane_base'` | `airports[(airports.iso_region == 'US-CA') & (airports.type == 'seaplane_base')]` |
| `select ident, name, municipality from airports where iso_region = 'US-CA' and type = 'large_airport'` | `airports[(airports.iso_region == 'US-CA') & (airports.type == 'large_airport')][['ident', 'name', 'municipality']]` |

| SQL | Pandas |
| --- | --- |
| `select * from airports` | `airports` |
| `select * from airports limit 3` | `airports.head(3)` |
| `select id from airports where ident = 'KLAX'` | `airports[airports.ident == 'KLAX'].id` |
| `select distinct type from airport` | `airports.type.unique()` |

# Сравнение Pandas и SQL. IS IN и HAVING

| SQL | Pandas |
| --- | --- |
| `select * from airports where type in ('heliport', 'balloonport')` | `airports[airports.type.isin(['heliport', 'balloonport'])]` |
| `select * from airports where type not in ('heliport', 'balloonport')` | `airports[~airports.type.isin(['heliport', 'balloonport'])]` |

| SQL | Pandas |
| --- | --- |
| `select type, count(*) from airports where iso_country = 'US' group by type having count(*) > 1000 order by count(*) desc` | `airports[airports.iso_country == 'US'].groupby('type').filter(lambda g: len(g) > 1000).groupby('type').size().sort_values(ascending=False)` |

# Сравнение Pandas и SQL. INSERT

| SQL | Pandas |
| --- | --- |
| `create table heroes (id integer, name text);` | `df1 = pd.DataFrame({'id': [1, 2], 'name': ['Harry Potter', 'Ron Weasley']})` |
| `insert into heroes values (1, 'Harry Potter');` | `df2 = pd.DataFrame({'id': [3], 'name': ['Hermione Granger']})` |
| `insert into heroes values (2, 'Ron Weasley');` | |
| `insert into heroes values (3, 'Hermione Granger');` | `pd.concat([df1, df2]).reset_index(drop=True)` |

# Сравнение Pandas и SQL.UPDATE и DELETE

| SQL | Pandas |
| --- | --- |
| `update airports set home_link = 'http://www.lawa.org/welcomelax.aspx' where ident == 'KLAX'` | `airports.loc[airports['ident'] == 'KLAX', 'home_link'] = 'http://www.lawa.org/welcomelax.aspx'` |

| SQL | Pandas |
| --- | --- |
| `delete from lax_freq where type = 'MISC'` | `lax_freq = lax_freq[lax_freq.type != 'MISC']` |
| | `lax_freq.drop(lax_freq[lax_freq.type == 'MISC'].index)` |

# Слияние датафреймов. Append.Примеры

```
1 df1.append(df2)
```

| | 0 | 1 | 2 | 1 | 2 |
|---|---|---|---|---|---|
| **a** | 1.0 | 2.0 | 3.0 | NaN | NaN |
| **b** | 1.0 | 2.0 | 3.0 | NaN | NaN |
| **a** | NaN | NaN | NaN | 4.0 | 5.0 |
| **b** | NaN | NaN | NaN | 6.0 | 7.0 |
| **c** | NaN | NaN | NaN | 8.0 | 9.0 |

# Слияние датафреймов. Concat.Примеры

```
1 pd.concat([pd.DataFrame([[1,2,3]]),pd.DataFrame([[1],[2],[3]])],axis=0)
```

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 1 | 2.0 | 3.0 |
| **0** | 1 | NaN | NaN |
| **1** | 2 | NaN | NaN |
| **2** | 3 | NaN | NaN |

# Слияние датафреймов. Join.Примеры

```
1 df1=pd.DataFrame([[1,2,3],[1,2,3]],index=['a','b'])
2 df2=pd.DataFrame([[4,5],[6,7],[8,9]],index=['a','b','c'],columns=['1','2'])
```

```
1 df1.join(df2)
```

| | 0 | 1 | 2 | 1 | 2 |
|---|---|---|---|---|---|
| **a** | 1 | 2 | 3 | 4 | 5 |
| **b** | 1 | 2 | 3 | 6 | 7 |

# Слияние датафреймов. Merge.Примеры

```
1 df1=pd.DataFrame([[1,2,3],[1,2,3]],index=['a','b'])
2 df2=pd.DataFrame([[4,5],[6,7],[8,9]],index=['a','b','c'],columns=['1','2'])
```

```
1 df1.assign(d3=range(2)).merge(df2.assign(d4=range(3)),left_on='d3',right_on='d4')
```

| | 0 | 1 | 2 | d3 | 1 | 2 | d4 |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 3 | 0 | 4 | 5 | 0 |
| **1** | 1 | 2 | 3 | 1 | 6 | 7 | 1 |

# Группировка

Разбиение - это процесс, в котором мы разбиваем данные на группы, применяя к наборам данных определенные условия. Для того чтобы разделить данные, мы применяем определенные условия к наборам данных. Для разбиения данных мы используем функцию groupby(), которая позволяет разбить данные на группы по некоторым критериям.

Объекты Pandas могут быть разбиты по любой из своих осей. Абстрактное определение группировки состоит в том, чтобы обеспечить отображение меток на имена групп. Наборы данных Pandas могут быть разбиты на любые их объекты.

# Группировка таблиц в Pandas

```
[ ]  1 df['clean_name']=df.name.apply(lambda x : re.split('[-, a-zA-Z]',x)[0].lower())
```

```
     1 df.groupby(['clean_name'])['salary_from'].mean()
```

```
clean_name
                          NaN
администратор       52200.000000
аналитик           150000.000000
бариста             26100.000000
бортпроводник      104400.000000
бухгалтер           59900.000000
ведущий             85000.000000
водитель            70130.000000
```

# Функции агрегации

| **Функция** | **Описание** |
|---|---|
| .count() | Количество ненулевых записей |
| .sum() | Сумма |
| .mean() | Среднее |
| .median() | Медиана |
| .min() | Минимум |
| .max() | Максимум |
| .mode() | Мода |
| .std() | С.К.О. |
| .var() | Дисперсия |

# Стандартная группировка по одному столбцу

Стандартная группировка по одному столбцу

```
1 df1.groupby([df1.date.dt.day])['bid'].mean()
```

```
date
1     149.953106
2     112.823085
3     100.080212
4     100.830441
5     123.472790
6     400.000000
7     112.641938
8     113.138338
9     123.749251
10    132.570183
```

# Множественная группировка

Множественная группировка позволяет применять сразу несколько функций.

```
1 df1.groupby([df1.date.dt.day])['bid'].agg({'mean','max'})
```

| date | max | mean |
|---|---|---|
| 1 | 149.953106 | 149.953106 |
| 2 | 112.823085 | 112.823085 |
| 3 | 100.080212 | 100.080212 |
| 4 | 100.830441 | 100.830441 |
| 5 | 123.472790 | 123.472790 |
| 6 | 400.000000 | 400.000000 |
| 7 | 112.641938 | 112.641938 |
| 8 | 113.138338 | 113.138338 |
| 9 | 123.749251 | 123.749251 |

# Группировка с помощью lambda-функций

Группировка с помощью lambda-функций.

```
1 df1.groupby([df1.date.dt.day])['bid'].apply(lambda x: np.mean(x))
```

```
date
1     149.953106
2     112.823085
3     100.080212
4     100.830441
5     123.472790
6     400.000000
7     112.641938
8     113.138338
9     123.749251
10    132.570183
```

# Получение нескольких значений в группе

Получение нескольких значений в группе.

```
1 df1.groupby([df1.date.dt.month,'bid'])['bid'].count().groupby(level=0).head(2)
```

```
date  bid
1     100    1
      101    3
2     102    1
      103    1
Name: bid, dtype: int64
```

# Трансформация в группировке

Метод transform позволяет получить процент роста в каждой группе.

```
1 df['рост']/df.groupby(['пол'])['рост'].transform('sum')
```

```
0      0.006368
1      0.008491
2      0.009579
3      0.008105
4      0.014369
         ...
195    0.007863
196    0.013685
197    0.014198
198    0.016250
199    0.008298
Name: рост, Length: 200, dtype: float64
```

# Получение строковых значений

Lambda –функция позволяет получать объединение строк в каждой группе

```
1 df.groupby(['пол'])['образование'].apply(lambda x: ' '.join(x))
```

```
пол
жен    среднее среднее высшее среднее среднее среднее...
муж    среднее специальное среднее среднее специально...
Name: образование, dtype: object
```

# Ранжирование в каждой группе

Ранжирование в каждой группе

```
1 df.assign(rank=df.groupby(['пол'])['вес'].rank(ascending=False))
```

| | код респондента | дата опроса | пол | образование | социальный статус | возраст | рост | вес | rank |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | 2022-02-03 | жен | среднее | разведен/разведена | 41.0 | 132.0 | 41.40 | 120.0 |
| **1** | 31 | 2020-01-31 | жен | среднее | холост/незамужем | 59.0 | 176.0 | 83.20 | 28.5 |
| **2** | 45 | 2019-06-06 | муж | среднее специальное | холост/незамужем | 48.0 | 112.0 | 20.40 | 72.0 |
| **3** | 54 | 2019-06-06 | жен | высшее | разведен/разведена | 49.0 | 168.0 | 73.60 | 57.0 |
| **4** | 53 | 2022-02-03 | муж | среднее | холост/незамужем | 62.0 | 168.0 | 73.60 | 29.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **195** | 14 | 2022-02-03 | жен | высшее | холост/незамужем | 40.0 | 163.0 | 70.85 | 63.5 |
| **196** | 1 | 2020-01-31 | муж | незаконченное высшее | разведен/разведена | 63.0 | 160.0 | 65.00 | 43.0 |

# Создание сводной таблицы

Данные часто хранятся в виде нормализованных таблиц, для обеспечения меньшего занимаемого объема памяти. Однако модели машинного обучения требуют формы – объект (по строкам) и значения его характеристик по столбцам. Для этого можно использовать сводную таблицу (pivot table). Сводная таблица также используется для статистических тестов.

```
1 pd.pivot_table(data=df, index=['пол'],columns=['образование'],values=['возраст'],aggfunc='mean')
```

| | возраст | | | |
|---|---|---|---|---|
| образование | высшее | незаконченное высшее | среднее | среднее специальное |
| пол | | | | |
| жен | 39.944444 | 43.500000 | 47.431818 | 46.148148 |
| муж | 43.375000 | 49.444444 | 47.083333 | 45.580645 |

# Объекты Pandas. Сводная таблица

```
P=pd.DataFrame({'пол':np.random.choice(['муж','жен'],size=10),
                'возраст':np.random.randint(18,45,size=10),
                'id':range(10)})
pd.pivot_table(data=P,index=['пол'],columns=['возраст'],aggfunc='count',margins=True)
```

| | id | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| возраст | 20 | 21 | 24 | 31 | 38 | 41 | 42 | All |
| пол | | | | | | | | |
| жен | 1.0 | 2.0 | NaN | NaN | 1.0 | 2.0 | 1.0 | 7 |
| муж | NaN | 1.0 | 1.0 | 1.0 | NaN | NaN | NaN | 3 |
| All | 1.0 | 3.0 | 1.0 | 1.0 | 1.0 | 2.0 | 1.0 | 10 |

# Расплав (melt)

Расплав (melt) таблицы – это приведение ее к форме когда все значения ее индексов помещаются в один столбец, а ее значения – в другой. Часто применяется как промежуточный этап преобразования данных.

```
1 df.melt()
```

| | variable | value |
|---|---|---|
| 0 | код респондента | 58 |
| 1 | код респондента | 31 |
| 2 | код респондента | 45 |
| 3 | код респондента | 54 |
| 4 | код респондента | 53 |
| ... | ... | ... |
| 1595 | вес | 70.85 |
| 1596 | вес | 65.0 |
| 1597 | вес | 69.7 |

# Разрыв (explode)

```
1 pd.DataFrame([[[1,2,3]],[[5,6,7]]])
```

| | 0 |
|---|---|
| **0** | [1, 2, 3] |
| **1** | [5, 6, 7] |

```
1 pd.DataFrame([[[1,2,3]],[[5,6,7]]]).explode(0)
```

| | 0 |
|---|---|
| **0** | 1 |
| **0** | 2 |
| **0** | 3 |
| **1** | 5 |
| **1** | 6 |
| **1** | 7 |

# Спасибо за внимание!