Что будет

- сводные таблицы
- фильтры и вычисления с помощью метода loc
- фильтрация пустых значений через isnull
- время в pandas
- строковые методы
- немного про учет форм слов

Сводные таблицы

Прям как в экселе

In [1]:

```
import pandas as pd
```

In [2]:

```
ratings = pd.read_csv('ml-latest-small/ratings.csv')
ratings.head()
```

Out[2]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

Вопрос аналитику

Какие оценки поставил пользователь №1 и в каком количестве?

Что сделаем:

1. Фильтруем датафрейм ratings для userId = 1
2. Считаем для этого пользователя сколько он выставил единиц, двоек итд

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

Как это сделать для всех пользователей сразу:

In [3]:

```
ratings.pivot_table(index = 'userId', columns = 'rating', values = 'timestamp', aggfunc = '
```

Out[3]:

rating	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
userId										
1	0	2	0	7	3	4	1	3	0	0
2	0	2	0	4	0	36	0	23	0	11
3	0	0	0	1	3	18	9	11	4	5
4	0	5	0	5	0	23	0	52	0	119
5	0	0	1	0	3	3	27	42	19	5

In [4]:

```
# можно итоги добавить
ratings.pivot_table(index = 'userId', columns = 'rating', values = 'timestamp', aggfunc = '
                    margins = True).head()
```

Out[4]:

rating	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	All
userId											
1	0	2	0	7	3	4	1	3	0	0	20
2	0	2	0	4	0	36	0	23	0	11	76
3	0	0	0	1	3	18	9	11	4	5	51
4	0	5	0	5	0	23	0	52	0	119	204
5	0	0	1	0	3	3	27	42	19	5	100

Фильтры и вычисления с помощью loc и iloc

In [5]:

```
log = pd.read_csv('visit_log.csv', sep=';')
log.head()
```

Out[5]:

	timestamp	visit_id	url	region	user_id	traffic_source
0	1549980692	e3b0c44298	https://host.ru/3c19b4ef7371864fa3	Russia	b1613cc09f	yandex
1	1549980704	6e340b9cff	https://host.ru/c8d9213a31839f9a3a	Russia	4c3ec14bee	direct
2	1549980715	96a296d224	https://host.ru/b8b58337d272ee7b15	Russia	a8c40697fb	yandex
3	1549980725	709e80c884	https://host.ru/b8b58337d272ee7b15	Russia	521ac1d6a0	yandex
4	1549980736	df3f619804	https://host.ru/b8b58337d272ee7b15	Russia	d7323c571c	yandex

In [6]:

```
# метод loc позволяет выбрать строки и столбцы в соответствии с условиями
# двоеточие означает выбор всех значений

log.loc[:, ['user_id', 'region']].head()
```

Out[6]:

	user_id	region
0	b1613cc09f	Russia
1	4c3ec14bee	Russia
2	a8c40697fb	Russia
3	521ac1d6a0	Russia
4	d7323c571c	Russia

In [7]:

```
# iloc работает аналогично, но с указанием номера строки / столбца

log.iloc[:, [1, -3]].head()
```

Out[7]:

	visit_id	region
0	e3b0c44298	Russia
1	6e340b9cff	Russia
2	96a296d224	Russia
3	709e80c884	Russia
4	df3f619804	Russia

In [8]:

```
# пример фильтра на страну
# если столбцы не надо фильтровать, то второй параметр можно не указывать

log.loc[log.region == 'Russia'].head()
```

Out[8]:

	timestamp	visit_id	url	region	user_id	traffic_source
0	1549980692	e3b0c44298	https://host.ru/3c19b4ef7371864fa3	Russia	b1613cc09f	yandex
1	1549980704	6e340b9cff	https://host.ru/c8d9213a31839f9a3a	Russia	4c3ec14bee	direct
2	1549980715	96a296d224	https://host.ru/b8b58337d272ee7b15	Russia	a8c40697fb	yandex
3	1549980725	709e80c884	https://host.ru/b8b58337d272ee7b15	Russia	521ac1d6a0	yandex
4	1549980736	df3f619804	https://host.ru/b8b58337d272ee7b15	Russia	d7323c571c	yandex

In [9]:

```
# пример вычисления нового столбца с НДС для страны

log.loc[log.region == 'Russia', 'VAT'] = 1.2
log.head(10)
```

Out[9]:

	timestamp	visit_id	url	region	user_id	traffic_source
0	1549980692	e3b0c44298	https://host.ru/3c19b4ef7371864fa3	Russia	b1613cc09f	yandex
1	1549980704	6e340b9cff	https://host.ru/c8d9213a31839f9a3a	Russia	4c3ec14bee	direct
2	1549980715	96a296d224	https://host.ru/b8b58337d272ee7b15	Russia	a8c40697fb	yandex
3	1549980725	709e80c884	https://host.ru/b8b58337d272ee7b15	Russia	521ac1d6a0	yandex
4	1549980736	df3f619804	https://host.ru/b8b58337d272ee7b15	Russia	d7323c571c	yandex
5	1549980742	8855508aad	https://host.ru/df646c3676cc259fa0	Russia	fc43898e47	yandex
6	1549980742	b0f66adc83	https://host.ru/b8b58337d272ee7b15	Russia	13fc55e781	paid
7	1549980754	837885c8f8	https://host.ru/108ce4b365afb7b88e	Russia	cb5082b6f6	direct
8	1549980760	af5570f5a1	https://host.ru/3004a8273caeef2867	China	45664f7af2	direct
9	1549980765	3e7077fd2f	https://host.ru/df646c3676cc259fa0	Russia	6f9de8c8b6	email

In [10]:

```
# вариант с вычисляемым фильтром

log.loc[lambda row: row.region == 'Russia'].head(10)
```

Out[10]:

	timestamp	visit_id	url	region	user_id	traffic_sourc
0	1549980692	e3b0c44298	https://host.ru/3c19b4ef7371864fa3	Russia	b1613cc09f	yande
1	1549980704	6e340b9cff	https://host.ru/c8d9213a31839f9a3a	Russia	4c3ec14bee	dire
2	1549980715	96a296d224	https://host.ru/b8b58337d272ee7b15	Russia	a8c40697fb	yande
3	1549980725	709e80c884	https://host.ru/b8b58337d272ee7b15	Russia	521ac1d6a0	yande
4	1549980736	df3f619804	https://host.ru/b8b58337d272ee7b15	Russia	d7323c571c	yande
5	1549980742	8855508aad	https://host.ru/df646c3676cc259fa0	Russia	fc43898e47	yande
6	1549980742	b0f66adc83	https://host.ru/b8b58337d272ee7b15	Russia	13fc55e781	pa
7	1549980754	837885c8f8	https://host.ru/108ce4b365afb7b88e	Russia	cb5082b6f6	dire
9	1549980765	3e7077fd2f	https://host.ru/df646c3676cc259fa0	Russia	6f9de8c8b6	ema
12	1549980789	15ec7bf0b5	https://host.ru/8e88d4703848cc0ec4	Russia	6d1d81f7f8	pa

Скорость метода loc

Посчитаем средний рейтинг разных жанров. Метод merge взят из материалов следующего занятия ¯_(ツ)_/¯

In [11]:

```
ratings = pd.read_csv('ml-latest-small/ratings.csv')
movies = pd.read_csv('ml-latest-small/movies.csv')
joined = ratings.merge(movies, how='left', on='movieId')
joined.head()
```

Out[11]:

	userId	movieId	rating	timestamp	title	genres
0	1	31	2.5	1260759144	Dangerous Minds (1995)	Drama
1	1	1029	3.0	1260759179	Dumbo (1941)	Animation|Children|Drama|Musical
2	1	1061	3.0	1260759182	Sleepers (1996)	Thriller
3	1	1129	2.0	1260759185	Escape from New York (1981)	Action|Adventure|Sci-Fi|Thriller
4	1	1172	4.0	1260759205	Cinema Paradiso (Nuovo cinema Paradiso) (1989)	Drama

In [12]:

```
joined['Adventure'] = joined.apply(lambda row: row.rating if 'Adventure' in row.genres else
joined.head()
```

Out[12]:

	userId	movieId	rating	timestamp	title	genres	Adventure
0	1	31	2.5	1260759144	Dangerous Minds (1995)	Drama	NaN
1	1	1029	3.0	1260759179	Dumbo (1941)	Animation|Children|Drama|Musical	NaN
2	1	1061	3.0	1260759182	Sleepers (1996)	Thriller	NaN
3	1	1129	2.0	1260759185	Escape from New York (1981)	Action|Adventure|Sci-Fi|Thriller	2.0
4	1	1172	4.0	1260759205	Cinema Paradiso (Nuovo cinema Paradiso) (1989)	Drama	NaN

Можно и через loc

In [13]:

```
joined['Adventure'] = joined.loc[joined.genres.str.contains('Adventure'), 'rating']
joined.head()
```

Out[13]:

	userId	movieId	rating	timestamp	title	genres	Adventure
0	1	31	2.5	1260759144	Dangerous Minds (1995)	Drama	NaN
1	1	1029	3.0	1260759179	Dumbo (1941)	Animation|Children|Drama|Musical	NaN
2	1	1061	3.0	1260759182	Sleepers (1996)	Thriller	NaN
3	1	1129	2.0	1260759185	Escape from New York (1981)	Action|Adventure|Sci-Fi|Thriller	2.0
4	1	1172	4.0	1260759205	Cinema Paradiso (Nuovo cinema Paradiso) (1989)	Drama	NaN

Для замера времени сделаем эти операции для набор жанров

In [14]:

```
genres = ['Adventure', 'Animation', 'Children', 'Drama', 'Musical', 'Thriller']
```

In [15]:

```
%%time
joined = ratings.merge(movies, how='left', on='movieId')

for genre in genres:
    joined[genre] = joined.apply(lambda row: row.rating if genre in row.genres else None, a
```

```
CPU times: user 12.7 s, sys: 196 ms, total: 12.9 s
Wall time: 16.1 s
```

In [16]:

```
%%time
joined = ratings.merge(movies, how='left', on='movieId')

for genre in genres:
    joined[genre] = joined.loc[joined.genres.str.contains(genre), 'rating']
```

```
CPU times: user 382 ms, sys: 12.6 ms, total: 395 ms
Wall time: 502 ms
```

Методы isnull, isna

Определение пустых или None значений. По сути одинаковые методы

In [17]:

```
import numpy as np
```

In [18]:

```
df = pd.DataFrame({'value': [123, None, np.nan, np.NaN, np.NAN, 456]})
df
```

Out[18]:

	value
0	123.0
1	NaN
2	NaN
3	NaN
4	NaN
5	456.0

In [19]:

```
# фильтр на пустые значения в столбце value

df.loc[pd.isnull(df.value), :]
```

Out[19]:

	value
1	NaN
2	NaN
3	NaN
4	NaN

Дата и время в pandas

In [20]:

```
log['date'] = pd.to_datetime(log['timestamp'], unit='s')
log.head()
```

Out[20]:

	timestamp	visit_id	url	region	user_id	traffic_source
0	1549980692	e3b0c44298	https://host.ru/3c19b4ef7371864fa3	Russia	b1613cc09f	yande
1	1549980704	6e340b9cff	https://host.ru/c8d9213a31839f9a3a	Russia	4c3ec14bee	direc
2	1549980715	96a296d224	https://host.ru/b8b58337d272ee7b15	Russia	a8c40697fb	yande
3	1549980725	709e80c884	https://host.ru/b8b58337d272ee7b15	Russia	521ac1d6a0	yande
4	1549980736	df3f619804	https://host.ru/b8b58337d272ee7b15	Russia	d7323c571c	yande

In [21]:

```
# столбец datetime64[ns] теперь имеет тип даты
log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18938 entries, 0 to 18937
Data columns (total 8 columns):
timestamp         18938 non-null int64
visit_id          18938 non-null object
url               18938 non-null object
region            18938 non-null object
user_id           18938 non-null object
traffic_source    18938 non-null object
VAT               10913 non-null float64
date              18938 non-null datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 1.2+ MB
```

In [22]:

```
# получим час визита

log['hour'] = log.date.dt.hour
log.head()
```

Out[22]:

	timestamp	visit_id	url	region	user_id	traffic_source
0	1549980692	e3b0c44298	https://host.ru/3c19b4ef7371864fa3	Russia	b1613cc09f	yandex
1	1549980704	6e340b9cff	https://host.ru/c8d9213a31839f9a3a	Russia	4c3ec14bee	direct
2	1549980715	96a296d224	https://host.ru/b8b58337d272ee7b15	Russia	a8c40697fb	yandex
3	1549980725	709e80c884	https://host.ru/b8b58337d272ee7b15	Russia	521ac1d6a0	yandex
4	1549980736	df3f619804	https://host.ru/b8b58337d272ee7b15	Russia	d7323c571c	yandex

Методы работы со строками

In [23]:

```
stats = pd.read_csv('keywords.csv')
stats.head()
```

Out[23]:

	keyword	shows
0	вк	64292779
1	одноклассники	63810309
2	порно	41747114
3	ютуб	39995567
4	вконтакте	21014195

Проверка наличия подстроки в строке в питоне:

In [24]:

```
'охотник' in 'каждый охотник желает знать...'
```

Out[24]:

```
True
```

Аналог в pandas:

In [25]:

```
stats[stats.keyword.str.contains('охотник')].head()
```

Out[25]:

	keyword	shows
3072	сумеречные охотники 2 сезон	71965
3474	сумеречные охотники	66083
3654	белоснежка и охотник 2 фильм 2016	63154
4178	последний охотник на ведьм	57560
6127	последний охотник на ведьм фильм 2015	42213

[Документация (https://www.geeksforgeeks.org/python-pandas-series-str-contains/)](https://www.geeksforgeeks.org/python-pandas-series-str-contains/)

Syntax: Series.str.contains(pat, case=True, flags=0, na=nan, regex=True)

Parameter :

- pat : Character sequence or regular expression.
- case : If True, case sensitive.
- flags : Flags to pass through to the re module, e.g. re.IGNORECASE.
- na : Fill value for missing values.
- regex : If True, assumes the pat is a regular expression.

In [26]:

```
# поиск одного из нескольких слов

stats[stats.keyword.str.contains('охотник|фильм|2016')].head()
```

Out[26]:

	keyword	shows
20	фильмы 2016	4486635
51	фильмы	2156641
54	фильмы онлайн	2305540
68	смотреть фильмы онлайн	1928484
86	порно фильмы	1458031

replace

In [27]:

```
'отпуск начнется завтра'.replace('завтра', 'через месяц')
```

Out[27]:

```
'отпуск начнется через месяц'
```

Аналог в pandas на запросах про сериалы:

In [28]:

```
serial = stats[stats.keyword.str.contains('сериалы')]
serial.head()
```

Out[28]:

	keyword	shows
246	сериалы	587779
304	сериалы тут	503967
555	турецкие сериалы на русском языке	296403
881	русские сериалы	234262
890	сериалы онлайн	204812

In [29]:

```
serial.keyword.str.replace('сериалы', 'книги').head()
```

Out[29]:

```
246                            книги
304                       книги тут
555    турецкие книги на русском языке
881                     русские книги
890                      книги онлайн
Name: keyword, dtype: object
```

Как учитывать разное написание слов

Самое простое - методы upper и lower

In [30]:

```
serial.keyword.str.upper().head()
```

Out[30]:

```
246                            СЕРИАЛЫ
304                       СЕРИАЛЫ ТУТ
555    ТУРЕЦКИЕ СЕРИАЛЫ НА РУССКОМ ЯЗЫКЕ
881                     РУССКИЕ СЕРИАЛЫ
890                      СЕРИАЛЫ ОНЛАЙН
Name: keyword, dtype: object
```

In [31]:

```
serial.keyword.str.lower().head()
```

Out[31]:

```
246                            сериалы
304                       сериалы тут
555    турецкие сериалы на русском языке
881                     русские сериалы
890                      сериалы онлайн
Name: keyword, dtype: object
```

Что делать если нужно учесть формы написания слов?

In [32]:

```
stats[stats.keyword.str.contains('рубл')].head()
```

Out[32]:

	keyword	shows
1202	алиэкспресс на русском в рублях официальный сайт	161553
1602	курс гривны к рублю	125076
1736	али экспресс русская версия на русском в рубля...	117260
2132	доллары в рубли	114173
2172	100 долларов в рублях	97534

Почему нельзя просто оставить str.contains('рубл'):

In [33]:

```
non_financial_search = 'рубленая котлетка'
```

Библиотека pymystem (https://pypi.org/project/pymystem3/)

In [34]:

```
from pymystem3 import Mystem
```

In [35]:

```
search = 'курс гривны к рублю рубли рублях'
```

In [36]:

```
m = Mystem()
lemmas = m.lemmatize(search)
lemmas
```

Out[36]:

```
['курс',
 ' ',
 'гривна',
 ' ',
 'к',
 ' ',
 'рубль',
 ' ',
 'рубль',
 ' ',
 'рубль',
 '\n']
```

In [37]:

```
' '.join(lemmas)
```

Out[37]:

```
'курс   гривна   к   рубль   рубль   рубль \n'
```

Домашнее задание

В датафрейме data создайте столбец lemmas, в котором вычислите леммы поисковых запросов из столбца keyword. Леммы должны иметь строковый тип.

In [38]:

```
data = pd.DataFrame({
    'keyword': ['курс гривны к рублю', 'доллары в рубли', '100 долларов в рублях', 'курс ру
    'shows': [125076, 114173, 97534, 53546],
})
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```