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 [ ]:

```
import pandas as pd
```

In [ ]:

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

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 [ ]:

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

In [ ]:

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

Фильтры и вычисления с помощью loc и iloc

In [ ]:

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

In [ ]:

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

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

In [ ]:

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

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

In [ ]:

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

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

In [ ]:

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

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

In [ ]:

```
# вариант с вычисляемым фильтром

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

Скорость метода loc

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

In [ ]:

```
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()
```

In [ ]:

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

Можно и через loc

In [ ]:

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

Для замера времени сделаем эти операции для набор жанров

In [ ]:

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

In [ ]:

```
%%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
```

In [ ]:

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

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

Упражнение

Какие варианты источников трафика есть в столбце traffic_source?

Создайте столбец traffic_type, в котором для источников 'yandex' и 'google' будет стоять значение 'organic'. А для остальных вариантов - NaN.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

Методы isnull, isna

Определение пустых или None значений. По сути одинаковые методы

In [ ]:

```
import numpy as np
```

In [ ]:

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

In [ ]:

```
# фильтр на пустые значения в столбце value

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

Упражнение

Для пустых значений в столбце traffic_type выставьте значение 'other'

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

Дата и время в pandas

In [ ]:

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

In [ ]:

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

In [ ]:

```
# получим час визита

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

Методы работы со строками

In [ ]:

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

Проверка наличия подстроки в строке в питоне:

In [ ]:

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

Аналог в pandas:

In [ ]:

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

Документация (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 [ ]:

```
# поиск одного из нескольких слов

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

Упражнение

Отфильтруйте датафрейм stats по поисковым запросам, которые содержат строку "погода в" и упоминают один из городов: Москва, Новосибирск, Краснодар.

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

replace

In [ ]:

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

Аналог в pandas на запросах про сериалы:

In [ ]:

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

In [ ]:

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

Как учитывать разное написание слов

Самое простое - методы upper и lower

In [ ]:

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

In [ ]:

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

Что делать если нужно учесть формы написания слов?

In [ ]:

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

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

In [ ]:

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

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

In [ ]:

```
from pymystem3 import Mystem
```

In [ ]:

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

In [ ]:

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

In [ ]:

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

Домашнее задание 3

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

In [ ]:

```
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 [ ]:

```

```