Pandas и большие файлы

In [ ]:

```
import pandas as pd
```

Упражнение

Для каждого пользователя user_id из файла sales_db.csv посчитайте самую дорогую покупку (в столбце cost)

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Объединение датафреймов

Данные со слайдов

In [ ]:

```
visits = pd.DataFrame(
    {
        'user_id': [11, 22, 55, 11, 77],
        'source': ['ad', 'yandex', 'email', 'google', 'ad']
    }
)

visits = visits[['user_id', 'source']]
visits
```

In [ ]:

```
purchases = pd.DataFrame(
    {
        'user_id': [11, 22, 55, 11, 99],
        'category': ['Спорт', 'Авто', 'Дача', 'Спорт', 'Авто'],
    }
)

purchases = purchases[['user_id', 'category']]
purchases
```

In [ ]:

```
visits_grouped = visits.groupby('user_id').count()
visits_grouped.rename(columns={'source': 'visits'}, inplace=True)
visits_grouped
```

In [ ]:

```
visits.groupby('user_id').count().reset_index()
```

In [ ]:

```
purchases_pivot = purchases.pivot_table(index='user_id', columns='category', values='user_i
                                        aggfunc='size', fill_value=0)
purchases_pivot
```

In [ ]:

```
purchases_pivot.reset_index()
```

In [ ]:

```
visits_grouped.join(purchases_pivot)
```

LEFT join

Каждой строчке в левой таблице ищет соответствие в правой

In [ ]:

```
visits_grouped.join(purchases_pivot, how='left')
```

RIGHT join

Каждой строчке в правой таблице ищет соответствие в левой

In [ ]:

```
visits_grouped.join(purchases_pivot, how='right')
```

Упражнение

Дана статистика:

- ID клиентов и их имена (датафрейм clients)
- статистика доходов (earnings)
- статистика расходов (spending)

Определите имена клиентов, расходы которых превышают доходы.

In [ ]:

```
# подсказка - по умолчанию в методе merge объединение НЕ left join

?pd.DataFrame.merge
```

In [ ]:

```
clients = pd.DataFrame(
    {
        'id': [43018, 48329, 51043, 74943, 75029],
        'name': ['Марков Илья', 'Зарицкая Елизавета', 'Благова Дарья', 'Слепова Елена', 'Го
    }
)

clients
```

In [ ]:

```
earnings = pd.DataFrame(
    {
        'id': [51043, 48329, 74943, 75029, 43018],
        'debit': [34500, 12400, 89044, 5355, 19800],
    }
)

earnings
```

In [ ]:

```
spending = pd.DataFrame(
    {
        'id': [51043, 48329, 74943, 75029, 43018],
        'credit': [22990, 2500, 69880, 6000, 29000],
    }
)

spending
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

INNER join

Оставляет строчки, которые есть в обеих таблицах

In [ ]:

```
visits_grouped.join(purchases_pivot, how='inner')
```

Outer join

Оставляет все строчки

In [ ]:

```
visits_grouped.join(purchases_pivot, how='outer')
```

Конкатенация таблиц

In [ ]:

```
a = pd.DataFrame({'date': ['2020-01-01', '2020-01-02', '2020-01-03'], 'value_a': [1, 2, 3]}
b = pd.DataFrame({'date': ['2020-01-01', '2020-01-02', '2020-01-03'], 'value_b': [3, 4, 5]}
```

In [ ]:

```
a
```

In [ ]:

```
b
```

In [ ]:

```
pd.concat([a, b])
```

In [ ]:

```
# объединение по горизонтали
pd.concat([a, b], axis=1)
```

Дубликаты при объединении таблиц

In [ ]:

```
ratings = pd.read_csv('ratings_example.txt', sep = '\t')
ratings.head()
```

In [ ]:

```
movies = pd.read_csv('movies_example.txt', sep = '\t')
movies.head()
```

In [ ]:

```
# ¯\_(ツ)_/¯

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

In [ ]:

```
movies.drop_duplicates(subset = 'movieId', keep = 'first', inplace = True)
movies.head()
```

In [ ]:

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

In [ ]:

```
ratings.merge(movies, how = 'right', on = 'movieId')
```

Упражнение

Объедините датафреймы с визитами и покупками на сайте по ключу date. Обратите внимание, что в датафрейме визитов имеются дубликаты по дате.

In [ ]:

```
visits = pd.DataFrame(
    {'date': ['2019-11-01', '2019-11-01', '2019-11-02', '2019-11-02', '2019-11-03'],
     'source': ['organic', 'paid', 'organic', 'paid', 'organic'],
     'visits': [16825, 1952, 21890, 376, 19509]}
)

visits
```

In [ ]:

```
orders = pd.DataFrame(
    {'date': ['2019-11-01', '2019-11-02', '2019-11-03'],
     'orders': [198, 225, 201]}
)

orders
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Оптимизация хранения данных

In [ ]:

```
# 2.4mb
ratings = pd.read_csv('ml-latest-small/ratings.csv')

# 0.5mb
movies = pd.read_csv('ml-latest-small/movies.csv')
joined = ratings.merge(movies, how='left', on='movieId')
```

In [ ]:

```
joined.head()
```

In [ ]:

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

In [ ]:

```
len(ratings)
```

In [ ]:

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

In [ ]:

```
len(movies)
```

In [ ]:

```
len(ratings) == len(joined)
```

In [ ]:

```
joined.to_csv('joined_ratings.csv', index=False)
```

In [ ]:

```
joined.head()
```

In [ ]:

```
logs = joined[['userId', 'movieId', 'rating']].head()
```

In [ ]:

```
len(joined[['movieId', 'title', 'genres']].drop_duplicates())
```

Какой жанр имеет самые высокие рейтинги?

In [ ]:

```
import numpy as np
```

In [ ]:

```
genres = ['Drama', 'Action', 'Thriller']
```

In [ ]:

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

In [ ]:

```
len(ratings)
```

In [ ]:

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

In [ ]:

```
len(movies)
```

In [ ]:

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

In [ ]:

```
# рекомендуемая проверка на возможные дубликаты

len(ratings) == len(joined)
```

Считаем рейтинг жанров

In [ ]:

```
# еще раз список жанров

genres = ['Drama', 'Action', 'Thriller']
```

In [ ]:

```
def genres_ratings(row):
    """Возвращает рейтинг, если он есть в списке жанров данного фильма"""

    return pd.Series([row['rating'] if genre in row['genres'] else np.NaN for genre in genr
```

In [ ]:

```
%%time
joined[genres] = joined.apply(genres_ratings, axis=1)
```

In [ ]:

```
def genres_ratings_version_2(row):
    """Возвращает рейтинг, если он есть в списке жанров данного фильма"""

    for genre in genres:
        if genre in row.genres:
            row[genre] = row.rating

    return rating
```

In [ ]:

```
joined[genres] = joined.apply(genres_ratings, axis=1)
joined.head()
```

Упражнение

Выведите средний рейтинг каждого жанра из списка genres

In [ ]:

In [ ]:

In [ ]:

К домашнему заданию, задача 2

Дана статистика услуг перевозок клиентов компании по типам:

- rzd - железнодорожные перевозки
- auto - автомобильные перевозки
- air - воздушные перевозки
- client_base - адреса клиентов

In [ ]:

```
rzd = pd.DataFrame(
    {
        'client_id': [111, 112, 113, 114, 115],
        'rzd_revenue': [1093, 2810, 10283, 5774, 981]
    }
)
rzd
```

In [ ]:

```
auto = pd.DataFrame(
    {
        'client_id': [113, 114, 115, 116, 117],
        'auto_revenue': [57483, 83, 912, 4834, 98]
    }
)
auto
```

In [ ]:

```
air = pd.DataFrame(
    {
        'client_id': [115, 116, 117, 118],
        'air_revenue': [81, 4, 13, 173]
    }
)
air
```

In [ ]:

```
client_base = pd.DataFrame(
    {
        'client_id': [111, 112, 113, 114, 115, 116, 117, 118],
        'address': ['Комсомольская 4', 'Энтузиастов 8а', 'Левобережная 1а', 'Мира 14', 'ЗЖБ
                    'Строителей 18', 'Панфиловская 33', 'Мастеркова 4']
    }
)
client_base
```

In [ ]:

In [ ]:

In [ ]: