```
# Чтение и запись файлов
# csv #- comma-separated values - данные разделенные запятой
# df = pd.read_csv('file.csv', # функция считывания внешнего файла формата csv (можно выбрать необходимый формат)
#                  encoding='windows-1251',
#                  sep = ';',
#                  index_col='название_столбца',
#                  parse_dates=['Date'],
#                  dayfirst=True)
# 'file.csv' - путь к файлу,
# sep - разделитель sep(по умолчаниию ',')
# encoding – параметр в read_csv, отвечает за кодировку текста, которая может быть различной. Самая распространённая – utf
# index_col='название_столбца' - название столбца, который будет выступать как столбец индексов
# index_col=[0] - индекс столбца, который будет выступать как столбец индексов
# parse_dates – указывает, стоит ли воспринимать даты как даты (по умолчанию они воспринимаются пандасом как строки).
    # пример pd.read_csv(path, parse_dates=['some_date', 'another_date'])
    # Параметр с датами может принимать несколько значений:
    # True – пытается перевести в дату первую колонку
    # список колонок – parse_dates=['some_date', 'another_date']
    # пытается перевести в дату указанные в списке колонки и столбцы create_data, payment_data
    # будут обрабатываться как даты
# dayfirst=True  - первое значение в дате это день или нет - True/False
# df['Date'].dt.name - номер дня недели в соответствии с данными в колонке с датами
# df['Date'].dt.name() - название дня недели в соответствии с данными в колонке с датами
# df['Date'].dt.month - номер месяца в соответствии с данными в колонке с датами
# df['Date'].dt.month() - название месяца в соответствии с данными в колонке с датами
```

Задача №57. Решение в группах

1. Прочесть с помощью pandas файл california_housing_test.csv, который находится в папке sample_data
2. Посмотреть сколько в нем строк и столбцов
3. Определить какой тип данных имеют столбцы

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('/content/sample_data/california_housing_test.csv')
df.sample(n = 5)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
1584	-120.42	34.91	4.0	6986.0	1217.0	2801
1077	-122.30	37.81	52.0	572.0	109.0	274
2562	-117.82	33.81	25.0	2662.0	402.0	1247
2584	-118.53	34.44	19.0	1285.0	195.0	650
2759	-116.99	33.20	17.0	2980.0	539.0	1531

```
df.shape
```

```
(3000, 9)
```

```
df.dtypes
```

```
longitude             float64
latitude              float64
housing_median_age    float64
total_rooms           float64
total_bedrooms        float64
population            float64
households            float64
median_income         float64
median_house_value    float64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           3000 non-null   float64
 1   latitude            3000 non-null   float64
 2   housing_median_age  3000 non-null   float64
 3   total_rooms         3000 non-null   float64
 4   total_bedrooms      3000 non-null   float64
```

```
 5   population          3000 non-null   float64
 6   households          3000 non-null   float64
 7   median_income       3000 non-null   float64
 8   median_house_value  3000 non-null   float64
dtypes: float64(9)
memory usage: 211.1 KB
```

Задача №59. Решение в группах

1. Проверить есть ли в файле пустые значения
2. Показать median_house_value где median_income < 2
3. Показать данные в первых 2 столбцах
4. Выбрать данные где housing_median_age < 20 и median_house_value > 70000

```
# Проверить есть ли в файле пустые значения
df.isna().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

```
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.05	37.37	27.0	3885.0	661.0	1537.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0
3	-118.36	33.82	28.0	67.0	15.0	49.0
4	-119.67	36.33	19.0	1241.0	244.0	850.0

```
# Показать median_house_value где median_income < 2
df[df['median_income'] < 2]['median_house_value']
```

```
5        67000.0
6        67000.0
16      181300.0
28      350000.0
43       79300.0
          ...
2943     57200.0
2964     91300.0
2985    109400.0
2986     85400.0
2995    225000.0
Name: median_house_value, Length: 360, dtype: float64
```

```
df.loc[df['median_income'] < 2, 'median_house_value']
```

```
5        67000.0
6        67000.0
16      181300.0
28      350000.0
43       79300.0
          ...
2943     57200.0
2964     91300.0
2985    109400.0
2986     85400.0
2995    225000.0
Name: median_house_value, Length: 360, dtype: float64
```

```
# Показать данные в первых 2 столбцах
df[['longitude', 'latitude']]
```

	longitude	latitude
0	-122.05	37.37
1	-118.30	34.26
2	-117.81	33.78
3	-118.36	33.82
4	-119.67	36.33
...	...	...
2995	-119.86	34.42
2996	-118.14	34.06
2997	-119.70	36.30
2998	-117.12	34.10
2999	-119.63	34.42

3000 rows × 2 columns

```
df.columns[:2]
```

```
Index(['longitude', 'latitude'], dtype='object')
```

```
df[df.columns[:2]]
```

	longitude	latitude
0	-122.05	37.37
1	-118.30	34.26
2	-117.81	33.78
3	-118.36	33.82
4	-119.67	36.33
...	...	...
2995	-119.86	34.42
2996	-118.14	34.06
2997	-119.70	36.30
2998	-117.12	34.10
2999	-119.63	34.42

3000 rows × 2 columns

```
df.loc[:,['longitude', 'latitude']]
```

	longitude	latitude
0	-122.05	37.37
1	-118.30	34.26
2	-117.81	33.78
3	-118.36	33.82
4	-119.67	36.33
...	...	...
2995	-119.86	34.42
2996	-118.14	34.06
2997	-119.70	36.30
2998	-117.12	34.10
2999	-119.63	34.42

3000 rows × 2 columns

```
df.loc[:, :'latitude']
```

	longitude	latitude
0	-122.05	37.37
1	-118.30	34.26
2	-117.81	33.78
3	-118.36	33.82
4	-119.67	36.33
...	...	...
2995	-119.86	34.42
2996	-118.14	34.06
2997	-119.70	36.30
2998	-117.12	34.10
2999	-119.63	34.42

3000 rows × 2 columns

```
df.iloc[:3, :2]
```

	longitude	latitude
0	-122.05	37.37
1	-118.30	34.26
2	-117.81	33.78

```
# Выбрать данные где housing_median_age < 20 и median_house_value > 70000
df[(df['housing_median_age'] < 20) & (df['median_house_value'] > 70000)]
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_valı
4	-119.67	36.33	19.0	1241.0	244.0	850.0	237.0	2.9375	81700
7	-120.65	35.48	19.0	2310.0	471.0	1341.0	441.0	3.2250	166900
8	-122.84	38.40	15.0	3080.0	617.0	1446.0	599.0	3.6696	194400
13	-117.03	32.97	16.0	3936.0	694.0	1935.0	659.0	4.5625	231200
16	-120.81	37.53	15.0	570.0	123.0	189.0	107.0	1.8750	181300
...	...	...	...	...	...	...	...	...	
2978	-121.34	38.64	17.0	2761.0	501.0	1128.0	482.0	3.7562	139700
2981	-120.66	35.49	17.0	4422.0	945.0	2307.0	885.0	2.8285	171300
2984	-117.59	33.88	13.0	3239.0	849.0	2751.0	813.0	2.6111	107000
2985	-120.47	34.94	17.0	1368.0	308.0	642.0	303.0	1.8633	109400
2991	-117.17	34.28	13.0	4867.0	718.0	780.0	250.0	7.1997	253800

792 rows × 9 columns

Задача №61. Решение в группах

1. Определить какое максимальное и минимальное значение median_house_value
2. Показать максимальное median_house_value, где median_income = 3.1250
3. Узнать какая максимальная population в зоне минимального значения median_house_value

```
df['median_house_value'].min(), df['median_house_value'].max()
```

```
(22500.0, 500001.0)
```

```
df[df['median_income'] == 3.1250]['median_house_value'].max()
```

```
233300.0
```

```
df[df['median_house_value'] == df['median_house_value'].min()]['population'].max()
```

```
1230.0
```

Домашнее задание

```
df[df['population'] <= 500]['median_house_value'].mean()
```

```
190157.0918032787
```

```
df[df['population'] <= df['population'].min()]['households'].max()
```

```
3.0
```