In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
df = pd.read_csv('data/water_potability.csv')
```

Анализ датасета

In [ ]:
```
df.shape
```

Out[ ]:
```
(3276, 10)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

In [ ]:
```
from utils import get_df_info

get_df_info(df)
```

Столбец ph (тип float64) имеет 491 пропусков из 3276 значений, 14.987789987789988% (индекс 0)

Столбец Hardness (тип float64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 1)

Столбец Solids (тип float64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 2)

Столбец Chloramines (тип float64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 3)

Столбец Sulfate (тип float64) имеет 781 пропусков из 3276 значений, 23.84004884004884% (индекс 4)

Столбец Conductivity (тип float64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 5)

Столбец Organic_carbon (тип float64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 6)

Столбец Trihalomethanes (тип float64) имеет 162 пропусков из 3276 значений, 4.945054945054945% (индекс 7)

Столбец Turbidity (тип float64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 8)

Столбец Potability (тип int64) имеет 0 пропусков из 3276 значений, 0.0% (индекс 9)

In [ ]:
```
df.head(20)
```

Out[ ]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12
8	NaN	118.988579	14285.583854	7.804174	268.646941	389.375566	12
9	11.180284	227.231469	25484.508491	9.077200	404.041635	563.885481	17
10	7.360640	165.520797	32452.614409	7.550701	326.624353	425.383419	15
11	7.974522	218.693300	18767.656682	8.110385	NaN	364.098230	14
12	7.119824	156.704993	18730.813653	3.606036	282.344050	347.715027	15.
13	NaN	150.174923	27331.361962	6.838223	299.415781	379.761835	19
14	7.496232	205.344982	28388.004887	5.072558	NaN	444.645352	13
15	6.347272	186.732881	41065.234765	9.629596	364.487687	516.743282	11
16	7.051786	211.049406	30980.600787	10.094796	NaN	315.141267	20
17	9.181560	273.813807	24041.326280	6.904990	398.350517	477.974642	13
18	8.975464	279.357167	19460.398131	6.204321	NaN	431.443990	12
19	7.371050	214.496610	25630.320037	4.432669	335.754439	469.914551	12

Как видно, в данном датасете есть 3 столбца с пустыми значениями. Столбец "Sulfate" трогать нет смысла, т.к. количество пропусков составляет почти 25%, и таким образом после 'лёгкой' обработки данные сильно исказятся. Попробуем обработать пропуски в столбцах "ph" и "Trihalomethanes".

Обработка пропусков

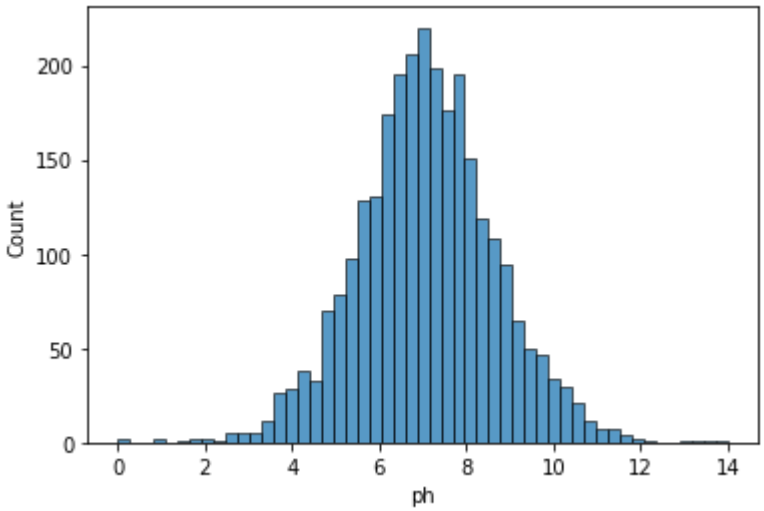
In [ ]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [ ]:
```
import math

def get_values_info(df: pd.DataFrame, column_name: str, lines: int = 10):
    value_count_list = list()
    print("Для датафрейма, столбец {0}:".format(column_name))
    sum = 0
    unique_count = 0

    # Формируем список из кортежей
    for value in df[column_name].unique():
        unique_count = unique_count + 1
        if (isinstance(value, float) and math.isnan(value)):
            temp = df[pd.isna(df[column_name])]
        else:
            temp = df[df[column_name] == value]
            sum = sum + temp.shape[0]

        value_count_list.append((value, temp.shape[0]))

    # Сортируем по убыванию на основе второго поля кортежа
    value_count_list = sorted(value_count_list, reverse = True, key = lambda

    line = 0
    # Смотрим на результат
    for element in value_count_list:
        if line == lines:
            break
        line += 1
        print(element[0], "->", element[1])

    print("Заполненных значений:", sum, "(из {})".format(df.shape[0]))
    print("Уникальных значений:", unique_count)
```

In [ ]:
```
def column_imputer(df: pd.DataFrame, column_name, strategy_name):
    column = df[[column_name]]
    mask = MissingIndicator().fit_transform(column)
    imputer = SimpleImputer(strategy = strategy_name)
    column_imputed = imputer.fit_transform(column)

    return column_imputed
```

In [ ]:
```
sns.histplot(df['ph'])
```

Out[ ]:
```
<AxesSubplot:xlabel='ph', ylabel='Count'>
```



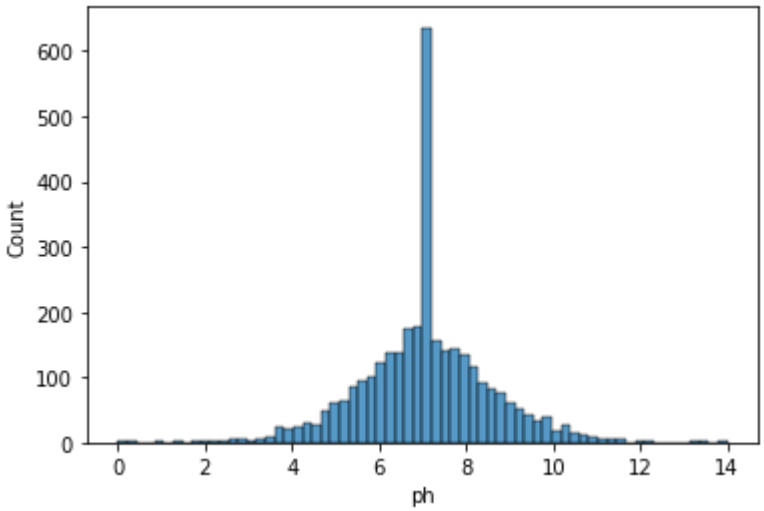
In [ ]:
```
get_values_info(df, 'ph')
```

```
Для датафрейма, столбец ph:
nan -> 491
3.71608007538699 -> 1
8.099124189298397 -> 1
8.316765884214679 -> 1
9.092223456290965 -> 1
5.584086638456089 -> 1
10.223862164528772 -> 1
8.635848718500734 -> 1
11.180284470721592 -> 1
7.360640105838258 -> 1
Заполненных значений: 2785 (из 3276)
Уникальных значений: 2786
```

In [ ]:
```
mean_df = df.copy()
mean_ph = column_imputer(mean_df, "ph", "mean")
mean_df["ph"] = mean_ph

get_values_info(mean_df, "ph")
sns.histplot(mean_df["ph"])
```

```
Для датафрейма, столбец ph:
7.080794504276835 -> 491
3.71608007538699 -> 1
8.099124189298397 -> 1
8.316765884214679 -> 1
9.092223456290965 -> 1
5.584086638456089 -> 1
10.223862164528772 -> 1
8.635848718500734 -> 1
11.180284470721592 -> 1
7.360640105838258 -> 1
Заполненных значений: 3276 (из 3276)
Уникальных значений: 2786
```

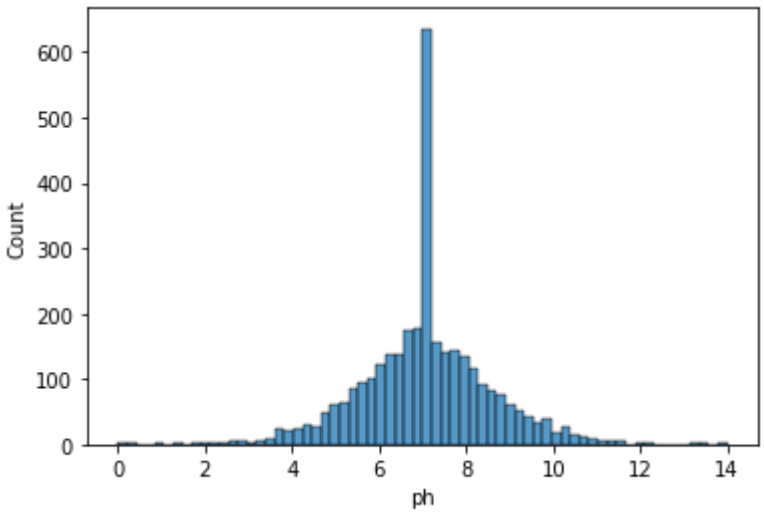
Out[ ]:
```
<AxesSubplot:xlabel='ph', ylabel='Count'>
```



In [ ]:
```
median_df = df.copy()
median_ph = column_imputer(median_df, "ph", "median")
median_df["ph"] = median_ph

get_values_info(median_df, "ph")
sns.histplot(median_df["ph"])
```

```
Для датафрейма, столбец ph:
7.036752103833548 -> 492
3.71608007538699 -> 1
8.099124189298397 -> 1
8.316765884214679 -> 1
9.092223456290965 -> 1
5.584086638456089 -> 1
10.223862164528772 -> 1
8.635848718500734 -> 1
11.180284470721592 -> 1
7.360640105838258 -> 1
Заполненных значений: 3276 (из 3276)
Уникальных значений: 2785
```

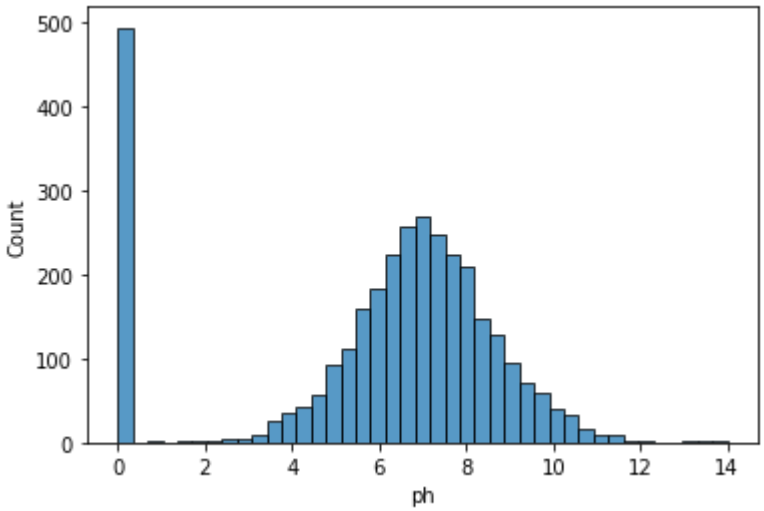
Out[ ]:
```
<AxesSubplot:xlabel='ph', ylabel='Count'>
```



In [ ]:
```
most_frequent_df = df.copy()
most_frequent_ph = column_imputer(most_frequent_df, "ph", "most_frequent")
most_frequent_df["ph"] = most_frequent_ph

get_values_info(most_frequent_df, "ph")
sns.histplot(most_frequent_df["ph"])
```

```
Для датафрейма, столбец ph:
0.0 -> 492
3.71608007538699 -> 1
8.099124189298397 -> 1
8.316765884214679 -> 1
9.092223456290965 -> 1
5.584086638456089 -> 1
10.223862164528772 -> 1
8.635848718500734 -> 1
11.180284470721592 -> 1
7.360640105838258 -> 1
Заполненных значений: 3276 (из 3276)
Уникальных значений: 2785
```

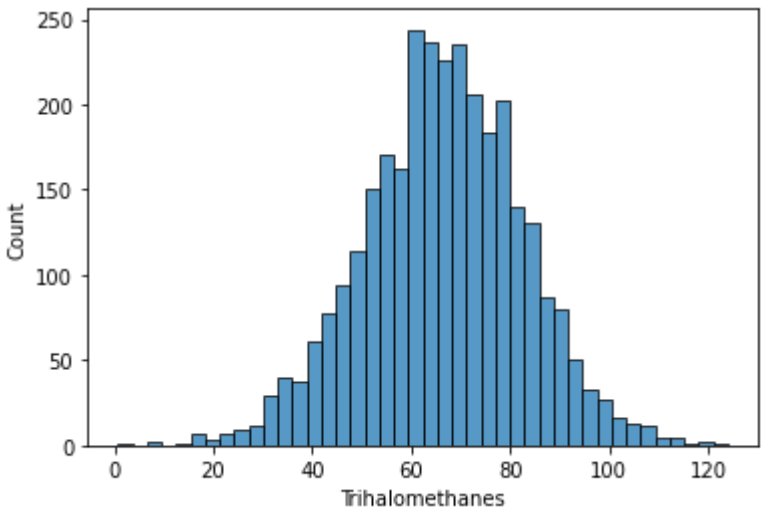
Out[ ]:
```
<AxesSubplot:xlabel='ph', ylabel='Count'>
```



Простая импьютация в данном случае сильно искажает данные, а заполнение нулями и вовсе даёт абсолютно неправильную статистику.

In [ ]:
```
tr = 'Trihalomethanes'
sns.histplot(df[tr])
```

Out[ ]:
```
<AxesSubplot:xlabel='Trihalomethanes', ylabel='Count'>
```



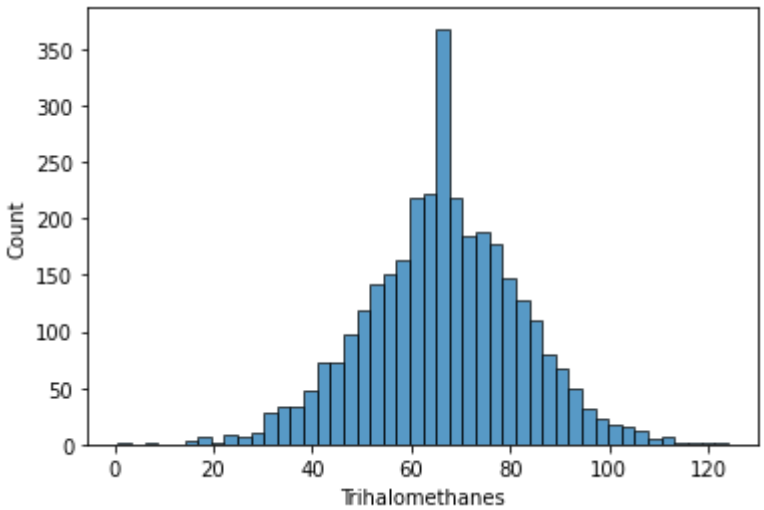
In [ ]:
```
get_values_info(df, tr)
```

```
Для датафрейма, столбец Trihalomethanes:
nan -> 162
86.9909704615088 -> 1
56.32907628451764 -> 1
66.42009251176368 -> 1
100.34167436508008 -> 1
31.997992727424737 -> 1
54.91786184199447 -> 1
84.60355617402357 -> 1
62.79830896292516 -> 1
53.92884576751224 -> 1
Заполненных значений: 3114 (из 3276)
Уникальных значений: 3115
```

In [ ]:
```
mean_df = df.copy()
mean_tr = column_imputer(mean_df, tr, "mean")
mean_df[tr] = mean_tr

get_values_info(mean_df, tr)
sns.histplot(mean_df[tr])
```

```
Для датафрейма, столбец Trihalomethanes:
66.39629294676803 -> 162
86.9909704615088 -> 1
56.32907628451764 -> 1
66.42009251176368 -> 1
100.34167436508008 -> 1
31.997992727424737 -> 1
54.91786184199447 -> 1
84.60355617402357 -> 1
62.79830896292516 -> 1
53.92884576751224 -> 1
Заполненных значений: 3276 (из 3276)
Уникальных значений: 3115
```

Out[ ]:
```
<AxesSubplot:xlabel='Trihalomethanes', ylabel='Count'>
```



Для данного столбца график после импьютации исказился меньше, но все равно изменился значительно.

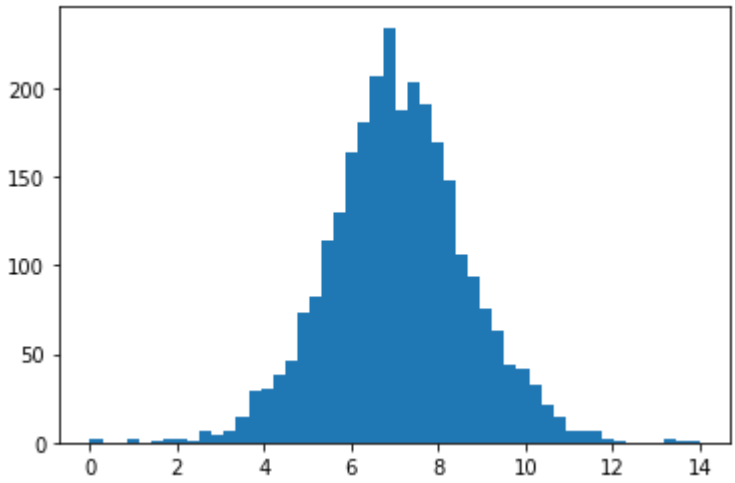
Масштабирование

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

In [ ]:
```
sc1 = MinMaxScaler()
```

```
sc1_data = sc1.fit_transform(df[['ph']])
```

In [ ]:
```
plt.hist(df['ph'], 50)
plt.show()
```



In [ ]:
```
plt.hist(sc1_data, 50)
plt.show()
```