Setup

In [3]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy import *

# Baca data dari CSV
df = pd.read_csv('water_potability.csv')

# Hapus data yang ada nullnya
df.dropna(inplace = True)

# Tambah nama Kolom
df.columns= ["id","pH","Hardness","Solids","Chloramines","Sulfate","Conductivity",
             "OrganicCarbon","Trihalomethanes","Turbidity","Potability"]
```

Soal 1

Membuat Visualisasi plot distribusi, dalam bentuk histogram dan boxplot untuk setiap kolom numerik. Berikan uraian penjelasan kondisi setiap kolom berdasarkan kedua plot tersebut.

In [4]:

```
#tipe data yang bersifat numerik
accepted_data_type = ["int","float","complex"]
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
'''
mean
median
modus
standar deviasi
variansi
range
minimum
maksimum
kuartil
IQR
skewness
kurtosis
'''
dataDesc = pd.DataFrame(columns = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
cols = df.columns[1:]
mean=[]
#mean.append("mean")
median=[]
#median.append("median")
modus=[]
#modus.append("modus")
std=[]
#std.append("standar deviasi")
var=[]
#var.append("variansi")
range=[]
#range.append("range")
min=[]
#min.append("minimum")
max=[]
#max.append("maksimum")
q1=[]
```

```
#q1.append("kuartil 1")
q3=[]
#q3.append("kuartil 3")
iqr=[]
#iqr.append("IQR")
skew=[]
#skew.append("skewness")
kurtosis=[]
#kurtosis.append("kurtosis")
for col in cols:
    mean.append(df[col].mean())
    median.append(df[col].median())
    modus.append(df[col].mode().iloc[0])
    std.append(df[col].std())
    var.append(df[col].var())
    range.append(df[col].max()-df[col].min())
    min.append(df[col].min())
    max.append(df[col].max())
    q1.append(df[col].quantile(0.25))
    q3.append(df[col].quantile(0.75))
    iqr.append(df[col].quantile(0.75)-df[col].quantile(0.25))
    skew.append(df[col].skew())
    kurtosis.append(df[col].kurt())

tempDf = pd.Series(mean, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(median, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate'
, 'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(modus, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(std, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', '
Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(var, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', '
Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(range, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(min, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', '
Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(max, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', '
Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(q1, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'C
onductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(q3, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'C
onductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(iqr, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', '
Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(skew, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
tempDf = pd.Series(kurtosis, index = ['pH', 'Hardness', 'Solids', 'Chloramines', 'Sulfat
e', 'Conductivity', 'OrganicCarbon', 'Trihalomethanes', 'Turbidity', 'Potability'])
dataDesc = dataDesc.append(tempDf, ignore_index = True)
dataDesc.index = ["mean", "median", "modus", "std", "var", "range", "min", "max", "q1",
"q3", "iqr", "skew", "kurtosis"]
print(dataDesc)
```

```
                pH     Hardness           Solids Chloramines      Sulfate  \
mean      7.086581   195.960048     21904.616822    7.133862   333.199592
median    7.027297   197.191839     20920.251561    7.140122   332.196048
modus     0.227499    73.492234       320.942611    1.390871        129.0
std       1.572955    32.648709      8627.545029    1.585474    41.217984
```

var	2.474188	1065.938177	74434533.222481	2.513729	1698.922197
range	13.772501	243.84589	56167.729801	11.736129	352.030642
min	0.227499	73.492234	320.942611	1.390871	129.0
max	14.0	317.338124	56488.672413	13.127	481.030642
q1	6.090016	176.736376	15613.160533	6.137757	307.621462
q3	8.052894	216.454108	27172.893573	8.11014	359.268543
iqr	1.962878	39.717732	11559.73304	1.972383	51.647081
skew	0.049475	-0.084567	0.590884	0.013777	-0.044964
kurtosis	0.6272	0.524698	0.335676	0.549179	0.78611

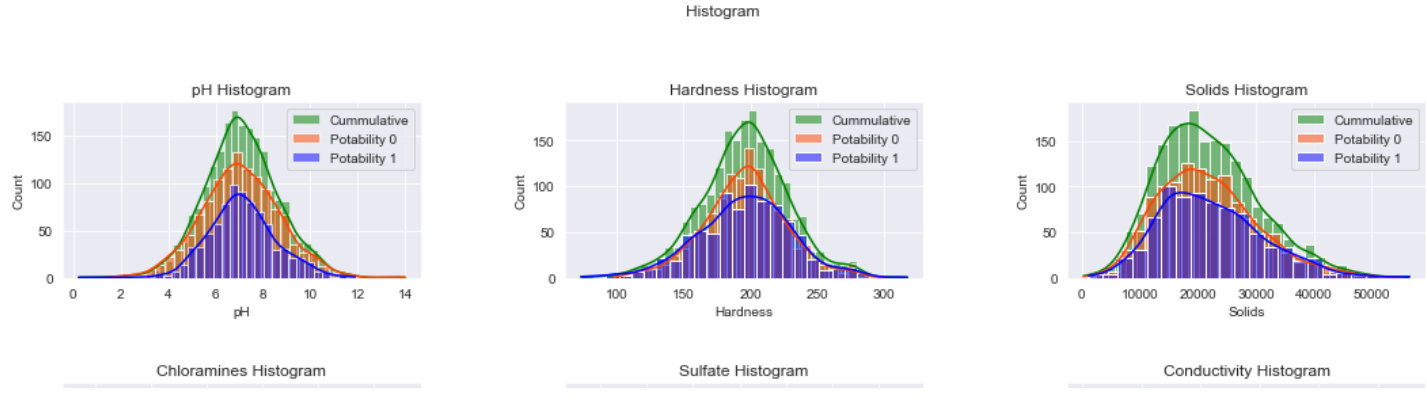
	Conductivity	OrganicCarbon	Trihalomethanes	Turbidity	Potability
mean	426.508172	14.35591	66.383822	3.969169	0.403186
median	423.455906	14.322019	66.421884	3.966571	0.0
modus	201.619737	2.2	8.577013	1.45	0.0
std	80.709632	3.325352	16.067261	0.780527	0.49066
var	6514.04474	11.057966	258.156867	0.609222	0.240747
range	551.722883	24.806707	115.422987	5.044749	1.0
min	201.619737	2.2	8.577013	1.45	0.0
max	753.34262	27.006707	124.0	6.494749	1.0
q1	366.802484	12.120956	55.947322	3.442848	0.0
q3	482.294404	16.682025	77.286551	4.513201	1.0
iqr	115.491921	4.561069	21.339228	1.070353	1.0
skew	0.267137	-0.019324	-0.053083	-0.031313	0.395023
kurtosis	-0.237497	0.03288	0.225597	-0.049507	-1.845796

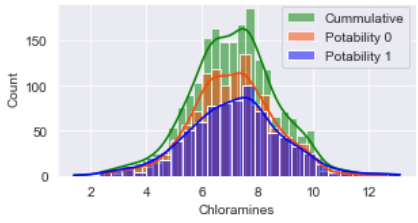
Soal 2

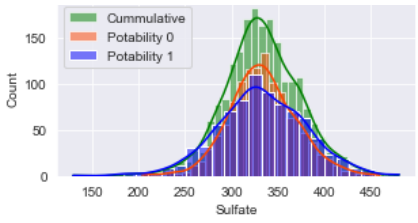
Membuat Visualisasi plot distribusi, dalam bentuk histogram dan boxplot untuk setiap kolom numerik. Berikan uraian penjelasan kondisi setiap kolom berdasarkan kedua plot tersebut.

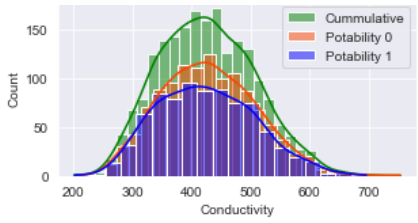
In [66]:

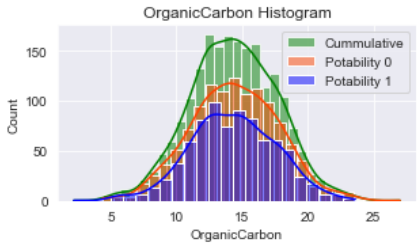
```
df0 = df[df["Potability"]==0]
df1 = df[df["Potability"]==1]

sns.set_style("darkgrid")
fig, axes = plt.subplots(3, 3, figsize=(18, 10))


fig.suptitle('Histogram')

i=0
for cols in df:
    if(df[cols].dtype in accepted_data_type):
        sns.color_palette("rocket_r", as_cmap=True)
        sns.histplot(ax=axes[i//3, i % 3], data=df, x=cols,color = 'green' ,label="Cummulative", kde=True).set_title(cols + " Histogram")
        sns.histplot(ax=axes[i//3, i % 3], data=df0, x=cols, color="orangered", label="Potability 0", kde=True)
        sns.histplot(ax=axes[i//3, i % 3], data=df1, x=cols,color = 'blue' ,label="Potability 1", kde=True)

        axes[i//3, i % 3].legend()
        i += 1
plt.subplots_adjust(hspace=0.6, wspace = 0.4)
plt.legend()
plt.show()
```

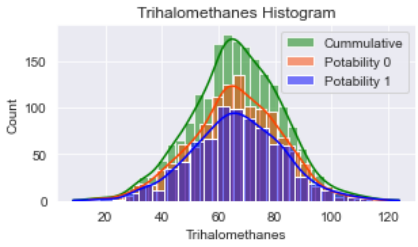


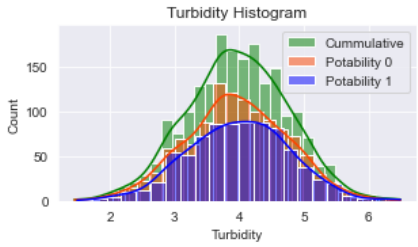






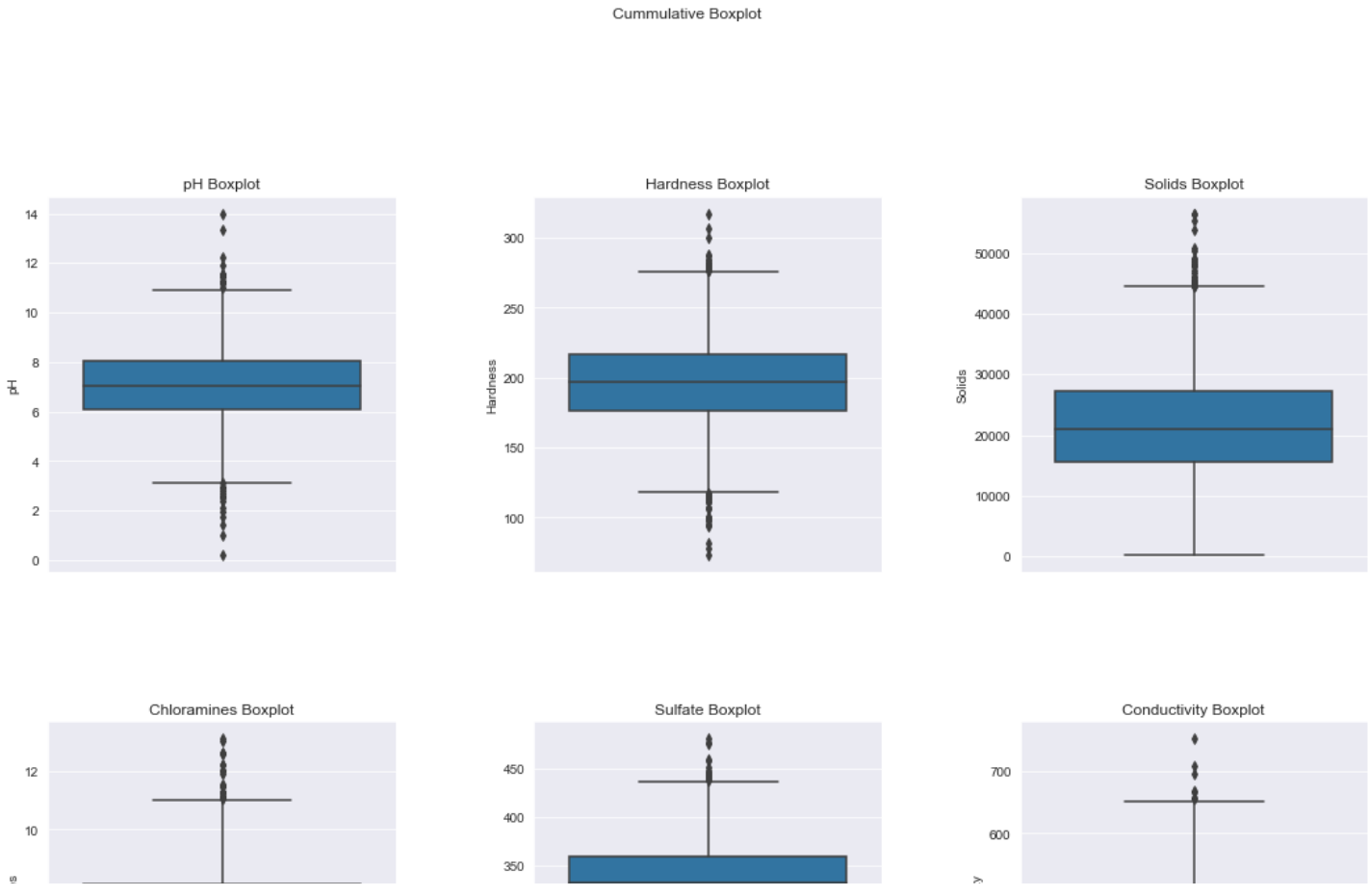


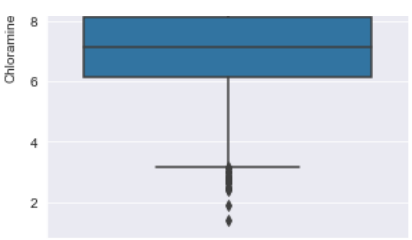


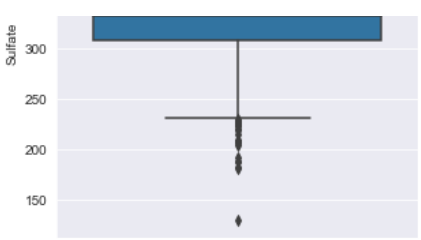


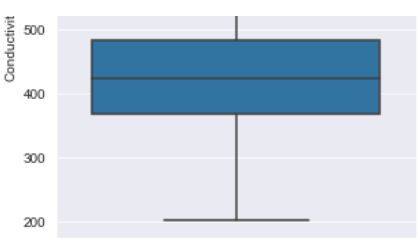
In [5]:

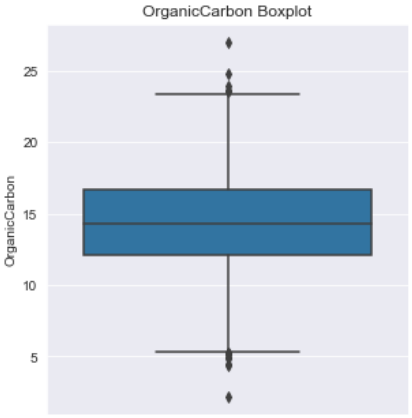
```
fig, axes = plt.subplots(3, 3, figsize=(18, 20))

fig.suptitle('Cummulative Boxplot')

i=0
for cols in df:
    if(df[cols].dtype in accepted_data_type):
        sns.boxplot(ax=axes[i//3, i % 3], y=cols, data=df).set_title(cols + " Boxplot")
        i += 1
plt.subplots_adjust(hspace=0.4, wspace = 0.4)

fig, axes = plt.subplots(3, 3, figsize=(18, 20))

fig.suptitle('Potability Grouped Boxplot')

i=0
for cols in df:
    if(df[cols].dtype in accepted_data_type):
        sns.boxplot(ax=axes[i//3, i % 3], x="Potability", y=cols, data=df).set_title(col
s + " Boxplot")
        i += 1
plt.subplots_adjust(hspace=0.4, wspace = 0.4)
```

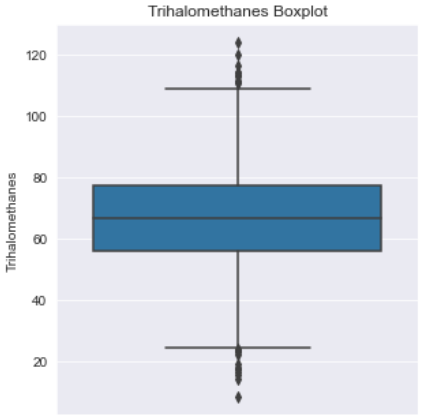


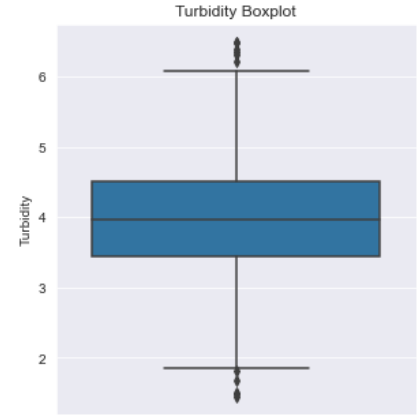




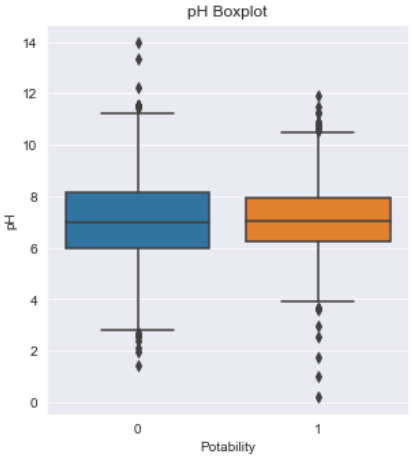


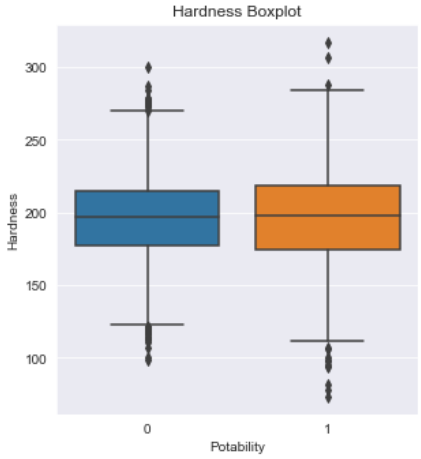


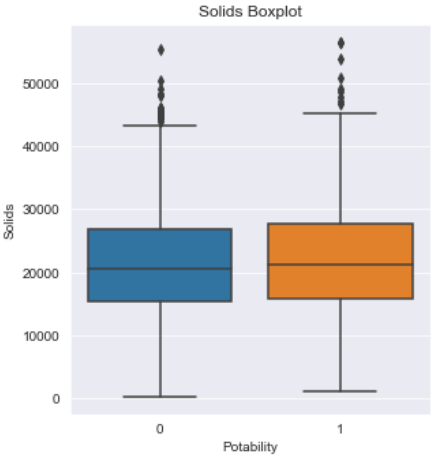


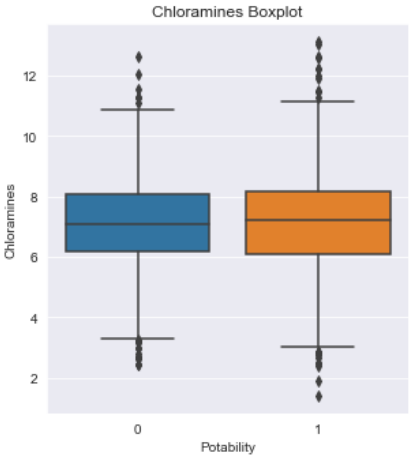


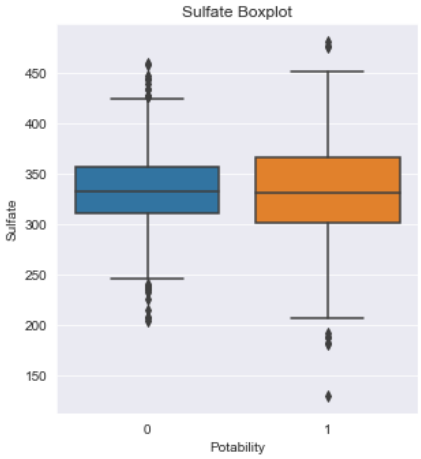
Potability Grouped Boxplot

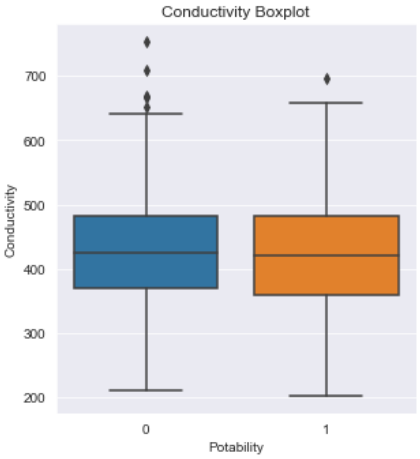


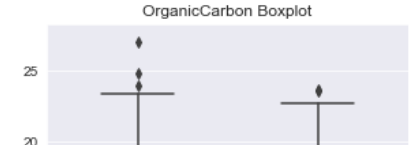


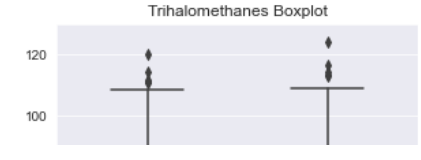


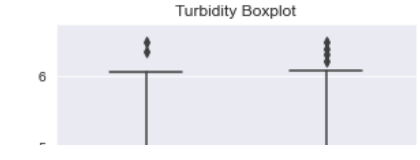


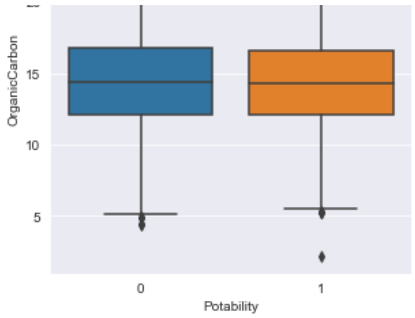


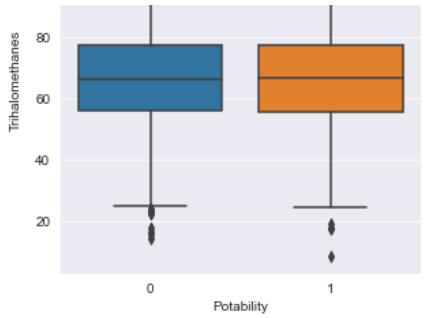


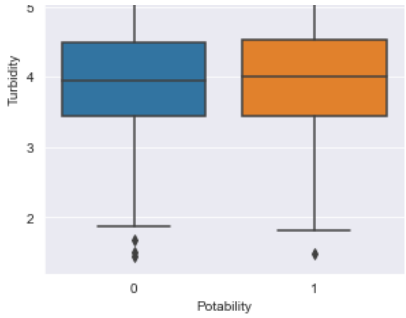












Soal 3

Menggunakan Test Shapiro (referensi: https://towardsdatascience.com/normality-tests-in-python-31e04aa4f411) didapat hasil (dengan keterangan pemerhatian dari boxplot dan histogram diagram)

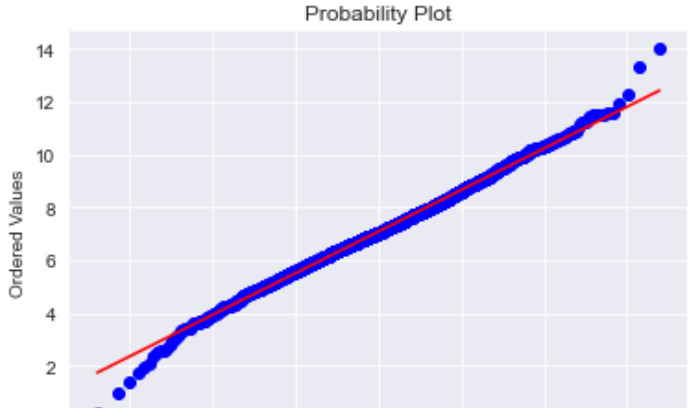
1. **pH : Tidak berdistribusi normal, terlihat terdapat banyak pencilan pada boxplot**
2. **Hardness : Tidak berdistribusi normal, terlihat banyak pencilan pada boxplot**
3. **Solids : Tidak berdistribusi normal, terlihat negative skewed pada histogram**
4. **Chloramines : Tidak berdistribusi normal, terlihat terlalu landai pada histogram dan tidak memperlihatkan distribusi normal.**
5. **Sulfate : Tidak berdistribusi normal, terlihat tidak simetris dan banyak pencilan**
6. **Conductivity : Tidak berdistribusi normal, terlihat positived skewed pada histogram**
7. **OrganicCarbon : Berdistribusi normal**
8. **Trihalomethanes : Berdistribusi normal**
9. **Turbidity : Berdistribusi normal**

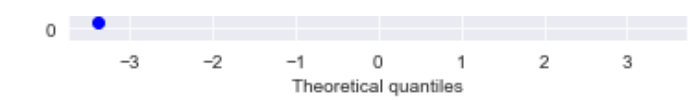
Pada hasil pemrograman di bawah diberikan QQ plot distributsi normal beserta nilai evaluasi Shapiro Normality Test

In [78]:

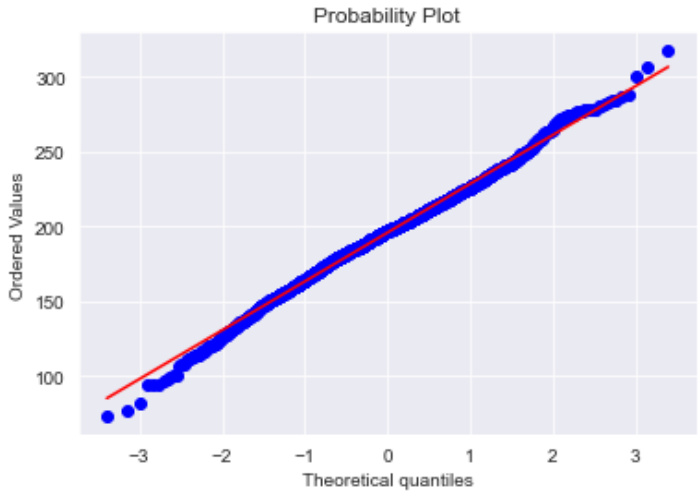
```
from scipy.stats import shapiro, probplot
import pylab

def print_result(stat,p):
    print("stat=%.3f, p=%.10f" %(stat, p), end=" ")
    if p>0.05:
        print("Normal")
    else:
        print("Tidak")

for cols in df:
    if(df[cols].dtype in accepted_data_type):
        print('=================='+cols+'================')
        data = df[cols]
        stat, p = shapiro(data)
        print("Shapiro Test : ", end="")
        print_result(stat, p)
        probplot(data, dist="norm", plot=pylab)
        pylab.show()
```

```
==================pH================
Shapiro Test : stat=0.996, p=0.0000850717 Tidak
```

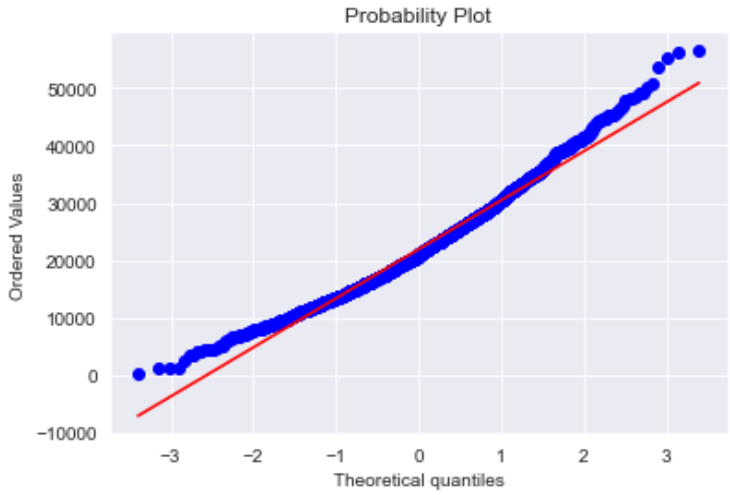




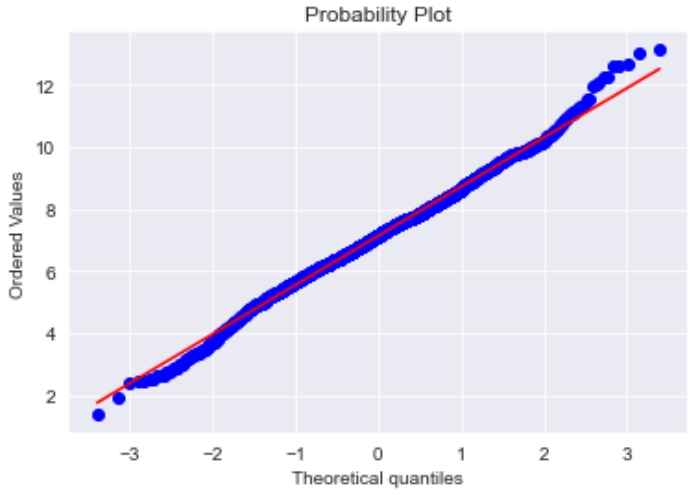
==========Hardness==========
Shapiro Test : stat=0.996, p=0.0000137997 Tidak



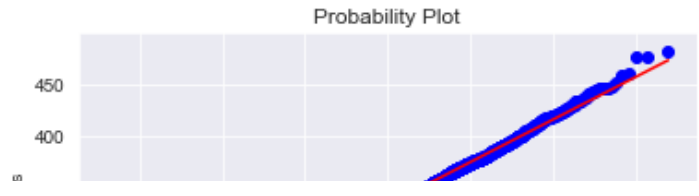
==========Solids==========
Shapiro Test : stat=0.979, p=0.0000000000 Tidak

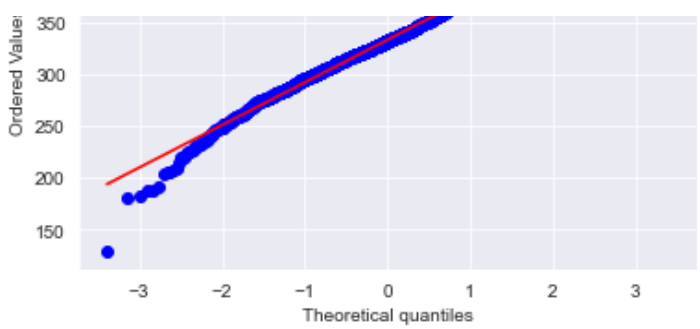


==========Chloramines==========
Shapiro Test : stat=0.996, p=0.0000364615 Tidak

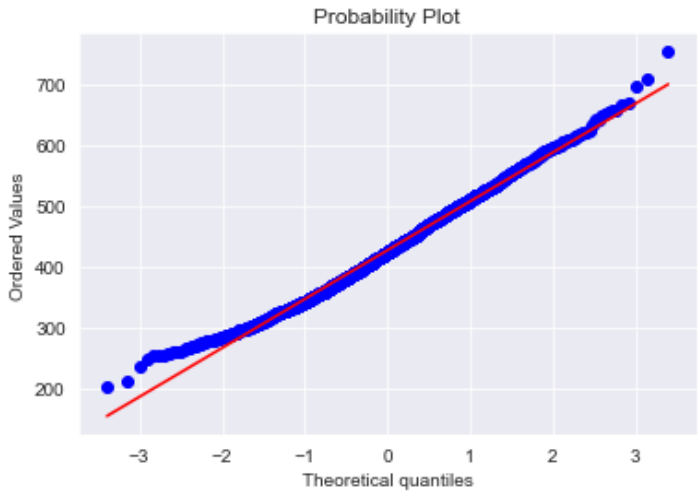


==========Sulfate==========
Shapiro Test : stat=0.995, p=0.0000008844 Tidak

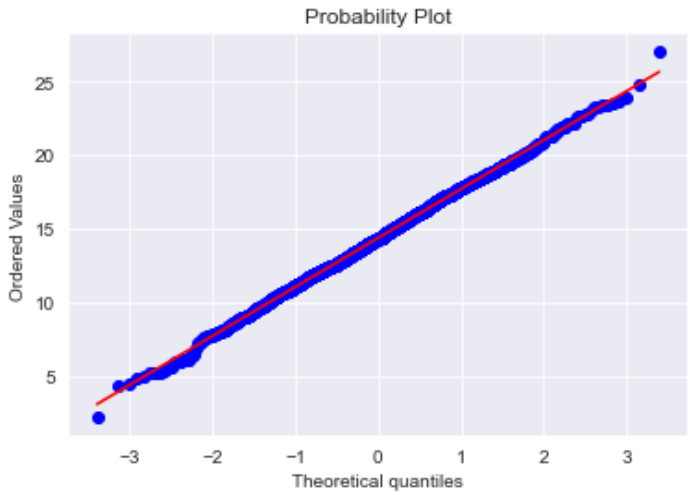




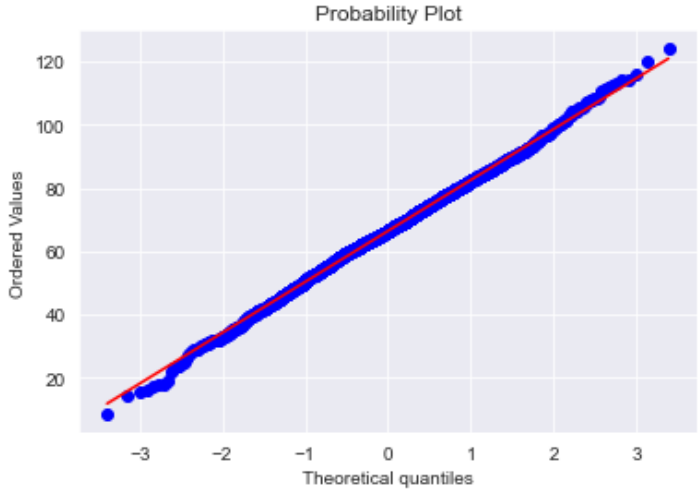
==========Conductivity==========
Shapiro Test : stat=0.993, p=0.0000000225 Tidak



==========OrganicCarbon==========
Shapiro Test : stat=0.999, p=0.6474561691 Normal

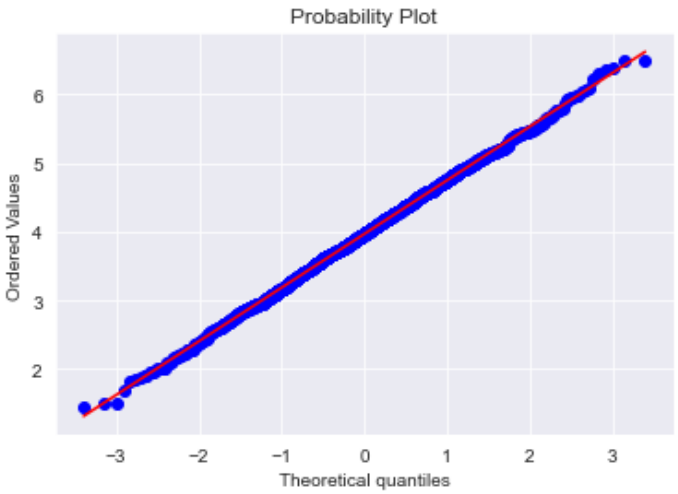


==========Trihalomethanes==========
Shapiro Test : stat=0.999, p=0.1062980294 Normal



==========Turbidity==========

```
Shapiro Test : stat=0.999, p=0.702239275 Normal
```



Soal 4

Melakukan test hipotesis 1 sampel, dengan menuliskan 6 langkah testing dan menampilkan juga boxplotnya untuk kolom/bagian yang bersesuaian.

a. Nilai Rata-rata pH di atas 7?

b. Nilai Rata-rata Hardness tidak sama dengan 205?

c. Nilai Rata-rata 100 baris pertama kolom Solids bukan 21900?

d. Proporsi nilai Conductivity yang lebih dari 450, adalah tidak sama dengan 10%?

e. Proporsi nilai Trihalomethanes yang kurang dari 40, adalah kurang dari 5%?

Sumber yang Kami Gunakan Untuk Menentukan Jenis Test yang digunakan: https://sonalake.com/latest/an-introduction-to-hypothesis-testing/

Soal 4.a

Nilai Rata-rata pH di atas 7?

1. **H0: pH = 7.0**
2. **H1: pH > 7.0**
3. **α = 0.05**
4. **1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai**
5. **p-value = 0.006851179743798378**
6. **Tolak H0**

Kesimpulan: Rata-rata pH di atas 7

In [7]:

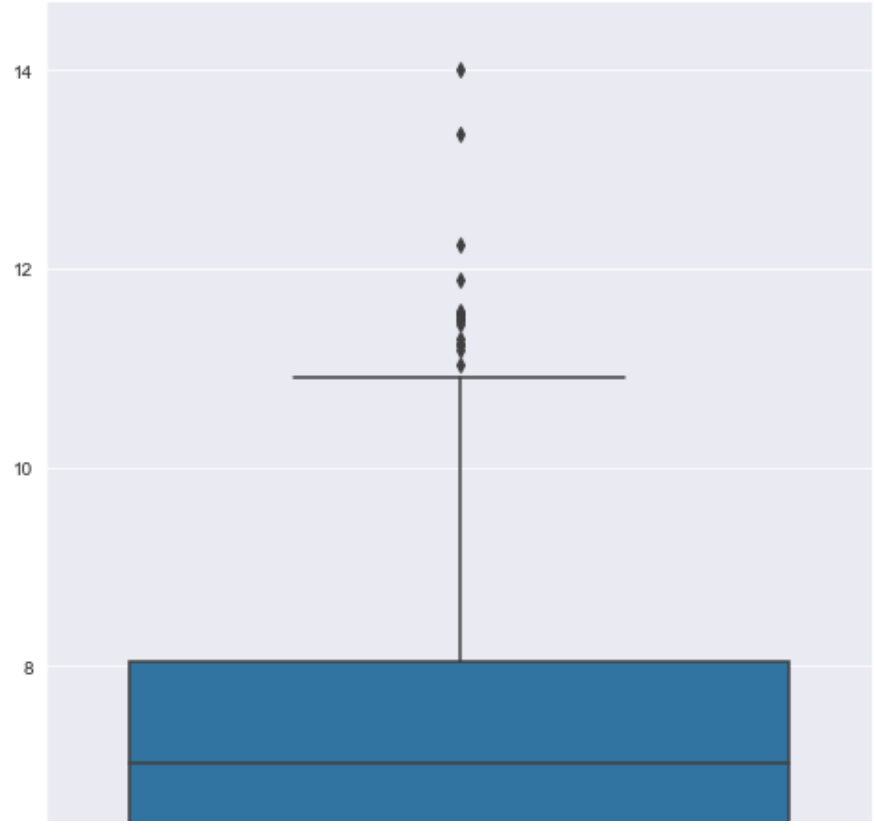
```
from scipy import stats

print("Soal 4.a")
mean = 7.0
print("Using Scipy Stats Ttest_1samp")
print("1. H0: pH =", mean)
print("2. H1: pH >", mean)
alpha = 0.05
print("3. α =",alpha)
print("4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
")
t_value,p_value=stats.ttest_1samp(df["pH"],mean, alternative='greater')
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
```

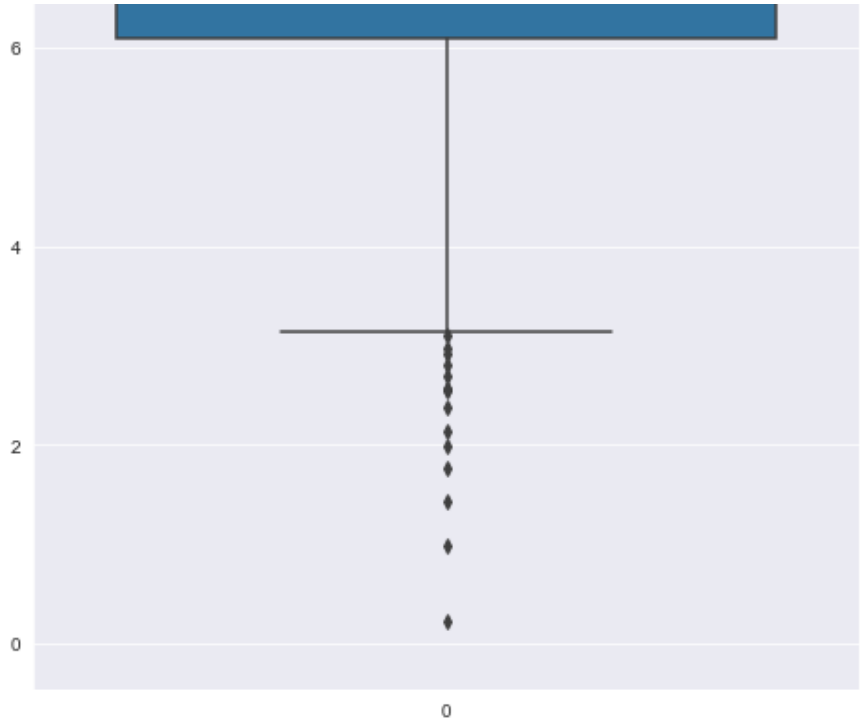
```
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManually")
print("1. H0: pH =", mean)
print("2. H1: pH >", mean)
alpha = 0.05
print("3. α =",alpha)
print("4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
")
t_value = (df["pH"].mean() - mean)/(df["pH"].std() / np.sqrt(len(df["pH"])))
print("5. t-value = ",float(t_value))
p_value = stats.t.sf(np.abs(t_value), len(df["pH"])-1)
print("   p-value = ",p_value)
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["pH"])
```

```
Soal 4.a
Using Scipy Stats Ttest_1samp
1. H0: pH = 7.0
2. H1: pH > 7.0
3. α = 0.05
4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
5. t-value =  2.4671463768689637
   p-value =  0.006851179743798378
6. Reject H0

Manually
1. H0: pH = 7.0
2. H1: pH > 7.0
3. α = 0.05
4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
5. t-value =  2.4671463768689637
   p-value =  0.006851179743798378
6. Reject H0
```

Out[7]:

<AxesSubplot:>





Soal 4.b

Nilai Rata-rata Hardness tidak sama dengan 205?

1. **H0: Hardness = 205.0**
2. **H1: Hardness != 205.0**
3. **α = 0.05**
4. **1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai**
5. **p-value = 3.951588599098364e-34**
6. **Tolak H0**

Kesimpulan: Hardness tidak sama dengan 205

In [8]:

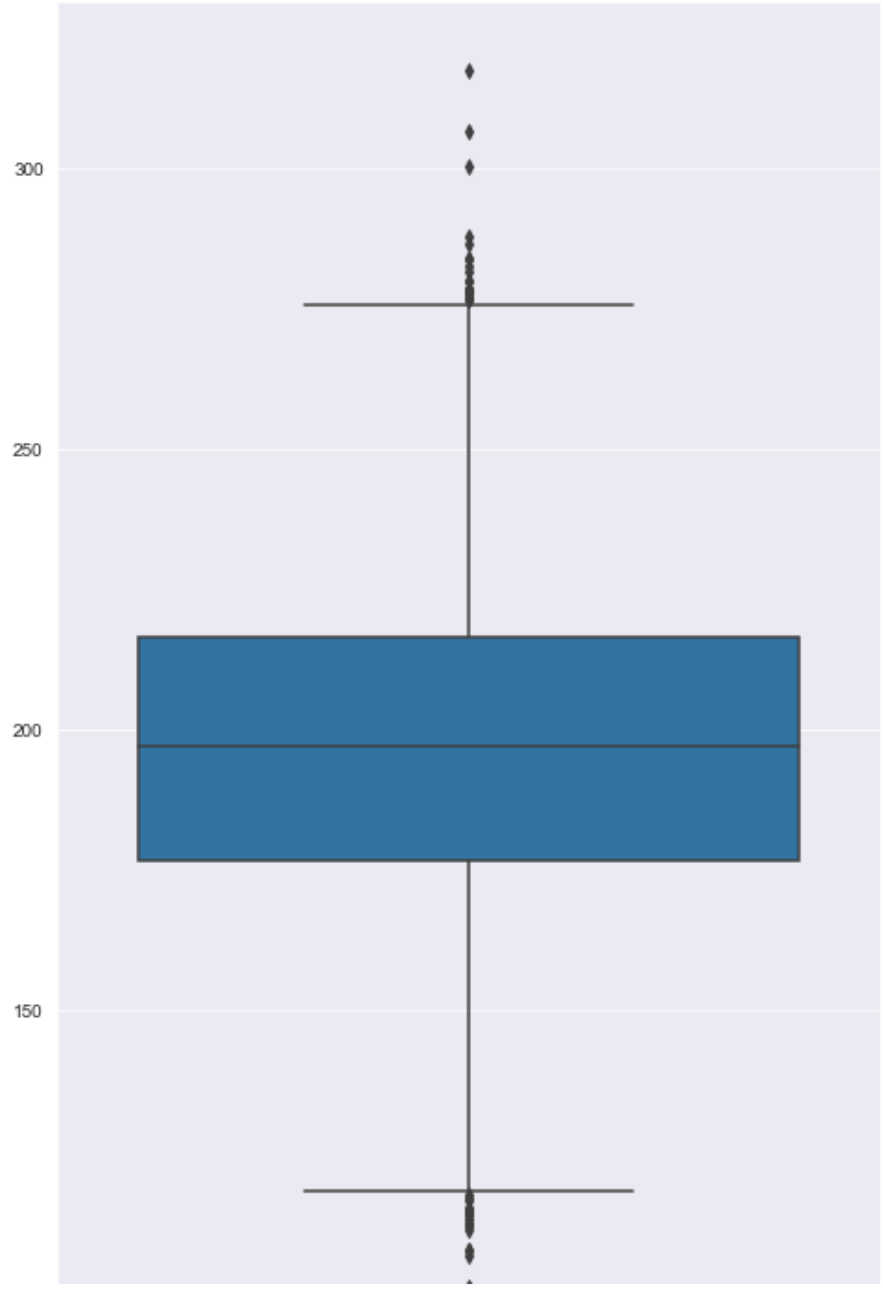
```
from scipy import stats

print("Soal 4.b")
mean = 205.0
print("1. H0: Hardness =", mean)
print("2. H1: Hardness !=", mean)
alpha = 0.05
print("3. α =",alpha)
print("4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
")
t_value,p_value=stats.ttest_1samp(df["Hardness"],mean, alternative = "two-sided")
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManually")
print("1. H0: Hardness =", mean)
print("2. H1: Hardness !=", mean)
alpha = 0.05
print("3. α =",alpha)
print("4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
")
t_value = (df["Hardness"].mean() - mean)/(df["Hardness"].std() / np.sqrt(len(df["Hardnes
s"])))
print("5. t-value = ",float(t_value))
p_value = stats.t.sf(np.abs(t_value), len(df["Hardness"])-1) * 2
```

```
print("   p-value = ",p_value)
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["Hardness"])
```

```
Soal 4.b
1. H0: Hardness = 205.0
2. H1: Hardness != 205.0
3. α = 0.05
4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
5. t-value =  -12.410522788078028
   p-value =  3.951588599098364e-34
6. Reject H0

Manually
1. H0: Hardness = 205.0
2. H1: Hardness != 205.0
3. α = 0.05
4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
5. t-value =  -12.410522788078028
   p-value =  3.951588599098364e-34
6. Reject H0
```

Out[8]:

<AxesSubplot:>



Soal 4.c

Nilai Rata-rata 100 baris pertama kolom Solids bukan 21900?

1. **H0: Solids = 21900.0**
2. **H1: Solids != 21900.0**
3. **α = 0.05**
4. **1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai**
5. **p-value = 0.5373441230987632**
6. **Tidak menolak H0**

Kesimpulan: Rata-rata 100 barus pertama kolom Solids 21900

In [9]:

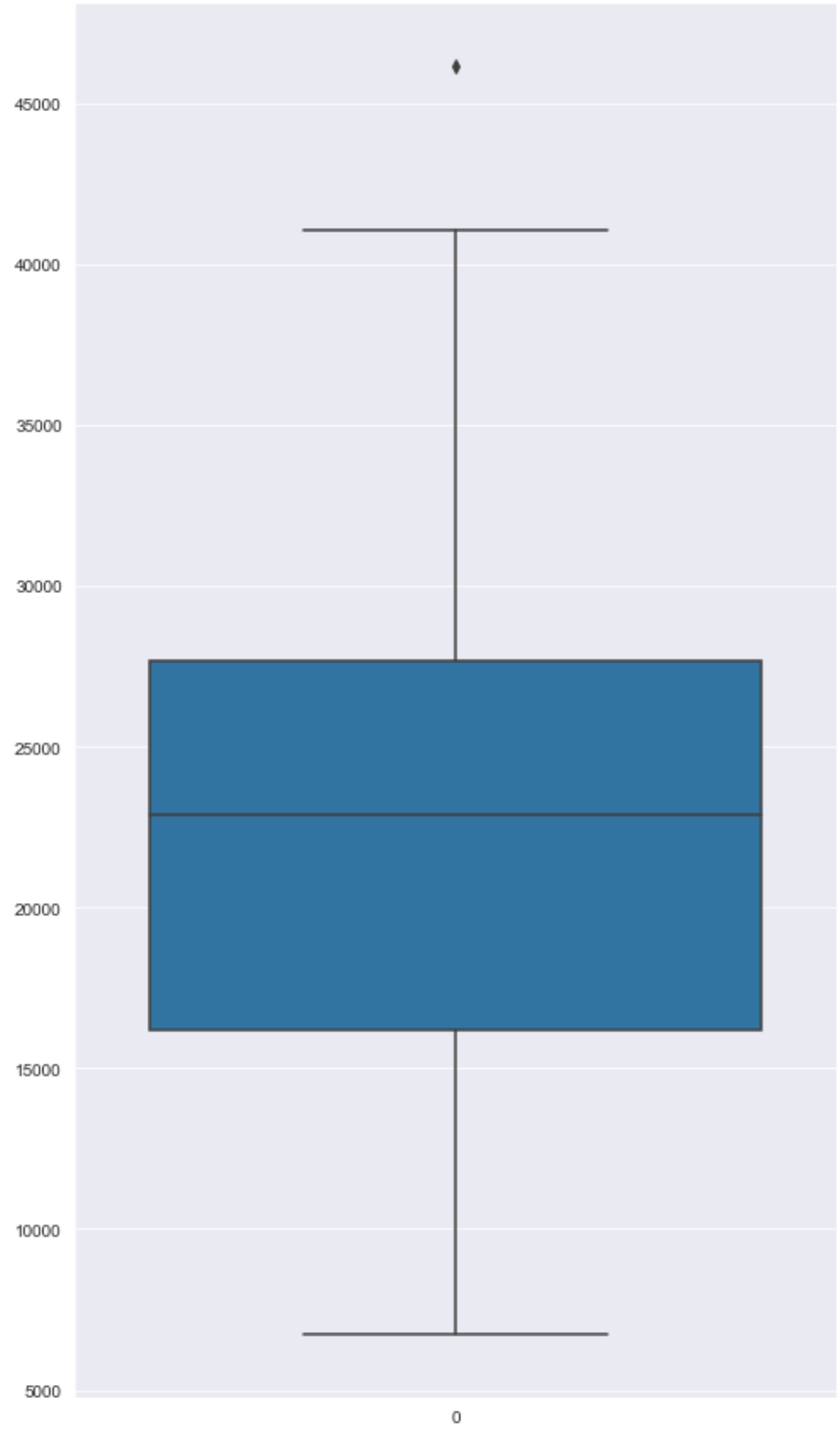
```
from scipy import stats

print("Soal 4.c")
mean = 21900.0
print("1. H0: Solids =",mean)
print("2. H1: Solids !=", mean)
alpha = 0.05
print("3. α =",alpha)
print("4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
")
t_value,p_value=stats.ttest_1samp(df["Solids"].head(100),mean, alternative="two-sided")
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")


print("\nManually")
print("1. H0: Solids =",mean)
print("2. H1: Solids !=", mean)
alpha = 0.05
print("3. α =",alpha)
print("4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
")
t_value = (df["Solids"].head(100).mean() - mean)/(df["Solids"].head(100).std() / np.sqrt
(100))
print("5. t-value = ",float(t_value))
p_value = stats.t.sf(np.abs(t_value), 100-1) * 2
print("   p-value = ",p_value)
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["Solids"].head(100))
```

```
Soal 4.c
1. H0: Solids = 21900.0
2. H1: Solids != 21900.0
3. α = 0.05
4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
5. t-value =  0.6189909029696404
```

```
   p-value =  0.5373441230987632
6. Can't / Don't Reject H0

Manually
1. H0: Solids = 21900.0
2. H1: Solids != 21900.0
3. α = 0.05
4. 1-Sample T-Test karena membandingkan Rata-rata sebuah sample dengan suatu nilai
5. t-value =  0.6189909029696403
   p-value =  0.5373441230987632
6. Can't / Don't Reject H0
```

Out[9]:

<AxesSubplot:>



Soal 4.d

Proporsi nilai Conductivity yang lebih dari 450, adalah tidak sama dengan 10%?

1. **H0: Proporsi Conductivity yang Lebih dari 450 = 0.1**
2. **H1: Proporsi Conductivity yang Lebih dari 450 != 0.1**
3. **α = 0.05**
4. **1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nilai**
5. **p-value = 0**
6. **Tolak H0**

Kesimpulan: Proporsi nilai Conductivity yang lebih dari 450 tidak sama dengan 10%

In [10]:

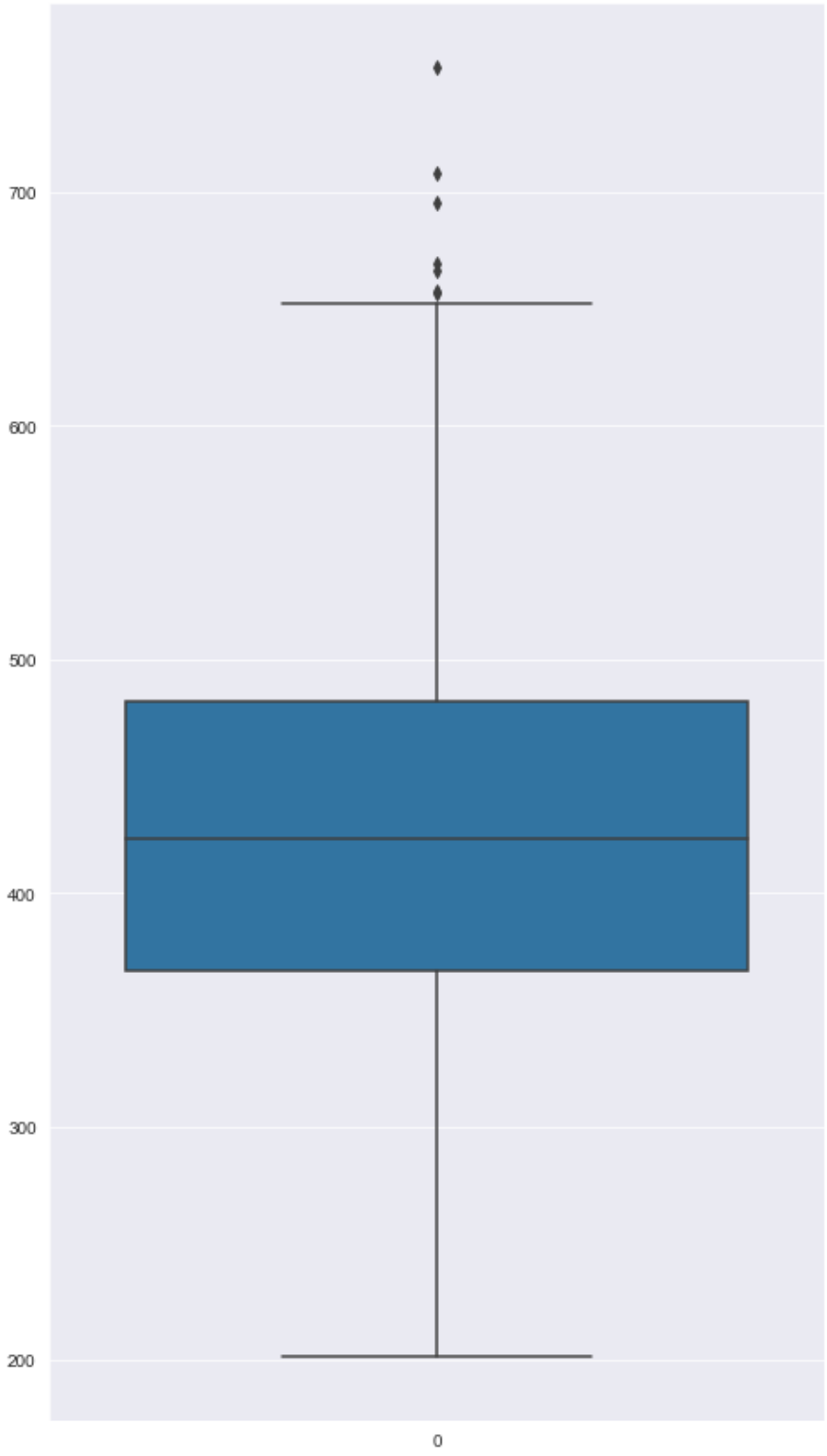
```
from statsmodels.stats.proportion import proportions_ztest

null_hypothesis = 0.1
conduct = df["Conductivity"]
sample_success = conduct[conduct>450.0].count()
sample_size = df["Conductivity"].count()
alpha = 0.05

print(sample_size)
print("Soal 4.d")
print("1. H0: Proporsi Conductivity yang Lebih dari 450 =", null_hypothesis)
print("2. H1: Proporsi Conductivity yang Lebih dari 450 !=", null_hypothesis)
print("3. α =",alpha)
print("4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nila
i")
z_value,p_value= proportions_ztest(count=sample_success, nobs=sample_size, value=null_hy
pothesis, alternative='two-sided', prop_var = null_hypothesis)
print("5. z-value = ",float(z_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManually")
print("1. H0: Proporsi Conductivity yang Lebih dari 450 =", null_hypothesis)
print("2. H1: Proporsi Conductivity yang Lebih dari 450 !=", null_hypothesis)
print("3. α =",alpha)
print("4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nila
i")
z_value = (sample_success - sample_size*null_hypothesis)/(np.sqrt(sample_size*null_hypot
hesis*(1-null_hypothesis)))
p_value = stats.norm.sf(abs(z_value))*2
print("5. z-value = ",float(z_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["Conductivity"])
```

```
2009
Soal 4.d
1. H0: Proporsi Conductivity yang Lebih dari 450 = 0.1
2. H1: Proporsi Conductivity yang Lebih dari 450 != 0.1
3. α = 0.05
4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nilai
5. z-value =  40.463878020757406
   p-value =  0.0
6. Reject H0

Manually
1. H0: Proporsi Conductivity yang Lebih dari 450 = 0.1
2. H1: Proporsi Conductivity yang Lebih dari 450 != 0.1
3. α = 0.05
4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nilai
5. z-value =  40.46387802075741
   p-value =  0.0
```

```
6. Reject H0
```

Out[10]:

```
<AxesSubplot:>
```



Soal 4.e

Proporsi nilai Trihalomethanes yang kurang dari 40, adalah kurang dari 5%?

1. **H0: Proporsi Trihalomethanes yang Lebih dari 40 = 0.05**
2. **H1: Proporsi Trihalomethanes yang Lebih dari 40 < 0.05**
3. **α = 0.05**
4. **1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nilai**
5. **p-value = 0.7150304199895099**
6. **Tidak menolak H0**

Kesimpulan: Proporsi nilai Trihalomethanes yang kurang dari 40 adalah kurang dari 5%

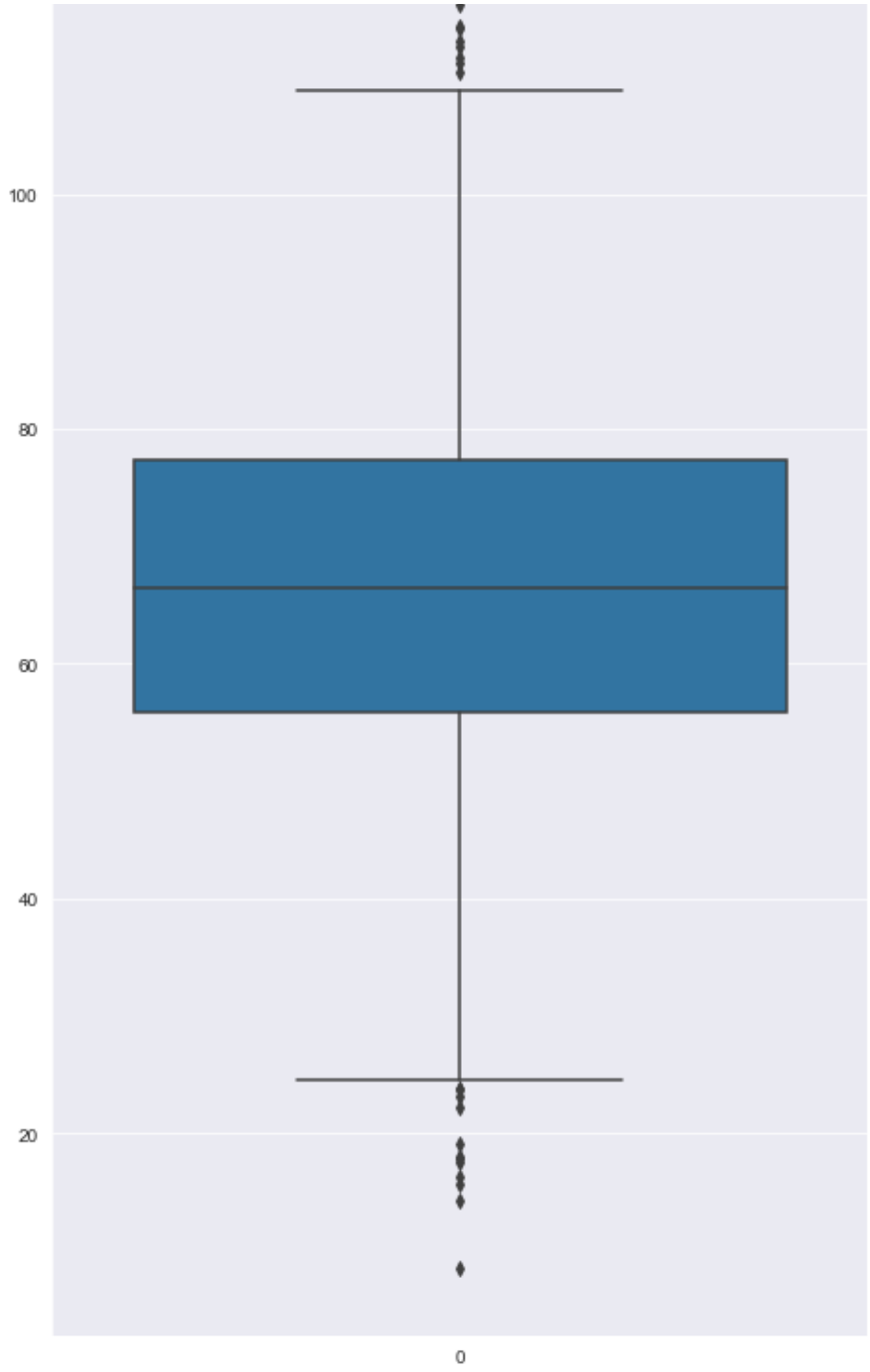
In [11]:

```
from statsmodels.stats.proportion import proportions_ztest

null_hypothesis = 0.05
Trihalomethanes = df["Trihalomethanes"]
sample_success = Trihalomethanes[Trihalomethanes<40].count()
sample_size = df["Trihalomethanes"].count()
alpha = 0.05


print("Soal 4.e")
print("1. H0: Proporsi Conductivity yang Lebih dari 40 =", null_hypothesis)
print("2. H1: Proporsi Conductivity yang Lebih dari 40 <", null_hypothesis)
print("3. α =",alpha)
print("4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nila
i")
z_value,p_value= proportions_ztest(count=sample_success, nobs=sample_size, value=null_hy
pothesis, alternative='smaller', prop_var = null_hypothesis)
print("5. z-value = ",float(z_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManually")
print("1. H0: Proporsi Conductivity yang Lebih dari 450 =", null_hypothesis)
print("2. H1: Proporsi Conductivity yang Lebih dari 450 !=", null_hypothesis)
print("3. α =",alpha)
print("4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nila
i")
z_value = (sample_success - sample_size*null_hypothesis)/(np.sqrt(sample_size*null_hypot
hesis*(1-null_hypothesis)))
p_value = 1- stats.norm.sf(abs(z_value))
print("5. z-value = ",float(z_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["Trihalomethanes"])
```

```
Soal 4.e
1. H0: Proporsi Conductivity yang Lebih dari 40 = 0.05
2. H1: Proporsi Conductivity yang Lebih dari 40 < 0.05
3. α = 0.05
4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nilai
5. z-value =  0.5681411028064428
   p-value =  0.7150304199895099
6. Can't / Don't Reject H0

Manually
1. H0: Proporsi Conductivity yang Lebih dari 450 = 0.05
2. H1: Proporsi Conductivity yang Lebih dari 450 != 0.05
3. α = 0.05
4. 1-Sample Z-test karena membandingkan proporsi sebuah variable dengan suatu nilai
5. z-value =  0.5681411028064431
   p-value =  0.7150304199895101
6. Can't / Don't Reject H0
```

Out[11]:

```
<AxesSubplot:>
```



Soal 5

Melakukan test hipotesis 2 sampel, dengan menuliskan 6 langkah testing dan menampilkan juga boxplotnya untuk kolom/bagian yang bersesuaian.

a. Data kolom Sulfate dibagi 2 sama rata: bagian awal dan bagian akhir kolom. Benarkah rata-rata kedua bagian tersebut sama?

b. Data kolom OrganicCarbon dibagi 2 sama rata: bagian awal dan bagian akhir kolom. Benarkah rata-rata bagian awal lebih besar dari pada bagian akhir sebesar 0.15?

c. Rata-rata 100 baris pertama kolom Chloramines sama dengan 100 baris terakhirnya?

d. Proporsi nilai bagian awal Turbidity yang lebih dari 4, adalah lebih besar daripada, proporsi nilai yang sama di bagian akhir Turbidity ?

e. Bagian awal kolom Sulfate memiliki variansi yang sama dengan bagian akhirnya?

Soal 5.a

Data kolom Sulfate dibagi 2 sama rata: bagian awal dan bagian akhir kolom. Benarkah rata-rata kedua bagian tersebut sama?

1. **H0: Rata-rata bagian awal = Rata-rata bagian akhir**
2. **H1: Rata-rata bagian awal != Rata-rata bagian akhir**
3. **α = 0.05**
4. **2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama**
5. **p-value = 0.0368965161184979**
6. **Tolak H0**

Kesimpulan: Data kolom Sulfate bagian awal dan bagian akhir memiliki mean yang berbeda

In [18]:

```
from scipy.stats import ttest_ind

midpoint = int(df["Sulfate"].count()/2)
awal = df.iloc[:midpoint,:]
akhir = df.iloc[midpoint:,:]
awal = awal["Sulfate"]
akhir = akhir["Sulfate"]


#d_hypotesis = perbedaan rata-rata dari kedua sample
d_hypothesis = 0
alpha = 0.05

print("Soal 5.a")
print("1. H0: Rata-rata bagian awal = Rata-rata bagian akhir")
print("2. H1: Rata-rata bagian awal != Rata-rata bagian akhir")
print("3. α =",alpha)
print("4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi ya
ng sama")
t_value,p_value= ttest_ind(awal,akhir, equal_var=True, alternative="two-sided")
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManually")
print("1. H0: Rata-rata bagian awal = Rata-rata bagian akhir")
print("2. H1: Rata-rata bagian awal != Rata-rata bagian akhir")
print("3. α =",alpha)
print("4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi ya
ng sama")
sp2 = ((awal.count() -1) * awal.var() + (akhir.count() -1) * akhir.var())   /   (awal.co
unt() + akhir.count() -2)
sp = np.sqrt(sp2)
t_value = ((awal.mean() - akhir.mean()) - d_hypothesis) / (sp* np.sqrt( 1/awal.count() +
1/akhir.count()))
p_value = stats.t.sf(abs(t_value), df= awal.count() + akhir.count() -2) * 2
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")
```

```
Soal 5.a
1. H0: Rata-rata bagian awal = Rata-rata bagian akhir
2. H1: Rata-rata bagian awal != Rata-rata bagian akhir
3. α = 0.05
4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama
5. t-value =  -2.088300289684085
   p-value =  0.0368965161184979
6. Reject H0

Manually
1. H0: Rata-rata bagian awal = Rata-rata bagian akhir
2. H1: Rata-rata bagian awal != Rata-rata bagian akhir
3. α = 0.05
```

```
4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama
5. t-value =  -2.088300289684085
   p-value =  0.0368965161184979
6. Reject H0
```

Soal 5.b

Data kolom OrganicCarbon dibagi 2 sama rata: bagian awal dan bagian akhir kolom. Benarkah rata-rata bagian awal lebih besar dari pada bagian akhir sebesar 0.15?

1. **H0: Rata-rata bagian awal - Rata-rata bagian akhir = 0.15**
2. **H1: Rata-rata bagian awal - Rata-rata bagian akhir != 0.15**
3. **α = 0.05**
4. **2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama**
5. **p-value = 0.01473341680663031**
6. **Reject H0**

Kesimpulan: Data kolom OrganicCarbon bagian awal dan bagian akhir memiliki perbedaan mean yang tidak sama dengan 0.15

In [24]:

```
from scipy.stats import ttest_ind

midpoint = int(df["OrganicCarbon"].count()/2)
awal = df.iloc[:midpoint,:]
akhir = df.iloc[midpoint:,:]
awal = awal["OrganicCarbon"]
akhir = akhir["OrganicCarbon"]

#d_hypotesis = perbedaan rata-rata dari kedua sample
d_hypothesis = 0.15
alpha = 0.05

print("Soal 5.b")
print("\nManually")
print("1. H0: Rata-rata bagian awal - Rata-rata bagian akhir = 0.15")
print("2. H1: Rata-rata bagian awal - Rata-rata bagian akhir != 0.15")
print("3. α =",alpha)
print("4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi ya
ng sama")
sp2 = ((awal.count() -1) * awal.var() + (akhir.count() -1) * akhir.var())   /   (awal.co
unt() + akhir.count() -2)
sp = np.sqrt(sp2)
t_value = ((awal.mean() - akhir.mean()) - d_hypothesis) / (sp* np.sqrt( 1/awal.count() +
1/akhir.count()))
p_value = stats.t.sf(abs(t_value), df= awal.count() + akhir.count() -2) * 2
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")
```

```
Soal 5.b

Manually
1. H0: Rata-rata bagian awal - Rata-rata bagian akhir = 0.15
2. H1: Rata-rata bagian awal - Rata-rata bagian akhir != 0.15
3. α = 0.05
4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama
5. t-value =  -2.4409788594021316
   p-value =  0.01473341680663031
6. Reject H0
```

Soal 5.c

Rata-rata 100 baris pertama kolom Chloramines sama dengan 100 baris terakhirnya?

1. **H0: Rata-rata bagian awal = Rata-rata bagian akhir**
2. **H1: Rata-rata bagian awal != Rata-rata bagian akhir**
3. **α = 0.05**
4. **2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama**
5. **p-value = 0.44676017233305654**
6. **Can't / Don't Reject H0**

Kesimpulan: 100 Data kolom Chloramines bagian awal dan 100 data bagian akhir kolom Chloramies memiliki mean yang berbeda

In [30]:

```
from scipy.stats import ttest_ind

end = df["Chloramines"].count()
awal = df.iloc[:100,:]
akhir = df.iloc[end-100:,:]
awal = awal["Chloramines"]
akhir = akhir["Chloramines"]

#d_hypotesis = perbedaan rata-rata dari kedua sample
d_hypothesis = 0
alpha = 0.05

print("Soal 5.c")
print("1. H0: Rata-rata bagian 100 awal = Rata-rata bagian 100 akhir")
print("2. H1: Rata-rata bagian 100 awal != Rata-rata bagian 100 akhir")
print("3. α =",alpha)
print("4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi ya
ng sama")
t_value,p_value= ttest_ind(awal,akhir, equal_var=True, alternative="two-sided")
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManually")
print("1. H0: Rata-rata bagian awal = Rata-rata bagian akhir")
print("2. H1: Rata-rata bagian awal != Rata-rata bagian akhir")
print("3. α =",alpha)
print("4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi ya
ng sama")
sp2 = ((awal.count() -1) * awal.var() + (akhir.count() -1) * akhir.var())   /   (awal.co
unt() + akhir.count() -2)
sp = np.sqrt(sp2)
t_value = ((awal.mean() - akhir.mean()) - d_hypothesis) / (sp* np.sqrt( 1/awal.count() +
1/akhir.count()))
p_value = stats.t.sf(abs(t_value), df= awal.count() + akhir.count() -2) * 2
print("5. t-value = ",float(t_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")
```

```
Soal 5.c
1. H0: Rata-rata bagian 100 awal = Rata-rata bagian 100 akhir
2. H1: Rata-rata bagian 100 awal != Rata-rata bagian 100 akhir
3. α = 0.05
4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama
5. t-value =  -0.7623469233489559
   p-value =  0.44676017233305654
6. Can't / Don't Reject H0

Manually
1. H0: Rata-rata bagian awal = Rata-rata bagian akhir
2. H1: Rata-rata bagian awal != Rata-rata bagian akhir
3. α = 0.05
```

```
4. 2-Sample paired T-test karena membandingkan rata-rata 2 sample dari populasi yang sama
5. t-value =  -0.7623469233489559
   p-value =  0.44676017233305654
6. Can't / Don't Reject H0
```

Soal 5.d

Proporsi nilai bagian awal Turbidity yang lebih dari 4, adalah lebih besar daripada, proporsi nilai yang sama di bagian akhir Turbidity ?

1. **H0: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal = bagian akhir**
2. **H1: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal > bagian akhir**
3. **α = 0.05**
4. **2-Sample Z-test karena membandingkan proporsi 2 buah sample**
5. **z-value = -0.1569235575631641 p-value = 0.5623474531912108**
6. **Terima H0**

Kesimpulan: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal sama dengan bagian akhir

In [37]:

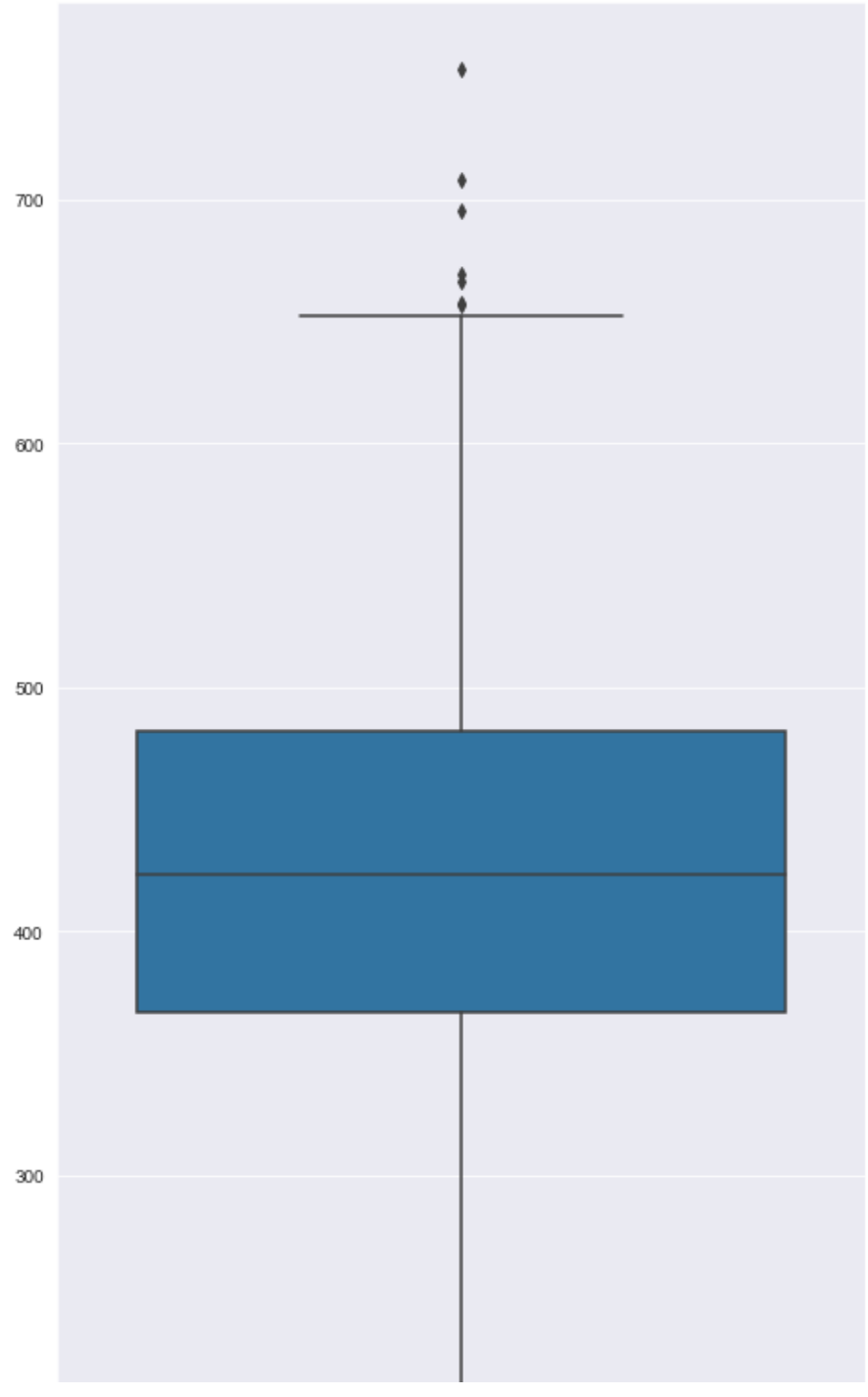
```
from statsmodels.stats.proportion import proportions_ztest

midpoint = int(df["Turbidity"].count()/2)
awal = df.iloc[:midpoint,:]
akhir = df.iloc[midpoint:,:]
awal_count = awal["Turbidity"].count()
awal_success = awal[awal["Turbidity"]>4.0]
awal_success_count = awal_success["Turbidity"].count()
akhir_count = akhir["Turbidity"].count()
akhir_success = akhir[akhir["Turbidity"]>4.0]
akhir_success_count = akhir_success["Turbidity"].count()
sample_size = df["Turbidity"].count()
alpha = 0.05
successes = np.array([awal_success_count, akhir_success_count])
samples = np.array([awal["id"].count(), akhir["id"].count()])

print("Soal 4.d")
print("1. H0: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal = bagian akhir"
)
print("2. H1: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal > bagian akhir"
)
print("3. α =",alpha)
print("4. 2-Sample Z-test karena membandingkan proporsi 2 buah sample")
z_value, p_value = proportions_ztest(count=successes, nobs=samples,  alternative='larger
', prop_var= 0)
print("5. z-value = ",float(z_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

print("\nManual")
print("1. H0: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal = bagian akhir"
)
print("2. H1: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal > bagian akhir"
)
print("3. α =",alpha)
print("4. 2-Sample Z-test karena membandingkan proporsi 2 buah sample")
phat1 = awal_success_count / awal_count
phat2 = akhir_success_count / akhir_count
phat = (awal_success_count+akhir_success_count) / (awal_count+akhir_count)
z_value = (phat1 - phat2) / np.sqrt(phat * (1-phat) * (1/awal_count + 1/akhir_count))
print("5. z-value = ",float(z_value))
print("   p-value = ",float(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
```

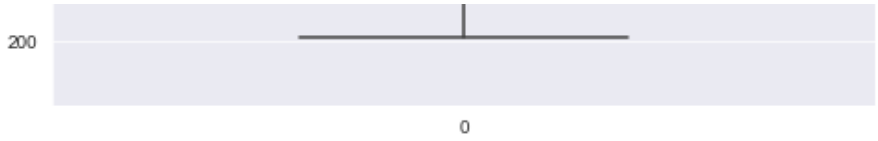
```
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["Conductivity"])
```

```
Soal 4.d
1. H0: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal = bagian akhir
2. H1: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal > bagian akhir
3. α = 0.05
4. 2-Sample Z-test karena membandingkan proporsi 2 buah sample
5. z-value =  -0.1569235575631641
   p-value =  0.5623474531912108
6. Can't / Don't Reject H0

Manual
1. H0: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal = bagian akhir
2. H1: Proporsi nilai Turbidity yang Lebih dari 4 pada bagian awal > bagian akhir
3. α = 0.05
4. 2-Sample Z-test karena membandingkan proporsi 2 buah sample
5. z-value =  -0.1569235575631641
   p-value =  0.5623474531912108
6. Can't / Don't Reject H0
```

Out[37]:

<AxesSubplot:>





Soal 5.e

Bagian awal kolom Sulfate memiliki variansi yang sama dengan bagian akhirnya?

1. **H0: Variansi bagian awal Sulfat = Variansi bagian akhir**
2. **H1: Variansi bagian awal Sulfat != Variansi bagian akhir**
3. **α = 0.05**
4. **f-test karena membandingkan variansi 2 buah sample**
5. **f-value = 1.0158752179644388 p-value = 0.8030355701576082**
6. **Tidak menolak H0**

Kesimpulan: Bagian awal kolom Sulfat memiliki variansi yang sama dengan bagian akhirnya

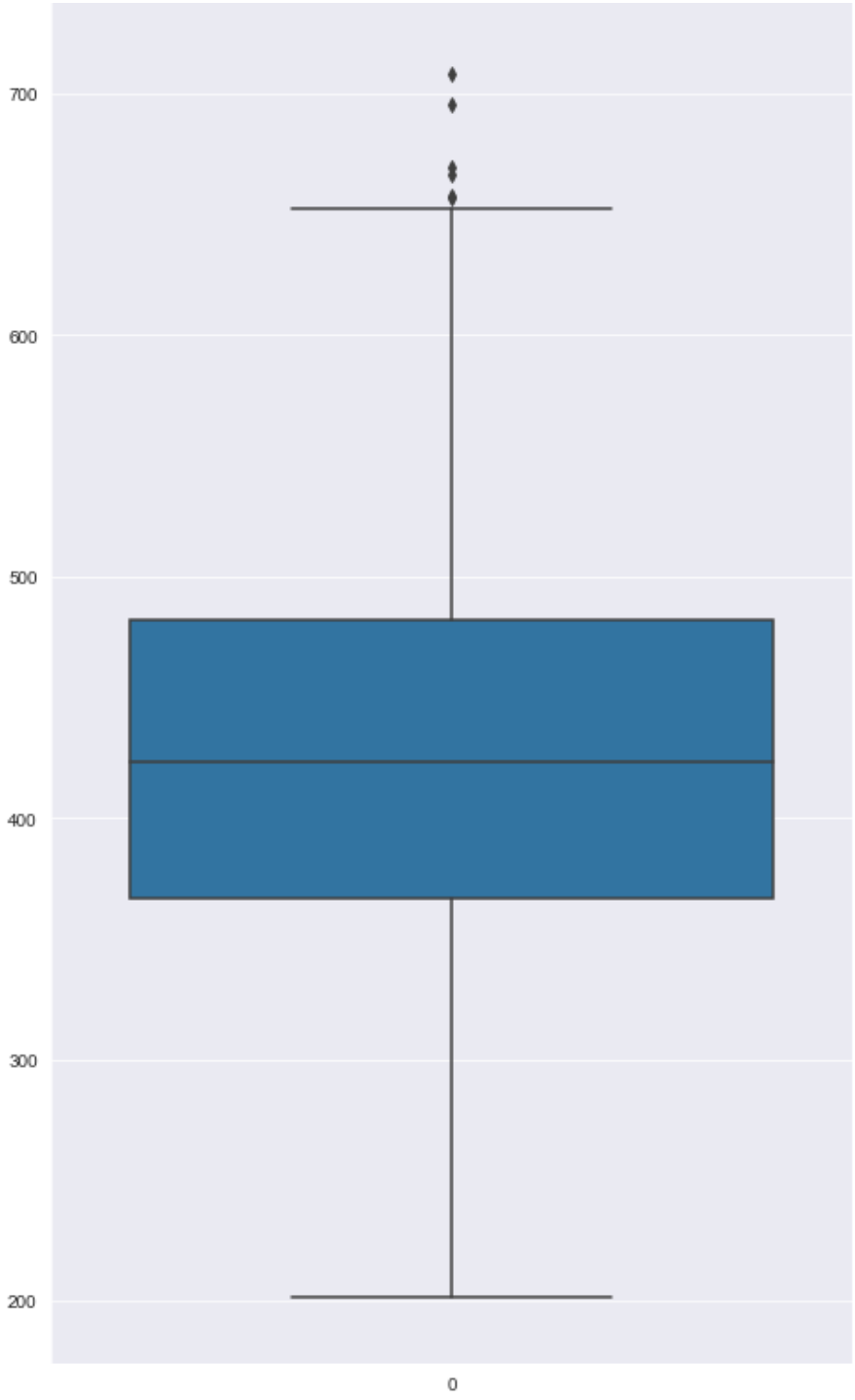
In [47]:

```
import scipy.stats

def f_test(x, y):
    x = np.array(x)
    y = np.array(y)
    f = np.var(x, ddof=1)/np.var(y, ddof=1) #calculate F test statistic
    dfn = x.size-1 #define degrees of freedom numerator
    dfd = y.size-1 #define degrees of freedom denominator
    p = 1-scipy.stats.f.cdf(f, dfn, dfd) #find p-value of F test statistic
    return f, p

midpoint = int(df["Sulfate"].count()/2)
awal = df.iloc[:midpoint,:]
akhir = df.iloc[midpoint:,:]

print("Soal 5.e")
print("1. H0: Variansi bagian awal Sulfat = Variansi bagian akhir")
print("2. H1: Variansi bagian awal Sulfat != Variansi bagian akhir")
print("3. α =",alpha)
print("4. f-test karena membandingkan variansi 2 buah sample")
f_value = awal["Sulfate"].var() / akhir["Sulfate"].var()
p_value = (1-scipy.stats.f.cdf(f_value, awal["Sulfate"].count()-1, akhir["Sulfate"].coun
t()-1))*2
print("5. f-value = ",(f_value))
print("   p-value = ",(p_value))
if p_value<alpha:
    print("6. Reject H0")
else:
    print("6. Can't / Don't Reject H0")

plt.figure(figsize=(8,15))
sns.boxplot(data=df["Conductivity"])
```

```
Soal 5.e
1. H0: Variansi bagian awal Sulfat = Variansi bagian akhir
2. H1: Variansi bagian awal Sulfat != Variansi bagian akhir
3. α = 0.05
4. f-test karena membandingkan variansi 2 buah sample
5. f-value =  1.0158752179644388
   p-value =  0.8030355701576082
6. Can't / Don't Reject H0
```

Out[47]:

```
<AxesSubplot:>
```



Soal 6

Test korelasi: tentukan apakah setiap kolom non-target berkorelasi dengan kolom target, dengan menggambarkan juga scatter plot nya. Gunakan correlation test.

Data dinilai berkolesai positif jika nilai korelasi mendekati 1 dan dinilai berkorelasi negatif jika nilai korelasi mendekati -1

1. **pH : Tidak berkorelasi**
2. **Hardness : Tidak berkorelasi**
3. **Solids : Tidak berkorelasi**
4. **Chloramines : Tidak berkorelasi**
5. **Sulfate : Tidak berkorelasi**
6. **Conductivity : Tidak berkorelasi**
7. **OrganicCarbon : Tidak berkorelasi**
8. **Trihalomethanes : Tidak berkorelasi**
9. **Turbidity : Tidak berkorelasi**

	Attribute	Correlation Score	Kesimpulan
	pH	0.015799939636415246	Tidak Berkorelasi

Hardness	**-0.0012328336798452782**	**Tidak Berkorelasi**
Solids	**0.03898852448814413**	**Tidak Berkorelasi**
Chloramines	**0.021022928450692695**	**Tidak Berkorelasi**
Sulfate	**-0.01547204411330092**	**Tidak Berkorelasi**
Conductivity	**-0.016582832508648802**	**Tidak Berkorelasi**
OrganicCarbon	**-0.01499495085173149**	**Tidak Berkorelasi**
Trihalomethanes	**0.010112984530356858**	**Tidak Berkorelasi**
Turbidity	**0.022684416432633126**	**Tidak Berkorelasi**

In [77]:

```
#Menggunakan Correlation Test Pearson
sns.set_style("darkgrid")
fig, axes = plt.subplots(3, 3, figsize=(18, 10))


fig.suptitle('Correlation and Scatter Plot', fontsize=20)

i = 0
for columns in df:
    if columns not in ["id", "Potability"]:
        r2, p = stats.pearsonr(df[columns], df["Potability"])
        sns.scatterplot(ax=axes[i//3, i % 3] ,data=df, x=columns, y="Potability", palett
e="deep").set_title(columns + " Correlation Score = " + str(r2))
        i += 1

plt.subplots_adjust(hspace=0.6, wspace = 0.4)
plt.show()
```