Water Consumption Amount by District in Istanbul

Assignment No – 02

Course Name & Code
Fundamental of Data Science (SWE – 685)

Submitted To
Mr. Fazly Rabbi
Faculty, Department of Software Engineering

Submitted by

Md. Ashraful Alam
ID: 0242220005343011
Program: M. Sc in Software Engineering
Department of Software Engineering

Daffodil International University
Daffodil Smart City (DSC),
Ashulia, Dhaka-1341
Bangladesh

Date: October 07, 2022

Research Problem: Will it still be possible for Istanbul to find further water from other metropolises indeed though climate change affects them all? Or are Istanbul's current water force programs and practices eventually coming to an end? Explore the relationship among 5 years water consumption of districts in Istanbul. Find out changes of the water consumption of all given districts since 2015 to 2019. Using those years as attributes and also total water consumption. Some of mathematical and statical libraries in Python programming language are using for dataset analysis and visualization.

About the Dataset: This is a public Dataset and this dataset is available at Kaggle data science link, https://www.kaggle.com/datasets/yasergirit/water-consumption-amount-by-district-in-istanbul. This dataset related to water consumption of districts in Istanbul since 2015 to 2019.

Domain Knowledge & Attributes:

District – Name of districts in Istanbul. (This attribute fully dependent on all attributes those are later come.)

2019 (Consumption-m3) – water consumption amount of 2019. (Water supply S. I unit of cubic meter as m3. It is listed by district list.)

2018 (Consumption-m3) – water consumption amount of 2018. (Water supply S. I unit of cubic meter as m3. It is listed by district list.)

2017 (Consumption-m3) – water consumption amount of 2017. (Water supply S. I unit of cubic meter as m3. It is listed by district list.)

2016 (Consumption-m3) – water consumption amount of 2016. (Water supply S. I unit of cubic meter as m3. It is listed by district list.)

2015 (Consumption-m3) – water consumption amount of 2015. (Water supply S. I unit of cubic meter as m3. It is listed by district list.)

Total – Sum water consumption of a district since 2015 to 2019. (This attribute created by using python code.)

EDA, Report Summary

Organizing Dataset and Analysis

- Reading dataset
- Rearrange the dataset
- Reading Data Tail
- Data set Shape
- Showing Columns
- How many districts are there?
- Creating a column that includes total water consumption of every year would be better
- Check Null
- Data set info
- Dataset means
- Dataset median
- Checking kurtosis
- Checking skewness
- Checking correlation
- Checking covariance
- Descriptive Statistics
- Finding out most water consumption

Visualization of Water Consumption in İstanbul

- Heatmap of water consumption
- District wise parameter checking
- District wise Water Consumption in those years
- District wise distribution
- Water Consumption by Stacked Bar Chart
- Total and Mean Water Consumption by Years
- Box Plot of Total Water Consumption
- Box Plots of Water Consumption
- Histogram chart
- 10. Categorical Scaller Chart

Organizing Dataset and Analysis

Environment Setup: We need to import following libraries for exploring this dataset,

```
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.mlab as mlab
import seaborn as sns
import pandas as pd
import numpy as np
import plotly.express as px
from plotly.subplots
import make_subplots
import plotly.graph_objects as go
```

Importing Dataset: We will load the Dataset directly from repository using URL and load this Dataset to a data frame variable *df*.

```
url='https://raw.githubusercontent.com/Sagor96/ds/main/dataset/water-consumption.csv'
df = pd.read_csv(url)
```

Describing the Dataset

Reading dataset: Firstly we will see the first 5 rows of our dataset using df.*head()*

	District	2019 (Consumption-m3)	2018 (Consumption-m3)	2017 (Consumption-m3)	2016 (Consumption-m3)	2015 (Consumption-m3)
0	ADALAR	1371291	1399182	1366581	1472276	1432494
1	ARNAVUTKÖY	13818204	11404878	10176132	9178953	8002123
2	ATAŞEHİR	22428468	21496185	21205911	19974097	18597049
3	AVCILAR	19485453	18312736	17558403	17052253	16220476
4	B.ÇEKMECE	13606571	12222470	11172787	10788924	9915435

Rearrange the dataset: we see this dataset arrange year 2019 to 2015. For analysis, we rearrange dataset year 2015 to 2019.

df = pd.concat((df[["District"]],df[df.columns[1::][::-1]]), axis=1)
 # Now the dataset will look better
df.head()

	District	2015 (Consumption-m3)	2016 (Consumption-m3)	2017 (Consumption-m3)	2018 (Consumption-m3)	2019 (Consumption-m3)
0	ADALAR	1432494	1472276	1366581	1399182	1371291
1	ARNAVUTKÖY	8002123	9178953	10176132	11404878	13818204
2	ATAŞEHİR	18597049	19974097	21205911	21496185	22428468
3	AVCILAR	16220476	17052253	17558403	18312736	19485453
4	B.ÇEKMECE	9915435	10788924	11172787	12222470	13606571

Showing Columns: For checking all attributes/ columns run *df.columns* then we find:

```
Index(['District', '2015 (Consumption-m3)', '2016 (Consumption-m3)',
       '2017 (Consumption-m3)', '2018 (Consumption-m3)',
       '2019 (Consumption-m3)'],
      dtype='object')
```

How many districts are there? we get all attributes data list count. For example- District column data number.

print("Number of districts: {}".format(df.District.nunique()))

```
Number of districts: 39
```

Exploring Attributes type Cleaning the Dataset

Creating a column that includes total water consumption of every year would be better: For better analysis with visualization, we can add a column name as total. It contains a district water consumption since 2015 to 2019 amount sum number.

	District	2015 (Consumption-m3)	2016 (Consumption-m3)	2017 (Consumption-m3)	2018 (Consumption-m3)	2019 (Consumption-m3)	Total
0	ADALAR	1432494	1472276	1366581	1399182	1371291	7041824
1	ARNAVUTKÖY	8002123	9178953	10176132	11404878	13818204	52580290
2	ATAŞEHİR	18597049	19974097	21205911	21496185	22428468	103701710
3	AVCILAR	16220476	17052253	17558403	18312736	19485453	88629321
4	B.ÇEKMECE	9915435	10788924	11172787	12222470	13606571	57706187

Reading Data Tail: we can also check last rows data from dataset by using df.tail(row num).

df.tail(5)

	District	2015 (Consumption-m3)	2016 (Consumption-m3)	2017 (Consumption-m3)	2018 (Consumption-m3)	2019 (Consumption-m3)	Total
34	ÇEKMEKÖY	8951936	9705483	10205702	10808688	11658154	51329963
35	ÜMRANİYE	28085173	30065065	31061816	32858425	33918205	155988684
36	ÜSKÜDAR	25142059	26110542	26353986	27205893	28298739	133111219
37	ŞİLE	1835302	2098928	2196246	2522410	2965979	11618865
38	ŞİŞLİ	15717630	15948709	16076563	17063291	18875509	83681702

Check Null Values: *df.isnull().values.any()* this function check dataset have any null value and given result True/False. True represent null value.

```
False
```

Our Dataset has no null values, so we don't need to treat any null values.

Data set info: To explore attributes data type and null value information we will use *df.info().*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   District                39 non-null     object
 1   2015 (Consumption-m3)   39 non-null     int64
 2   2016 (Consumption-m3)   39 non-null     int64
 3   2017 (Consumption-m3)   39 non-null     int64
 4   2018 (Consumption-m3)   39 non-null     int64
 5   2019 (Consumption-m3)   39 non-null     int64
 6   Total                   39 non-null     int64
dtypes: int64(6), object(1)
memory usage: 2.3+ KB
```

Exploring Statical Analysis the Dataset

Dataset mean:

```
df.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions

2015 (Consumption-m3)    1.587994e+07
2016 (Consumption-m3)    1.665096e+07
2017 (Consumption-m3)    1.728420e+07
2018 (Consumption-m3)    1.844990e+07
2019 (Consumption-m3)    1.978694e+07
Total                    8.805193e+07
dtype: float64
```

Dataset median:

```
df.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions

2015 (Consumption-m3)    16174934.0
2016 (Consumption-m3)    16584976.0
2017 (Consumption-m3)    17162651.0
2018 (Consumption-m3)    17804043.0
2019 (Consumption-m3)    19175326.0
Total                    86581261.0
dtype: float64
```

Dataset kurtosis:

```
df.kurtosis()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions

2015 (Consumption-m3)   -0.560966
2016 (Consumption-m3)   -0.327526
2017 (Consumption-m3)   -0.071568
2018 (Consumption-m3)    0.101722
2019 (Consumption-m3)    0.356175
Total                   -0.118692
dtype: float64
```

Dataset skewness

```
df.skew()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions

2015 (Consumption-m3)    0.017916
2016 (Consumption-m3)    0.198999
2017 (Consumption-m3)    0.266959
2018 (Consumption-m3)    0.293341
2019 (Consumption-m3)    0.337764
Total                    0.215699
dtype: float64
```

Checking correlation:

```
df.corr()
```

	2015 (Consumption-m3)	2016 (Consumption-m3)	2017 (Consumption-m3)	2018 (Consumption-m3)	2019 (Consumption-m3)	Total
2015 (Consumption-m3)	1.000000	0.983279	0.974886	0.967549	0.957332	0.981266
2016 (Consumption-m3)	0.983279	1.000000	0.997902	0.993500	0.987106	0.998189
2017 (Consumption-m3)	0.974886	0.997902	1.000000	0.997715	0.993190	0.998944
2018 (Consumption-m3)	0.967549	0.993500	0.997715	1.000000	0.996521	0.997551
2019 (Consumption-m3)	0.957332	0.987106	0.993190	0.996521	1.000000	0.993627
Total	0.981266	0.998189	0.998944	0.997551	0.993627	1.000000

Checking Covariance:

```
df.cov()
```

	2015 (Consumption-m3)	2016 (Consumption-m3)	2017 (Consumption-m3)	2018 (Consumption-m3)	2019 (Consumption-m3)	Total
2015 (Consumption-m3)	5.318456e+13	5.573382e+13	5.749504e+13	6.028044e+13	6.329532e+13	2.899892e+14
2016 (Consumption-m3)	5.573382e+13	6.040851e+13	6.272214e+13	6.596712e+13	6.955511e+13	3.143867e+14
2017 (Consumption-m3)	5.749504e+13	6.272214e+13	6.539849e+13	6.892882e+13	7.281698e+13	3.273615e+14
2018 (Consumption-m3)	6.028044e+13	6.596712e+13	6.892882e+13	7.298289e+13	7.718156e+13	3.453408e+14
2019 (Consumption-m3)	6.329532e+13	6.955511e+13	7.281698e+13	7.718156e+13	8.219263e+13	3.650416e+14
Total	2.899892e+14	3.143867e+14	3.273615e+14	3.453408e+14	3.650416e+14	1.642120e+15

Descriptive Statistics:

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
2015 (Consumption-m3)	39.0	1.587994e+07	7.292774e+06	1432494.0	11338749.5	16174934.0	20530603.5	29188561.0
2016 (Consumption-m3)	39.0	1.665096e+07	7.772291e+06	1472276.0	11552462.0	16584976.0	20894335.0	32799007.0
2017 (Consumption-m3)	39.0	1.728420e+07	8.086933e+06	1366581.0	11964930.0	17162651.0	21594798.5	36083030.0
2018 (Consumption-m3)	39.0	1.844990e+07	8.543002e+06	1399182.0	12717852.5	17804043.0	22844318.5	39205682.0
2019 (Consumption-m3)	39.0	1.978694e+07	9.066015e+06	1371291.0	13871993.5	19175326.0	25464289.0	43576187.0
Total	39.0	8.805193e+07	4.052308e+07	7041824.0	61613240.0	86581261.0	109764043.0	179597303.0

Finding out most water consumption:

```
df.groupby('District')['Total'].mean().sort_values(ascending=False)
```

```
District
ESENYURT          179597303.0
K.ÇEKMECE         161891430.0
ÜMRANİYE          155988684.0
PENDİK            150142554.0
BAĞCILAR          147871648.0
ÜSKÜDAR           133111219.0
KADIKÖY           128314577.0
BAHÇELİEVLER      126808760.0
MALTEPE           117599521.0
FATİH             115826376.0
ATAŞEHİR          103701710.0
KARTAL            103262043.0
BAŞAKŞEHİR        100765141.0
SARIYER            98248423.0
KAĞITHANE          97606023.0
SALTANGAZİ         96383158.0
GOP                95263805.0
EYÜPSULTAN         93823600.0
AVCILAR            88629321.0
ESENLER            86581261.0
ŞİŞLİ              83681702.0
SANCAKTEPE         76458060.0
BEYLİKDÜZÜ         75050593.0
GÜNGÖREN           71606564.0
TUZLA              69753877.0
ZEYTİNBURNU        68214183.0
BAKIRKÖY           67433576.0
BEŞİKTAŞ           65008521.0
BAYRAMPAŞA         62833200.0
BEYOĞLU            60393280.0
B.ÇEKMECE          57706187.0
BEYKOZ             57148082.0
SULTANBEYLİ        54240920.0
ARNAVUTKÖY         52580290.0
ÇEKMEKÖY           51329963.0
SİLİVRİ            43298547.0
ÇATALCA            17210496.0
ŞİLE               11618865.0
ADALAR              7041824.0
Name: Total, dtype: float64
```

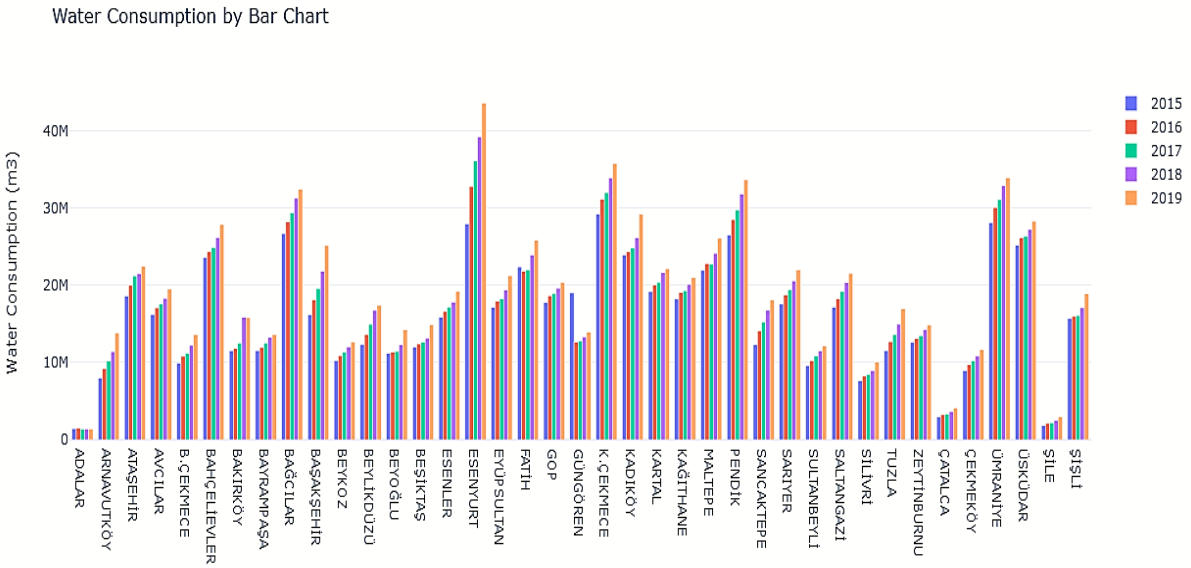
Visualization of Water Consumption in İstanbul

Visualizing the Dataset: We will visualize this given Dataset using different visualizing tools.

District wise Bar Chart- We can plot the count of different years water consumption of given districts in Bar Chart.

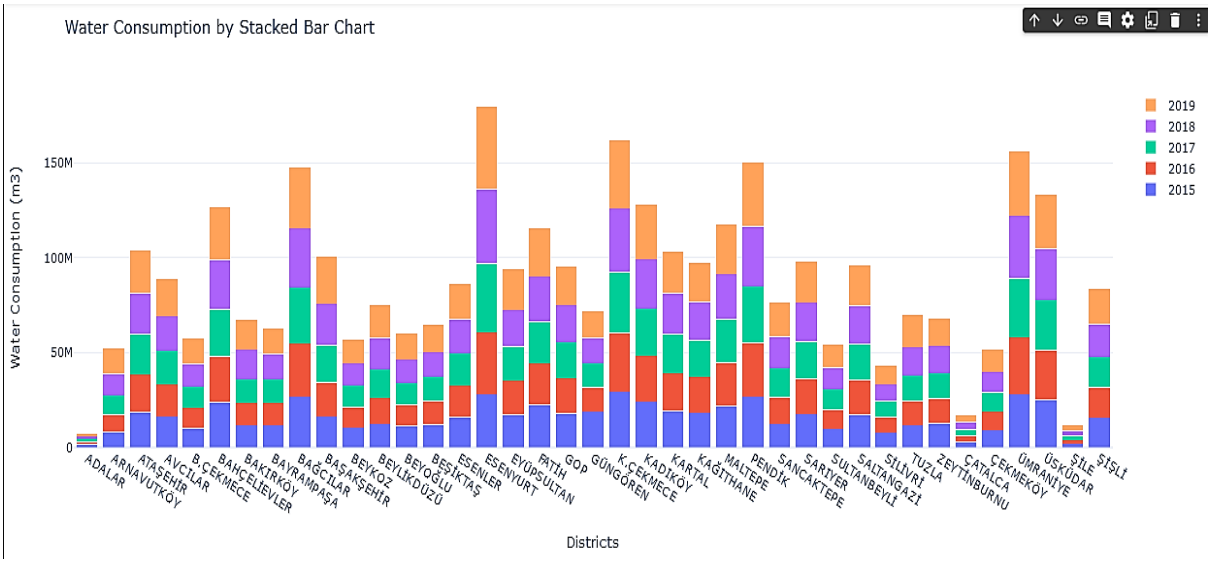
- **District wise Water Consumption in those years**

```
districts = df["District"].tolist()
fig = go.Figure(data=[
            go.Bar(name="2015", x=districts, y=df["2015 (Consumption-m3)"]),
            go.Bar(name="2016", x=districts, y=df["2016 (Consumption-m3)"]),
            go.Bar(name="2017", x=districts, y=df["2017 (Consumption-m3)"]),
            go.Bar(name="2018", x=districts, y=df["2018 (Consumption-m3)"]),
            go.Bar(name="2019", x=districts, y=df["2019 (Consumption-m3)"])
            ])
# Change the bar mode
fig.update_layout(
    title={"text": "Water Consumption by Bar Chart", },
    template = "plotly_white",
    xaxis = dict(title="Districts"),
     yaxis = dict(title="Water Consumption (m3)"))

fig.show()
```



- **Water Consumption by Stacked Bar Chart**

```
# Change the bar mode
 fig.update_layout(barmode="stack",
title={"text": "Water Consumption by Stacked Bar Chart", },
            template = "plotly_white",
           xaxis = dict(title="Districts"),
            yaxis = dict(title="Water Consumption (m3)"))
 fig.show()
```

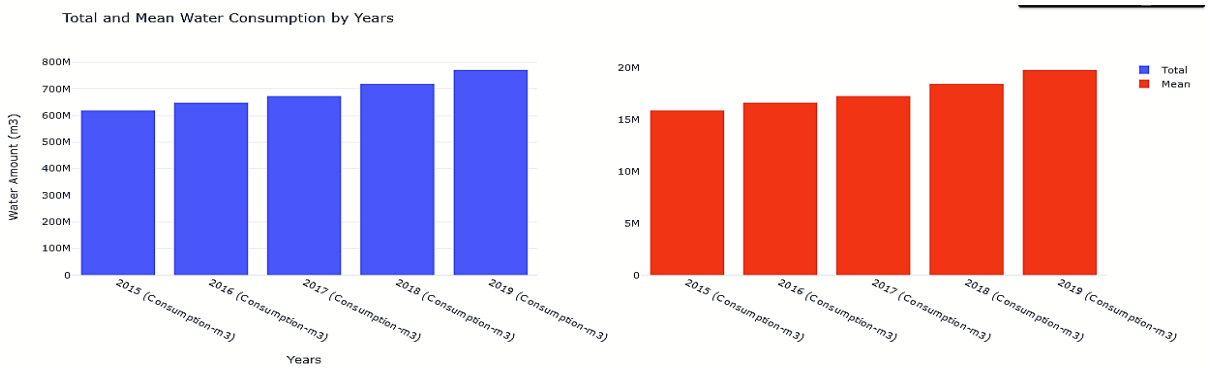


- **Total and Mean Water Consumption by Years**

fig = make_subplots(rows=1, cols=2)

#Total water consumption by years
df["2019 (Consumption-m3)"].sum()
fig.add_trace(go.Bar(
x=df.columns[1:6],
y=df[df.columns[1:6]].sum(),
orientation="v",
name="Total"), row=1, col=1)
 # Mean of water consumption by years

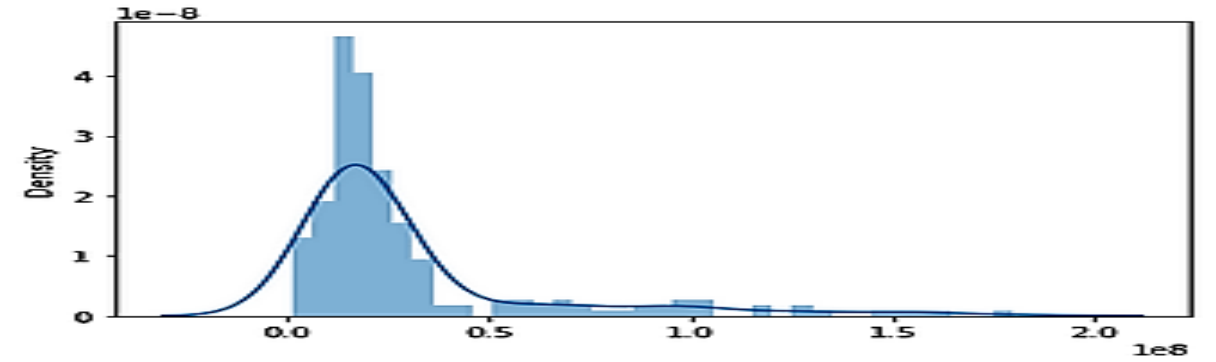
df["2019 (Consumption-m3)"].sum()
fig.add_trace(go.Bar(
x=df.columns[1:6],
y=df[df.columns[1:6]].mean(),
orientation="v",
name="Mean"), row=1, col=2)
fig.update_layout(title={"text": "Total and Mean Water Consumption by Years"},
template = "plotly_white",
xaxis = dict(title="Years"),
yaxis = dict(title="Water Amount (m3)"))
fig.show()



- **District wise parameter checking**

tot=df.groupby('District')['Total','2015 (Consumption-m3)','2016 (Consumption-m3)', '2017 (Consumption-m3)', '2018 (Consumption-m3)','2019 (Consumption-m3)'].median()

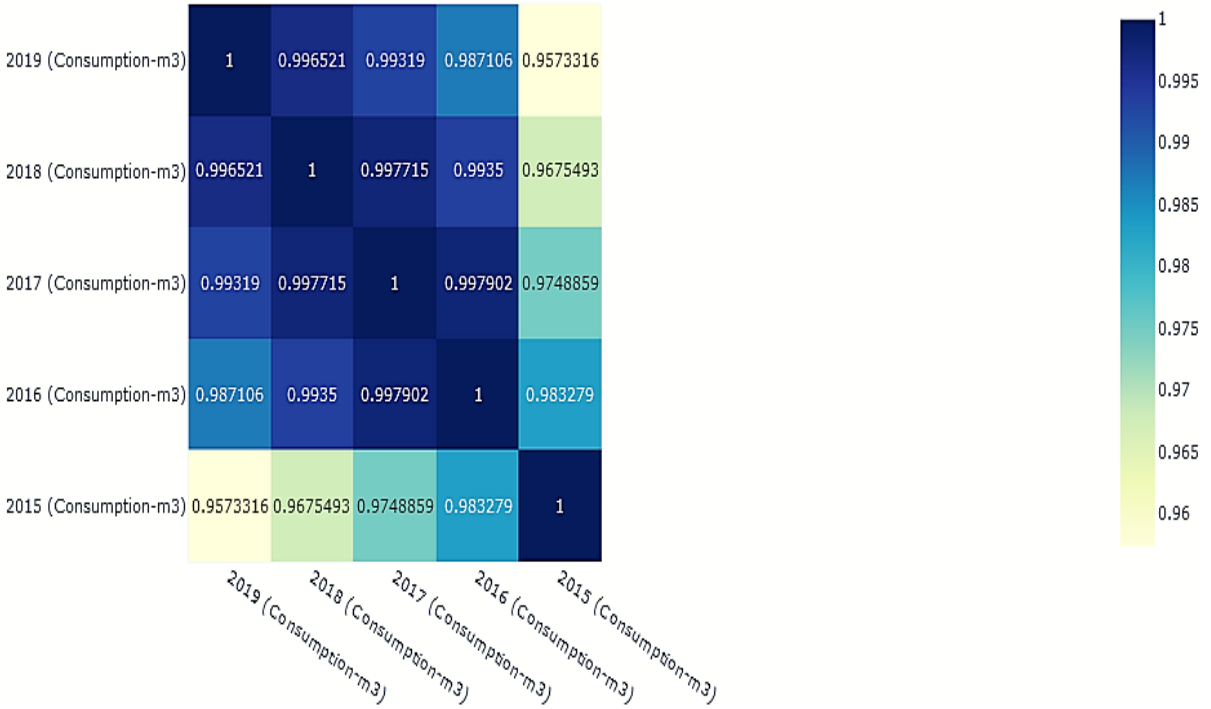
sns.distplot(tot)



Investigating Correlated Variables: We need to know the correlations of different variables with Target variables. That will indicate which variables are behind the water consumption of Istanbul.

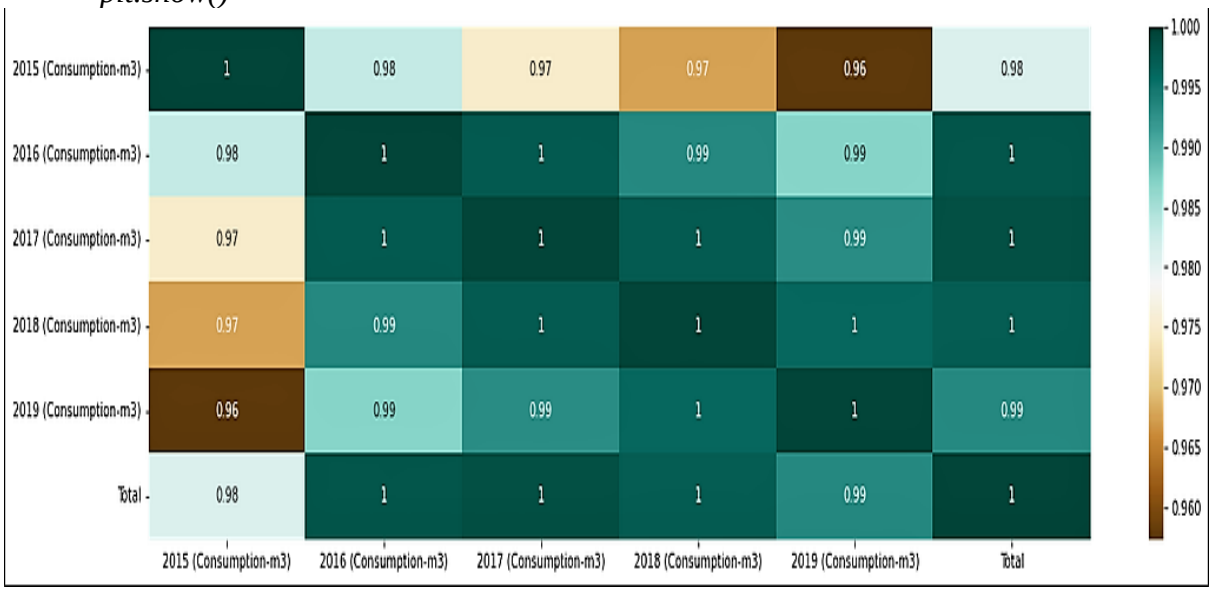
- **Heatmap by using plotly**

px.imshow(df.corr(), text_auto=True, color_continuous_scale="ylgnbu")



- **Heatmap by using Seaborn**

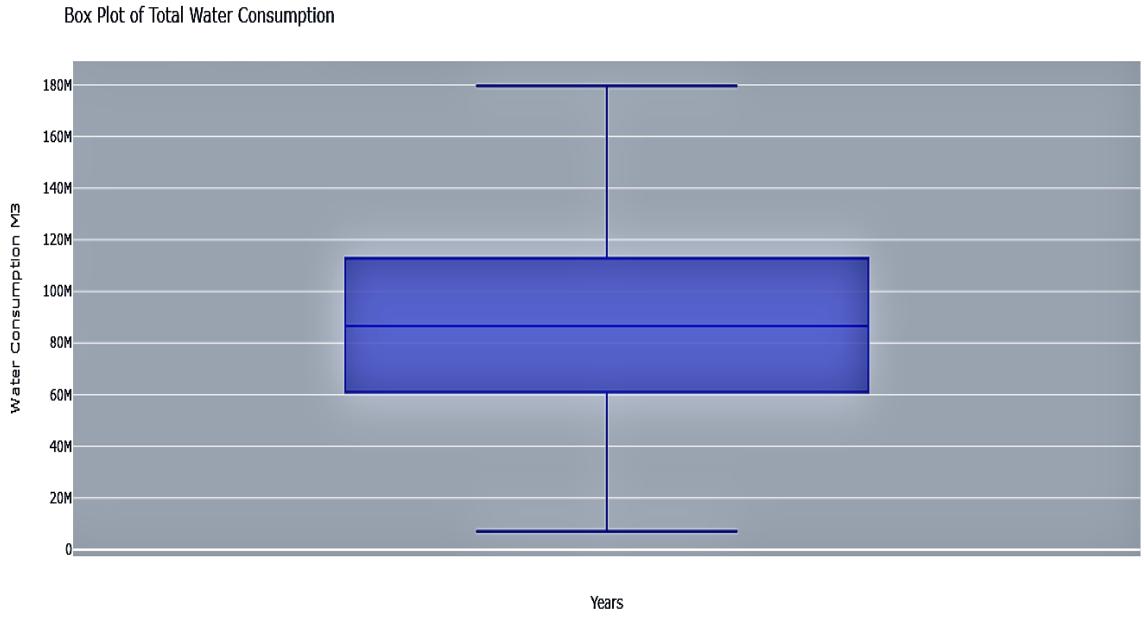
correlation = df.corr()
plt.figure(figsize=(20, 5))
sns.heatmap(correlation, annot=True, cmap="BrBG")
plt.show()



Comparing among significant variables and target variables: We can visualize the data range between any significance variable and target variable to visualize the relationship.

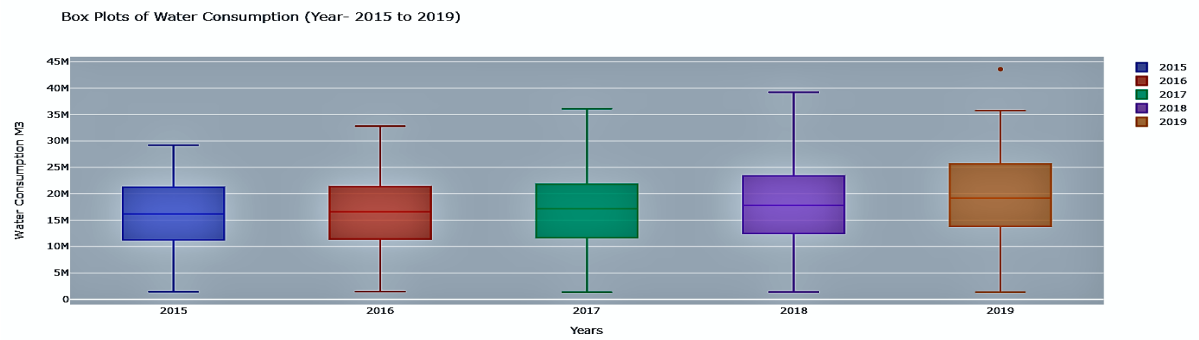
- **Box Plot of Total Water Consumption**

fig = px.box(df, y=df["Total"])
fig.update_layout(title="Box Plot of Total Water Consumption ",
xaxis_title="Years",
yaxis_title="Water Consumption M3")
fig.show()

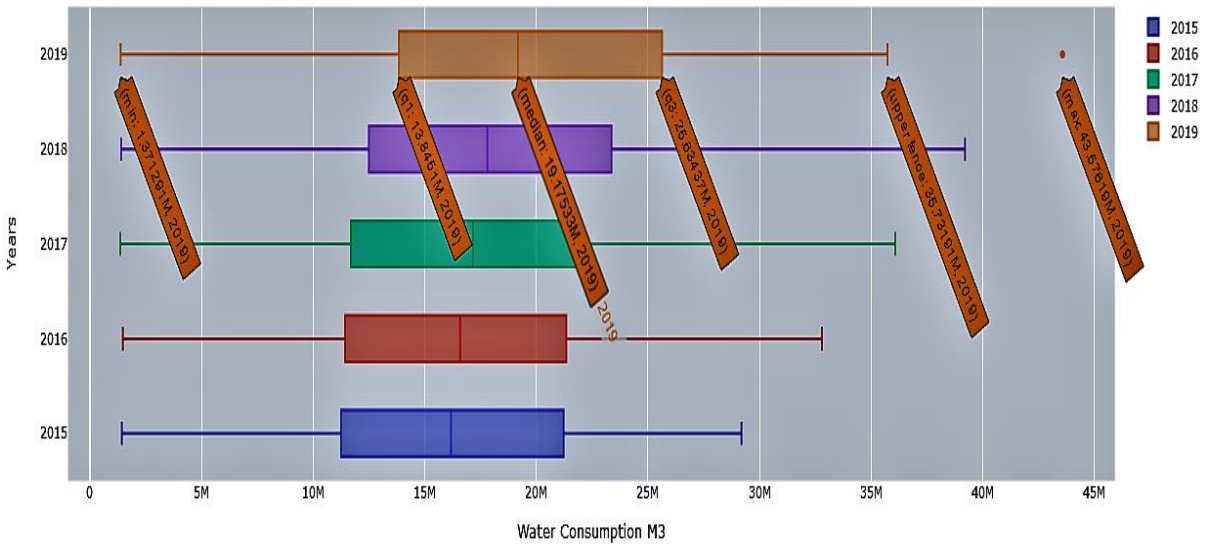


- **Box Plots of Water Consumption**

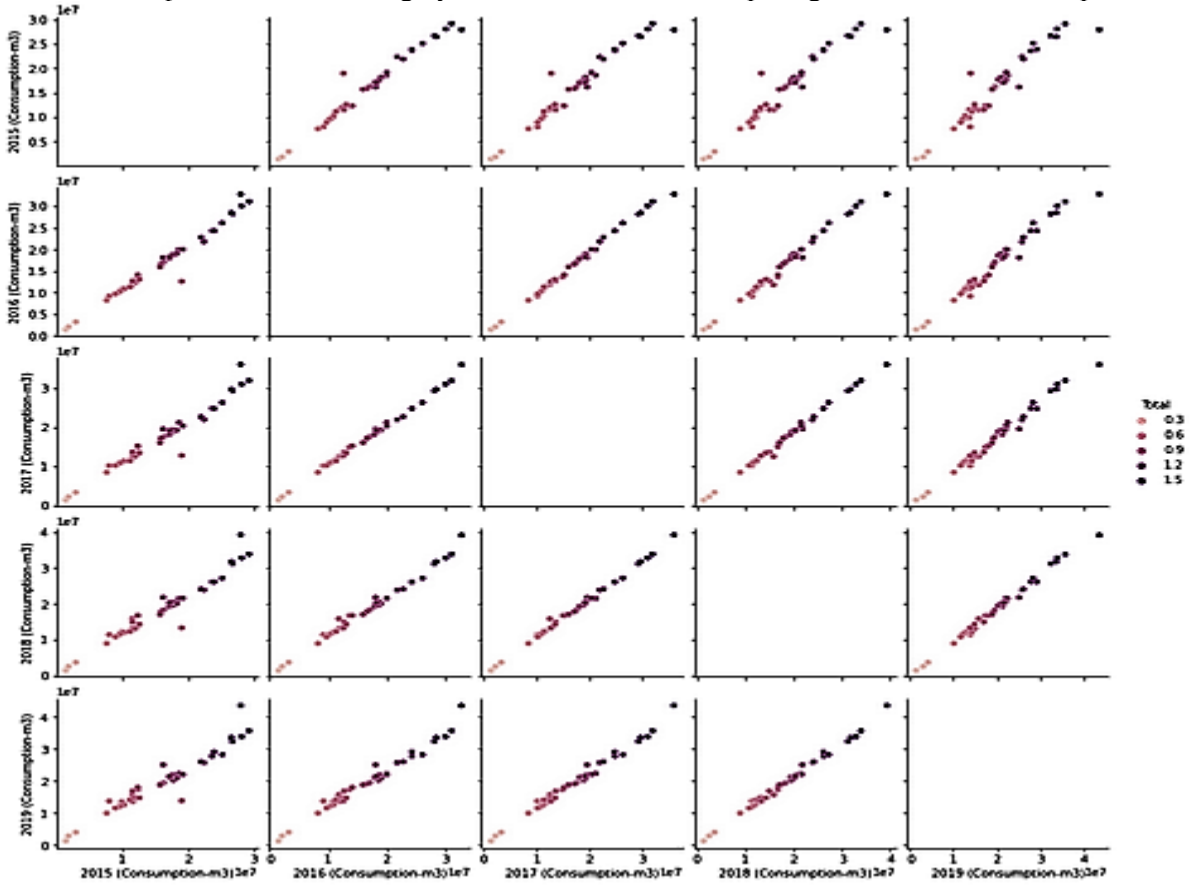
```
#Box Plots of Water Consumption
import plotly.graph_objects as go
y0 = df['2015 (Consumption-m3)']
y1 = df['2016 (Consumption-m3)']
y2 = df['2017 (Consumption-m3)']
y3 = df['2018 (Consumption-m3)']
y4 = df['2019 (Consumption-m3)']

fig = go.Figure()
fig.add_trace(go.Box(y=y0, name="2015"))
fig.add_trace(go.Box(y=y1, name="2016"))
fig.add_trace(go.Box(y=y2, name="2017"))
fig.add_trace(go.Box(y=y3, name="2018"))
fig.add_trace(go.Box(y=y4, name="2019"))

fig.update_layout(title="Box Plots of Water Consumption (Year- 2015 to 2019) ",
xaxis_title="Years",
yaxis_title="Water Consumption M3")

fig.show()
```



Water Consumption Box Plots



Internal correlation: As per our Heat map we found ‘water consumption 2015 to 2019’ and ‘Total water consumption’ is the most highly correlated variables comparing with other relationship.



Categorical Scaller chart: we found ‘water consumption 2015 to 2019’ and ‘District’ is the most highly correlated variables comparing with other relationship and water consumption increases when Year increases.