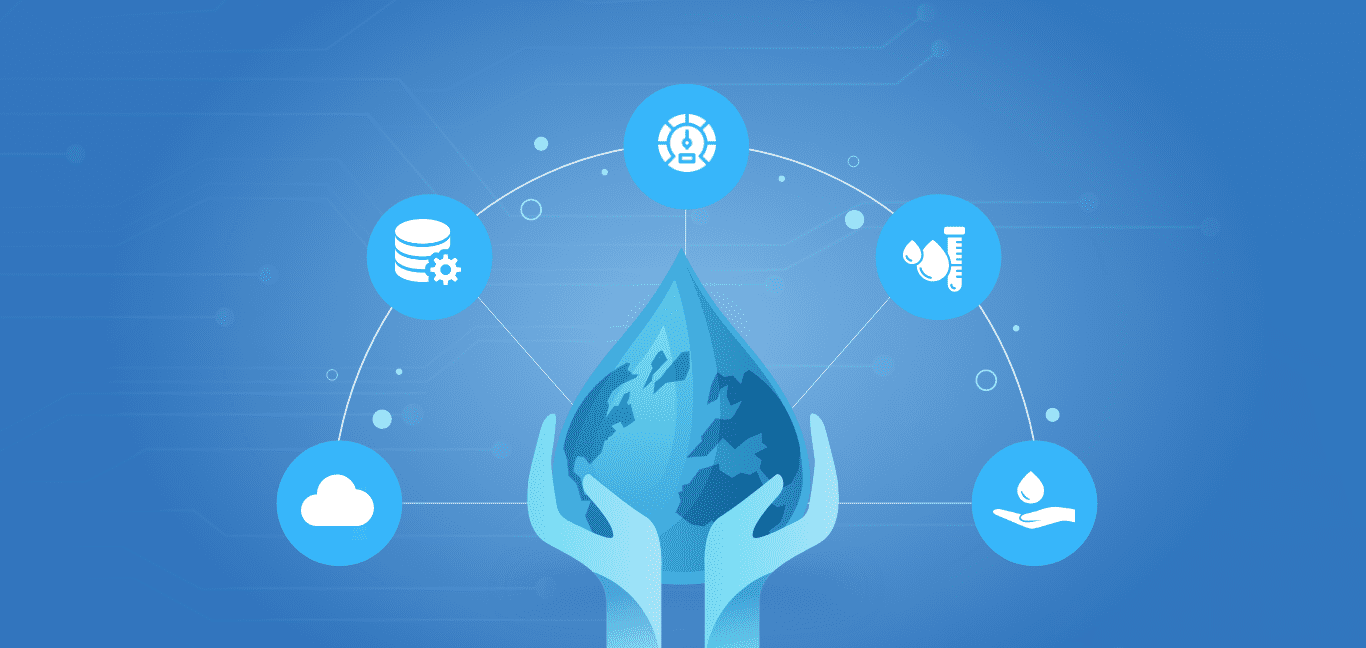
**SMART WATER MANAGEMENT in IoT**

**(Start building the smart water management by loading and pre-processing the dataset)**

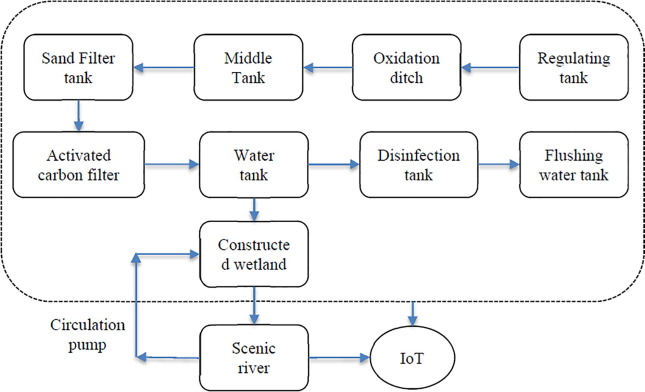
****

**SMART WATER MANAGEMENT-IoT**

**Phase 3:** **Development Part**

**INTRODUCTION**

The IoT has the ability to lessen this worrying picture. Smart Water Management Systems, based on the combination of sensors, big data and AI technologies, can provide to water utility operators, farmers and companies the ability to measure, monitor and control their water distribution networks as well as the quality of the water distributed. Less waste, less consumption, and a better management of the water’s quality can improve dramatically the preservation of our planet’s resources.



**Introduction to IoT Smart Water Management with Python**

* Automation in the pumping system for filling up the water tank. A sensor is placed on the top of the tank which constantly monitors the level of the water being supplied to the tank.
* As the water reaches the limiting level the water pump is automatically turned off. It also calculates the running time and the power consumed by the motor. The data is thereafter stored in the cloud.
* The data can be easily fetched to have budget estimation per month. It also notifies the weather conditions, if the weather is predicted to be bad the notification about filling up the tank would come which in response helps up in filling up the tank when there is a power supply using IoT.
* **Importing Libraries**

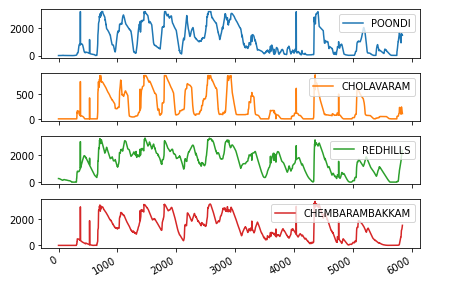
import pandas as pd  
import numpy as np  
import math  
import matplotlib.pyplot as plt  
import matplotlib.patches as mpatches  
import datetime  
import seaborn as sns  
import plotly.graph\_objs as go  
import plotly.express as px  
from plotly.offline import download\_plotlyjs, init\_notebook\_mode, plot, iplot  
init\_notebook\_mode(connected=True)  
%matplotlib inline  
from IPython.display import display  
import statsmodels.api as sm  
from pylab import rcParams  
from statsmodels.tsa.stattools import adfuller  
import itertools  
colors = plt.rcParams['axes.prop\_cycle'].by\_key()['color']

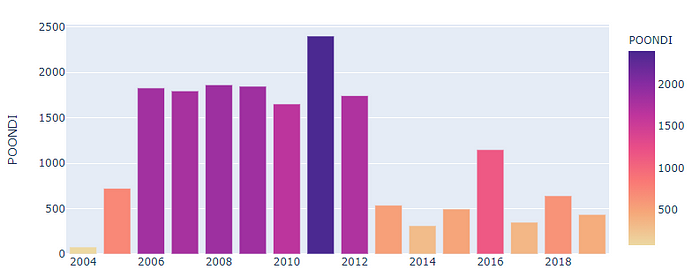
* **Loading Dataset**

df\_reservoir = pd.read\_csv("/content/gdrive/My Drive/ECO6800\_PROJECT/chennai\_reservoir\_levels.csv")df\_rainfall = pd.read\_csv("/content/gdrive/My Drive/ECO6800\_PROJECT/chennai\_reservoir\_rainfall.csv")df\_Groundwater = pd.read\_csv("/content/gdrive/My Drive/ECO6800\_PROJECT/Groundwater.csv")df\_population = pd.read\_csv("/content/gdrive/My Drive/ECO6800\_PROJECT/chennai\_pop.csv", index\_col=0, parse\_dates=True)rainfall\_time = pd.read\_csv("/content/gdrive/My Drive/ECO6800\_PROJECT/chennai\_reservoir\_rainfall.csv", index\_col=0, parse\_dates=True, dayfirst=True)reservoir\_time = pd.read\_csv("/content/gdrive/My Drive/ECO6800\_PROJECT/chennai\_reservoir\_levels.csv", index\_col=0, parse\_dates=True, dayfirst=True)

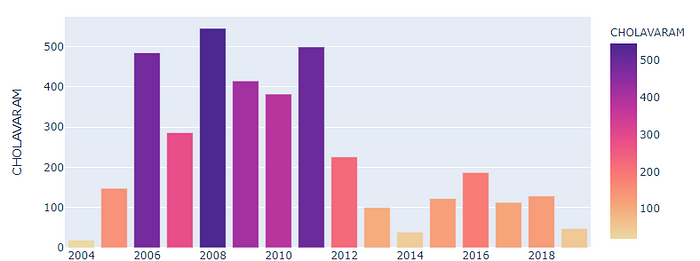
* **Data Exploration**

df\_reservoir.plot(subplots=True)  
plt.tight\_layout()  
plt.show()

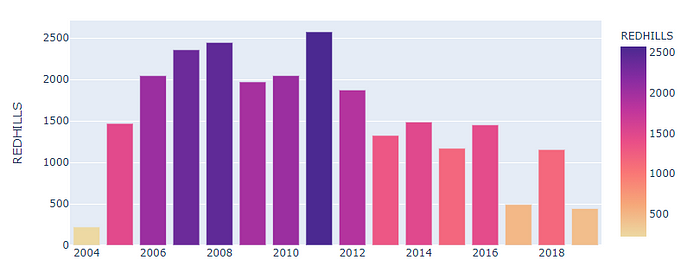


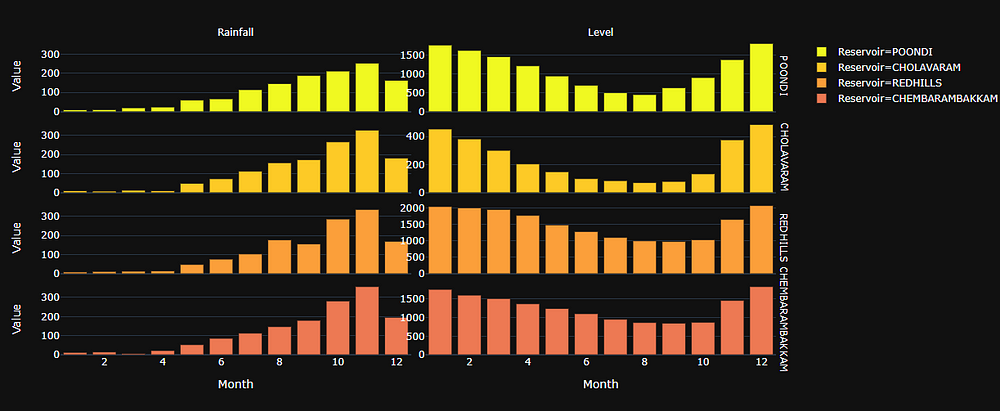


* This is an interactive plot and values can be seen by hovering over it in python.

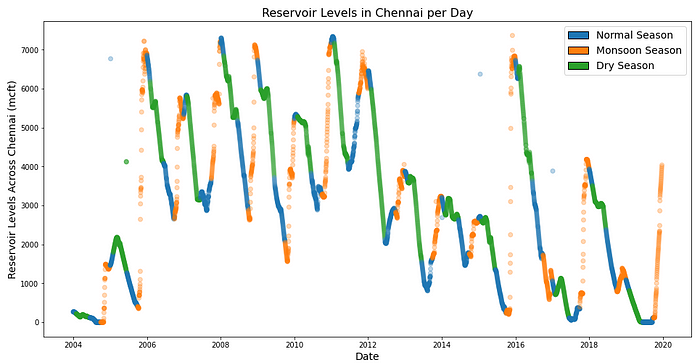


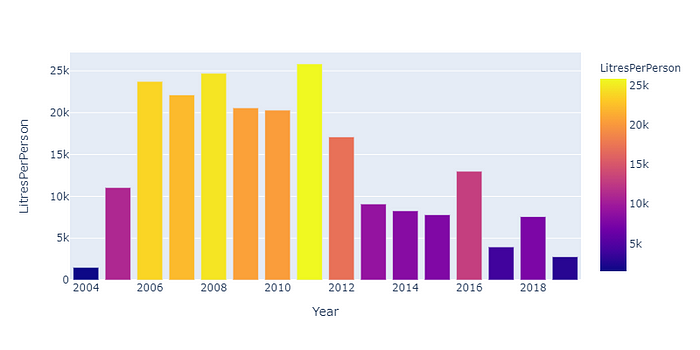
* This is an interactive plot and values can be seen by hovering over it in python.



* This is an interactive plot and values can be seen by hovering over it in python.
* **Season-wise Analysis**

def read\_input\_file(filepath):  
frame = pd.read\_csv(filepath)  
frame['Date'] = pd.to\_datetime(frame['Date'], format='%d-%m-%Y')  
frame = frame.sort\_values('Date', ascending=True).reset\_index(drop=True)  
return framedef assign\_temporal\_categories(date):  
"""Given a date, determines whether to color for 'Monsoon', 'Dry', or 'Regular' season."""  
if date.month in [2, 3, 4, 5] or (date.month == 6 and date.day <= 15):  
return colors[2]  
elif date.month in [10, 11, 12]:  
return colors[1]  
else:  
return colors[0]reservoir\_level = read\_input\_file("/content/gdrive/My Drive/ECO6800\_PROJECT/chennai\_reservoir\_levels.csv")  
# Plot the reservoir levels per day summed across all 4 reservoirs  
c = reservoir\_level['Date'].apply(assign\_temporal\_categories, colors)  
chennai\_reservoirs = reservoir\_time.iloc[:, 1:].sum(axis=1)plt.figure(figsize=(16, 8))  
plt.scatter(reservoir\_level['Date'].values, chennai\_reservoirs, color=c, alpha=0.3)  
plt.ylabel('Reservoir Levels Across Chennai (mcft)', size=14)  
plt.xlabel('Date', size=14)  
legend\_labels = {'Normal Season':colors[0], 'Monsoon Season':colors[1], 'Dry Season':colors[2]}  
leg\_el = [mpatches.Patch(facecolor=value, edgecolor='k', label=key) for key, value in legend\_labels.items()]  
plt.legend(handles=leg\_el, prop={'size':14})  
plt.title('Reservoir Levels in Chennai per Day', size=16)plt.show(renderer="colab")

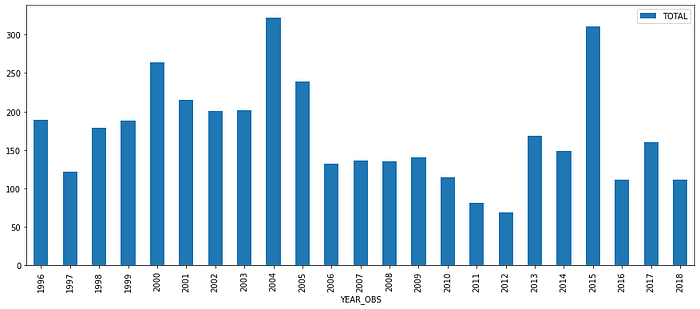




* This is an interactive plot and values can be seen by hovering over it in python.

**Ground Water Analysis**

df\_Groundwater=df\_Groundwater[['YEAR\_OBS','TOTAL']]  
df\_Groundwater=df\_Groundwater.groupby('YEAR\_OBS').sum()  
df\_Groundwater.plot(kind='bar')



* The persistent decline in the ground water trend further emphasizes the severity of the shortage crisis that the city is facing, and may continue to face. To check the future trends of water availability, we now use MLA algorithms and make predictions.

## ****Future Predictions****

We now aim to predict the future rainfall so as to see any specific trends or causes of concerns, and thereby try to comment on the future of scarcity of water in the city