```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import json
import folium
```

```
os.chdir("G:\Padhai\Mtech\Data Visualization Lab\Project\Datasets")
path =os.getcwd()+"/Final Datasets"
```

The data we have is of Rainfall - yearly data which will be used for time series data to show the rainfall changes over the years - this can be used to show the impact of Climate change on rainfall patterns

Rainfall _districtwise - is essentially a single time data, where it shows the data of some year - it will be used to show the rainfall pattern in any given year in the country This can be used to relate forest covers, agriculture, vegetation and various relief features in various parts of the country as rainfall affect it a lot.

Climate Change in India is something that we can consider

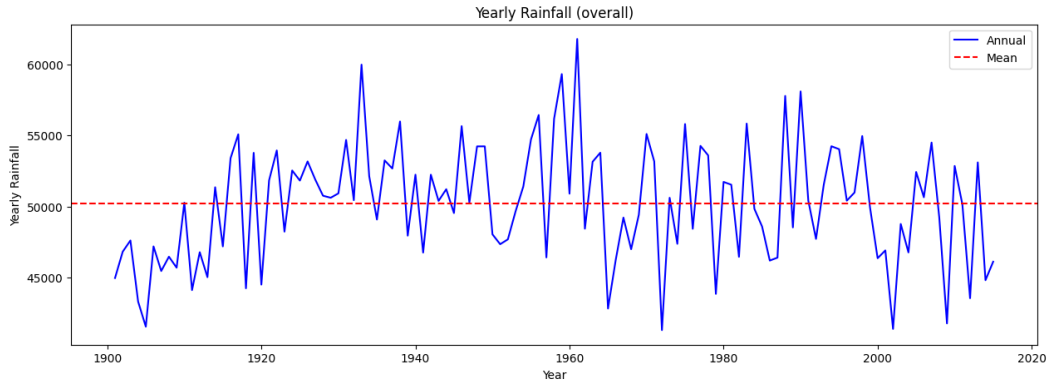
```
# Rainfall data, yearly for timeseries plots

df_1 = pd.read_csv(path+"/Rainfall_yearly.csv")

#df_1.head()
df_1 = df_1.rename(columns = {'SUBDIVISION':'State', 'JAN': 'Jan', 'FEB': 'Feb', 'MAR': 'Mar',
                       'APR': 'Apr' , 'MAY': 'May', 'JUN': 'Jun', 'JUL': 'Jul', 'AUG': 'Aug', 'SEP': 'Sep', 'OCT': 'Oct'
                      , 'NOV': 'Nov', 'DEC': 'Dec', 'ANNUAL': 'Annual', 'YEAR': 'Year'})
#df_1.columns

df_1_year = df_1.groupby("Year")
df_1_state = df_1.groupby("State")
```

```
df_2 = pd.read_csv(path+"/Rainfall_Districtwise.csv")
# df_2.head()

df_2 = df_2.rename(columns= {'STATE_UT_NAME' : "State",'JAN': 'Jan', 'FEB': 'Feb', 'MAR': 'Mar',
                       'APR': 'Apr' , 'MAY': 'May', 'JUN': 'Jun', 'JUL': 'Jul', 'AUG': 'Aug', 'SEP': 'Sep', 'OCT': 'Oct'
                      , 'NOV': 'Nov', 'DEC': 'Dec', 'ANNUAL': 'Annual'})

#df_2.columns
#df_2 = df_2.rename(columns="ST")
```

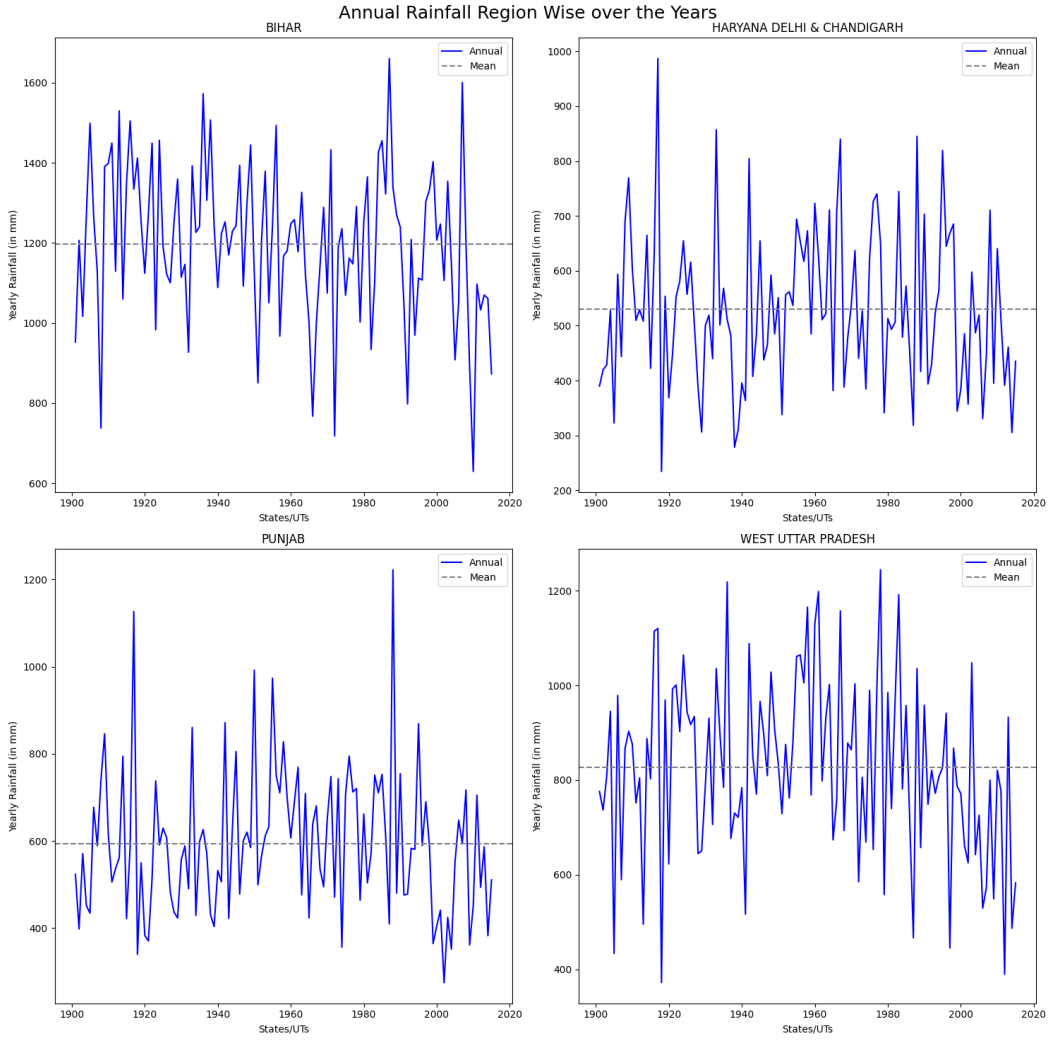
```
# p = list(df_1.columns[2:])
# #print(list(p))
# df_cpy = df_1.copy()
# piv_df = df_cpy.pivot(index="State",columns="Year",values=p)

# piv_df
```

```
# overall rainfall in India over the years

plt.figure(figsize=(15,5))
df_1.groupby(['Year'])['Annual'].sum().plot(kind='line', color = 'b')
plt.ylabel('Yearly Rainfall')
plt.xlabel("Year")
plt.title('Yearly Rainfall (overall)')
# Add a horizontal line for the mean annual rainfall
plt.axhline(df_1.groupby(['Year'])['Annual'].sum().mean(), color='red', linestyle='--', label='Mean')
plt.legend()
```

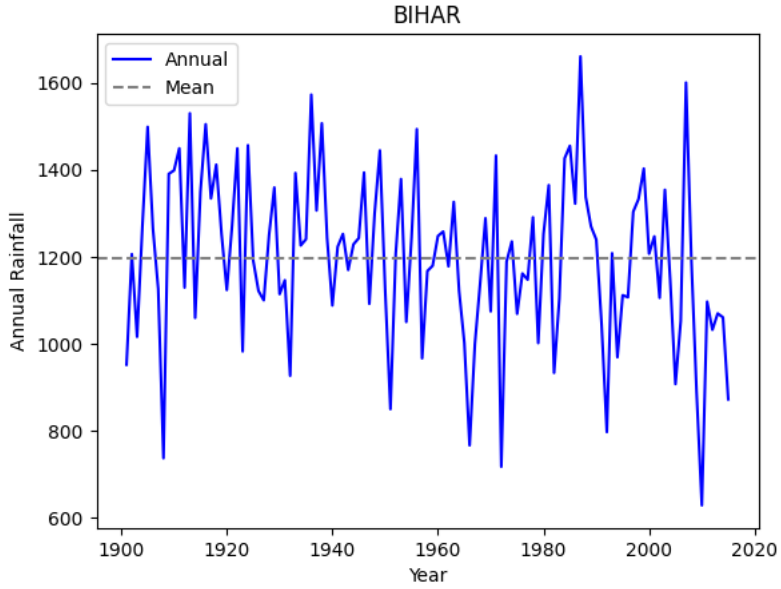
```
<matplotlib.legend.Legend at 0x2328e8f5a90>
```

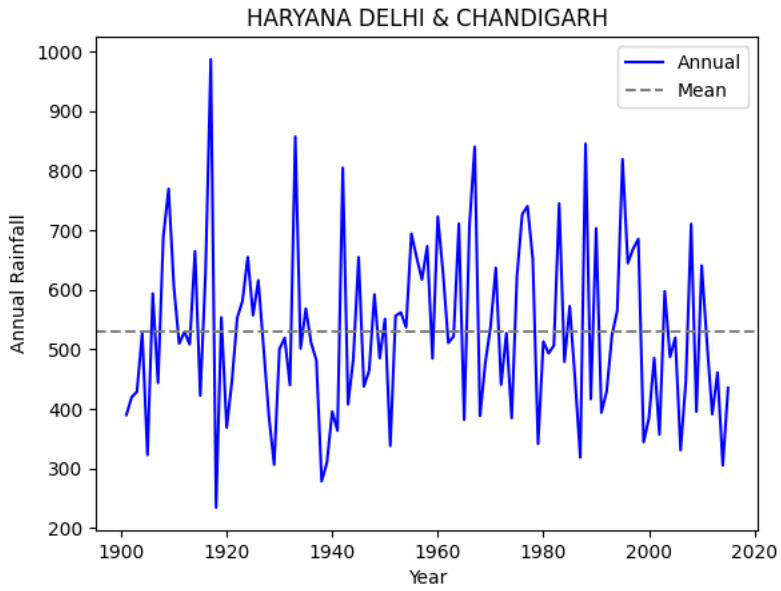


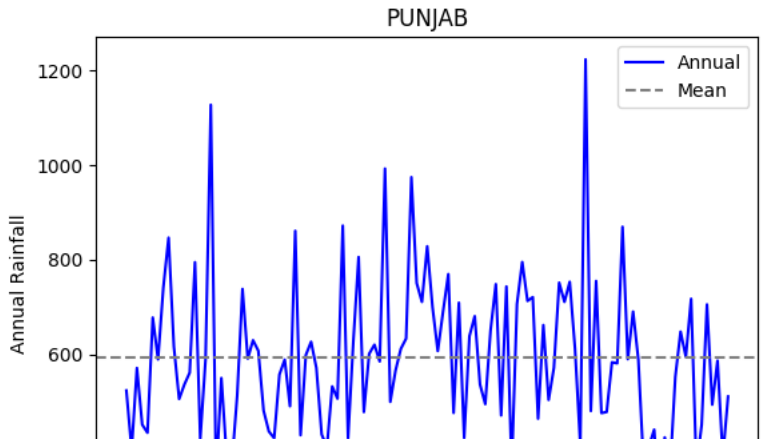

```
lst = ["BIHAR","HARYANA DELHI & CHANDIGARH","PUNJAB","WEST UTTAR PRADESH"]

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,15))

for ax, state in zip(axes.flatten(),lst):
    # Select only the data for the current subdivision
    data = df_1[df_1["State"] == state]
    ax = data.groupby(['Year'])['Annual'].sum().plot(kind='line', color = 'b',ax=ax)
    ax.axhline(data.groupby(['Year'])['Annual'].sum().mean(), color='gray', linestyle='--', label='Mean')
    ax.set_title(state, fontsize=12)
    ax.set_xlabel("States/UTs")
    ax.set_ylabel("Yearly Rainfall (in mm)")
    ax.legend()



fig.suptitle("Annual Rainfall Region Wise over the Years",fontsize=18)

# adjust the position of the main title
fig.subplots_adjust(top=0.9)
plt.tight_layout()
plt.show()
```




```
lst = ["BIHAR","HARYANA DELHI & CHANDIGARH","PUNJAB","WEST UTTAR PRADESH"]

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,15))

for ax, state in zip(axes.flatten(),lst):
    # Select only the data for the current subdivision
    data = df_1[df_1["State"] == state]
    ax = sns.lineplot()
    data.groupby(['Year'])['Annual'].sum().plot(kind='line', color = 'b')
    # # Extract the year and annual rainfall data
    # years = data['YEAR']
    # annual_rainfall = data['ANNUAL']

    # # Calculate the mean annual rainfall for the current subdivision
    # mean_rainfall = annual_rainfall.mean()

    # # Plot the annual rainfall over time
    # plt.plot(years, annual_rainfall)

    # Add a horizontal line for the mean annual rainfall
    plt.axhline(data.groupby(['Year'])['Annual'].sum().mean(), color='gray', linestyle='--', label='Mean')

    # Add a title and labels to the plot
    plt.title(state)
    plt.xlabel('Year')
    plt.ylabel('Annual Rainfall')
    plt.legend()

    # Show the plot
    plt.show()
```

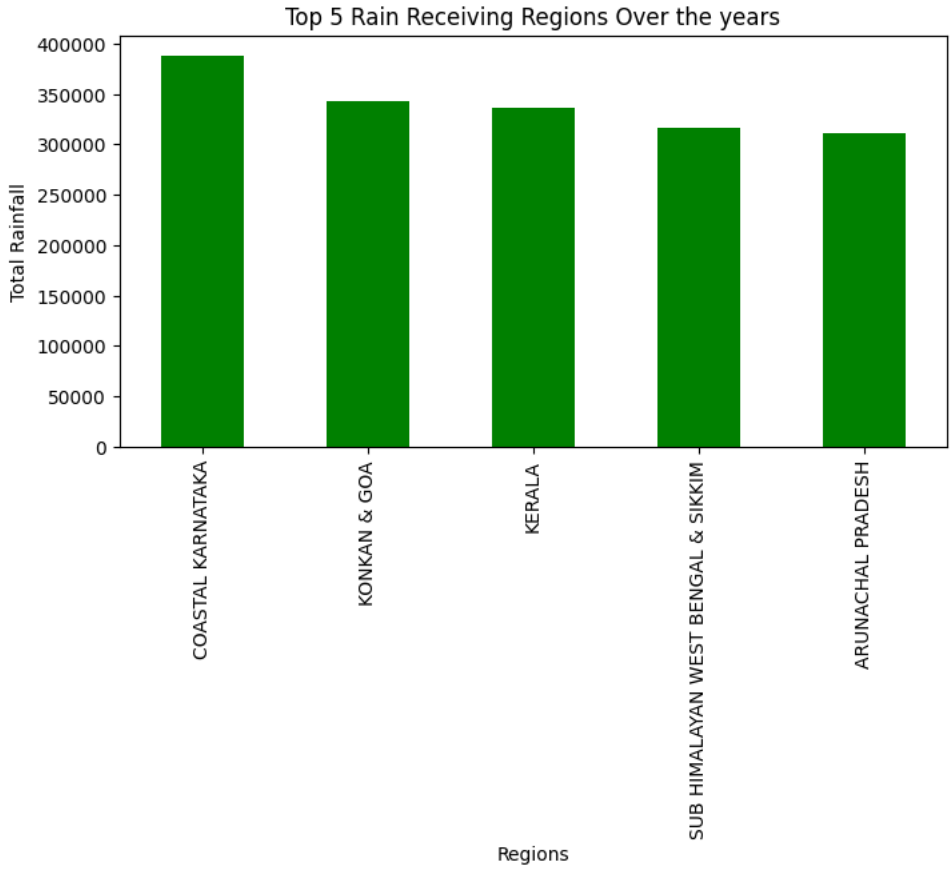

BIHAR
Annual
Mean
Annual Rainfall
Year
1600
1400
1200
1000
800
600
1900
1920
1940
1960
1980
2000
2020


HARYANA DELHI & CHANDIGARH
Annual
Mean
Annual Rainfall
Year
1000
900
800
700
600
500
400
300
200
1900
1920
1940
1960
1980
2000
2020


PUNJAB
Annual
Mean
Annual Rainfall
1200
1000
800
600

```
# top 5 rain recieving states

plt.figure(figsize=(8,4))
df_1.groupby(['state'])['annual'].sum().sort_values(ascending=False).head(5).plot(kind='bar', color = 'g')
plt.ylabel('Total Rainfall')
plt.xlabel("Regions")
plt.title('Top 5 Rain Receiving Regions Over the years')

# we didn't use the mean values because that was bringing the values too close and it wasn't a good measure to rank the
# states as per the rainfall received
```

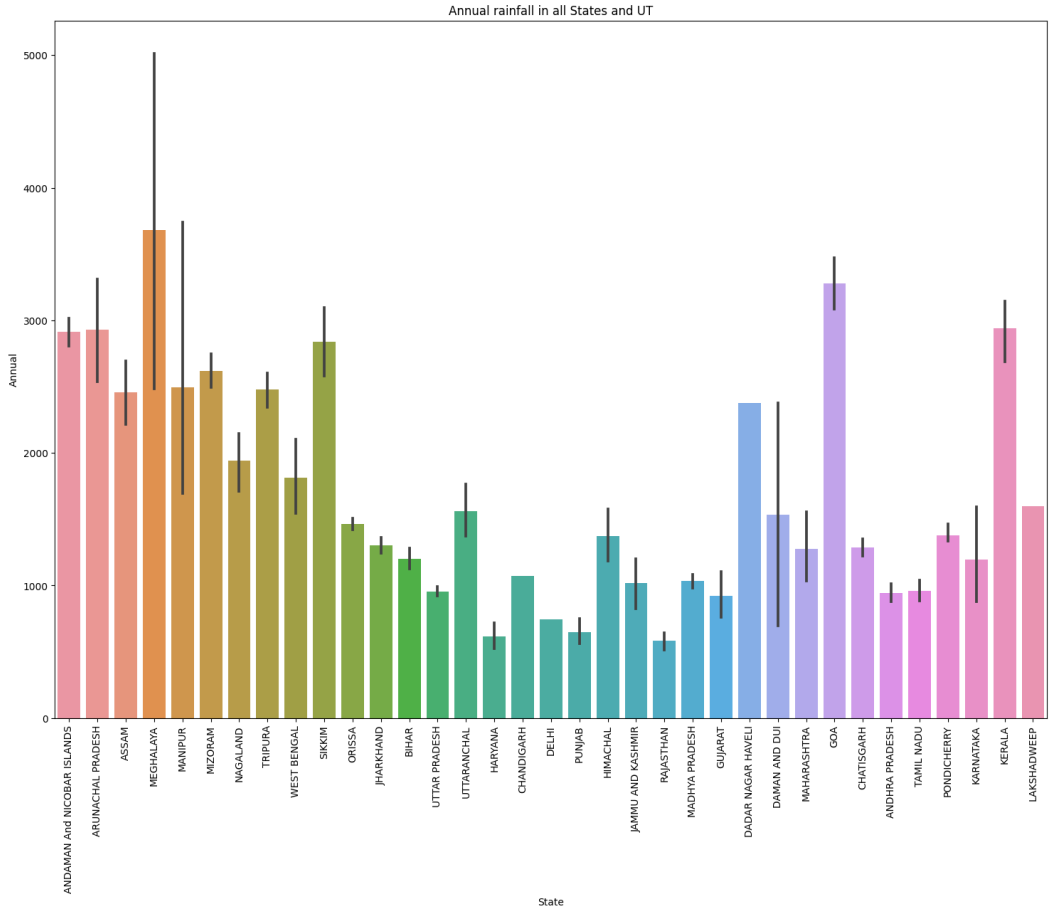
```
Text(0.5, 1.0, 'Top 5 Rain Receiving Regions Over the years')
```



The above insight of top five regions which receive highest rainfall seems to be correct. As Coastal Karnataka is in western ghats which are known to have Evergreen rainforest.

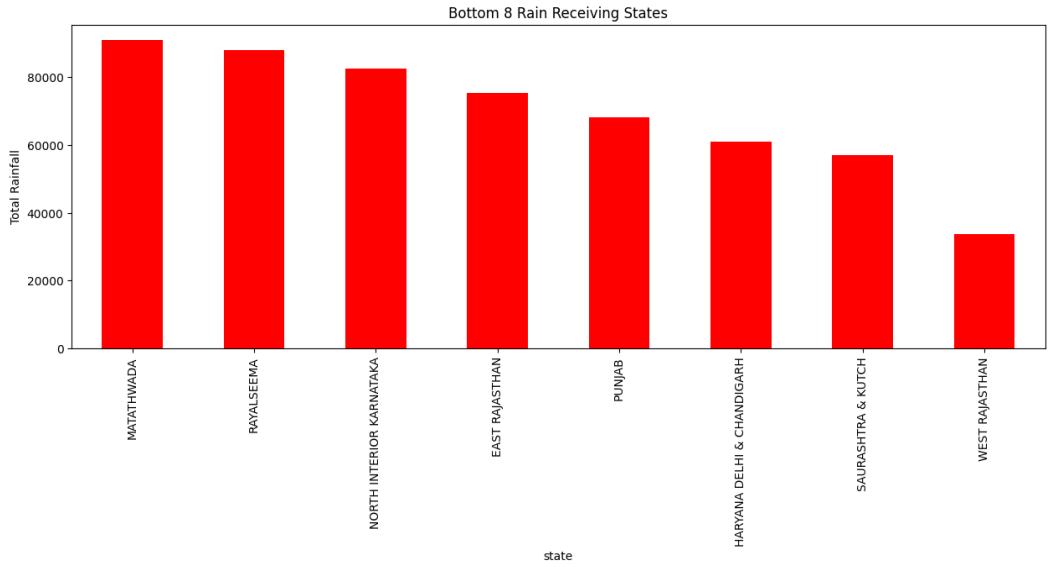
Double-click (or enter) to edit

```
# plt.figure(figsize=(15,5))
# df_2.groupby(['State'])['Annual'].sum().sort_values(ascending=False).plot(kind='bar', color = 'g')
# plt.ylabel('Total Rainfall')
# plt.title('Top 8 Rain Receiving States')  - this is showing wrong, coz UP is shown to have highest rainfall, which is not true, this must b
# region and more data on UP

fig = plt.figure(figsize=(18, 28))
ax = plt.subplot(2,1,1)
ax = plt.xticks(rotation=90)
ax = plt.title('Avg Annual rainfall in all States and UT')
ax = sns.barplot(x='State', y='Annual', data=df_2)    # sns plot by default takes avg of all the values of the group, which is state here
```




```
plt.figure(figsize=(15,5))
df_1.groupby(['state'])['annual'].sum().sort_values(ascending=False).tail(8).plot(kind='bar', color = 'r')
plt.ylabel('Total Rainfall')
plt.title('Bottom 8 Rain Receiving States')
```

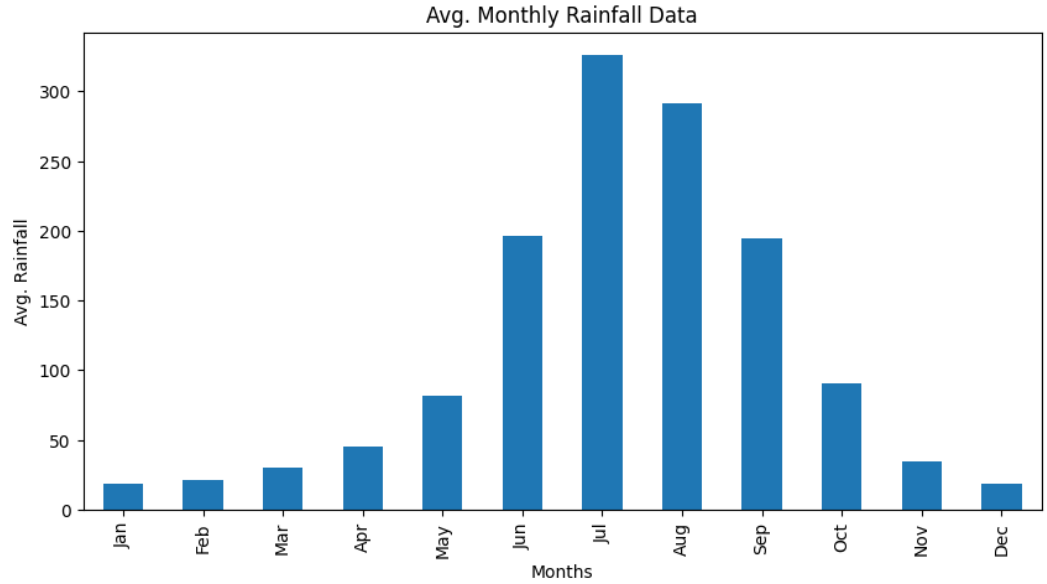
```
Text(0.5, 1.0, 'Bottom 8 Rain Receiving States')
```




```
df_2.columns
```

```
Index(['State', 'DISTRICT', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
       'Aug', 'Sep', 'Oct', 'Nov', 'Dec', 'Annual', 'Jan-Feb', 'Mar-May',
       'Jun-Sep', 'Oct-Dec', 'Region'],
      dtype='object')
```

```
plt.figure(figsize=(10,5))
df_2[['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
       'Sep', 'Oct', 'Nov', 'Dec']].mean().plot(kind= 'bar')
plt.xlabel('Months')
plt.ylabel('Avg. Rainfall')
plt.title('Avg. Monthly Rainfall Data')
plt.show()
```

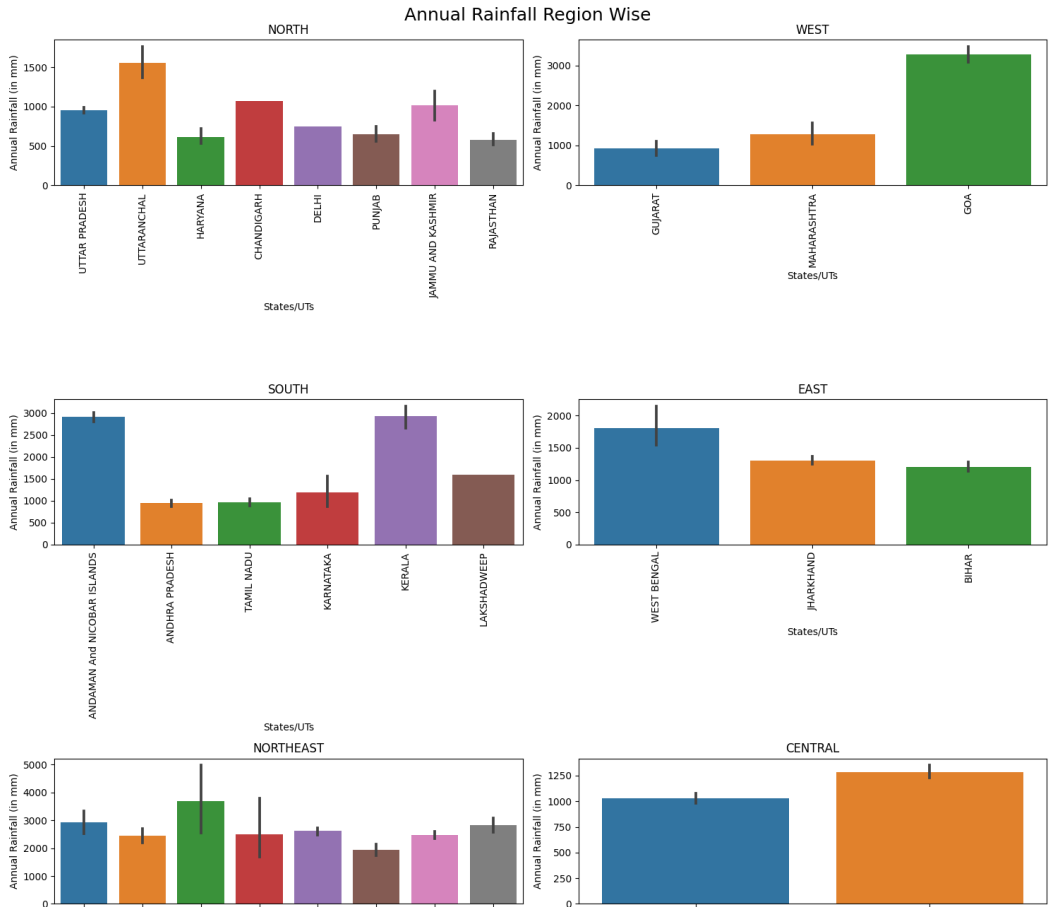



```
dic = {
    'NORTH': ['JAMMU AND KASHMIR', 'HIMACHAL PRADESH', 'PUNJAB', 'CHANDIGARH', 'UTTARANCHAL', 'HARYANA', 'DELHI', 'UTTAR PRADESH', 'RAJASTHAN
    'WEST': ['GUJARAT', 'DAMAN AND DIU', 'DADRA AND NAGAR HAVELI', 'MAHARASHTRA', 'GOA'],
    'SOUTH': ['KARNATAKA', 'ANDHRA PRADESH', 'TAMIL NADU', 'KERALA', 'PUDUCHERRY', 'TELANGANA','LAKSHADWEEP', 'ANDAMAN And NICOBAR ISLANDS'],
    'EAST': ['ODISHA', 'WEST BENGAL', 'JHARKHAND', 'BIHAR'],
    'NORTHEAST': ['ARUNACHAL PRADESH', 'ASSAM', 'MANIPUR', 'MEGHALAYA', 'MIZORAM', 'NAGALAND', 'SIKKIM', 'TRIPURA'],
    'CENTRAL': ['MADHYA PRADESH', 'CHATISGARH']
}
```

```
# Create a new column 'Region' in the existing DataFrame and map the states/union territories with their regions
#df_2['Region'] = df_2['State'].map({state: region for region, states in regions.items() for state in states})
df_2["Region"] = df_2["State"].map({state: region for region, states in dic.items() for state in states})
df_2["Region"]
```

```
0          SOUTH
1          SOUTH
2          SOUTH
3      NORTHEAST
4      NORTHEAST
          ...
636        SOUTH
637        SOUTH
638        SOUTH
639        SOUTH
640        SOUTH
Name: Region, Length: 641, dtype: object
```

```
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15,15))

for ax ,region in zip(axes.flatten(),dic.keys()):
    #plt.figure(figsize=(12, 6))
    ax = sns.barplot(x='State', y='Annual', data=df_2[df_2["Region"] == region],ax=ax)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha='center')
    ax.set_title(region, fontsize=12)
    ax.set_xlabel("States/UTs")
    ax.set_ylabel("Annual Rainfall (in mm)")
    # plt.title(f'Annual Rainfall in {region} Region')
    # plt.xlabel('State/UT')
    # plt.ylabel('Annual Rainfall (in mm)')
    # plt.tight_layout()
    # plt.show()

fig.suptitle("Annual Rainfall Region Wise",fontsize=18)

# adjust the position of the main title
fig.subplots_adjust(top=0.9)
plt.tight_layout()
plt.show()
```



Climate Change Part

The data used here has been sourced from Climate Change Knowledge Portal(CCKP) of World Bank, but we downloaded the curated data from the Kaggle. The data is in the form that we wanted it to be.

About Data: The data shows mean surface temperature of various States and UT's of the country from 1901-2020, and we also have these temperatures by Monthly, Annually and divided in the various seasons, such as Summer, Monsoon, Post Monsoon and Winter

```
df_c = pd.read_csv(path+"/Mean Temperature Data.csv")

#df_c


# Function for zonal-classification of states
def zones(state):
    if state in ['Chandigarh','Delhi','Haryana','Himachal Pradesh','Punjab','Rajasthan']: return 'North'
    elif state in ['Bihar', 'Orissa', 'Jharkhand', 'West Bengal']: return 'East'
    elif state in ['Dadra and Nagar Haveli','Daman and Diu','Goa','Gujarat','Maharashtra']: return 'West'
    elif state in ['Andhra Pradesh','Karnataka','Kerala','Puducherry','Tamil Nadu','Andaman and Nicobar','Lakshadweep']: return 'South'
    elif state in ['Chhattisgarh','Madhya Pradesh','Uttarakhand','Uttar Pradesh']: return 'Central'
    elif state in ['Assam','Sikkim','Nagaland','Meghalaya','Manipur','Mizoram','Tripura','Arunachal Pradesh']: return 'North East'
    else: return None


print(df_c.columns)

     Index(['States', 'Period', '1901', '1902', '1903', '1904', '1905', '1906',
            '1907', '1908',
            ...
            '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019',
            '2020'],
           dtype='object', length=122)


#creating a copy of data
df_c_1=df_c.copy()

#Calculate basline mean relative to the period 1961-1990
df_c_1['Baseline mean']= df_c.iloc[:,62:92].mean(axis=1).round(2)
```

```
#print(df_c_1.iloc[:,122])
print(df_c_1.groupby(["States"]))
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000232E382FD50>
```

```
#Temp. Change relative to the baseline mean

df_c_1= df_c_1.melt(id_vars= ['States', 'Period', 'Baseline mean'], var_name= 'Year', value_name= 'temp_change')
print(df_c_1)
```

```
                      States        Period  Baseline mean  Year  temp_change
0      Andaman and Nicobar           Jan          25.81  1901        27.68
1      Andaman and Nicobar           Feb          26.06  1901        28.50
2      Andaman and Nicobar           Mar          26.91  1901        27.03
3      Andaman and Nicobar           Apr          28.10  1901        28.52
4      Andaman and Nicobar           May          27.69  1901        28.28
...                    ...           ...            ...   ...          ...
71395          West Bengal        Annual          25.56  2020        25.51
71396          West Bengal        Winter          19.67  2020        19.03
71397          West Bengal        Summer          28.16  2020        27.56
71398          West Bengal       Monsoon          28.56  2020        29.02
71399          West Bengal  Post Monsoon          22.90  2020        23.10

[71400 rows x 5 columns]
```

```
df_c_1['Year']= df_c_1['Year'].astype(int)
df_c_1['Zone']= df_c_1['States'].apply(zones)
df_c_1['temp_change']= df_c_1['temp_change']-df_c_1['Baseline mean']
df_c_1= df_c_1[['States', 'Zone', 'Period', 'Year', 'temp_change', 'Baseline mean']]
```

```
# Which ten states/UTs suffered the most from temperature change in the last ten years?

df_c_2= df_c_1[ (df_c_1.Period=='Annual') & (df_c_1.Year.between(2011,2020)) ][['States', 'temp_change']]
df_c_2= df_c_2.groupby('States').mean().round(2).nlargest(10, 'temp_change')
#temp = df_c_2.groupby('States')["temp_change"].mean().sort_values(ascending=False).head(10)
#print(df_c_2)
#print(temp)
df_c_2.reset_index(inplace=True)

# temp.plot(kind='bar', color = 'g')
# plt.ylabel('Change in Temps')
# plt.xlabel("States")
# plt.title('Top 10 affected States/UTs with Cimate Change')
```

```
# we went with plotly because of aesthetic purposes

fig= px.bar(df_c_2, x='States', y='temp_change', text_auto='.2f',
            title= 'Ten States/UTs which suffered the most from temperature change in the last ten years (2011-2020)',
            color='temp_change',
            color_continuous_scale= 'ylorrd'
           )

fig.update_layout(
    width=1000,
    height=600,
    template='plotly_dark',
    coloraxis_showscale=False,
    title_x= 0.5

)
fig.update_traces(textposition='outside')
fig.update_xaxes( title='States/UTs')
fig.update_yaxes( title='Temperature Change (°C)')

fig
```

```
# Which ten states/UTs suffered the least from temperature change in the last ten years?

df_c_2= df_c_1[ (df_c_1.Period=='Annual') & (df_c_1.Year.between(2011,2020)) ][['States', 'temp_change']]
df_c_2= df_c_2.groupby('States').mean().round(2).nsmallest(10, 'temp_change')
```

```
df_c_2.reset_index(inplace=True)

fig= px.bar(df_c_2, x='States', y='temp_change', text_auto='.2f',
            title= 'Ten States/UTs which suffered the least from temperature change in the last ten years (2011-2020)',
            color='temp_change',
            color_continuous_scale= 'ylgnbu_r'
           )

fig.update_layout(
    width=1000,
    height= 600,
    template='plotly_dark',
    coloraxis_showscale=False,
    title_x= 0.5

)
fig.update_traces(textposition='outside')
fig.update_yaxes( title='Temperature Change (ºC)')

fig
```

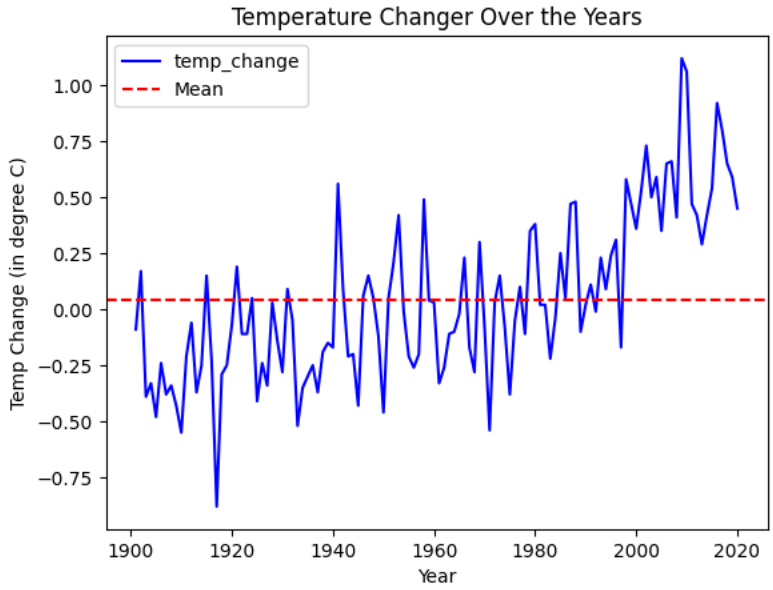
```
# Zone-wise temperature change in the last decade (2011-2020)

fig= px.bar(df_c_1[ (df_c_1.Year>2010) & (df_c_1.Period=='Annual')].groupby('Zone').mean().round(2).sort_values('temp_change',ascending=False
            text='temp_change',
            y='temp_change',
            title= 'Zone-wise temperature change in the last ten years (2011-2020)',
            color='temp_change',
            color_continuous_scale= 'teal'
           )

fig.update_layout(
    width=1000,
    height= 600,
    template='plotly_dark',
    coloraxis_showscale=False,
    title_x= 0.5

)
fig.update_traces(textposition='outside')
fig.update_yaxes( title='Temperature Change (ºC)')

fig
```

```
C:\Users\mayba\AppData\Local\Temp\ipykernel_4424\1036887146.py:3: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_
```

```
# Temperature change trend in India
fig= px.bar(df_c_1[(df_c_1.States=='India') & (df_c_1.Period=='Annual')],
            x= 'Year',
            y='temp_change',
            title= 'Temperature change in India (1901-2020)',
            color='temp_change',
            color_continuous_scale= 'ice_r',
            range_color=[-1,1.7]
           )

fig.update_layout(
    width=1000,
    height= 600,
    template='plotly_dark',
    coloraxis_showscale=False,
    title_x= 0.5
)

fig.update_xaxes( tickmode='linear',tick0=1901, dtick=10)
fig.update_yaxes( title='Temperature Change (ºC)')

fig
```

```
temp = df_c_1[(df_c_1.States=='India') & (df_c_1.Period=='Annual')]
temp.set_index('Year', inplace=True)
print(temp)
temp["temp_change"].plot(kind="line",color="b")
plt.ylabel('Temp Change (in degree C)')
plt.xlabel("Year")
plt.title('Temperature Changer Over the Years')
plt.axhline(temp["temp_change"].mean(), color='red', linestyle='--', label='Mean')
plt.legend()
plt.show()

# plt.figure(figsize=(15,5))
# temp.groupby(['year'])['annual'].sum().plot(kind='line', color = 'b')
# plt.ylabel('Yearly Rainfall')
# plt.xlabel("Year")
# plt.title('Yearly Rainfall (overall)')
# # Add a horizontal line for the mean annual rainfall
# plt.axhline(df_1.groupby(['year'])['annual'].sum().mean(), color='red', linestyle='--', label='Mean')
# plt.legend()
```

```
      States  Zone  Period  temp_change  Baseline mean
Year
1901  India  None  Annual        -0.09          24.35
1902  India  None  Annual         0.17          24.35
1903  India  None  Annual        -0.39          24.35
1904  India  None  Annual        -0.33          24.35
1905  India  None  Annual        -0.48          24.35
...     ...   ...     ...          ...            ...
2016  India  None  Annual         0.92          24.35
2017  India  None  Annual         0.80          24.35
2018  India  None  Annual         0.65          24.35
2019  India  None  Annual         0.59          24.35
2020  India  None  Annual         0.45          24.35

[120 rows x 5 columns]
```




```
temp1 = df_c_1[(df_c_1.States=='India') & (df_c_1.Period=='Annual')]
temp1
```

	States	Zone	Period	Year	temp_change	Baseline mean
250	India	None	Annual	1901	-0.09	24.35
845	India	None	Annual	1902	0.17	24.35
1440	India	None	Annual	1903	-0.39	24.35
2035	India	None	Annual	1904	-0.33	24.35

```
indian_states= json.load(open(path+ "/india_state.geojson",'r'))
x= df_c_1[ (df_c_1.Period =='Annual') & (df_c_1.Year.between(2011,2020)) ]
x= x.groupby('States').mean().round(2)
x.reset_index(inplace=True)

x.at[33,"States"] = "Uttaranchal"
#x
```

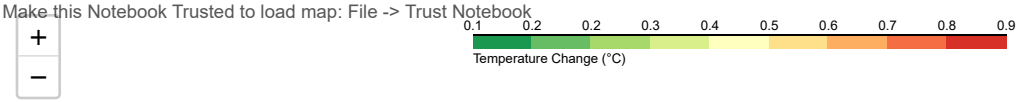
```
C:\Users\mayba\AppData\Local\Temp\ipykernel_4424\3093307714.py:3: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Eithe
```

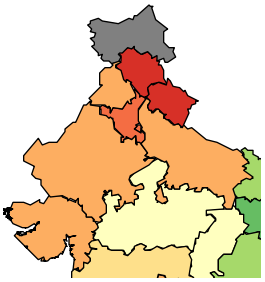
```
# following is the list which contain the list of the state names for the maps

state_lst = []

for feature in indian_states["features"]:
    state_lst.append(feature["properties"]["NAME_1"])

print(state_lst)
```

```
['Andaman and Nicobar', 'Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh', 'Dadra and Nagar Haveli',
```

```
fig = folium.Map( location=[21.9149, 78.0281], zoom_start=4, tiles=None)
folium.Choropleth(

    geo_data=indian_states,
    data=x,
    bins=9,
    highlight=True,
    columns=['States',"temp_change"],
    key_on="feature.properties.NAME_1",
    fill_color= 'RdYlGn_r',
    fill_opacity=1,
    legend_name="Temperature Change (°C)",
    nan_fill_color = "Grey"


).add_to(fig)

fig
```

Make this Notebook Trusted to load map: File -> Trust Notebook





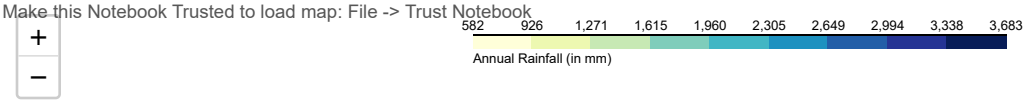
```
y_2 = df_2.loc[:,["State","Annual"]]
y_2 = y_2.groupby("State").mean().round(2)
y_2.reset_index(inplace=True)
#y_2
```

```
lst = list(x["States"])
lst.remove("India")
idx = lst.index("Himachal Pradesh")
lst.insert(idx+1,"Jammu and Kashmir")
#print(lst)
y_2['State'] = lst

y_2
```

	State	Annual
0	Andaman and Nicobar	2911.40
1	Andhra Pradesh	945.07
2	Arunachal Pradesh	2927.38
3	Assam	2454.36
4	Bihar	1200.56
5	Chandigarh	1070.60
6	Chhattisgarh	1286.35
7	Dadra and Nagar Haveli	2374.10
8	Daman and Diu	1535.70
9	Delhi	747.10
10	Goa	3278.50
11	Gujarat	924.34
12	Haryana	614.56
13	Himachal Pradesh	1371.59

```
fig = folium.Map( location=[21.9149, 78.0281], zoom_start=4, tiles=None)
folium.Choropleth(

    geo_data=indian_states,
    data=y_2,
    bins=9,
    highlight=True,
    columns=['State',"Annual"],
    key_on="feature.properties.NAME_1",
    fill_color= 'YlGnBu',
    fill_opacity=1,
    legend_name="Annual Rainfall (in mm)",
    nan_fill_color = "Grey"

).add_to(fig)

fig
```

Make this Notebook Trusted to load map: File -> Trust Notebook




```
df_carbon = pd.read_csv(path+"/CarbonEmissionIndia.csv")
lst1 = list(df_carbon["States"])
#print(lst1)
lst1[lst1.index("Arunachal")] = "Arunachal Pradesh"
lst1[lst1.index("Jammu & Kashmir")] = "Jammu and Kashmir"
lst1[lst1.index("Chattisgarh")] = "Chhattisgarh"
lst1[lst1.index("Odhisha")] = "Orissa"
lst1[lst1.index("Tamilnadu ")] = "Tamil Nadu"
lst1[lst1.index("Uttarakhand")] = "Uttaranchal"
#print(lst1)

df_carbon["States"] = lst1
df_carbon = df_carbon.rename(columns={"per capita CO2 (kg per person)":"CO2","per capita CO (kg per person)":"CO","per capita CH4 (kg per per
df_carbon
```

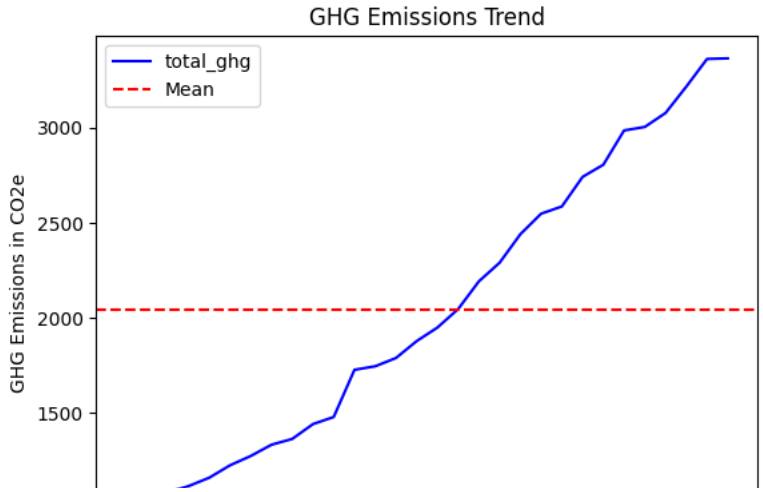
	States	CO2	CO	CH4	Population
0	Andhra Pradesh	974.17	27.18	16.97	84580777
1	Arunachal Pradesh	405.90	17.43	25.82	1383727
2	Assam	340.91	16.63	21.29	31205576
3	Bihar	179.01	8.83	9.59	104099452
4	Chhattisgarh	1963.88	17.56	22.37	25545198
5	Goa	2662.51	23.12	7.62	1458545
6	Gujarat	1310.58	24.01	12.26	60439692
7	Haryana	1381.86	17.90	21.57	25351462
8	Himachal Pradesh	784.16	16.98	18.28	6864602
9	Jammu and Kashmir	509.03	15.59	14.42	12541302
10	Jharkhand	1403.43	15.02	15.39	32988134
11	Karnataka	888.86	24.93	12.20	61095297
12	Kerala	780.12	18.29	4.52	33406061
13	Madhya Pradesh	656.37	16.14	15.15	72626809
14	Maharashtra	936.70	23.58	9.80	112374333
15	Manipur	379.20	10.80	22.63	2727749
16	Meghalaya	691.53	12.65	19.80	2966889
17	Mizoram	754.71	15.40	10.72	1097206
18	Nagaland	1275.27	24.13	29.08	1978502
19	Orissa	700.13	18.40	19.88	41974218
20	Punjab	1618.08	27.90	33.38	27743338
21	Rajasthan	793.69	14.33	14.18	68548473
22	Sikkim	711.39	11.68	10.04	610577
23	Tamil Nadu	985.70	26.60	10.40	72447030
24	Tripura	295.64	17.76	19.23	3673917
25	Uttar Pradesh	404.26	12.40	12.87	199812341
26	Uttaranchal	493.01	14.28	17.93	10086292
27	West Bengal	763.13	22.69	15.99	91276115

```
df_carbon["GHG per capita"] = df_carbon["CO2"] + 3.76*(df_carbon["CO"]) + 25*(df_carbon["CH4"])

df_carbon["GHG Emissions Overall"] = (df_carbon["GHG per capita"]*df_carbon["Population"])/1000   # divided by 1000 to convert it in tonnes

df_carbon
```

	States	CO2	CO	CH4	Population	GHG per capita	GHG Emissions Overall
0	Andhra Pradesh	974.17	27.18	16.97	84580777	1500.6168	1.269233e+08
1	Arunachal Pradesh	405.90	17.43	25.82	1383727	1116.9368	1.545536e+06
2	Assam	340.91	16.63	21.29	31205576	935.6888	2.919871e+07
3	Bihar	179.01	8.83	9.59	104099452	451.9608	4.704887e+07
4	Chhattisgarh	1963.88	17.56	22.37	25545198	2589.1556	6.614049e+07
5	Goa	2662.51	23.12	7.62	1458545	2939.9412	4.288037e+06
6	Gujarat	1310.58	24.01	12.26	60439692	1707.3576	1.031922e+08
7	Haryana	1381.86	17.90	21.57	25351462	1988.4140	5.040920e+07
8	Himachal Pradesh	784.16	16.98	18.28	6864602	1305.0048	8.958339e+06
9	Jammu and Kashmir	509.03	15.59	14.42	12541302	928.1484	1.164019e+07
10	Jharkhand	1403.43	15.02	15.39	32988134	1844.6552	6.085173e+07
11	Karnataka	888.86	24.93	12.20	61095297	1287.5968	7.866611e+07
12	Kerala	780.12	18.29	4.52	33406061	961.8904	3.213297e+07
13	Madhya Pradesh	656.37	16.14	15.15	72626809	1095.8064	7.958492e+07
14	Maharashtra	936.70	23.58	9.80	112374333	1270.3608	1.427559e+08
15	Manipur	379.20	10.80	22.63	2727749	985.5580	2.688355e+06
16	Meghalaya	691.53	12.65	19.80	2966889	1234.0940	3.661420e+06
17	Mizoram	754.71	15.40	10.72	1097206	1080.6140	1.185656e+06
18	Nagaland	1275.27	24.13	29.08	1978502	2092.9988	4.141002e+06
19	Orissa	700.13	18.40	19.88	41974218	1266.3140	5.315254e+07
20	Punjab	1618.08	27.90	33.38	27743338	2557.4840	7.095314e+07
21	Rajasthan	793.69	14.33	14.18	68548473	1202.0708	8.240012e+07
22	Sikkim	711.39	11.68	10.04	610577	1006.3068	6.144278e+05
23	Tamil Nadu	985.70	26.60	10.40	72447030	1345.7160	9.749313e+07
24	Tripura	295.64	17.76	19.23	3673917	843.1676	3.097728e+06
25	Uttar Pradesh	404.26	12.40	12.87	199812341	772.6340	1.543818e+08
26	Uttaranchal	493.01	14.28	17.93	10086292	994.9528	1.003538e+07
27	West Bengal	763.13	22.69	15.99	91276115	1248.1944	1.139303e+08

```
df_carbon_yearly = pd.read_csv(path+"/owid-co2-data.csv")

df_yearly_ind = df_carbon_yearly[(df_carbon_yearly.country=="India") & (df_carbon_yearly.year >= 1947)]

df_yearly_ind
df_yearly_ind.set_index("year",inplace=True)


df_yearly_ind["total_ghg"].plot(kind="line",color="b")
plt.ylabel('GHG Emissions in CO2e')
plt.xlabel("Year")
plt.title('GHG Emissions Trend')
# Add a horizontal line for the mean annual rainfall
plt.axhline(df_yearly_ind["total_ghg"].mean(), color='red', linestyle='--', label='Mean')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x232e3906010>
```



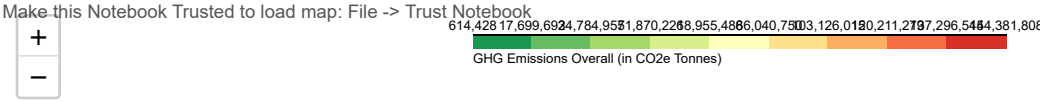

```
top_10 = df_carbon.nlargest(10,"GHG Emissions Overall")
top_10.reset_index(inplace=True)

#print(type(top_5))

fig= px.bar(top_10, x='States', y='GHG Emissions Overall', text_auto='.2f',
            title= 'Top 10 most GHG Emitting States',
            color='GHG Emissions Overall',
            color_continuous_scale= 'ylorrd'
           )

fig.update_layout(
    width=1000,
    height=600,
    template='plotly_dark',
    coloraxis_showscale=False,
    title_x= 0.5

)
fig.update_traces(textposition='outside')
fig.update_xaxes( title='States/UTs')
fig.update_yaxes( title='GHG Emissions in Tonnes of CO2e')

fig
```

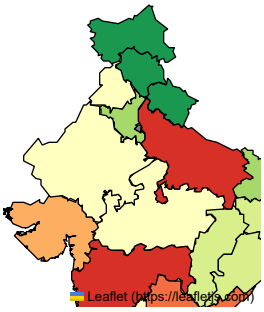
```
bot_10 = df_carbon.nsmallest(10,"GHG Emissions Overall")
bot_10.reset_index(inplace=True)

#print(type(top_5))

fig= px.bar(bot_10, x='States', y='GHG Emissions Overall', text_auto='.2f',
            title= 'Top 10 Least GHG Emitting States',
            color='GHG Emissions Overall',
            color_continuous_scale= 'ylgn_r'
           )

fig.update_layout(
    width=1000,
    height=600,
    template='plotly_dark',
    coloraxis_showscale=False,
    title_x= 0.5

)
fig.update_traces(textposition='outside')
fig.update_xaxes( title='States/UTs')
fig.update_yaxes( title='GHG Emissions in Tonnes of CO2e')
```

```
fig = folium.Map( location=[21.9149, 78.0281], zoom_start=4, tiles=None)
folium.Choropleth(
```

```
    geo_data=indian_states,
    data=df_carbon,
    bins=9,
    highlight=True,
    columns=['States',"GHG Emissions Overall"],
    key_on="feature.properties.NAME_1",
    fill_color= 'RdYlGn_r',
    fill_opacity=1,
    legend_name="GHG Emissions Overall (in CO2e Tonnes)",
    nan_fill_color = "Grey"

).add_to(fig)

fig

# this seems wrong because, UP is showing very less CO2 Emissions, which is not true, the data is wrong maybe
```

Make this Notebook Trusted to load map: File -> Trust Notebook



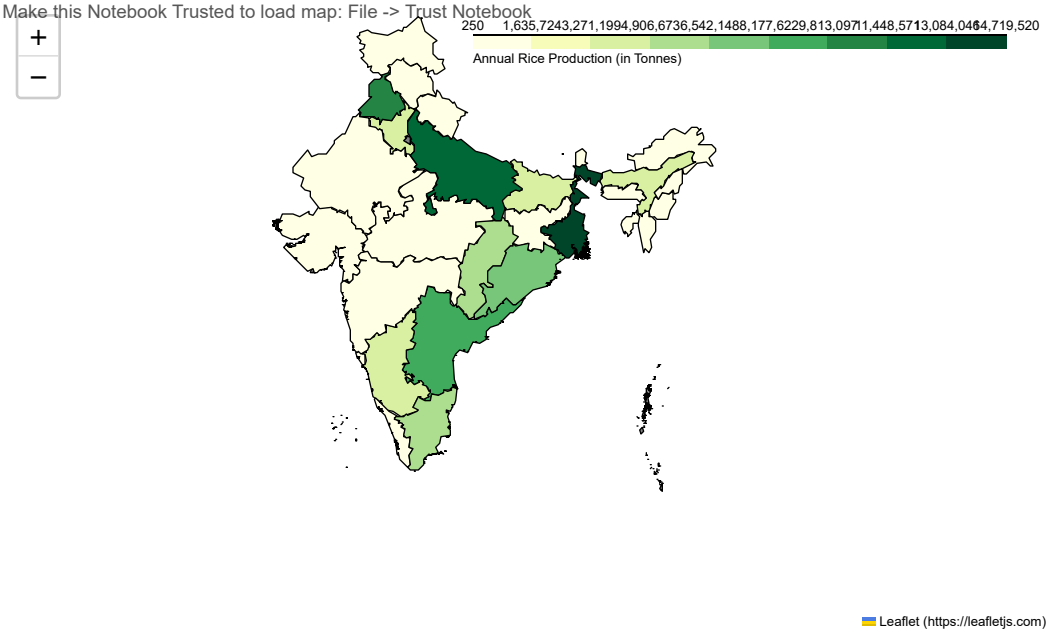



```
df_crop = pd.read_csv(path + "/crop_production.csv")
df_crop
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0

```
df_crop['State_Name'] = df_crop['State_Name'].str.upper()
#print(df_crop["State_Name"])
common_states = np.intersect1d(df_2["State"], df_crop["State_Name"])
#print(common_states)
# Filter dataframes to only include common states
df1_common = df_2[df_2["State"].isin(common_states)]
df2_common = df_crop[df_crop["State_Name"].isin(common_states)]

# Find common districts
common_districts = np.intersect1d(df1_common["DISTRICT"], df2_common["District_Name"])

# Filter dataframes to only include common districts
df1_common = df1_common[df1_common["DISTRICT"].isin(common_districts)]
df2_common = df2_common[df2_common["District_Name"].isin(common_districts)]

# Merge dataframes based on common states and districts
merged_df = pd.merge(df1_common, df2_common, how='inner', left_on=['State', 'DISTRICT'], right_on=['State_Name', 'District_Name'])



merged_df


crop_data_rice = df_crop[(df_crop.Crop=="Rice") & (df_crop.Crop_Year==2007)]
crop_data_rice = crop_data_rice.groupby("State_Name").sum()
crop_data_rice.reset_index(inplace=True)
crop_data_rice


crop_data_rice = crop_data_rice.drop([27])
lst = ['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh', 'Dadra and Nagar Haveli', 'Goa', 'Gujarat', 'Ha
crop_data_rice["State"] = lst

crop_data_rice


crop_data_rice.columns

     Index(['State_Name', 'Crop_Year', 'Area', 'Production', 'State'], dtype='object')


fig = folium.Map( location=[21.9149, 78.0281], zoom_start=4, tiles=None)
folium.Choropleth(

    geo_data=indian_states,
    data=crop_data_rice,
    bins=9,
    highlight=True,
    columns=['State',"Production"],
    key_on="feature.properties.NAME_1",
    fill_color= 'YlGn',
    fill_opacity=1,
    legend_name="Annual Rice Production (in Tonnes)",
    nan_fill_color = "grey"

).add_to(fig)

fig
```

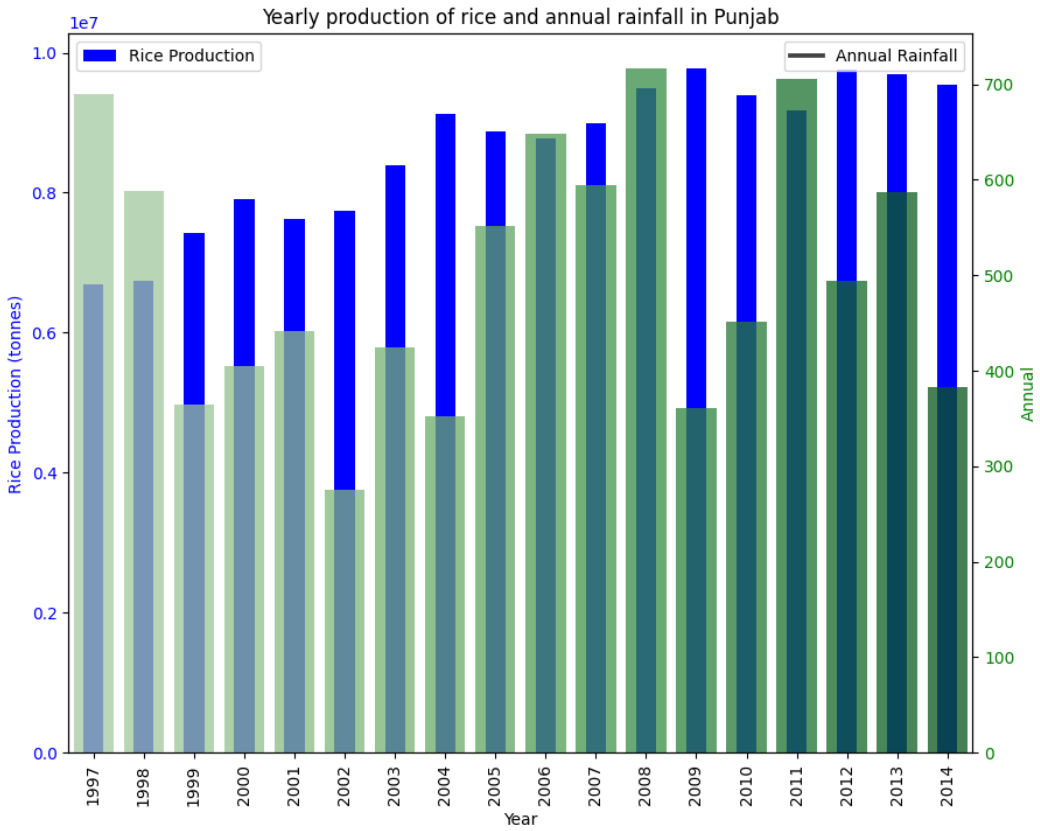
Make this Notebook Trusted to load map: File -> Trust Notebook




```
# Punjab and Rice data
punjab_rice_df = merged_df[(merged_df['State'] == 'PUNJAB') & (merged_df['Crop'] == 'Rice')]


punjab_rice_yearly = punjab_rice_df.groupby('Crop_Year')['Production'].sum()


punjab_rainfall_df = df_1[(df_1['State'] == 'PUNJAB') & (df_1['Year'] >= punjab_rice_yearly.index.min())]   # since we have less yearly data



punjab_rainfall_df.reset_index(inplace=True)
punjab_rainfall_df.drop(index=18,inplace=True)

punjab_rice_yearly = punjab_rice_yearly.reset_index().set_index('Crop_Year')['Production']


# Plot the yearly production and annual rainfall on the same plot
fig, ax1 = plt.subplots(figsize=(10, 8))

# Plot the yearly production on the left y-axis
punjab_rice_yearly.plot(kind='bar', ax=ax1, color='blue', width=0.4)
ax1.set_xlabel('Year')
ax1.set_ylabel('Rice Production (tonnes)', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Create a second y-axis on the right side
ax2 = ax1.twinx()

# Plot the annual rainfall on the right y-axis
sns.barplot(data=punjab_rainfall_df, x='Year', y='Annual', ax=ax2, color='green', alpha=0.5)
ax2.set_ylabel('Annual Rainfall (mm)', color='green')
ax2.tick_params(axis='y', labelcolor='green')

# Set the labels for the axes and legend
ax1.set_title('Yearly production of rice and annual rainfall in Punjab')
ax1.legend(['Rice Production'], loc='upper left')
ax2.legend(['Annual Rainfall'], loc='upper right')
```

```
sns.barplot(data=punjab_rainfall_df, x='Year', y='Annual', ax=ax2, palette='Greens', alpha=0.5)

# Display the graph
plt.show()
```

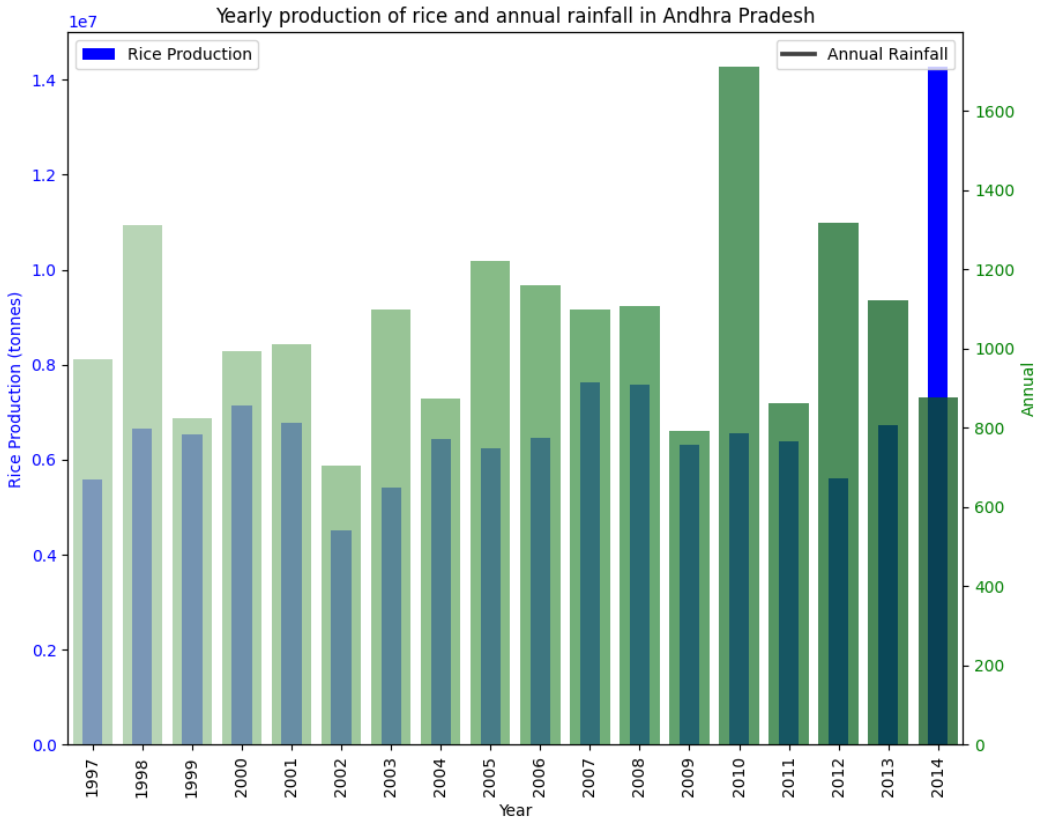
```
C:\Users\mayba\AppData\Local\Temp\ipykernel_4424\3323124918.py:13: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
```




```
# ap and Rice data
ap_rice_df = merged_df[(merged_df['State'] == 'ANDHRA PRADESH') & (merged_df['Crop'] == 'Rice')]


ap_rice_yearly = ap_rice_df.groupby('Crop_Year')['Production'].sum()


ap_rainfall_df = df_1[(df_1['State'] == 'COASTAL ANDHRA PRADESH') & (df_1['Year'] >= ap_rice_yearly.index.min())]   # since we have less year



ap_rainfall_df.reset_index(inplace=True)

ap_rainfall_df.drop(index=[18],inplace=True)

ap_rice_yearly = ap_rice_yearly.reset_index().set_index('Crop_Year')['Production']


# Plot the yearly production and annual rainfall on the same plot
fig, ax1 = plt.subplots(figsize=(10, 8))

# Plot the yearly production on the left y-axis
ap_rice_yearly.plot(kind='bar', ax=ax1, color='blue', width=0.4)
ax1.set_xlabel('Year')
ax1.set_ylabel('Rice Production (tonnes)', color='blue')
```

```
ax1.tick_params(axis='y', labelcolor='blue')

# Create a second y-axis on the right side
ax2 = ax1.twinx()

# Plot the annual rainfall on the right y-axis
sns.barplot(data=ap_rainfall_df, x='Year', y='Annual', ax=ax2, color='green', alpha=0.5)
ax2.set_ylabel('Annual Rainfall (mm)', color='green')
ax2.tick_params(axis='y', labelcolor='green')

# Set the labels for the axes and legend
ax1.set_title('Yearly production of rice and annual rainfall in Andhra Pradesh')
ax1.legend(['Rice Production'], loc='upper left')
ax2.legend(['Annual Rainfall'], loc='upper right')

sns.barplot(data=ap_rainfall_df, x='Year', y='Annual', ax=ax2, palette='Greens', alpha=0.5)

# Display the graph
plt.show()
```

```
C:\Users\mayba\AppData\Local\Temp\ipykernel_4424\3603561926.py:14: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
```




```
# Punjab and Rice data
wb_rice_df = merged_df[(merged_df['State'] == 'WEST BENGAL') & (merged_df['Crop'] == 'Rice')]
```

```
wb_rice_yearly = wb_rice_df.groupby('Crop_Year')['Production'].sum()
```

```
wb_rainfall_df = df_1[(df_1['State'] == 'GANGETIC WEST BENGAL') & (df_1['Year'] >= wb_rice_yearly.index.min())]   # since we have less yearly
```

```
wb_rainfall_df.reset_index(inplace=True)
wb_rainfall_df.drop(index=18,inplace=True)

wb_rice_yearly = wb_rice_yearly.reset_index().set_index('Crop_Year')['Production']


# Plot the yearly production and annual rainfall on the same plot
fig, ax1 = plt.subplots(figsize=(10, 8))

# Plot the yearly production on the left y-axis
wb_rice_yearly.plot(kind='bar', ax=ax1, color='blue', width=0.4)
ax1.set_xlabel('Year')
ax1.set_ylabel('Rice Production (tonnes)', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Create a second y-axis on the right side
ax2 = ax1.twinx()

# Plot the annual rainfall on the right y-axis
sns.barplot(data=wb_rainfall_df, x='Year', y='Annual', ax=ax2, color='green', alpha=0.5)
ax2.set_ylabel('Annual Rainfall (mm)', color='green')
ax2.tick_params(axis='y', labelcolor='green')

# Set the labels for the axes and legend
ax1.set_title('Yearly production of rice and annual rainfall in West Bengal')
ax1.legend(['Rice Production'], loc='upper left')
ax2.legend(['Annual Rainfall'], loc='upper right')

sns.barplot(data=wb_rainfall_df, x='Year', y='Annual', ax=ax2, palette='Greens', alpha=0.5)

# Display the graph
plt.show()
```

```
C:\Users\mayba\AppData\Local\Temp\ipykernel_4424\261457079.py:13: SettingWithCopyWarning:
```

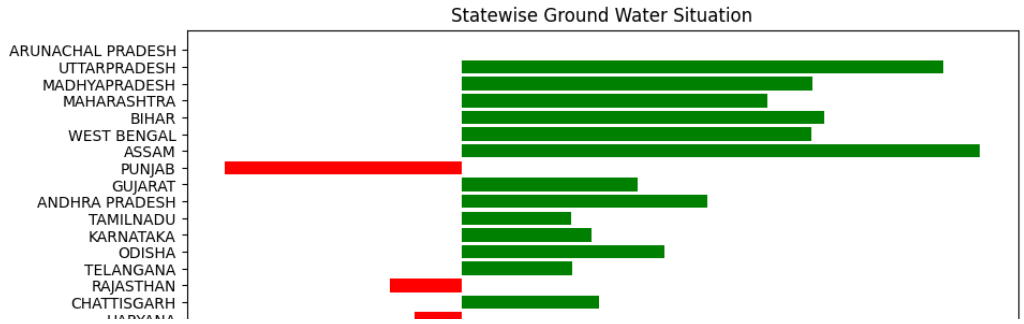
```
df_gw = pd.read_csv(path+"/ground_water.csv")
# df_gw.shape
df_gw.sample(10)
```

	S.no.	Name of State	Name of District	Recharge from rainfall During Monsoon Season	Recharge from other sources During Monsoon Season	Recharge from rainfall During Non Monsoon Season	Recharge from other sources During Non Monsoon Season	Total Annual Ground Water Recharge	Total Natural Discharges	E›
205	206	J&K	Budgam	4459.39	3521.72	5429.76	4716.56	18127.42	1812.74	
514	515	TAMILNADU	Madurai	20879.13	31368.31	4475.96	9738.96	66462.36	6646.24	
661	662	WEST BENGAL	North 24-Parganas	105327.29	8147.66	27207.49	25564.50	166246.93	16307.62	
423	424	ODISHA	Gajapati	12426.51	2564.56	4946.34	2536.35	22473.76	1681.69	
523	524	TAMILNADU	Thanjavur	32171.27	35411.26	5325.10	9286.96	82194.59	8219.46	
504	505	TAMILNADU	Chennai	1273.77	0.00	88.28	478.26	1840.30	184.03	
447	448	PUNJAB	Faridkot	12240.19	38379.86	2201.59	14398.61	67220.25	6722.03	
474	475	RAJASTHAN	Bundi	25607.90	3117.63	0.00	23490.04	52215.57	4842.20	
358	359	MAHARASHTRA	Nandurbar	39060.71	2624.03	0.00	8939.55	50624.30	3201.39	
341	342	MAHARASHTRA	Akola	32077.00	1388.27	300.60	8648.42	42414.28	2958.76	

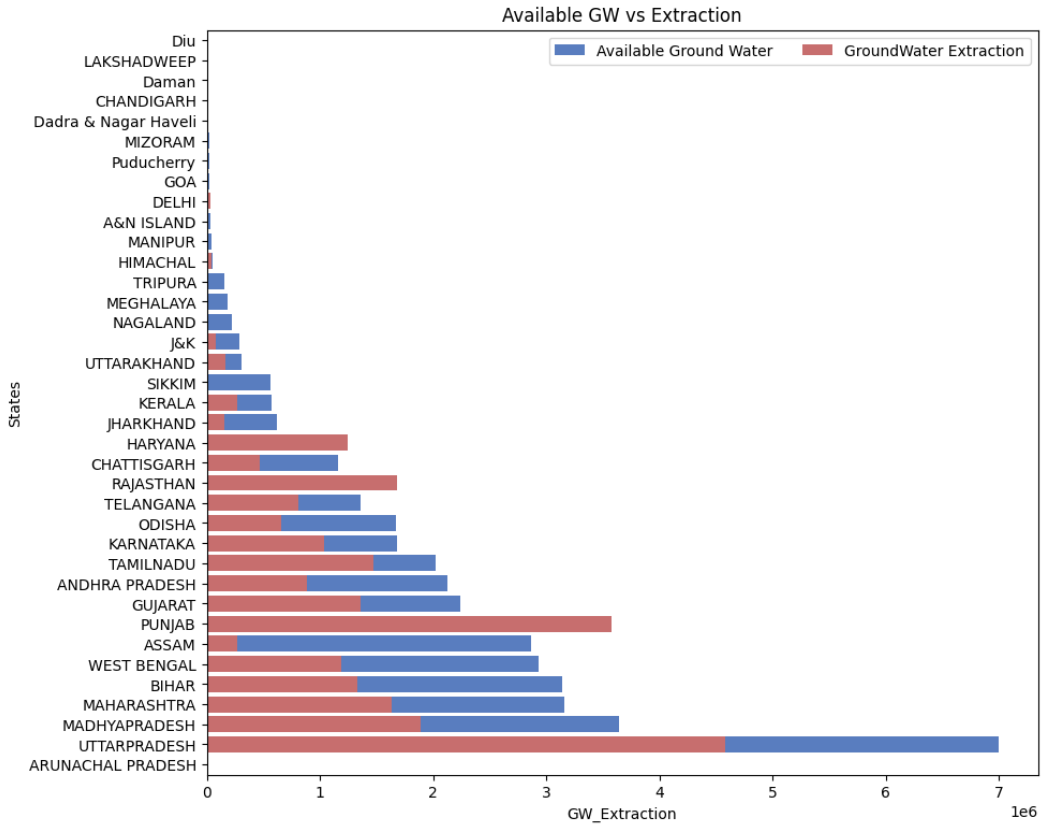

```
#statewise groundwater reserve
state_list = []
total_frnd_water_recharge = []
curr_gw_extr_list = []
future_available_GW_list = []
#Net Ground Water Availability for future use

for state, subset in df_gw.groupby('Name of State'):
    #print(state, sum(subset['Net Ground Water Availability for future use']))
    state_list.append(state)
    total_frnd_water_recharge.append(sum(subset['Total Annual Ground Water Recharge']))
    curr_gw_extr_list.append(sum(subset['Total Current Annual Ground Water Extraction']))
    future_available_GW_list.append(sum(subset['Net Ground Water Availability for future use']))

dfnew = pd.DataFrame({"State":state_list, "GW_Recharge":total_frnd_water_recharge, "GW_Extraction": curr_gw_extr_list, "Future_GW_Available":


dfnew.sort_values(['GW_Recharge','GW_Extraction'], inplace= True)
dfnew['annual_reserve'] = dfnew['GW_Recharge']-dfnew['GW_Extraction']
dfnew
```

	State	GW_Recharge	GW_Extraction	Future_GW_Available	annual_reserve
10	Diu	471.00	386.00	6.00	85.00
19	LAKSHADWEEP	1072.80	238.10	122.81	834.70
9	Daman	1304.00	635.00	480.00	669.00
5	CHANDIGARH	4216.00	3378.00	416.00	838.00
8	Dadra & Nagar Haveli	6861.59	2042.86	4475.73	4818.73
24	MIZORAM	21280.76	731.88	18092.91	20548.88
28	Puducherry	22633.18	15140.82	5473.84	7492.36
11	GOA	26722.34	5371.36	6992.81	21350.98
7	DELHI	32100.02	35990.29	2299.43	-3890.27
0	A&N ISLAND	36841.83	908.17	32131.52	35933.66
22	MANIPUR	42991.50	556.86	34290.53	42434.64
14	HIMACHAL	50576.83	39311.93	16444.25	11264.90
33	TRIPURA	152568.37	9759.24	111249.36	142809.13
23	MEGHALAYA	183132.59	3739.54	159096.77	179393.05
25	NAGALAND	220443.18	1960.12	196308.27	218483.06
15	J&K	289008.60	75674.34	184433.39	213334.26
34	UTTARAKHAND	304495.92	164382.46	124890.45	140113.46
30	SIKKIM	563216.75	87.40	150728.28	563129.35
18	KERALA	576923.21	267064.39	241449.10	309858.82
16	JHARKHAND	621342.87	157773.29	412800.47	463569.58
13	HARYANA	1014500.05	1250038.64	87367.91	-235538.59
6	CHATTISGARH	1157241.10	469527.41	576373.37	687713.69
29	RAJASTHAN	1320537.64	1677050.87	88012.78	-356513.23
32	TELANGANA	1361897.09	809396.05	426060.85	552501.04
26	ODISHA	1673807.67	656766.50	884649.93	1017041.17
17	KARNATAKA	1683850.45	1033652.36	541171.47	650198.09

```
f, ax = plt.subplots(figsize=(10,8))
plt.barh(dfnew['State'],dfnew['annual_reserve'], color=(dfnew['annual_reserve']>0).map({True: 'g',False: 'r'}))
plt.title("Statewise Ground Water Situation")
plt.ylabel("States")
plt.xlabel("Recharge - Extraction")
plt.show()
```




```
f, ax = plt.subplots(figsize=(10, 9))

sns.set_color_codes("muted")
sns.barplot(x='GW_Recharge', y= 'State', data = dfnew, label = 'Available Ground Water', color='b')
sns.barplot(x='GW_Extraction', y= 'State', data = dfnew, label = 'GroundWater Extraction', color='r')
plt.title("Available GW vs Extraction")
plt.ylabel("States")

ax.legend(ncol=2, loc="upper right", frameon=True)
plt.show()
```




```
dfnew.sort_values('Future_GW_Available', inplace = True)
f, ax = plt.subplots(figsize=(10, 9))
sns.barplot(x='Future_GW_Available', y = 'State',data = dfnew)
plt.title("Ground Water Potential for Future Generations")
plt.ylabel("States")
plt.xlabel("GW Availability for the Future")

plt.show()
```