GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY
UNIVERSITY

CSE2015

DATA ANALYSIS AND VISUALIZATION

REPORT

AIR POLLUTION

SUBMITTED TO:

MS. POORNIMA S
ASST.PROF
DEPARTMENT OF CSE

SUBMITTED BY:

E RAHUL -20211ISD0004
T ABHINAV KISHAN -20211ISD0008
T YESWANTH TEJA -20211ISD0041
SECTION: 6ISD-01

ABSTRACT

Air pollution is a critical environmental issue impacting public health and ecosystems globally. Effective visualization of air pollution data can enhance understanding and support decision- making processes. This report explores the conceptual framework, objectives, scope, and methodologies for visualizing air pollution data. It includes an examination of identified data sources and a presentation of various visualization techniques to convey air quality information effectively.

1. INTRODUCTION

1.1 CONCEPTUAL STUDY OF THE PROJECT:

The conceptual study involves understanding the nature of air pollution data, the types of pollutants, and the significance of visualizing this information. Air pollution data typically includes concentrations of various pollutants such as particulate matter (PM2.5 and PM10), nitrogen dioxide (NO_2), sulfur dioxide (SO_2), carbon monoxide (CO), and ozone (O_3). Visualization techniques aim to translate this data into accessible formats, such as maps, charts, and interactive dashboards, facilitating better comprehension and analysis.

1.2 OBJECTIVES OF THE PROJECT:

Enhance Understanding: Improve public and stakeholder understanding of air pollution levels and trends.
Support Decision-Making: Provide valuable insights to policymakers and environmental agencies for informed decision-making.
Raise Awareness: Increase public awareness about the sources and impacts of air pollution.
Promote Data Transparency: Ensure accessibility and transparency of air pollution data.

1.3. SCOPE OF THE PROJECT:

Geographical Coverage: Global, regional, and local air quality data visualization.
Temporal Coverage: Historical data analysis and real-time monitoring.
Pollutant Types: Visualization of key pollutants affecting air quality.
Audience: Targeted at policymakers, researchers, environmentalists, and the general public.
Outputs: Interactive dashboards, static maps, time series graphs, and infographics.

2. ABOUT DATASET:

2.1 DATA IDENTIFIED FROM:

Research Institutions: Data from academic studies and research projects

2.2 DETAILS ABOUT THE ATTRIBUTES IN DATASET :

0. Date (DD/MM/YYYY)

1. Time (HH.MM.SS)

2. CO_GT - True hourly averaged concentration CO in mg/m^3

3. PT08_S1_CO - PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)

4. C6H6_GT- True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)

5. True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)

6. PT08_S2_NMHC- PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)

7. Nox_GT- True hourly averaged Nitric oxide concentration in ppb (reference analyzer)

8. PT08_S3_Nox- PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)

9. NO2_GT- True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)

10. PT08_S4_NO2- PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO2 targeted)

11. PT08_S5_O3 - PT08.S5 (indium oxide) hourly averaged sensor response (nominally O3 targeted)

12. T - Temperature in Â°C

13. RH- Relative Humidity (%)

14. AH - Absolute Humidity

3. BASIC DATA EXPLORATION:

- df.head()
- df.info()
- df.describe()

4. VARIOUS ANALYSIS PERFORMED:

- Checking for null values.
- Handling missing values.
- Checking for outliers.
- Handlin Outliers that are present.

Air Quality Dataset Analysis

Loading and understanding the dataset

In [2]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

In [3]:
```
#Load the dataset
df = pd.read_csv('AIR_QUALITY.csv')
```

In [4]:
```
df = pd.read_csv('AIR_QUALITY.csv', parse_dates={'datetime': ['Date', '
```

In [5]:
```
df.head()
```

Out[5]:

	datetime	CO_GT	PT08_S1_CO	NMHC_GT	C6H6_GT	PT08_S2_NMHC	Nox_GT	PT0
0	2004-11-23 19:00:00	11.9	2008	-200	50.6	1980	1389	
1	2004-11-23 20:00:00	11.5	1918	-200	49.4	1958	1358	
2	2004-11-17 18:00:00	10.2	1802	-200	47.7	1924	748	
3	2004-11-23 18:00:00	10.2	1982	-200	49.5	1959	1369	
4	2004-11-26 18:00:00	10.1	1956	-200	45.2	1877	1389	

In [6]:
```
df.tail(10)
```

Out[6]:

	datetime	CO_GT	PT08_S1_CO	NMHC_GT	C6H6_GT	PT08_S2_NMHC	Nox_GT
9347	2005-03-13 07:00:00	-200.0	944	-200	1.6	551	98
9348	2005-03-13 08:00:00	-200.0	970	-200	2.1	592	190
9349	2005-03-14 04:00:00	-200.0	1036	-200	2.8	636	122
9350	2005-03-17 04:00:00	-200.0	959	-200	1.9	578	100
9351	2005-03-20 04:00:00	-200.0	993	-200	2.8	640	85
9352	2005-03-23 04:00:00	-200.0	993	-200	2.3	604	85
9353	2005-03-26 04:00:00	-200.0	1122	-200	6.0	811	181
9354	2005-03-29 04:00:00	-200.0	883	-200	1.3	530	63
9355	2005-04-01 04:00:00	-200.0	818	-200	0.8	473	47
9356	2005-04-04 04:00:00	-200.0	864	-200	0.8	478	52

In [7]:
```
df.describe()
```

Out[7]:

	CO_GT	PT08_S1_CO	NMHC_GT	C6H6_GT	PT08_S2_NMHC	Nox_G
count	9357.000000	9357.000000	9357.000000	9357.000000	9357.000000	9357.0000
mean	-34.207524	1048.990061	-159.090093	1.865683	894.595276	168.6169
std	77.657170	329.832710	139.789093	41.380206	342.333252	257.4338
min	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.0000
25%	0.600000	921.000000	-200.000000	4.000000	711.000000	50.0000
50%	1.500000	1053.000000	-200.000000	7.900000	895.000000	141.0000
75%	2.600000	1221.000000	-200.000000	13.600000	1105.000000	284.0000
max	11.900000	2040.000000	1189.000000	63.700000	2214.000000	1479.0000

In [8]:
```
df.info(
)
```

```
                     Non-Null Count          Dtype
---  ---------       ----------------------  --------
 0   datetime        9357 non-null           datetime64[ns]
 1   CO_GT           9357 non-null           float64
 2   PT08_S1_C       9357 non-null           int64
 3   O NMHC_GT       9357 non-null           int64
 4   C6H6_GT         9357 non-null           float64
 5   PT08_S2_NMH     9357 non-null           9357 non-null   int64
 6   Nox_GT          9357 non-null           int64
 7   PT08_S3_Nox     9357 non-                int64
 8   null                                    int64
 9   NO2_GT          9357 non-null           int64
 10  PT08_S5_O       9357 non-null           int64
 11  3 T             9357 non-null           float64
 12  RH              9357 non-null           float64
 13  AH              9357 non-null           float64
 14  CO_level        9357 non-null           object
                     9357 non-null
```

In [9]:
```
df.shape
```

Out[9]:
```
(9357, 15)
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['datetime', 'CO_GT', 'PT08_S1_CO', 'NMHC_GT', 'C6H6_GT', 'PT08_
S2_NMHC',
       'Nox_GT', 'PT08_S3_Nox', 'NO2_GT', 'PT08_S4_NO2', 'PT08_S5_O3',
'T',
       'RH', 'AH', 'CO_level'],
      dtype='object')
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
datetime        0
CO_GT           0
PT08_S1_CO      0
NMHC_GT         0
C6H6_GT         0
PT08_S2_NMHC    0
Nox_GT          0
PT08_S3_Nox     0
NO2_GT          0
PT08_S4_NO2     0
PT08_S5_O3      0
T               0
RH              0
```

```
AH          0
CO_level    0
dtype: int64
```

Cleaning the dataset

In [12]:
```
df['NMHC_GT'].value_counts()
```

Out[12]:
```
-200     8443
 66        14
 29         9
 40         9
 93         8
          ...
 206        1
 268        1
 320        1
 270        1
 10         1
Name: NMHC_GT, Length: 430, dtype: int64
```

In [13]:
```
#[illegible]
df.drop('NMHC_GT', axis=1, inplace=True)
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['datetime', 'CO_GT', 'PT08_S1_CO', 'C6H6_GT', 'PT08_S2_NMHC', 'Nox_GT',
       'PT08_S3_Nox', 'NO2_GT', 'PT08_S4_NO2', 'PT08_S5_O3', 'T', 'R H', 'AH',
       'CO_level'], dtype='object')
```

In [15]:
```
#[illegible]
df.replace(to_replace= -200, value= np.NaN, inplace= True)
```

In [16]:
```
#[illegible]
col_list = df.columns[2:13]

for i in col_list:
    df[i] = df[i].fillna(df[i].mean())
```

In [17]:
```
df.describe()
```

Out[17]:

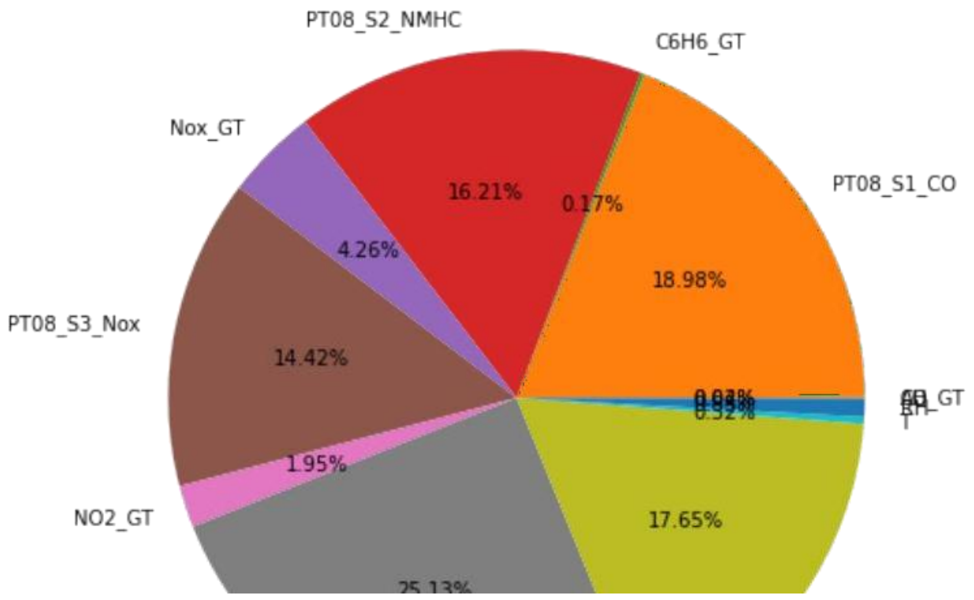
	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	Nox_GT	PT08_S3_
count	7674.000000	9357.000000	9357.000000	9357.000000	9357.000000	9357.00
mean	2.152750	1099.833166	10.083105	939.153376	246.896735	835.49
std	1.453252	212.791672	7.302650	261.560236	193.426632	251.74
min	0.100000	647.000000	0.100000	383.000000	2.000000	322.00
25%	1.100000	941.000000	4.600000	743.000000	112.000000	666.00
50%	1.800000	1075.000000	8.600000	923.000000	229.000000	818.00
75%	2.900000	1221.000000	13.600000	1105.000000	284.000000	960.00
max	11.900000	2040.000000	63.700000	2214.000000	1479.000000	2683.00

Performing EDA on the dataset Step 5: Perform an in-depth extensive EDA (Exploratory Data Analysis) on the dataset. The analysis should contain at least 5 different types of graphs/charts.

Finding co-realtion between different gases

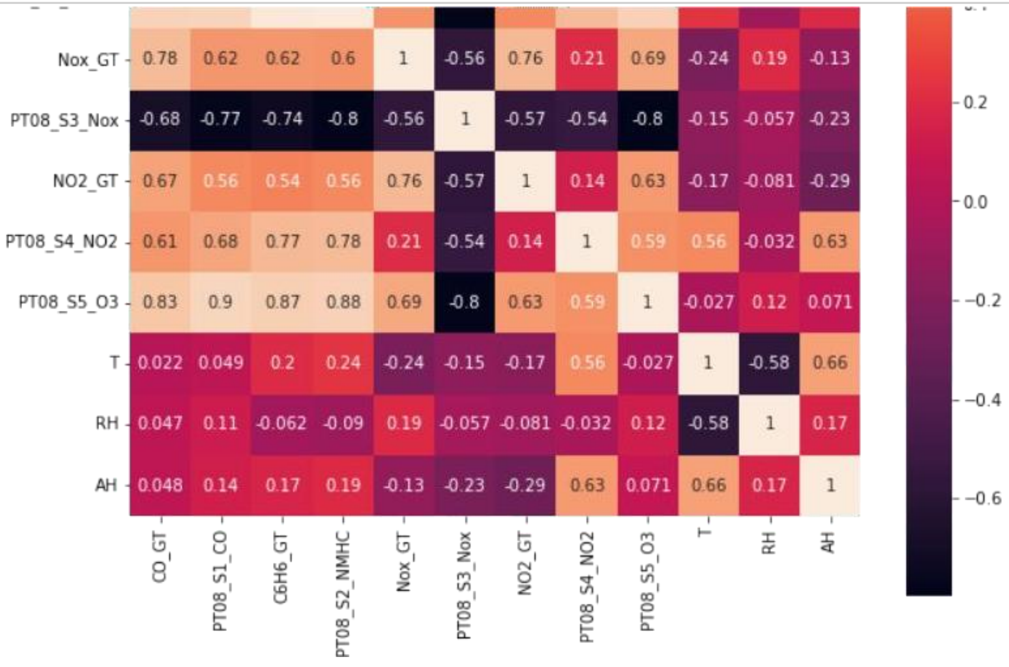
In [18]:
```
df.mean().plot.pie(ylabel='',radius = 2,autopct =
```

```
reduction.
  df.mean().plot.pie(ylabel='',radius = 2,autopct = "%.2f%%");
```



In [19]:

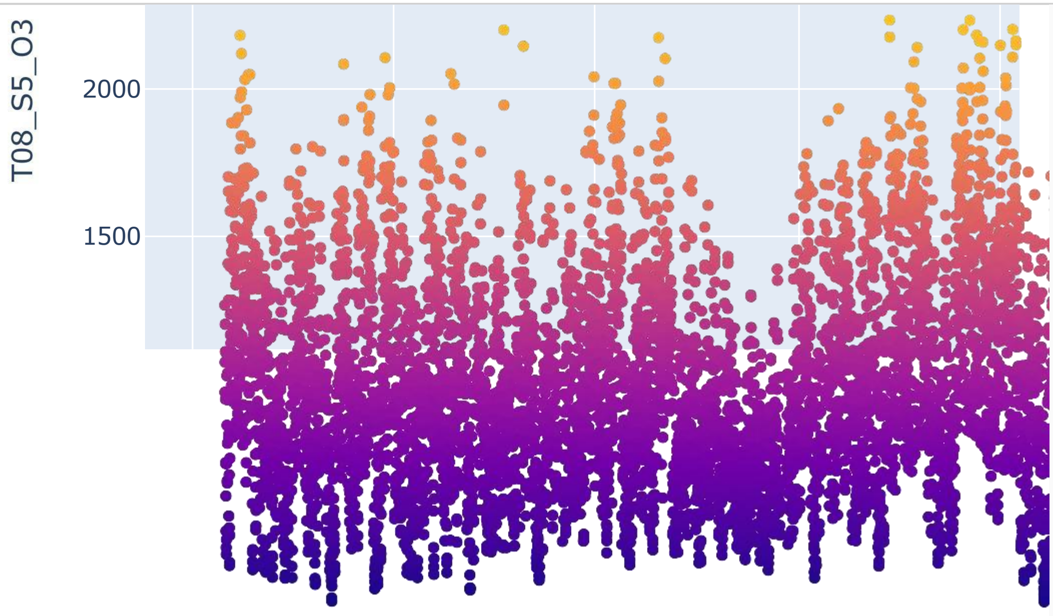
```
correlation = df.corr() fig_corr=
plt.gcf();
fig_corr.set_size_inches(10,10);
sns.heatmap(correlation,annot = True, square = True)

plt.show()
```



Finding concentration of air pollutants

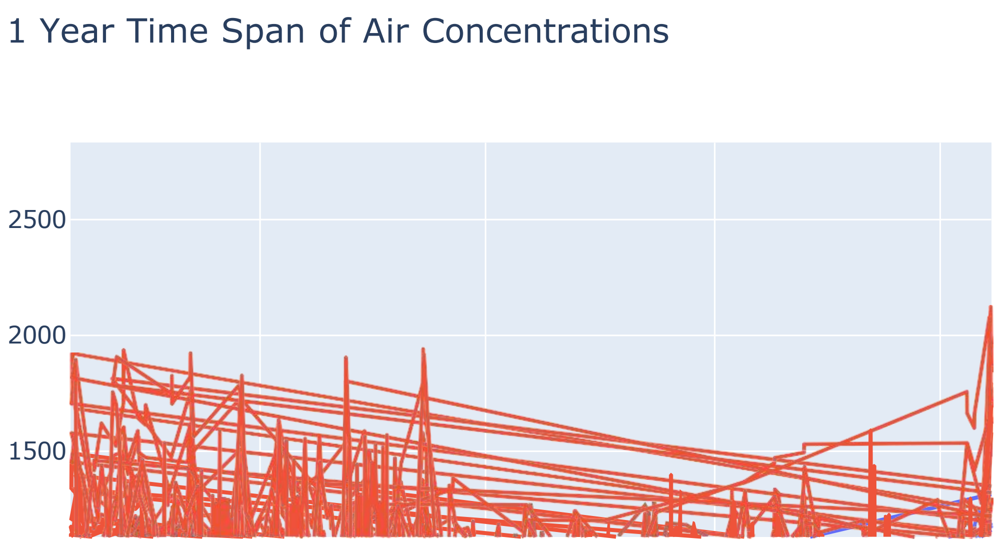
In [20]:

```
#[illegible]
for i in df.columns[2:15]:
    fig =px.scatter(df,x="datetime",y=i,color =i)
    fig.show()
```



In [21]:

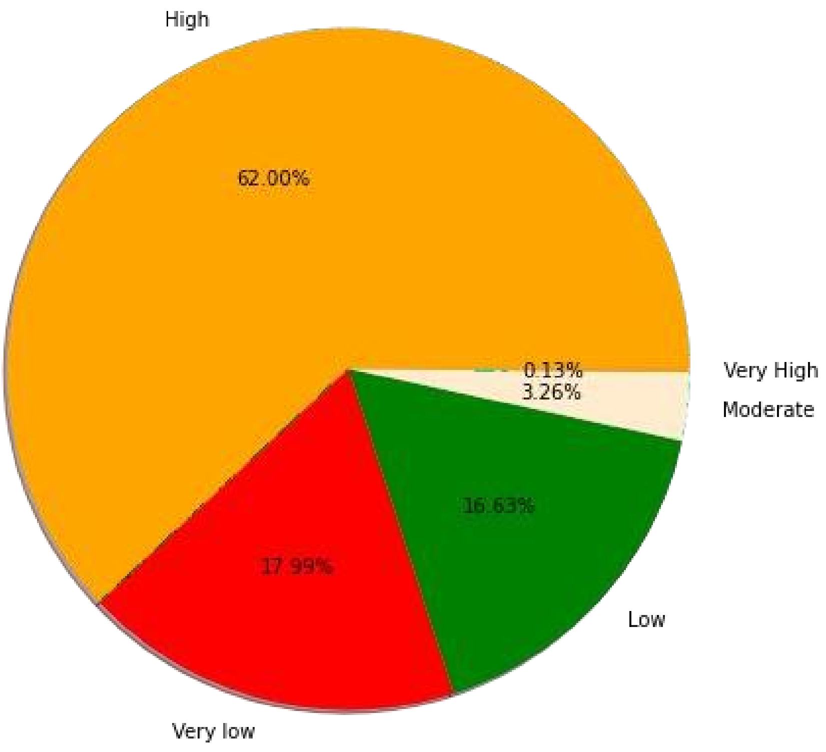
```
columns = ["Nox_GT", "PT08_S3_Nox", "NO2_GT"]
oneday = df[6:28].copy()

fig =go.Figure([{'x': df["datetime"],'y': df[col],'name': fig.show()        for  col
col}
```



Extracting features of Air depending upon the level of CO concentration.

In [22]:
```
df['CO_level'].unique()
CO_level_count = df['CO_level'].value_counts() print(CO_level_count)
plt.pie(CO_level_count, labels =
CO_level_count.index,colors=['orange',
```

```
High          5801
Very low      1683
Low           1556
Moderate       305
Very High       12
Name: CO_level, dtype: int64
```

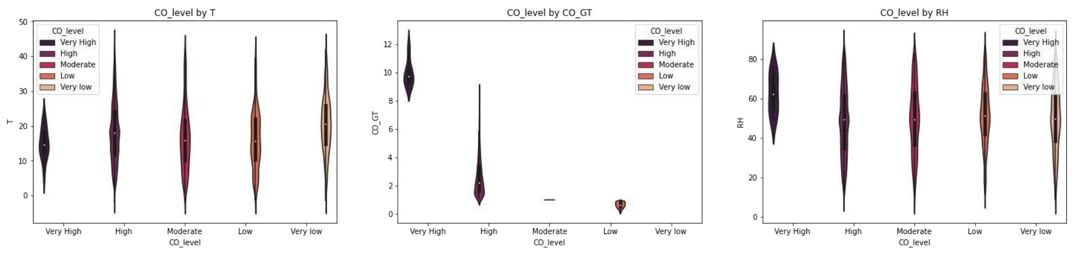


In [23]:

```
#[illegible]
fig, axes = plt.subplots(1,3, figsize=(25, 5))
sns.violinplot(x='CO_level', y='T', data=df, hue="CO_level", palette='r
axes[0].set_title("{} by {}".format("CO_level", "T"))



sns.violinplot(x="CO_level", y="CO_GT", data=df, hue="CO_level",
palett axes[1].set_title("{} by {}".format("CO_level", "CO_GT"))



sns.violinplot(x="CO_level", y="RH", data=df, hue="CO_level",
```

Out[23]:

```
Text(0.5, 1.0, 'CO_level by RH')
```



In [24]:

```
Very_High = df[df['CO_level'] == 'Very High']
Very_High.describe()
```

Out[24]:

	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	Nox_GT	PT08_S3_Nox
count	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000
mean	9.950000	1745.805528	41.155518	1753.692229	1209.166667	425.582267
std	0.918992	314.590744	15.107218	389.540394	233.443329	192.637656
min	9.100000	1099.833166	10.083105	939.153376	748.000000	322.000000
25%	9.275000	1758.750000	42.000000	1811.250000	1126.000000	331.750000
50%	9.700000	1848.500000	47.950000	1929.500000	1281.500000	344.000000
75%	10.200000	1927.500000	49.775000	1964.250000	1374.000000	367.500000
max	11.900000	2008.000000	52.100000	2007.000000	1479.000000	835.493605

In [25]:
```
High = df[df['CO_level'] == 'High'] High.describe()
```

Out[25]:

	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	Nox_GT	PT08_S3_
count	5801.000000	5801.000000	5801.000000	5801.000000	5801.000000	5801.00
mean	2.608033	1177.244318	12.390772	1036.044810	299.184801	743.54
std	1.322318	193.256641	6.845445	222.888739	212.652187	185.02
min	1.100000	667.000000	1.600000	554.000000	16.000000	328.00
25%	1.600000	1037.000000	7.500000	877.000000	146.000000	618.00
50%	2.200000	1132.000000	10.500000	995.000000	244.000000	741.00
75%	3.300000	1295.000000	15.800000	1174.000000	387.000000	842.00
max	8.700000	2040.000000	63.700000	2214.000000	1345.000000	2542.00

In [26]:
```
Moderate = df[df['CO_level'] == 'Moderate']
Moderate.describe()
```

Out[26]:

	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	Nox_GT	PT08_S3_Nox	
count	305.0	305.000000	305.000000	305.000000	305.000000	305.000000	3
mean	1.0	953.249170	4.675396	734.750035	125.305043	965.373393	
std	0.0	91.573017	2.186755	100.019418	76.809318	159.886460	
min	1.0	771.000000	1.600000	550.000000	23.000000	527.000000	
25%	1.0	892.000000	3.100000	659.000000	61.000000	852.000000	
50%	1.0	935.000000	4.100000	718.000000	116.000000	944.000000	
75%	1.0	1002.000000	5.600000	791.000000	162.000000	1045.000000	1
max	1.0	1432.000000	15.000000	1150.000000	547.000000	1678.000000	2

In [27]:
```
Low = df[df['CO_level'] == 'Low'] Low.describe()
```

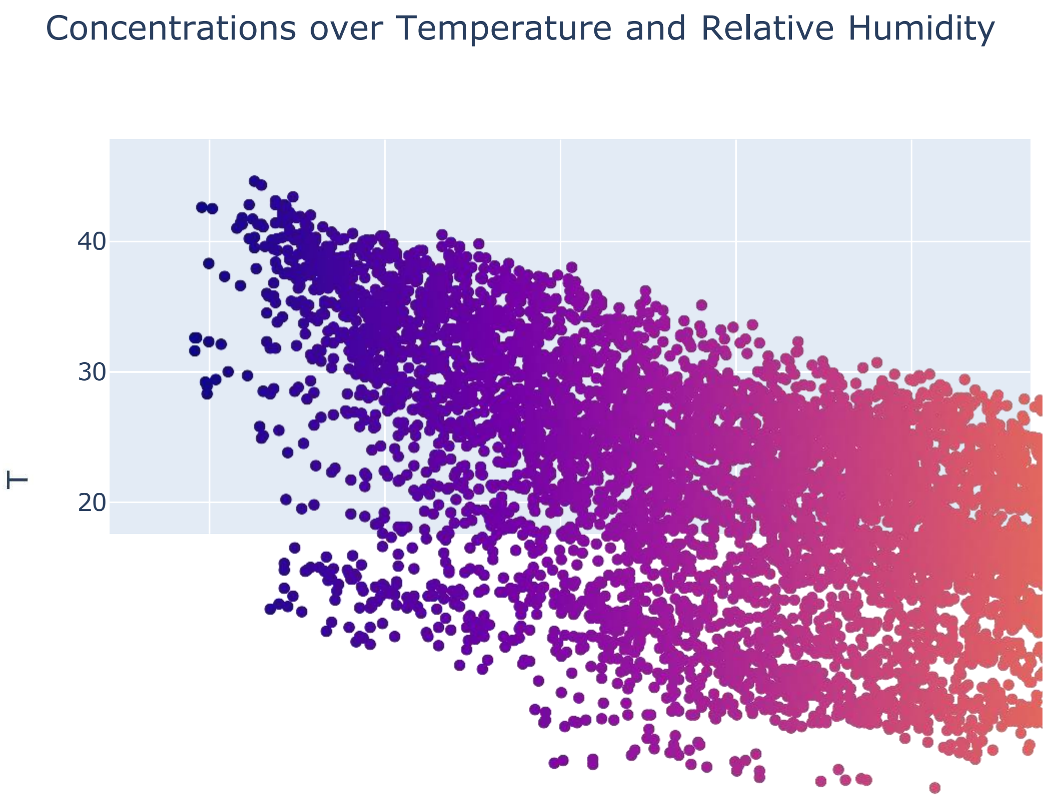
Out[27]:

	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	Nox_GT	PT08_S3_
count	1556.000000	1556.000000	1556.000000	1556.000000	1556.000000	1556.000
mean	0.621208	885.710151	3.209314	649.681620	107.649924	1115.528
std	0.207729	95.159206	1.993717	112.201268	83.603681	252.646
min	0.100000	647.000000	0.200000	387.000000	2.000000	522.000
25%	0.500000	825.000000	1.900000	576.000000	43.000000	944.000
50%	0.600000	874.000000	2.800000	640.000000	76.000000	1077.500
75%	0.800000	935.250000	4.100000	715.000000	156.000000	1237.000
max	0.900000	1390.000000	15.600000	1167.000000	588.000000	2683.000

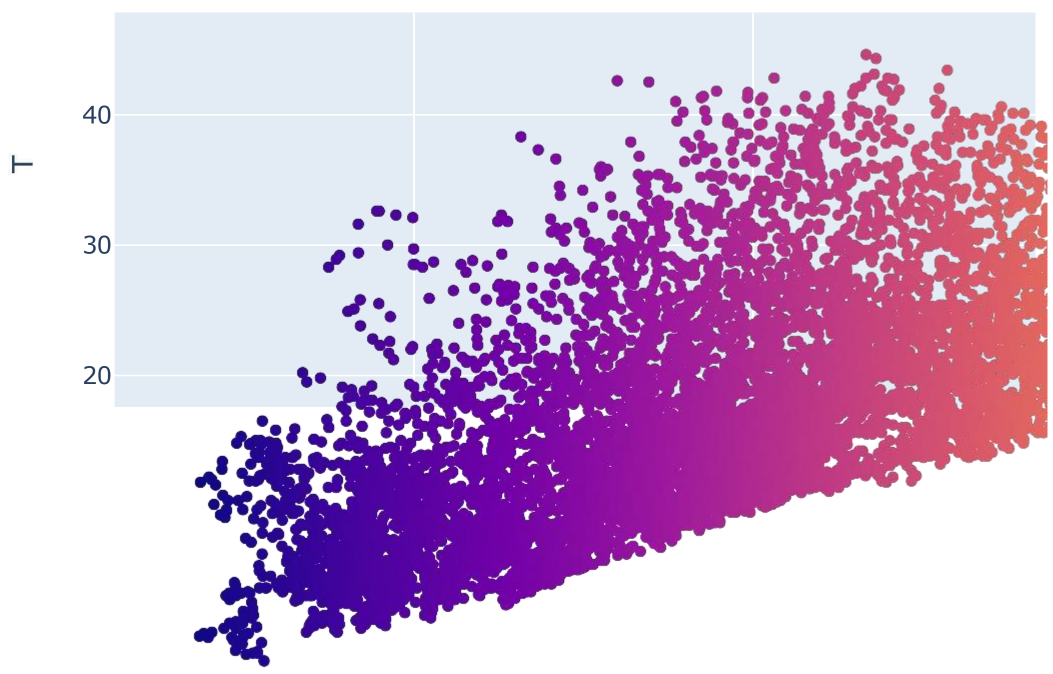
Realtion between relative humidity/ absolute humidity and temerature and it's concentration.

In [28]:

```
fig1 = px.scatter(df,x="RH",y="T",color ='RH',title = "Concentrations o fig2
= px.scatter(df,x="AH",y="T",color ='AH',title = "Concentrations o fig1.show()
fig2.show()
```

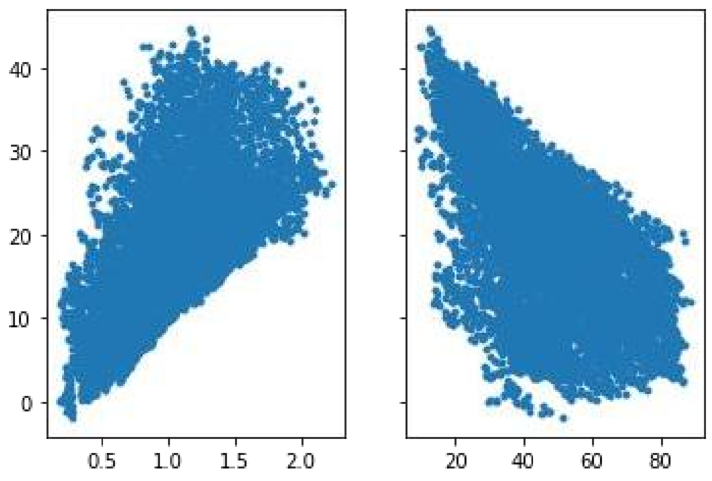


Concentrations over Temperature and Absolute Humidity



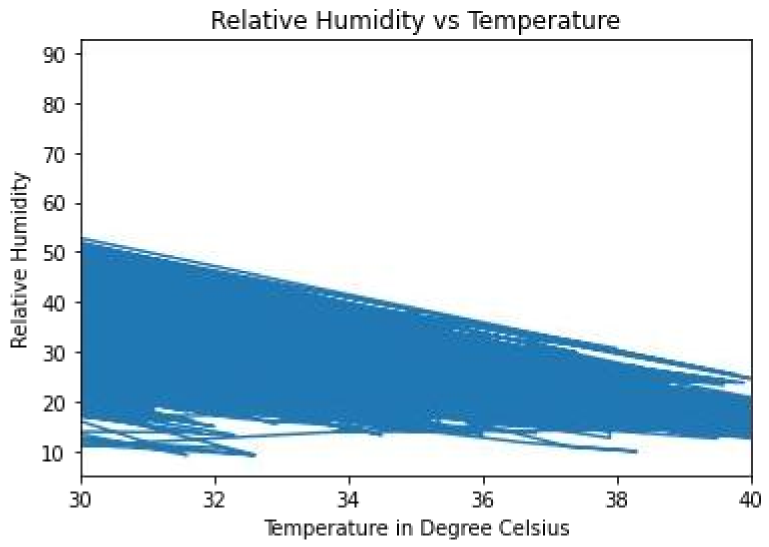
In [29]:

```
#[illegible]
fig,(ax1,ax2) = plt.subplots(1,2,sharey =True)
ax1.scatter(df['AH'],df["T"],marker=".")
ax2.scatter(df['RH'], df['T'],marker=".")  plt.show()
```


40
30
20
10
0
0.5
1.0
1.5
2.0
20
40
60
80

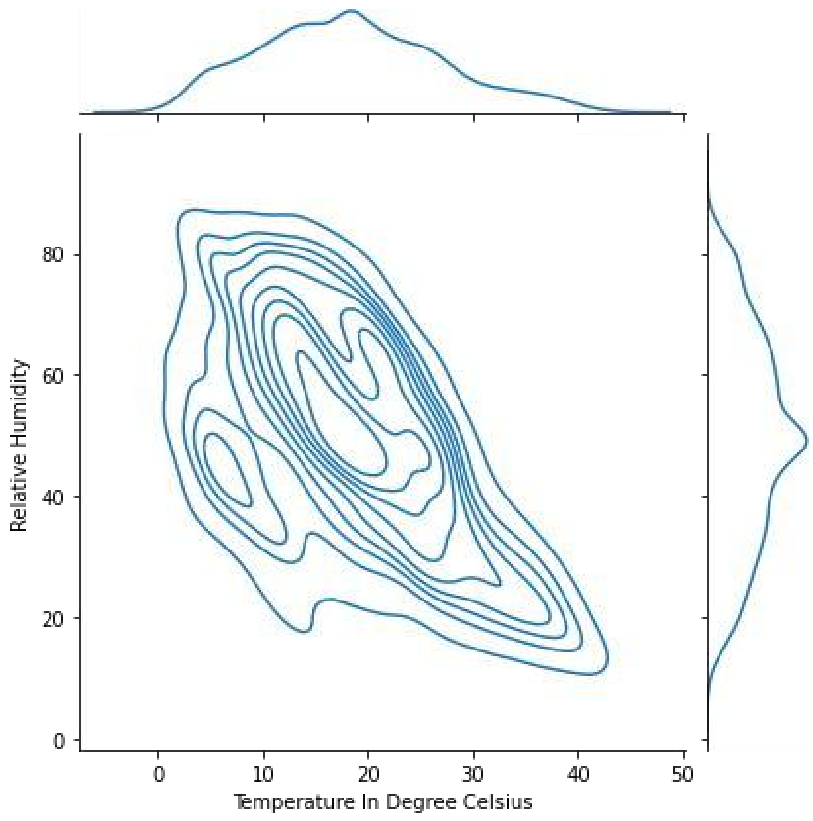
In [30]:
```
plt.xlabel("Temperature in Degree Celsius")
plt.ylabel('Relative Humidity')
plt.xlim(30,40)
plt.title("Relative Humidity vs Temperature")
plt.plot(df['T'],df["RH"])
```

Out[30]:
```
[<matplotlib.lines.Line2D at 0x15caa308b80>]
```



In [31]:
```
sns.jointplot(x="T", y="RH", data=df,kind= "kde").set_axis_labels("Temp
```

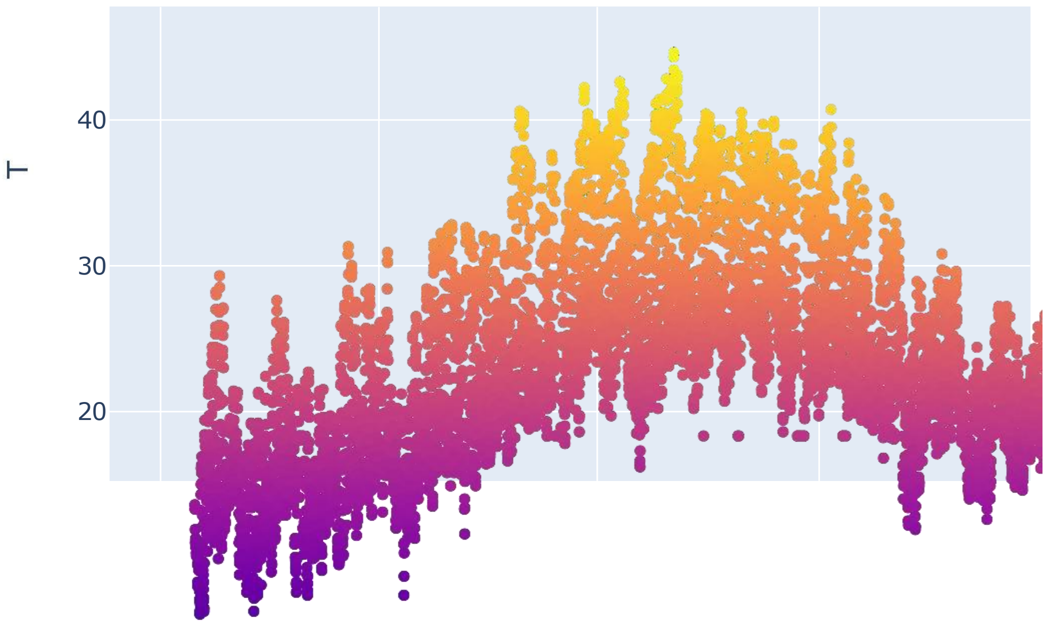
Out[31]:
```
<seaborn.axisgrid.JointGrid at 0x15caa4b12b0>
```



Varition of temperature over two years.

In [32]:

```
#[illegible]
fig =px.scatter(df,x="datetime",y="T",color ="T")
fig.show()
```



In [ ]:

CONCLUSION : Data visualization is a powerful tool in the fight against air pollution. By transforming complex datasets into clear, actionable insights, visualizations can enhance understanding, support policy decisions, and raise public awareness. The project outlined in this report leverages various data sources and visualization techniques to present air quality information effectively, aiming to contribute to the global effort in mitigating air pollution impacts.