In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
import nose.tools
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Air Pollution

Sourse:

1. Hypothesis

India has more conditions for air pollution compared to Taiwan.

2. Abstract

During the 2015 period, 27% Indian people consumed unhealthy air according to the AQI, during the remaining time the statistics show healthy levels of 58%-'Moderate' AQI, 12%-'poor', 12%-'satisfactory', 10%-'very poor' and the worst AQI readings being: 5%-'severe'. In Taiwan during 74% of the time the air had been 'good'(very little pollution level), 23% 'satisfactory' and 3% 'moderately'. India's population is 1.31 billion, compared to Taiwan's 23.57 million, which explains why there's a big difference in the AQI between

them. The main pollutants of the air come from the people and their social activities and needs industrial activity, operating motor vehicles, urban planning, etc.

Improved management of municipal and agricultural waste, including the capture of methane gas produced from waste sites as an alternative to incineration (for use as biogas); clean technology that lower industrial smokestack emissions. Assuring access to clean, affordable home energy options for lighting, heating, and cooking. Prioritizing quick urban transportation, walking and cycling networks in cities, as well as rail interurban freight and passenger service; switch to renewable energy generation; converting to low-emissions, cleaner Heavy-duty diesel cars and fuels, such as those with lower sulfur content. Enhancing building energy efficiency and making cities more compact and green to make them more energy efficient are some recommendations for urban planning. Increasing the use of low-emission fuels, renewable combustion-free energy sources (such as solar, wind, or hydropower), co-generation of heat and power, and distributed energy generation (such as mini-grids and rooftop solar power generation) are all important factors in the production of electricity. Strategies for waste reduction, waste separation, recycling, and reuse as well as waste reprocessing, as well as improved biological waste management techniques like anaerobic waste digestion to produce biogas, are workable, affordable alternatives to open incineration of solid waste for municipal and agricultural waste management. Combustion methods with tight emission controls are essential where incineration cannot be avoided.

Table of Contents

3. Introduction

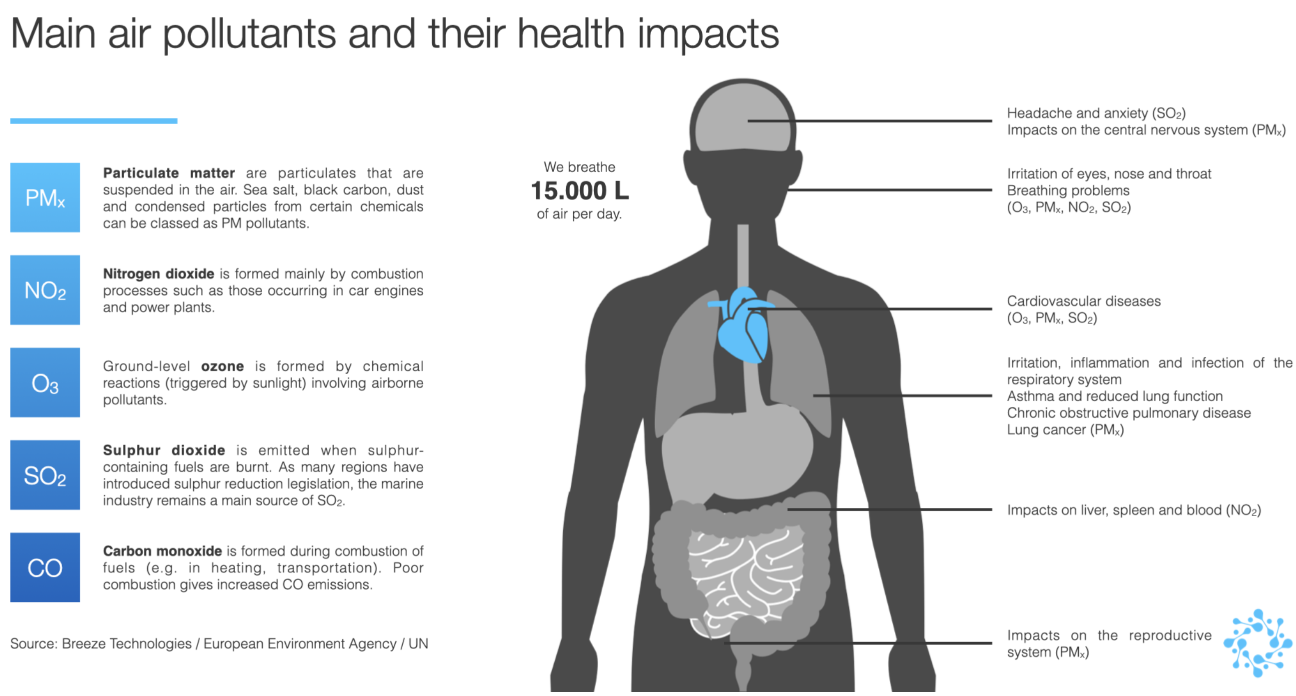
Any chemical, physical, or biological entity that alters the natural properties of the atmosphere is considered an air pollutant and can affect either the interior or outdoor environment.

Common causes of air pollution include motor vehicles, industrial operations, household combustion devices, and forest fires. Particulate matter, carbon monoxide, ozone, nitrogen dioxide, and sulfur dioxide are among the pollutants that pose the greatest threat to human health. Environmental and indoor air pollution are significant contributors to morbidity and death through causing respiratory and other illnesses.

According to the WHO data, low- and middle-income countries have the worst exposures, with 99 percent of the world's population breathing air that is polluted and exceeds WHO guideline levels.

Air quality around the globe has a direct impact on the earth's climate and ecosystems. One of several things that causes air pollution and emits greenhouse gases is the combustion of fossil fuels. Therefore, by lowering the burden of disease linked to air pollution and contributing in the short- and long-term mitigation of climate change, efforts to reduce air pollution offer a win-win strategy for both climate and health.

3.1. Impacts air pollution on our health



The pollutants that have the most reason to cause worry about the public's health are particulate matter (PM), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), and sulphur dioxide (SO2). Both short-term and long-term exposure to these different contaminants can have negative health effects. There are certain contaminants for which there are no limits beyond which negative effects do not happen.

Particulate Matter (PM): Sulphate, nitrates, ammonia, sodium chloride, black carbon, mineral dust, or water are examples of the inhalable particles that make up particulate matter (PM). PM10 and PM2.5 particulate matter's (PM10) and 2.5 microns' (PM2.5) respective health hazards are particularly well known. Deep lung penetration by PM allows it to enter the circulation and have an influence on the heart (ischaemic heart disease), brain (stroke), and lungs. The risk of developing cardiovascular and respiratory disorders as well as their mortality is increased by both short-term and long-term exposure to particle matter. Additional poor prenatal outcomes and lung cancer have been associated to long-term exposure. By 2013, the International Agency for Research on Cancer (IARC) of the WHO has identified it as a cause of lung cancer (IARC).It is also the most often used metric for determining how exposure to air pollution affects health.As a significant part of PM2.5 and a strong greenhouse gas, black carbon also contributes to regional environmental disturbance and hastens glacier melting.

Carbon monoxide(CO) is a colorless, odorless, and tasteless poisonous gas that is created when carbonaceous fuels like wood, gasoline, charcoal, natural gas, and kerosene are not completely burned. The circulation is invaded by carbon monoxide, which makes it challenging for the body's cells to bind oxygen. Cells and tissues are harmed by this oxygen deficiency. Breathing issues, tiredness, lightheadedness, and other flu-like symptoms can all result from carbon monoxide exposure. Death can result from extremely high CO exposure.

Nitrogen Dioxide (NO2): Nitrogen dioxide is a powerful oxidant and a reddish-brown gas that is soluble in water. Combustion processes used in power generation, heating, and transportation all result in the creation of NO2. Exposure to nitrogen dioxide can irritate the airways and exacerbate respiratory problems. NO2, a pollutant that is a significant precursor of ozone, is closely linked to asthma and other respiratory illnesses.

Ozone(O3): One of the main elements of smog is ground-level ozone. It is created by photochemical interactions with pollutants, such as nitrogen oxides (NOx) released by industry and cars. The largest quantities of ozone are seen during sunny periods because of the photochemical nature of the gas. Breathing issues, asthma flare-ups, lowered lung function, and lung illness can all be results of severe ozone exposure.

Sulfur dioxide (SO2) is a reactive, colorless air pollutant with a pungent smell. Sulfur dioxide irritates the mucous membranes of the skin, eyes, nose, throat, and lungs. High SO2 concentrations can irritate and inflame the respiratory system, especially during strenuous activity. Some of the indications and symptoms include pain while taking a deep breath, coughing, throat inflammation, and breathing difficulty. In those that are vulnerable, high SO2 levels can impede lung function, aggravate asthma episodes, and exacerbate underlying heart disease. The capacity of this gas to interact with other airborne molecules can result in the formation of a small particle that, if breathed, can have a similar detrimental effect on health.

According to estimates, ambient air pollution is to blame for 4.2 million deaths worldwide, mostly from heart disease, stroke, chronic obstructive pulmonary disease, lung cancer, and acute respiratory infections.

Converting from ppb to $\mu g/m^3$ is: $\frac{W \times C}{24.45}$

W - molecular weight

C - concentration

The number 24.45 in the equation is the volume (litres) of a mole (gram molecular weight) of a gas when the temperature is at 25°C and the pressure is at 1 atmosphere (1 atm = 1.01325 bar).

The same equations above can be used for conversion between mg/m^3 (milligrams per cubic metre) and ppm (parts per million) as well.

Thus, mg/m^3 represents milligrams (one-thousandth of a gram) per cubic metre of air, while $\mu g/m^3$ stands for micrograms (one-millionth of a gram) per cubic metre of air. However, these concentrations can also be expressed as parts per million (ppm) or parts per billion (ppb) by volume through a conversion factor.

1 ppm = 1000 ppb

Air Pollutant	Conversion Factor	Molecular Weight
Ammonia (NH3)	1 ppb = 0.7 $\mu g/m^3$	17.03 g/mol
Carbon monoxide (CO)	1 ppb = 1.15 $\mu g/m^3$	28.01 g/mol
Nitric oxide (NO)	1 ppb = 1.23 $\mu g/m^3$	30.01 g/mol
Nitrogen dioxide (NO2)	1 ppb = 1.88 $\mu g/m^3$	46.01 g/mol
Ozone (O3)	1 ppb = 1.96 $\mu g/m^3$	48 g/mol
Sulphur dioxide (SO2)	1 ppb = 2.62 $\mu g/m^3$	64.07 g/mol

4. Tidying and cleaning data

4.1. *About India data*

India data is from Kaggle

Particulate Matter 2.5(PM2.5) - ug/m^3

Particulate Matter 10(PM10) - ug/m^3

Nitric Oxide(NO) - ug/m^3

Nitric Dioxide(NO2) - ug/m^3

Any Nitric x(NOx)- ppb

Carbon Monoxide(CO) - ug/m^3

Sulphur Dioxide(SO2) - ug/m^3

In [3]:
```
india = pd.read_csv("city_hour.csv")
```

In [4]:
```
india.head()
```

Out[4]:

	City	Datetime	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylen
0	Ahmedabad	2015-01-01 01:00:00	NaN	NaN	1.00	40.01	36.37	NaN	1.00	122.07	NaN	0.0	0.0	0.(
1	Ahmedabad	2015-01-01 02:00:00	NaN	NaN	0.02	27.75	19.73	NaN	0.02	85.90	NaN	0.0	0.0	0.(
2	Ahmedabad	2015-01-01 03:00:00	NaN	NaN	0.08	19.32	11.08	NaN	0.08	52.83	NaN	0.0	0.0	0.(
3	Ahmedabad	2015-01-01 04:00:00	NaN	NaN	0.30	16.45	9.20	NaN	0.30	39.53	153.58	0.0	0.0	0.(
4	Ahmedabad	2015-01-01 05:00:00	NaN	NaN	0.12	14.90	7.85	NaN	0.12	32.63	NaN	0.0	0.0	0.(

In [5]:
```
india["year"] = india.Datetime.apply(lambda x: int(x.split()[0].split("-")[0]))
india = india[india.year == 2015]
```

We will make a column for 'year' and look at only the year '2015' matching it with the data for Taiwan.

In [6]:
```
def observations_and_features(dataset):
    """
    Returns the number of observations and features in the provided dataset
    """
    observations = dataset.shape[0]
    features = dataset.shape[1]
    return f"{observations} observations on {features} features"
```

In [7]:
```
(india.isna().sum() / india.shape[0]) * 100
```

```
Out[7]: City            0.000000
Datetime        0.000000
PM2.5          35.857623
PM10           82.368178
NO             18.778099
NO2            18.306190
NOx             5.509572
NH3            59.156817
CO             13.511180
SO2            21.679519
O3             20.363534
Benzene        29.947599
Toluene        27.851550
Xylene         55.171644
AQI            36.430762
AQI_Bucket     36.430762
year            0.000000
dtype: float64
```

P - percentage of missing values from total:

$$P = \frac{missing}{total} \times 100$$

We don't need the last columns (Benzene,Toluene,Xylene and NH3). Simply said we won't be using them because in the other data they are not present and it will unnecessary make formatting harder and less clean.

```
In [8]: india.drop(["NH3", "Benzene", "Xylene", "Toluene"], axis=1, inplace=True)
```

To understand how to fill the average(mean) values of the current column or to use the 'median function' we need to see the distribution.

```
In [9]: # for column in india.columns[2:-1]:
        #     sns.displot(india[column], kde=True)
        #     plt.show()
```

```
In [10]: india["AQI_Bucket"] = india["AQI_Bucket"].fillna("Moderate")
```

We fill the column 'AQI_Bucket' with 'Moderate' because it is the common quality of the polluted air.

The distributions are very similar. if I'm not mistaken the distribution is called 'Pareto'. We will use median of each column to fill the mising values.

```
In [11]: india = india.fillna(india[india.columns[2:-2]].median())
```

```
In [12]: for column in india.columns:
             nose.tools.assert_equal(
                 india[column].isna().sum(), 0
             )  # Checking if our work can run properly.
```

```
In [13]: india["month"] = india.Datetime.apply(lambda x: int(x.split()[0].split("-")[1]))
         india["day"] = india.Datetime.apply(lambda x: int(x.split()[0].split("-")[2]))
```

Let's make columns 'month' and 'day' because we will need them later to measue monthly daily hourly and annual pollution.

In [14]:
```
india.Datetime = india.Datetime.apply(lambda x: int(x.split()[1].split(":")[0]))
india = india.rename(columns={"Datetime": "hours"})
```

I wan't to leave only 'hours' in the column - 'Datetime' and rename it to match the new criteria(hourly).

In [15]:
```
nose.tools.assert_equal(india.year.dtype, "int64")
nose.tools.assert_equal(india.month.dtype, "int64")
nose.tools.assert_equal(india.day.dtype, "int64")
nose.tools.assert_equal(
    india.hours.dtype, "int64"
)  # Checking if the dtypes of each column match.
```

In [16]:
```
india.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 67174 entries, 0 to 577174
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        67174 non-null  object 
 1   hours       67174 non-null  int64  
 2   PM2.5       67174 non-null  float64
 3   PM10        67174 non-null  float64
 4   NO          67174 non-null  float64
 5   NO2         67174 non-null  float64
 6   NOx         67174 non-null  float64
 7   CO          67174 non-null  float64
 8   SO2         67174 non-null  float64
 9   O3          67174 non-null  float64
 10  AQI         67174 non-null  float64
 11  AQI_Bucket  67174 non-null  object 
 12  year        67174 non-null  int64  
 13  month       67174 non-null  int64  
 14  day         67174 non-null  int64  
dtypes: float64(9), int64(4), object(2)
memory usage: 8.2+ MB
```

In [17]:
```
print(observations_and_features(india))
```

```
67174 observations on 15 features
```

In [18]:
```
nose.tools.assert_equal(
    observations_and_features(india), "67174 observations on 15 features"
)
```

The dtypes look like they are ready for work.

4.2. *About Taiwan data*

Taiwan data is from Kaggle.

Particulate Matter 2.5(PM2.5) - micrometer in ug/m^3

Particulate Matter 10(PM10) - micrometer in ug/m^3

Nitric Oxide(NO) - ppb

Nitric Dioxide(NO2) - ppb

Any Nitric x(NOx)- ppb

Carbon Monoxide(CO) - ppm

Sulphur Dioxide(SO2) - *ppb*

In [19]:
```
air_taiwan = pd.read_csv("2015_Air_quality_in_northern_Taiwan.csv", low_memory=False)
```

In [20]:
```
air_taiwan.head()
```

Out[20]:

	time	station	AMB_TEMP	CH4	CO	NMHC	NO	NO2	NOx	O3	...	RAINFALL	RAIN_COND	RH	S
0	2015/01/01 00:00	Banqiao	16	2.1	0.79	0.14	1.2	16	17	37	...	NR	NR	57	
1	2015/01/01 01:00	Banqiao	16	2.1	0.8	0.15	1.3	16	17	36	...	NR	NR	57	
2	2015/01/01 02:00	Banqiao	16	2.1	0.71	0.13	1	13	14	38	...	NR	NR	57	
3	2015/01/01 03:00	Banqiao	15	2	0.66	0.12	0.8	11	12	39	...	NR	NR	58	
4	2015/01/01 04:00	Banqiao	15	2	0.53	0.11	0.6	10	11	38	...	NR	NR	58	

5 rows × 23 columns

We delete the unnecessary columns.

In [21]:
```
print(observations_and_features(air_taiwan))
```

```
218640 observations on 23 features
```

In [22]:
```
nose.tools.assert_equal(
    observations_and_features(air_taiwan), "218640 observations on 23 features"
)
```

In [23]:
```
air_taiwan.dtypes  # Checking each type of our data.
```

Out[23]:
```
time          object
station       object
AMB_TEMP      object
CH4           object
CO            object
NMHC          object
NO            object
NO2           object
NOx           object
O3            object
PH_RAIN       object
PM10          object
PM2.5         object
RAINFALL      object
RAIN_COND     object
RH            object
SO2           object
THC           object
UVB           object
WD_HR         object
WIND_DIREC    object
WIND_SPEED    object
WS_HR         object
dtype: object
```

In [24]:
```
def changing_dtype_of_data_to_numeric(data, column=str):
    data[column] = data[column].apply(lambda x: float(x.replace(",", ".")))
```

We will change some column's dtypes so we can plot them.

In [25]:
```
(air_taiwan.isna().sum() / air_taiwan.shape[0]) * 100
```

Out[25]:
```
time            0.000000
station         0.000000
AMB_TEMP        8.447677
CH4            56.173161
CO              0.607849
NMHC           56.268295
NO              0.645810
NO2             0.895993
NOx             0.645353
O3              8.587175
PH_RAIN        84.188621
PM10            1.316776
PM2.5           1.313575
RAINFALL        4.418679
RAIN_COND      84.188621
RH              8.413831
SO2             0.728595
THC            56.172704
UVB            88.120198
WD_HR          16.432492
WIND_DIREC     16.555068
WIND_SPEED     16.540889
WS_HR          16.634651
dtype: float64
```

In [26]:
```
# for column in air_taiwan.columns[2:]:
#     sns.displot(air_taiwan[column], kde=True)
#     plt.show()
```

In [27]:
```
def change_dtype_to_numeric(data, colunm=str):
    data[colunm] = data[colunm].apply(lambda x: float(x))
```

In [28]:
```
def changing_wrong_values(data, column):  # removing the unnecessary symbols
    data[column] = air_taiwan[column].str.replace("x", "")
    data[column] = air_taiwan[column].str.replace("#", "")
    data[column] = air_taiwan[column].str.replace("*", "", regex=True)
    data[column] = air_taiwan[column].str.replace("NR", "0", regex=True)
```

In [29]:
```
for column in air_taiwan.columns[
    2:
]:  # replecing missing values with median of each column and checking our work
    changing_wrong_values(air_taiwan, column)
    change_dtype_to_numeric(air_taiwan, column)
    median_of_current_column = air_taiwan[column].median()
    air_taiwan[column] = air_taiwan[column].mask(
        air_taiwan[column] < 0, median_of_current_column
    )
    # air_taiwan[column] = air_taiwan[column].replace(0, np.nan)
    # air_taiwan[column] = air_taiwan[column].replace(np.nan, median_of_current_column)
    air_taiwan[column] = air_taiwan[column].fillna(median_of_current_column)
    nose.tools.assert_equal(air_taiwan[column].dtype, "float64")
    nose.tools.assert_equal(air_taiwan[column].isna().sum(), 0)
```

We replace the `NaN` values with the median of each column because we have a lot of `NaN` values and if we delete all of them we will lose most of our data and we want to use it as much as possible.

Values smaller than 0 are missing values.

In [30]:
```
air_taiwan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 218640 entries, 0 to 218639
Data columns (total 23 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   time        218640 non-null  object
 1   station     218640 non-null  object
 2   AMB_TEMP    218640 non-null  float64
 3   CH4         218640 non-null  float64
 4   CO          218640 non-null  float64
 5   NMHC        218640 non-null  float64
 6   NO          218640 non-null  float64
 7   NO2         218640 non-null  float64
 8   NOx         218640 non-null  float64
 9   O3          218640 non-null  float64
 10  PH_RAIN     218640 non-null  float64
 11  PM10        218640 non-null  float64
 12  PM2.5       218640 non-null  float64
 13  RAINFALL    218640 non-null  float64
 14  RAIN_COND   218640 non-null  float64
 15  RH          218640 non-null  float64
 16  SO2         218640 non-null  float64
 17  THC         218640 non-null  float64
 18  UVB         218640 non-null  float64
 19  WD_HR       218640 non-null  float64
 20  WIND_DIREC  218640 non-null  float64
 21  WIND_SPEED  218640 non-null  float64
 22  WS_HR       218640 non-null  float64
dtypes: float64(21), object(2)
memory usage: 38.4+ MB
```

In [31]:
```
air_taiwan.time = air_taiwan.time.apply(lambda x: x.split())   # TO DO
```

We have to do a little work on the column `time`. Our job is to make new columns - `year, month, day` .From the column `time` we will leave only 'hours' in.

In [32]:
```
air_taiwan["year"] = air_taiwan.time.apply(lambda x: int(x[0].split("/")[0]))
air_taiwan["month"] = air_taiwan.time.apply(lambda x: int(x[0].split("/")[1]))
air_taiwan["day"] = air_taiwan.time.apply(lambda x: int(x[0].split("/")[2]))
```

In [33]:
```
air_taiwan.time = air_taiwan.time.apply(lambda x: int(x[1].split(":")[0]))
air_taiwan = air_taiwan.rename(columns={"time": "hours"})
```

In [34]:
```
print(observations_and_features(air_taiwan))
```

```
218640 observations on 26 features
```

In [35]:
```
air_taiwan.drop(
    [
        "AMB_TEMP",
        "PH_RAIN",
        "RAINFALL",
        "RAIN_COND",
        "RH",
        "THC",
        "UVB",
        "WD_HR",
        "WIND_DIREC",
        "WIND_SPEED",
        "WS_HR",
        "NMHC",
    ],
```

```
    axis=1,
    inplace=True,
)
```

We remove the unnecessary columns.

In [36]:
```
nose.tools.assert_equal(
    observations_and_features(air_taiwan), "218640 observations on 14 features"
)
```

5. Grouping and Presenting Data

Source:)

Pollutant	Averaging Time	2005 AQGs	2021 AQGs
$PM_{2.5}$, µg/m^3	Annual	10	5
	24-hour[a]	25	15
PM_{10}, µg/m^3	Annual	20	15
	24-hour[a]	50	45
O_3, µg/m^3	Peak season[b]	-	60
	8-hour[a]	100	100
NO_2, µg/m^3	Annual	40	10
	24-hour[a]	-	25
SO_2, µg/m^3	24-hour[a]	20	40
CO, mg/m^3	24-hour[a]	-	4

We will work with 2005 AQGs year bc our datas are for 2015 year. WHO dont released 2021 AQGs.

Since the WHO organization's data from 2005 depicts 03 and C0's values as 0, but in 2021 they're 4, 25 and 60 ,for the year 2015 we will use calculations from our own data(~2, ~45).

NOx in India data is in ppb we will transform in to ug/m^3. This will happend with formula from higher.

In [37]:
```
NOx_ug = 46 * 1 / 24.45
```

In [38]:
```
NOx_ug
```

Out[38]:
```
1.8813905930470347
```

In [39]:
```
india.NOx = india.NOx * NOx_ug
```

In [40]:
```
air_taiwan.NOx = air_taiwan.NOx * NOx_ug
```

For the column 'CO' we need to first divide it by 1000 ppm = ppb/1,000, and ppb = (1,000)ppm and then we will transform it to ug/m^3

In [41]:
```
air_taiwan.CO = air_taiwan.CO / 1000
```

In [42]:
```
india.CO = india.CO / 1000
```

In [43]:
```
air_taiwan.NO2 = air_taiwan.NO2 * 1.88
```

In [44]:
```
air_taiwan.NO = air_taiwan.NO * 1.23
```

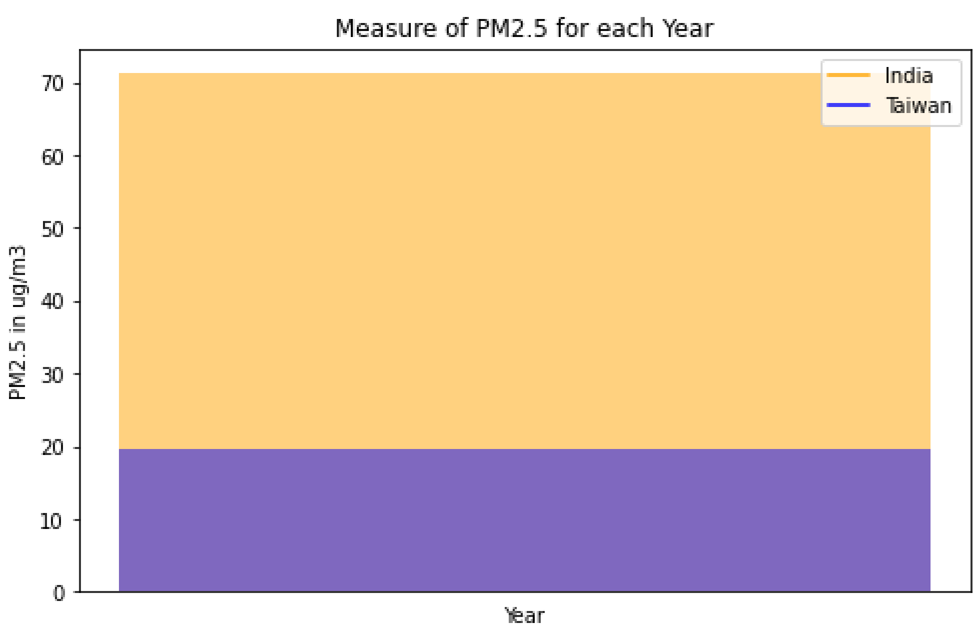
In [45]:
```
air_taiwan.SO2 = air_taiwan.SO2 * 2.62
```

we must ploting them separate bc values in current column in India data is too small from Taiwan

In [46]:
```
india_group = india.groupby("hours")["O3"].mean().iloc[::8]
```

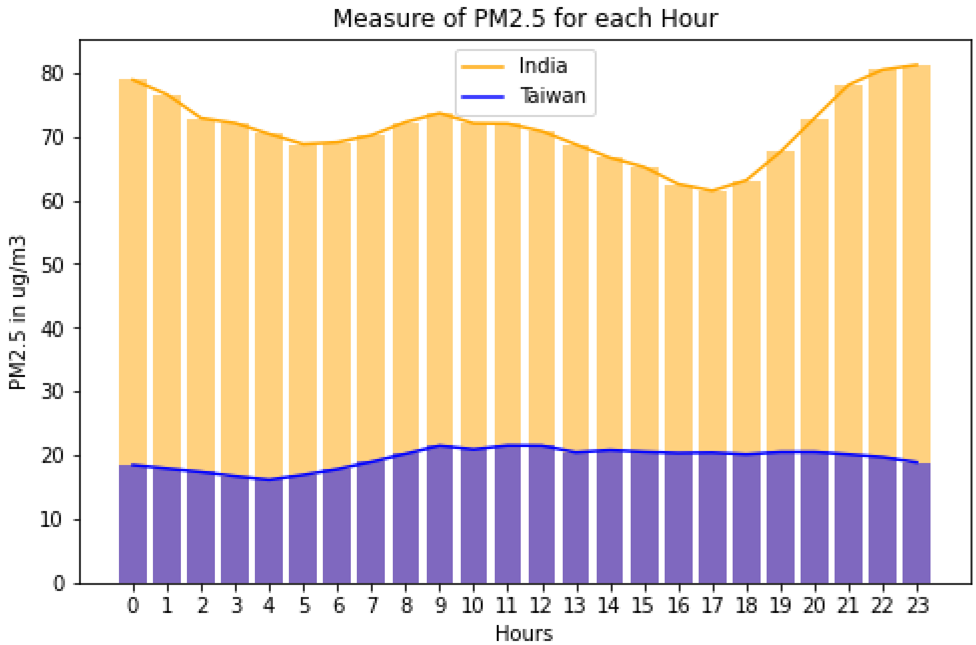
In [47]:
```
def make_bar_plot_using_two_datas(
    india_data, taiwan_data, india_column, taiwan_column, time_for_comparison
):
    india_group = india_data.groupby(time_for_comparison)[india_column].mean()
    taiwan_group = taiwan_data.groupby(time_for_comparison)[taiwan_column].mean()
    plt.figure(figsize=(8, 5))
    plt.bar(india_group.index, india_group, alpha=0.5, color="orange")
    plt.plot(india_group.index, india_group, color="orange")
    plt.bar(taiwan_group.index, taiwan_group, alpha=0.5, color="b")
    plt.plot(taiwan_group.index, taiwan_group, color="b")
    if time_for_comparison == "day":
        plt.xticks(np.arange(1, 32))
        plt.xlabel(f"Day")
        plt.title(f"Measure of {india_column} for each Day")
    elif time_for_comparison == "month":
        plt.xticks(np.arange(1, 13))
        plt.xlabel(f"Month")
        plt.title(f"Measure of {india_column} for each Month")
    elif time_for_comparison == "year":
        plt.xticks(np.arange(1))
        plt.xlabel(f"Year")
        plt.title(f"Measure of {india_column} for each Year")
    elif time_for_comparison == "hours":
        plt.xticks(np.arange(0, 24))
        plt.xlabel(f"Hours")
        plt.title(f"Measure of {india_column} for each Hour")
    plt.ylabel(f"{india_column} in ug/m3")
    plt.legend(labels=["India", "Taiwan"])
    plt.show()
```

In [48]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "PM2.5", "PM2.5", "year")
```



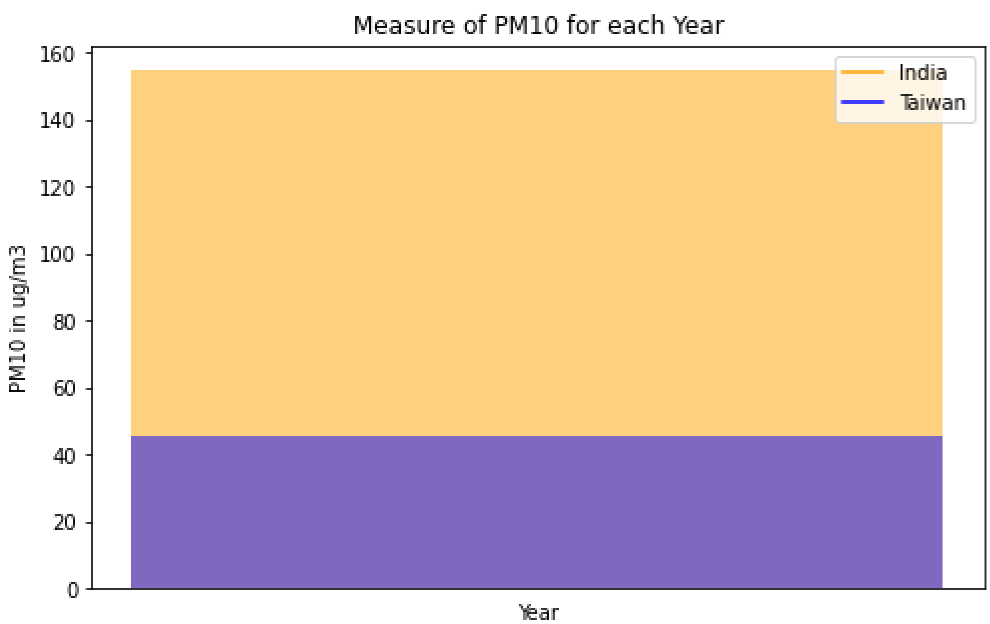
The annual measurements show us that Taiwan is over 2 times above the healthy levels whereas Inida is over 7.

In [49]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "PM2.5", "PM2.5", "hours")
```



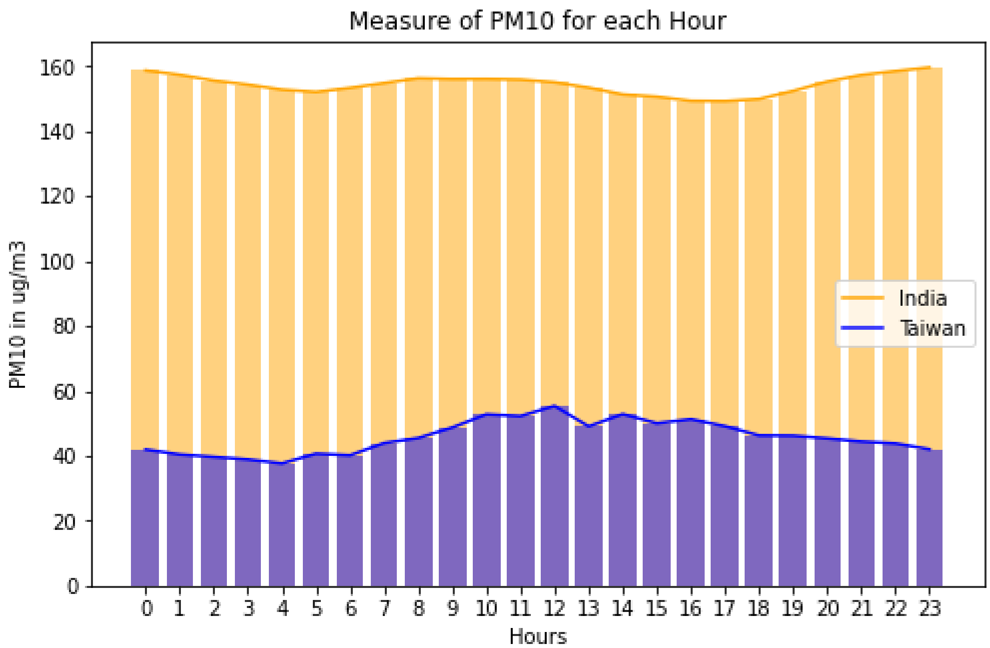
The 24 hour measurements shows that Taiwan is hovering in the healthy norms whereas india is 3.2 times above it.

In [50]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "PM10", "PM10", "year")
```



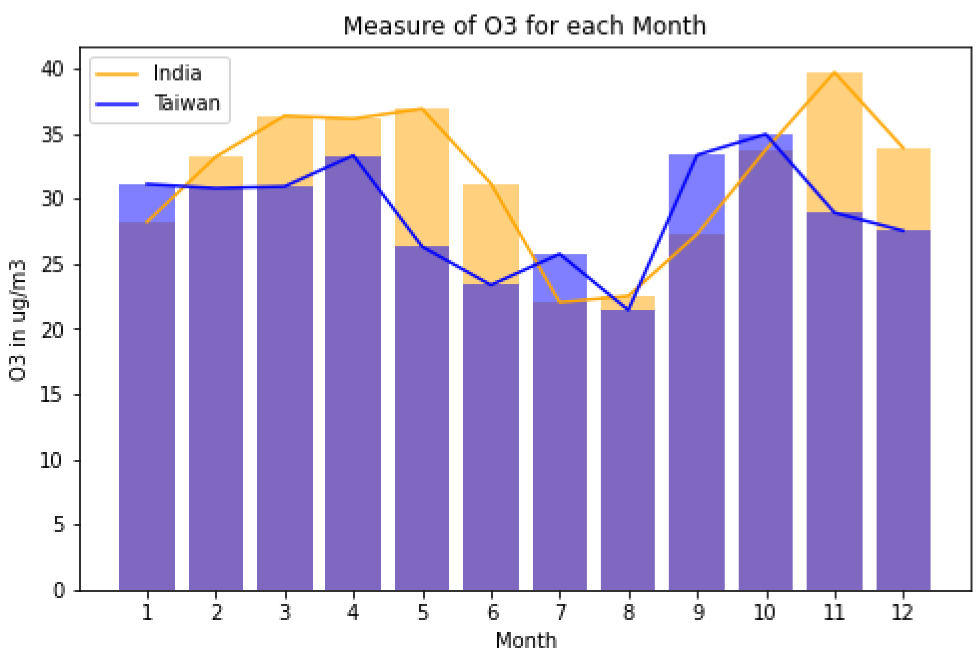
Annual average measure of PM10 reads Taiwan being a little over two times above the reccomended levels, opposed to India being ~8 times.

In [51]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "PM10", "PM10", "hours")
```

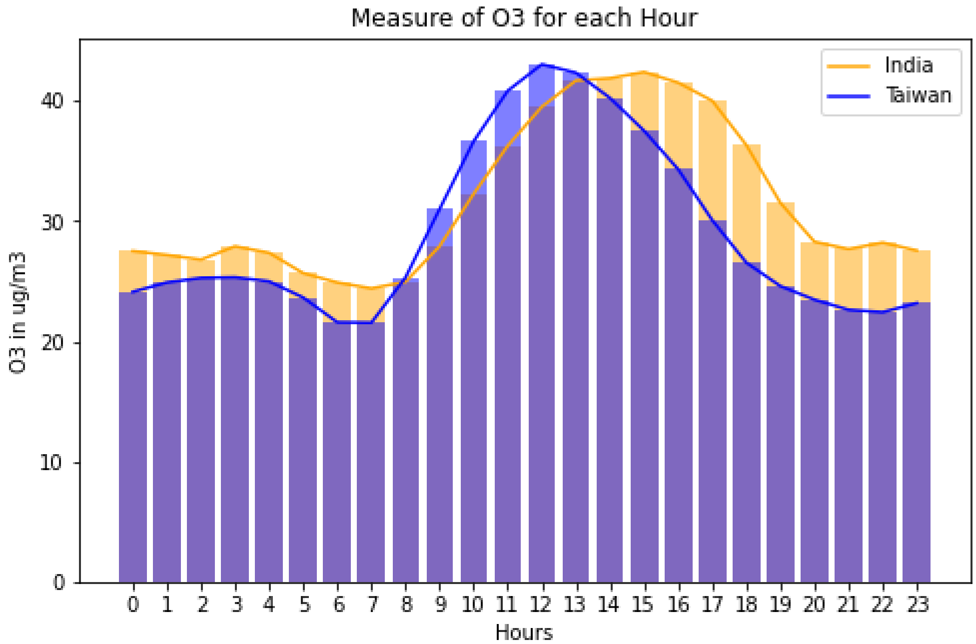


The 24 hour measue of PM10 allows us to see that Taiwan is in the normal levels and India is ~3 times over them.

In [52]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "O3", "O3", "month")
```

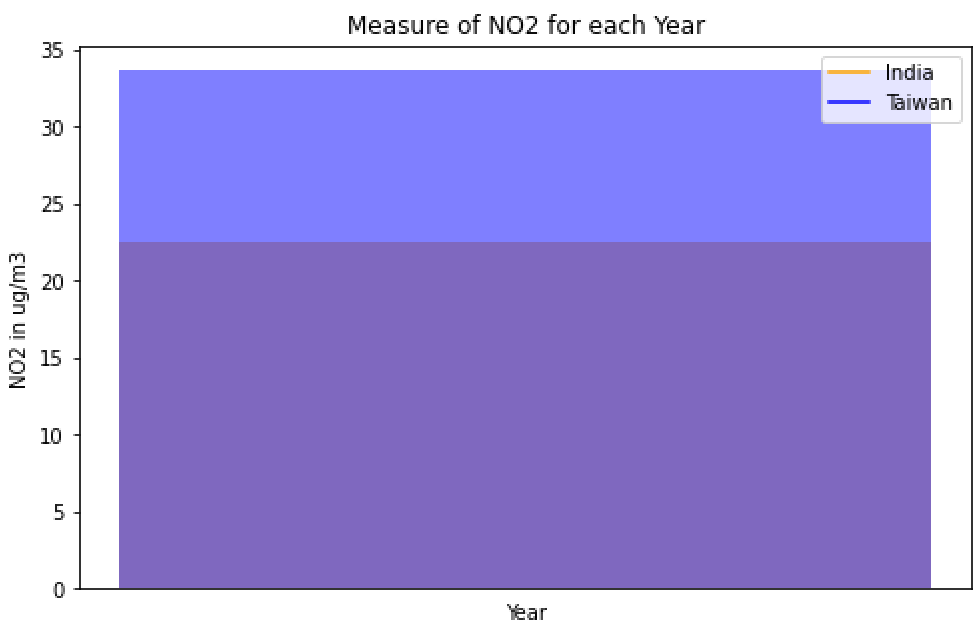


In [53]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "O3", "O3", "hours")
```



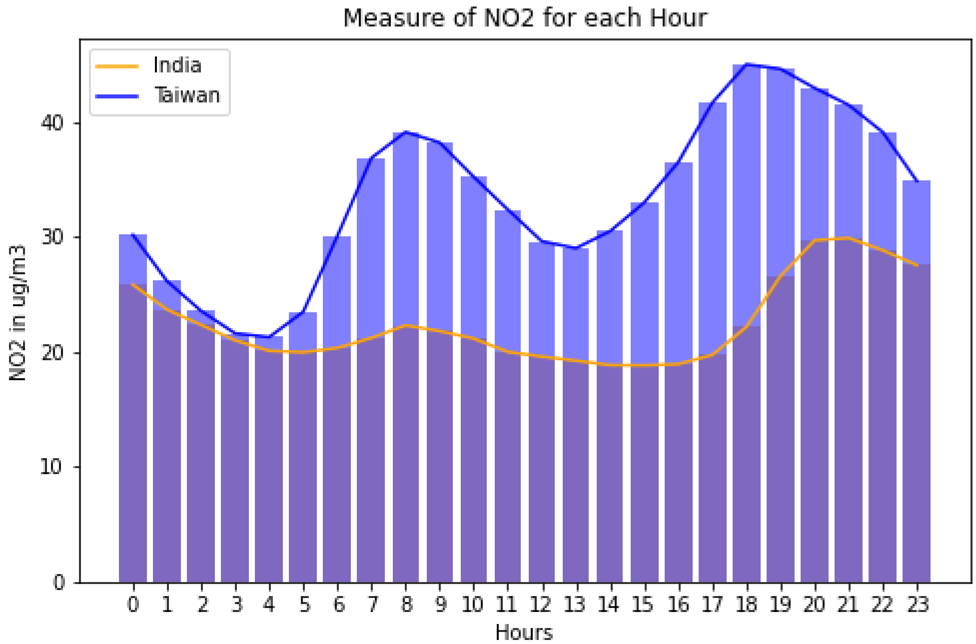
The hourly ozone measures shows us that there's a big peak in pollution between 8 and 18 o'clock(during the most public activity)

In [54]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "NO2", "NO2", "year")
```



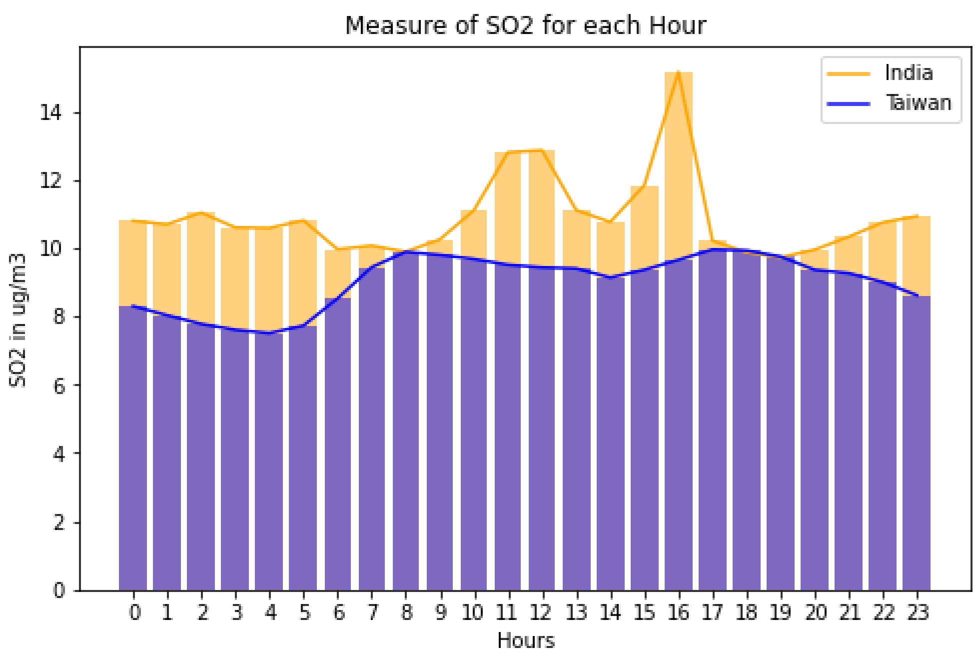
Annual measures of nitrogen dioxide are a little different as this time Taiwan is above India in polluting, although both are in the helthy levels.

In [55]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "NO2", "NO2", "hours")
```



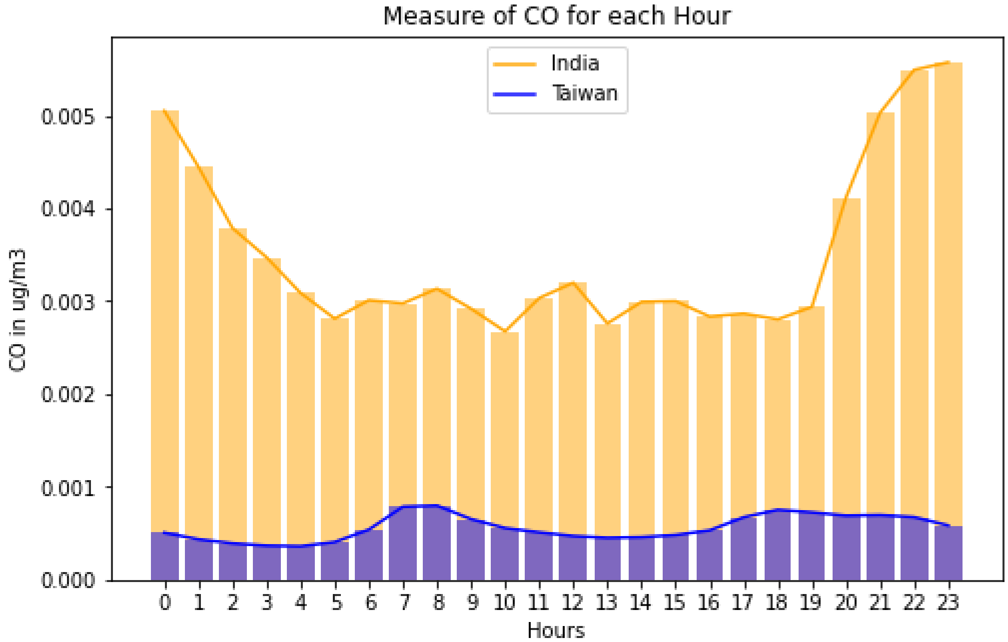
The 24 hour evaluations are again in the reccomended levels, again Taiwan being above India accompanied by some peaks during certain hours. These pollution induced activities are uknown to us but they might include some bush burning ;D

In [56]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "SO2", "SO2", "hours")
```



Both hourly graphs depict levels in the acceptable pollution range, with India being a little worse than Taiwan.

In [57]:
```
make_bar_plot_using_two_datas(india, air_taiwan, "CO", "CO", "hours")
```



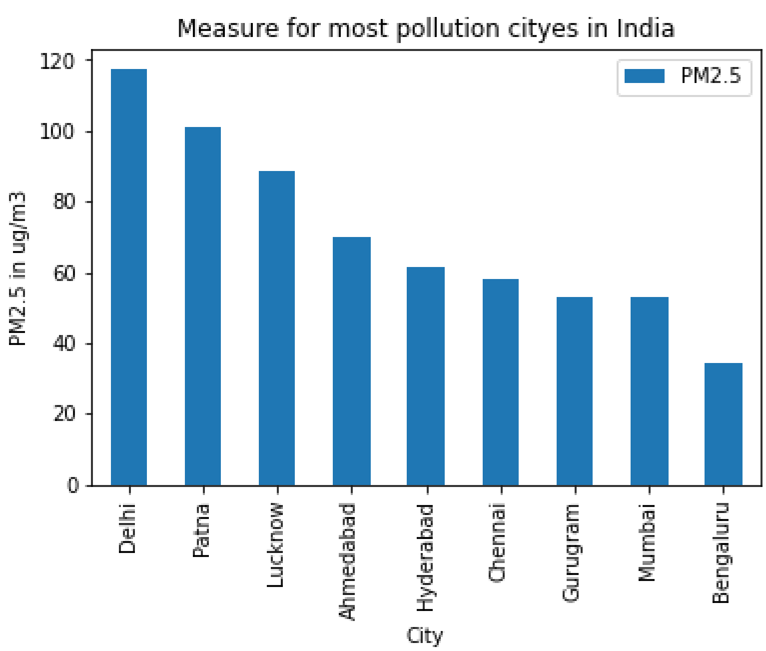
India is showing anywhere from 3 up to 6 times higher values than Taiwan, but they are still well in the clear.

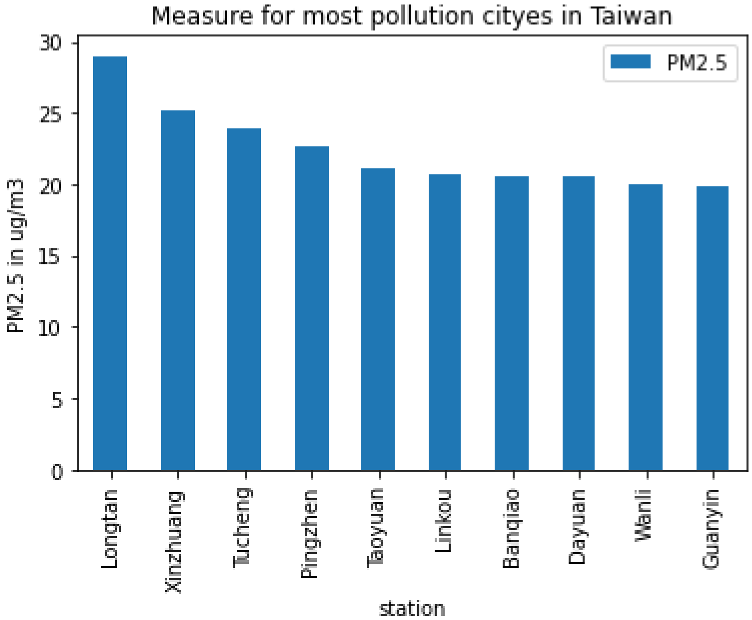
Conclusion: For such a big difference, it is understandable to expect a vastly different graph. Definitely the air of India is more polluted because the more people there are, the more needs they have.

I wonder if there is a difference in the most polluted cities. I assume that again India will be more polluted.

In [58]:
```
def plot_most_major_cities_of_current_pollution(pollution, city, data, name):
    data[[pollution, city]].groupby([city]).mean().sort_values(
        pollution, ascending=False
    )[:10].plot.bar()
    plt.ylabel(f"{pollution} in ug/m3")
    plt.title(f"Measure for most pollution cityes in {name}")
    plt.show()
```

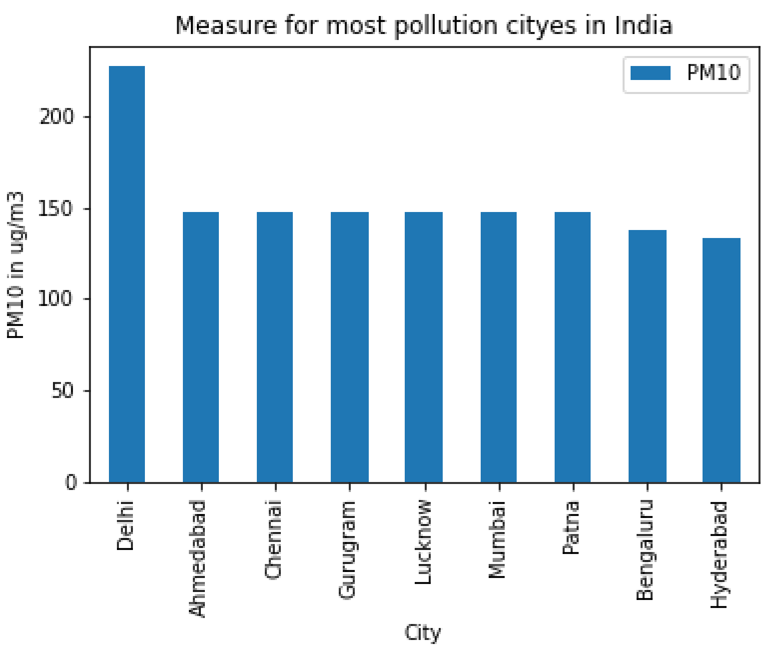
In [59]:
```
plot_most_major_cities_of_current_pollution("PM2.5", "City", india, "India")
plot_most_major_cities_of_current_pollution("PM2.5", "station", air_taiwan, "Taiwan")
```

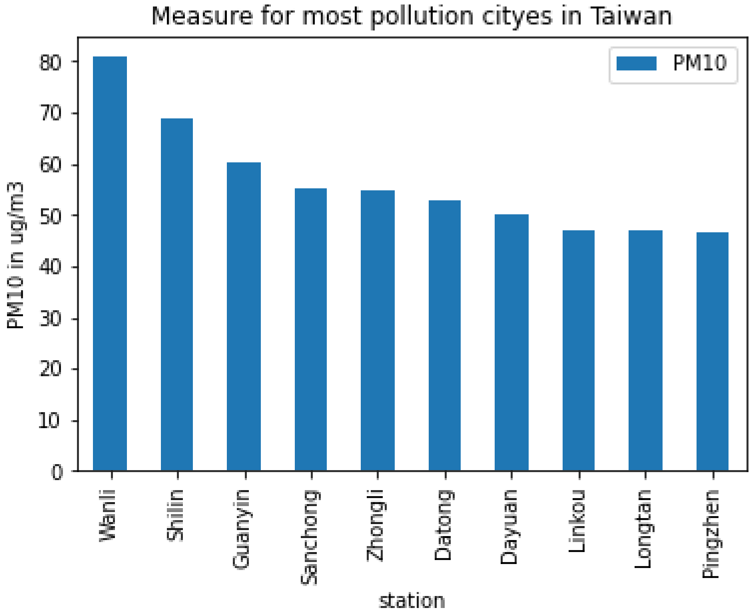




The most polluted cities is 4 times the dirty air of Taiwan for measure PM2.5.

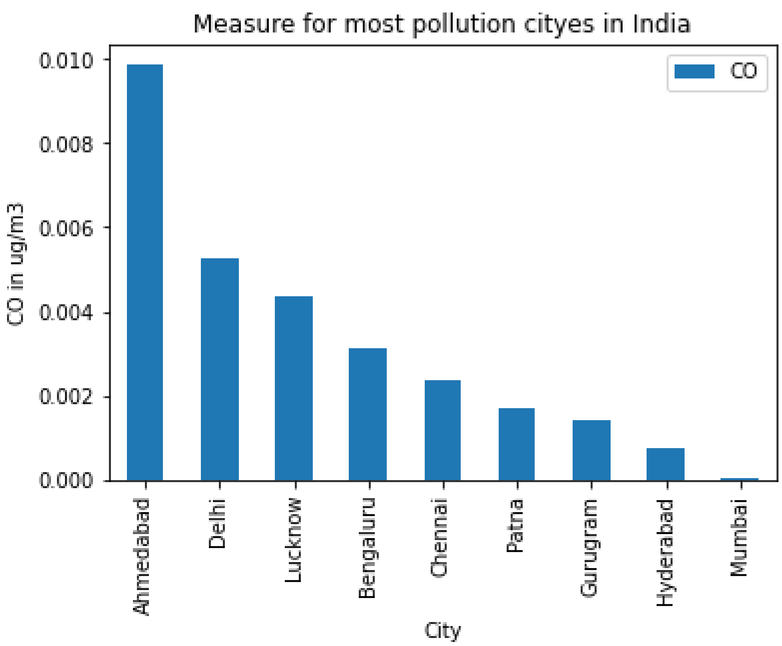
In [60]:
```
plot_most_major_cities_of_current_pollution("PM10", "City", india, "India")
plot_most_major_cities_of_current_pollution("PM10", "station", air_taiwan, "Taiwan")
```

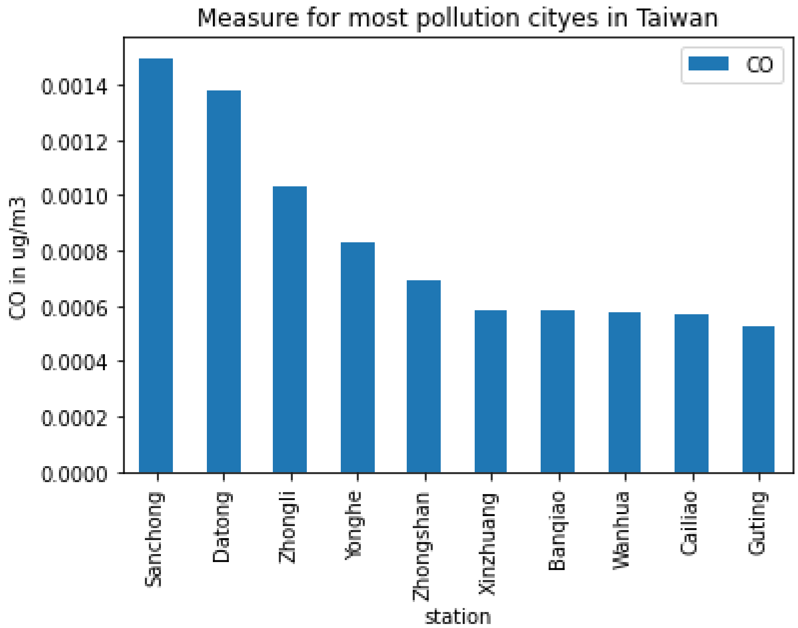




For the polluter PM10 it is double from Taiwan.

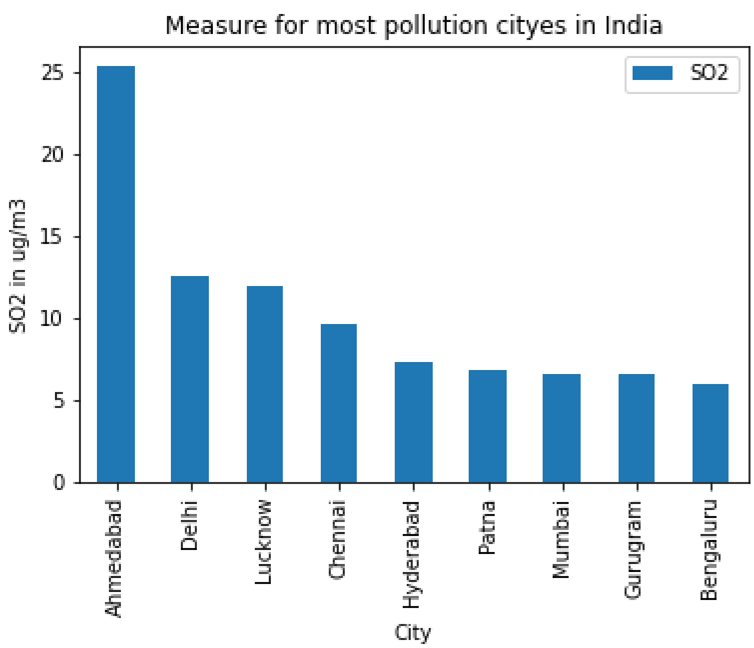
In [61]:
```
plot_most_major_cities_of_current_pollution("CO", "City", india, "India")
plot_most_major_cities_of_current_pollution("CO", "station", air_taiwan, "Taiwan")
```

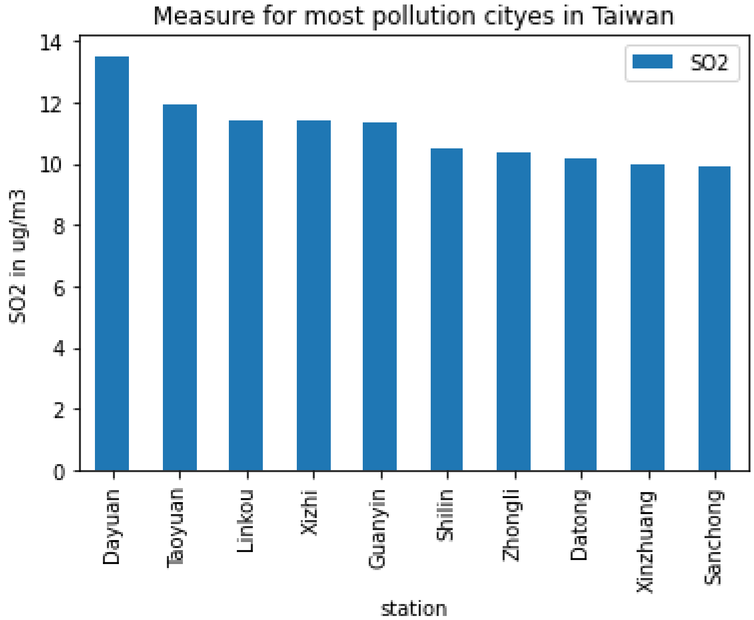




If we look a little more closely at the graph(CO), we will see that from the second most polluted city in India, cities from Taiwan become twice as polluted. But Amhedabad is 10 times more polluted then Sanchong.

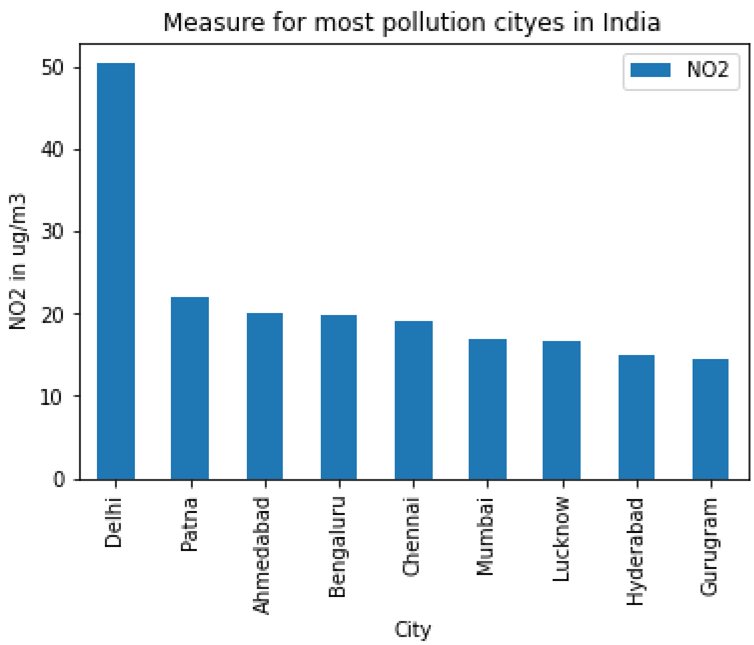
In [62]:
```
plot_most_major_cities_of_current_pollution("SO2", "City", india, "India")
plot_most_major_cities_of_current_pollution("SO2", "station", air_taiwan, "Taiwan")
```

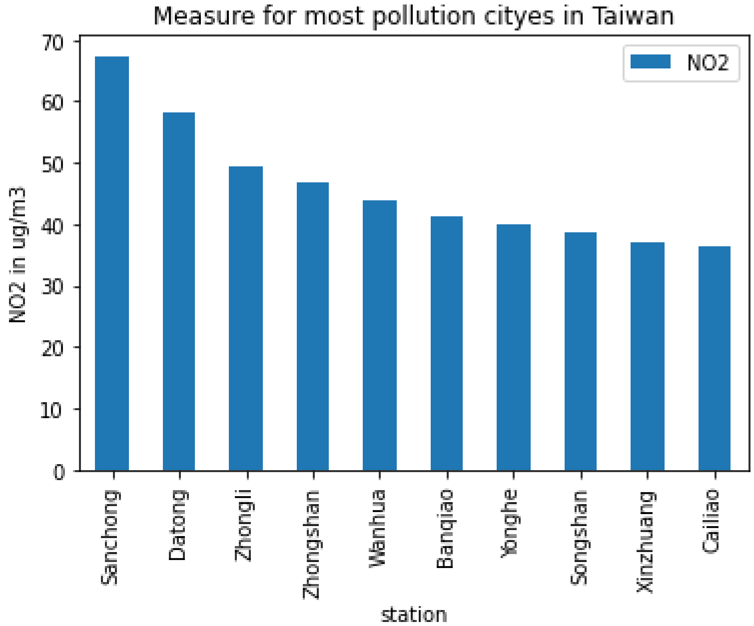




If we abstract from the first city, the other cities are almost equal. Measure - SO2

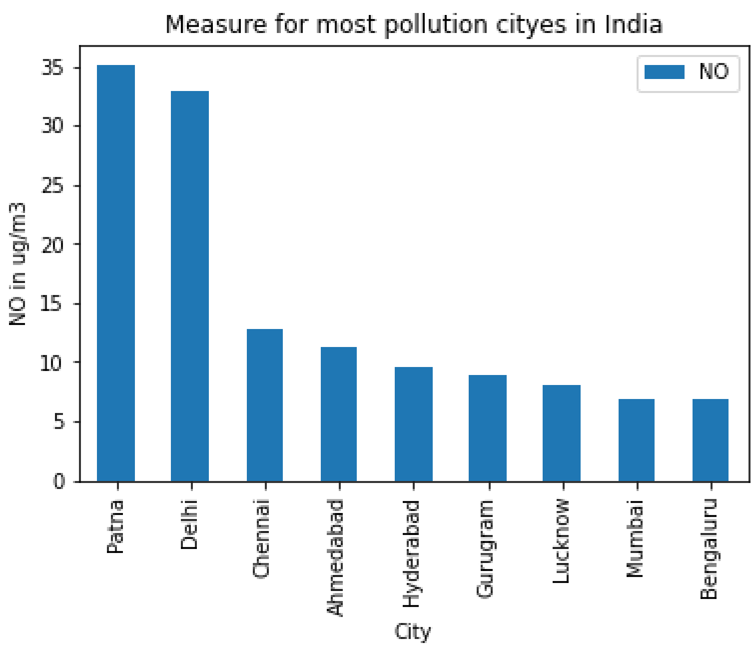
In [63]:
```
plot_most_major_cities_of_current_pollution("NO2", "City", india, "India")
plot_most_major_cities_of_current_pollution("NO2", "station", air_taiwan, "Taiwan")
```

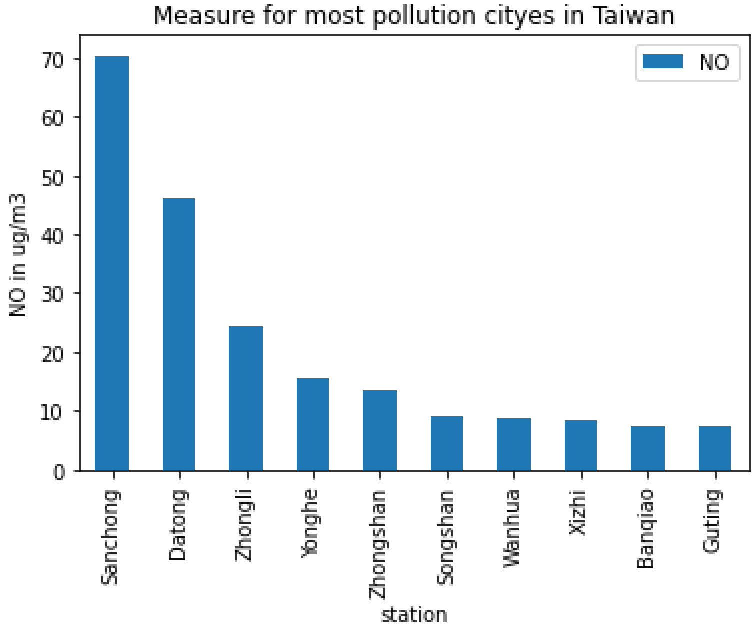




To measure the pollutant NO2. In this case India is a bit polluted.

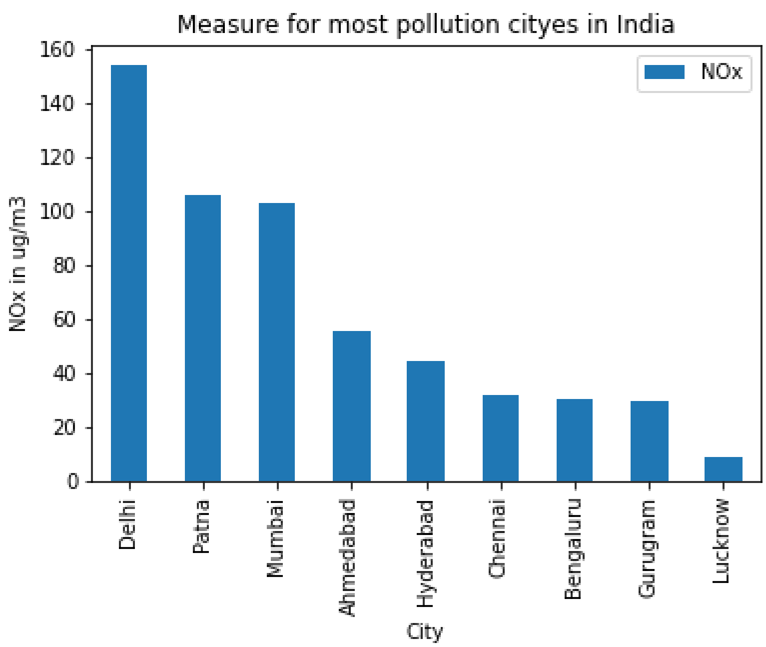
In [64]:
```
plot_most_major_cities_of_current_pollution("NO", "City", india, "India")
plot_most_major_cities_of_current_pollution("NO", "station", air_taiwan, "Taiwan")
```

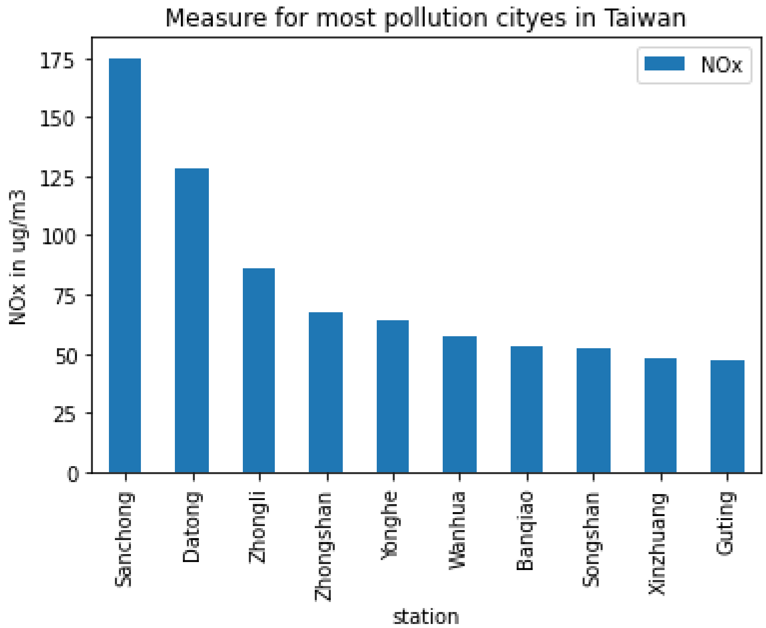




taiwan has dirtier air for this pollutant NO

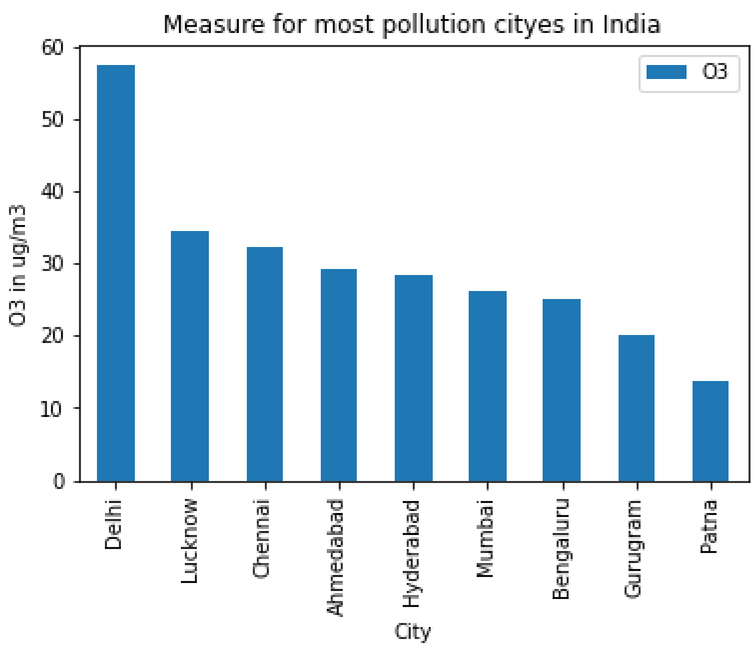
In [65]:
```
plot_most_major_cities_of_current_pollution("NOx", "City", india, "India")
plot_most_major_cities_of_current_pollution("NOx", "station", air_taiwan, "Taiwan")
```

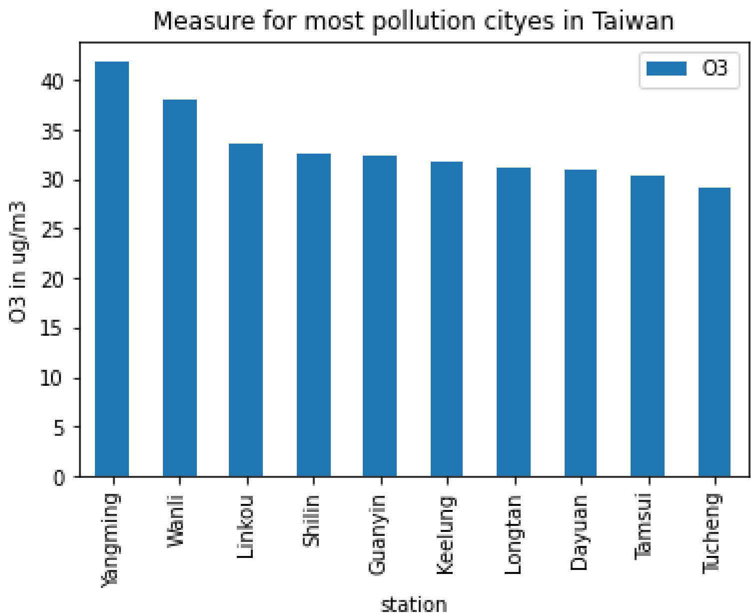




India has less pollution per pollutant NOx

In [66]:
```
plot_most_major_cities_of_current_pollution("O3", "City", india, "India")
plot_most_major_cities_of_current_pollution("O3", "station", air_taiwan, "Taiwan")
```





The first three cities india have more polluted cities but the others are less polluted than taiwan.This is for O3

Conclusion: India has a population of 1.32 billion while Taiwan has a population of 23.57 million. When the population decreases, pollution also decreases.

Let's look at and Air Quality Index.

What is Air Quality Index(AQI): The AQI system alerts people to harmful air pollution levels.

Source

AQI Category	AQI	Concentration range*							
		PM_{10}	$PM_{2.5}$	NO_2	O_3	CO	SO_2	NH_3	Pb
Good	**0 - 50**	**0 - 50**	**0 - 30**	**0 - 40**	**0 - 50**	**0 - 1.0**	**0 - 40**	**0 - 200**	**0 - 0.5**
Satisfactory	**51 - 100**	**51 - 100**	**31 - 60**	**41 - 80**	**51 - 100**	**1.1 - 2.0**	**41 - 80**	**201 - 400**	**0.5 - 1.0**
Moderately polluted	**101 - 200**	**101 - 250**	**61 - 90**	**81 - 180**	**101 - 168**	**2.1 - 10**	**81 - 380**	**401 - 800**	**1.1 - 2.0**
Poor	**201 - 300**	**251 - 350**	**91 - 120**	**181 - 280**	**169 - 208**	**10 - 17**	**381 - 800**	**801 - 1200**	**2.1 - 3.0**
Very poor	**301 – 400**	**351 - 430**	**121 - 250**	**281 - 400**	**209 - 748***	**17 - 34**	**801 - 1600**	**1200 -1800**	**3.1 - 3.5**
Severe	**401 - 500**	**430+**	**250+**	**400+**	**748+***	**34+**	**1600+**	**1800+**	**3.5+**

*** CO in mg/m^3 and other pollutants in $\mu g/m^3$; 2h-hourly average values for PM_{10}, $PM_{2.5}$, NO_2, SO_2, NH_3, and Pb, and 8-hourly values for CO and O_3.**

We don't have AQI in our taiwan data.Let's try making a new column 'AQI_Bucket' which contains AQI Category.

In [67]:
```
air_taiwan["AQI_Bucket"] = pd.cut(
    air_taiwan["PM10"],
    bins=[0, 50, 100, 250, 350, 430, 9999],
    labels=["Good", "Satisfactory", "Moderately", "Poor", "Very poor", "Severe"],
    include_lowest=True,
)
```

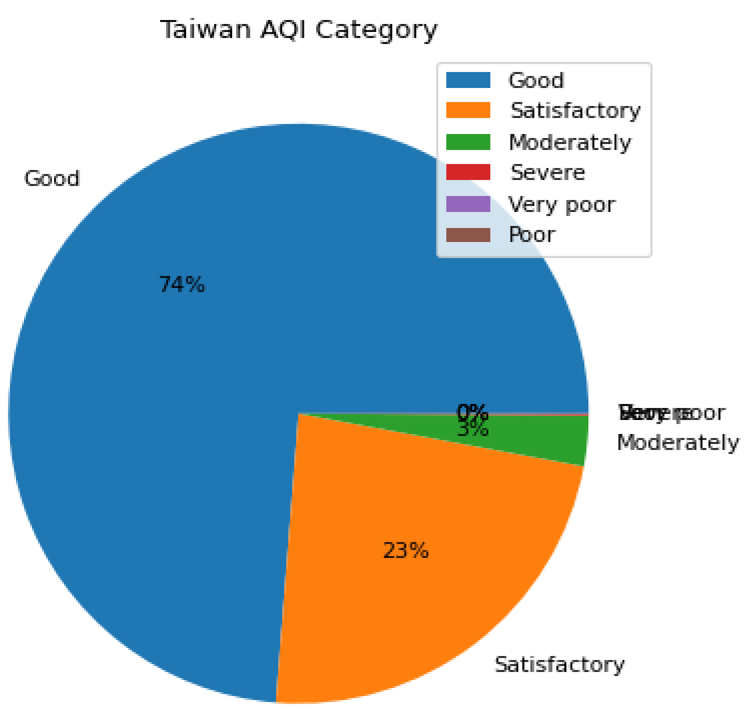
We replace the number with the given category.

In [68]:
```
group_taiwan_aqi_bucket = (
    air_taiwan.sort_values(by="AQI_Bucket")
    .groupby("AQI_Bucket")["AQI_Bucket"]
    .count()
    .sort_values()[::-1]
)
```

In [69]:
```
group_taiwan_aqi_bucket
```

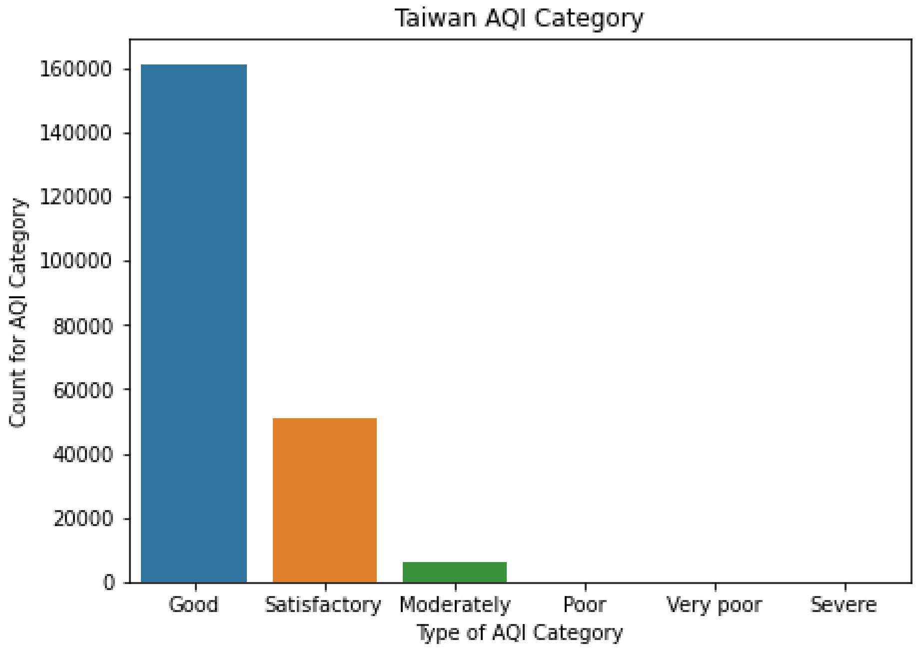
Out[69]:
```
AQI_Bucket
Good            161253
Satisfactory     51014
Moderately        6152
Severe             180
Very poor           21
Poor                20
Name: AQI_Bucket, dtype: int64
```

In [70]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.pie(
    group_taiwan_aqi_bucket, labels=group_taiwan_aqi_bucket.index, autopct="%0.0f%%"
)
# plt.xlabel(f"Type of AQI")
# plt.ylabel(f"Count for AQI")
plt.legend()
plt.title("Taiwan AQI Category")
plt.show()
```



In [71]:
```
plt.figure(figsize=(7, 5))
```

```
sns.barplot(x=group_taiwan_aqi_bucket.index, y=group_taiwan_aqi_bucket)
plt.xlabel(f"Type of AQI Category")
plt.ylabel(f"Count for AQI Category")
plt.title("Taiwan AQI Category")
plt.show()
```



Severe, very poor and poor approximately zero and we can't see them.

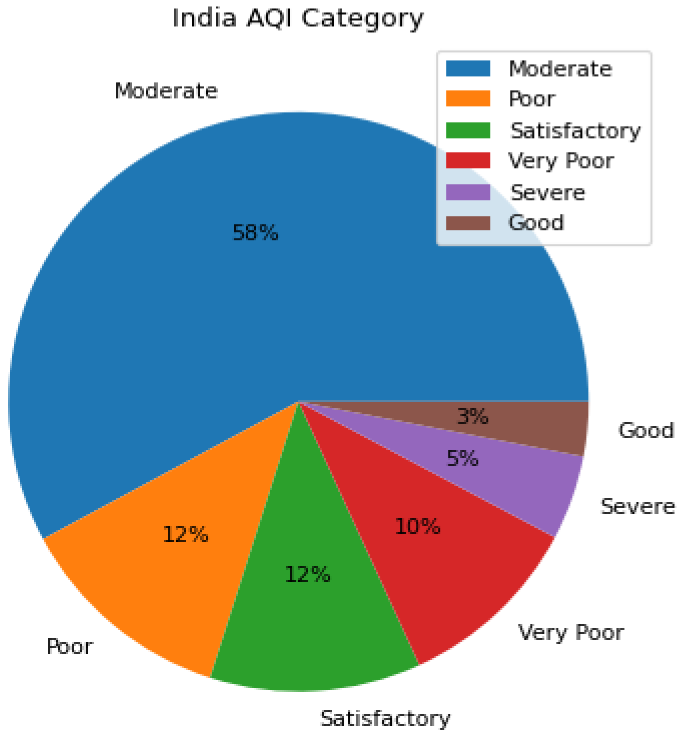
From pie and bar plot we understand that most of the time the air in Taiwan is 74% good, 23% satisfactory and 3% moderately.

In [72]:
```
group_india_aqi_bucket = (
    india.sort_values(by="AQI_Bucket")
    .groupby("AQI_Bucket")["AQI_Bucket"]
    .count()
    .sort_values()[::-1]
)
```

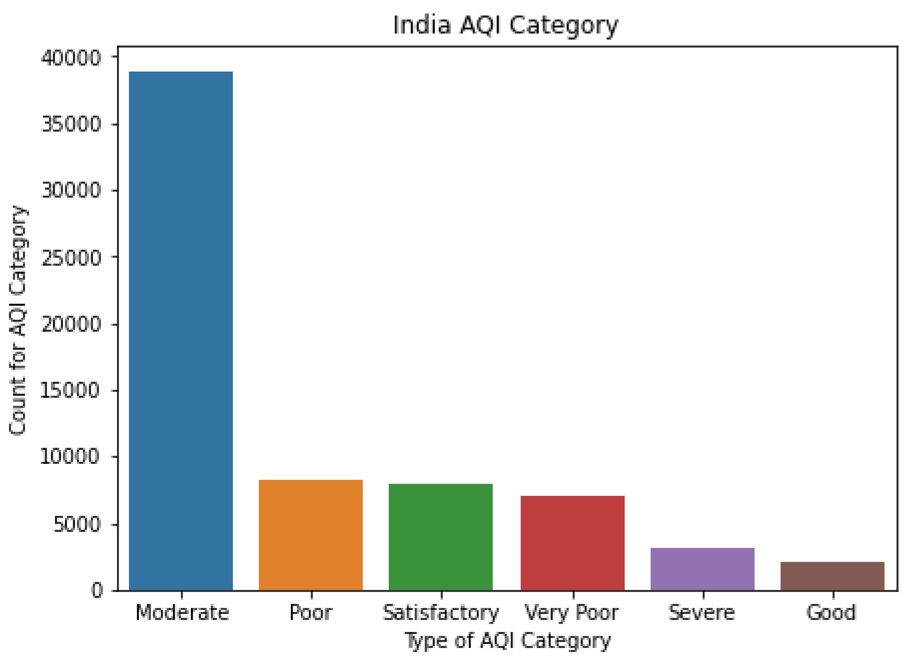
In [73]:
```
group_india_aqi_bucket
```

Out[73]:
```
AQI_Bucket
Moderate        38869
Poor             8203
Satisfactory     7908
Very Poor        6988
Severe           3168
Good             2038
Name: AQI_Bucket, dtype: int64
```

In [74]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.pie(
    group_india_aqi_bucket,
    labels=group_india_aqi_bucket.index,
    autopct="%0.0f%%",
)
plt.legend()
plt.title("India AQI Category")
plt.show()
```



In [75]:
```
plt.figure(figsize=(7, 5))
sns.barplot(x=group_india_aqi_bucket.index, y=group_india_aqi_bucket)
plt.xlabel(f"Type of AQI Category")
plt.ylabel(f"Count for AQI Category")
plt.title("India AQI Category")
plt.show()
```



Most common is moderate with 58%, poor-12%,satisfactory-12%,very poor-10%, severe-5% and least good.

Moderate: Air quality is acceptable; however, there may be some health concern for a small number of unusually sensitive people. While EPA cannot identify these people, studies indicate that there are people who experience health effects when air quality is in the moderate range.

Conclusion: According to AQI, India's air is 27% above the unhealthy norms, whereas people who live in Taiwan have cleaner air.

6. Machine Learning Model

In [76]:
```
india["AQI"] = np.log(india["AQI"])
```

In [77]:
```
X_train, X_test, y_train, y_test = train_test_split(
    india[[i for i in india.columns[2:-3] if i not in ["AQI", "AQI_Bucket"]]],
    india["AQI"],
    test_size=0.2,
    random_state=100,
)
```

In [78]:
```
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)

X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)
```

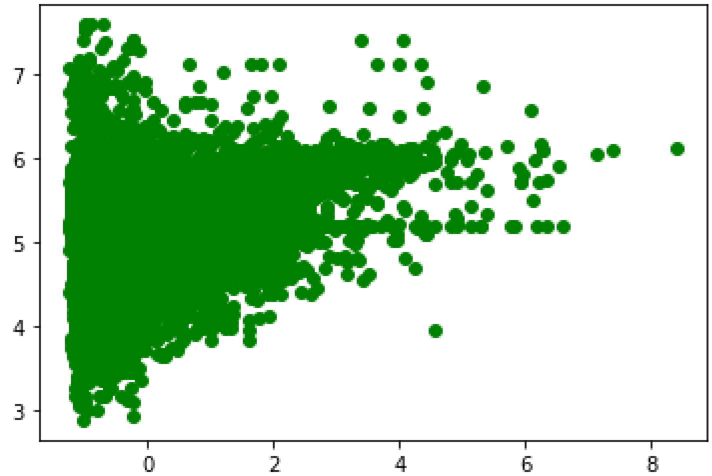
In [79]:
```
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
```

Out[79]:
```
▾ LinearRegression
LinearRegression()
```

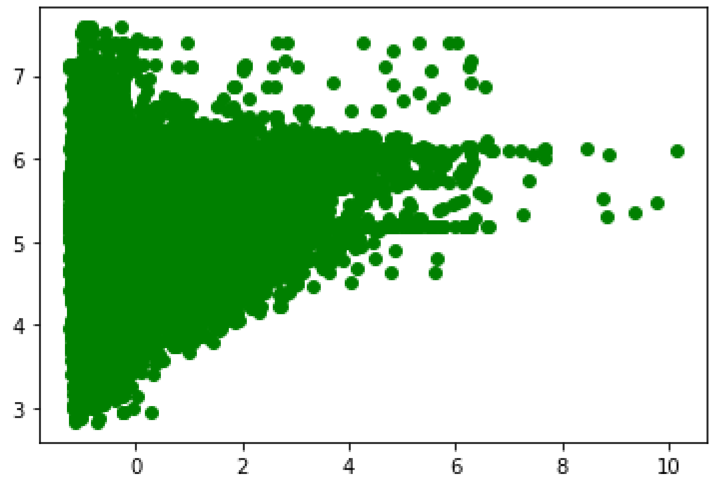
In [80]:
```
pred = model_lr.predict(X_train)

print("train mse: {}".format(mean_squared_error((y_train), (pred))))
print("train rmse: {}".format(mean_squared_error((y_train), (pred), squared=False)))
print("train r2: {}".format(r2_score((y_train), (pred))))
print()

# make predictions for test set
pred = model_lr.predict(X_test)

# determine mse, rmse and r2
print("test mse: {}".format(mean_squared_error((y_test), (pred))))
print("test rmse: {}".format(mean_squared_error((y_test), (pred), squared=False)))
print("test r2: {}".format(r2_score((y_test), (pred))))
```

```
train mse: 0.1899129053876505
train rmse: 0.43578997853054224
train r2: 0.35652742994184583

test mse: 0.1880787553142074
test rmse: 0.4336804760583619
test r2: 0.35334816219199394
```

In [81]:
```
plt.scatter(X_test["O3"], y_test, color="green")
plt.show()
```



In [82]:
```
plt.scatter(X_train["O3"], y_train, color="green")
plt.show()
```



7. Refferences

W.H.O.-air-quality-and-health)

Breeze Technologies

Prana Air

National Geographic

Explored by: Tihomir Dimitrov