1. The database(s) you plan to use for storage.

I will be using Mysql, Psql or google firebase for storing my dataset. Firstly, Mysql syntax is very similar to psql so adaptability and ease of use is two main reasons behind suggesting it. Secondly google firebase is a popular db which works quite fast and efficient compared to many cloud database systems.

Finally, talking about psql there is no doubt that I am well capable of handling such a sql language very similar to human language like python. Psql is very easy to learn if someone has the interest to work with it.

2. Where the data is loaded from.

Raw data is taken from the below website.

https://data.world/data-society/us-air-pollution-data/workspace/project-summary?agentid=data-society&datasetid=us-air-pollution-data

and the segregated data is loaded from my local computer folder. Github is used as repository for version control.

https://github.com/sujilkumarkm/da_assignment2_semester2_2021_dkit

3. Any transformations or scripts to load the data to the

database (e.g. cleansing in pandas)

dataset has been initially cleaned using pandas library In python. For time series analysis we only needed two columns date and carbon count do rest of the columns are removed from the csv file.

Current data contained multiple carbon values on the same day so using sql I have removed repeated dates to make the analysis accurate.

Below are codes used in pandas for transformations.

```
In [7]: df = df.iloc[:10000 , :]
        df1 = df
```

```
In [8]: df.shape
Out[8]: (10000, 28)
```

```
In [9]: df.describe()
Out[9]:
```

	State Code	County Code	Site Num	NO2 Mean	NO2 1st Max Value	NO2 1st Max Hour	NO2 AQI	O3 Mean	O3 1st Max Value	O3 1st Max Hour	
count	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000
mean	5.296800	14.67280	1504.333600	18.265320	35.613760	13.683600	32.613800	0.022896	0.037605	10.05660	33
std	0.954987	3.50425	1180.806768	13.197708	27.887599	7.676652	20.813226	0.010723	0.015856	3.31682	18
min	4.000000	13.00000	2.000000	0.739130	1.900000	0.000000	1.000000	0.001438	0.002000	0.00000	2
25%	4.000000	13.00000	1002.000000	9.826087	19.000000	6.000000	18.000000	0.014573	0.026000	9.00000	22
50%	6.000000	13.00000	1003.000000	15.250000	30.000000	18.000000	28.000000	0.022958	0.037000	10.00000	31
75%	6.000000	13.00000	3001.000000	23.168478	44.000000	20.000000	42.000000	0.030417	0.048000	11.00000	41
max	6.000000	25.00000	3003.000000	139.541667	267.000000	23.000000	132.000000	0.063167	0.113000	23.00000	195

Fig 1.0 (iLoc function to take last 10000 rows)

```
In [14]: df = df.dropna()
         df.isnull().sum()

Out[14]: State Code           0
         County Code          0
         Site Num             0
         Address              0
         State                0
         County               0
         City                 0
         Date Local           0
         NO2 Units            0
         NO2 Mean             0
         NO2 1st Max Value    0
         NO2 1st Max Hour     0
         NO2 AQI              0
         O3 Units             0
         O3 Mean              0
         O3 1st Max Value     0
         O3 1st Max Hour      0
         O3 AQI               0
         SO2 Units            0
         SO2 Mean             0
         SO2 1st Max Value    0
         SO2 1st Max Hour     0
         SO2 AQI              0
         CO Units             0
```

Fig 1.1(dropping null values from data)

```
In [15]: for i in df.columns[df.isnull().any(axis=0)]:     #---Applying mean to null Only on variables with NaN values
             df[i].fillna(df[i].mean(),inplace=True)
```

Fig 1.2(Replacing null values with mean)

Downsampling

```
In [42]: resample = ts_data.resample('Q')
         qrtly_mean_co = resample.mean()
         qrtly_mean_co.head(5)
```

```
Out[42]: date
         2000-03-31    1.460765
         2000-06-30    0.670598
         2000-09-30    0.567059
         2000-12-31    1.298862
         Freq: Q-DEC, Name: co_count, dtype: float64
```

Fig 1.3(downsampling to convert daily data to quarterly)

Square Root Transform

```
In [79]: from matplotlib import pyplot
         series = [i**2 for i in range(1,100)]
         # line plot
         pyplot.plot(series)
         pyplot.show()
```

Fig 1.4(transformation1 – square root method)

Log Transform

```
In [83]: from matplotlib import pyplot
         from math import exp

         series2 = ts_data


         series2 = [exp(i) for i in range(1,100)]
         pyplot.figure(1)
         # line plot
         pyplot.subplot(211)
         pyplot.plot(series2)
         # histogram
         pyplot.subplot(212)
         pyplot.hist(series2)
         pyplot.show()
```

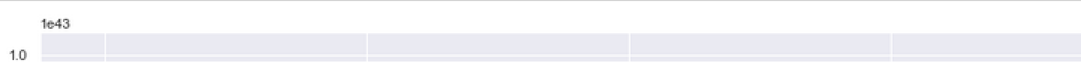


Fig 1.5(transformation2 – log method)

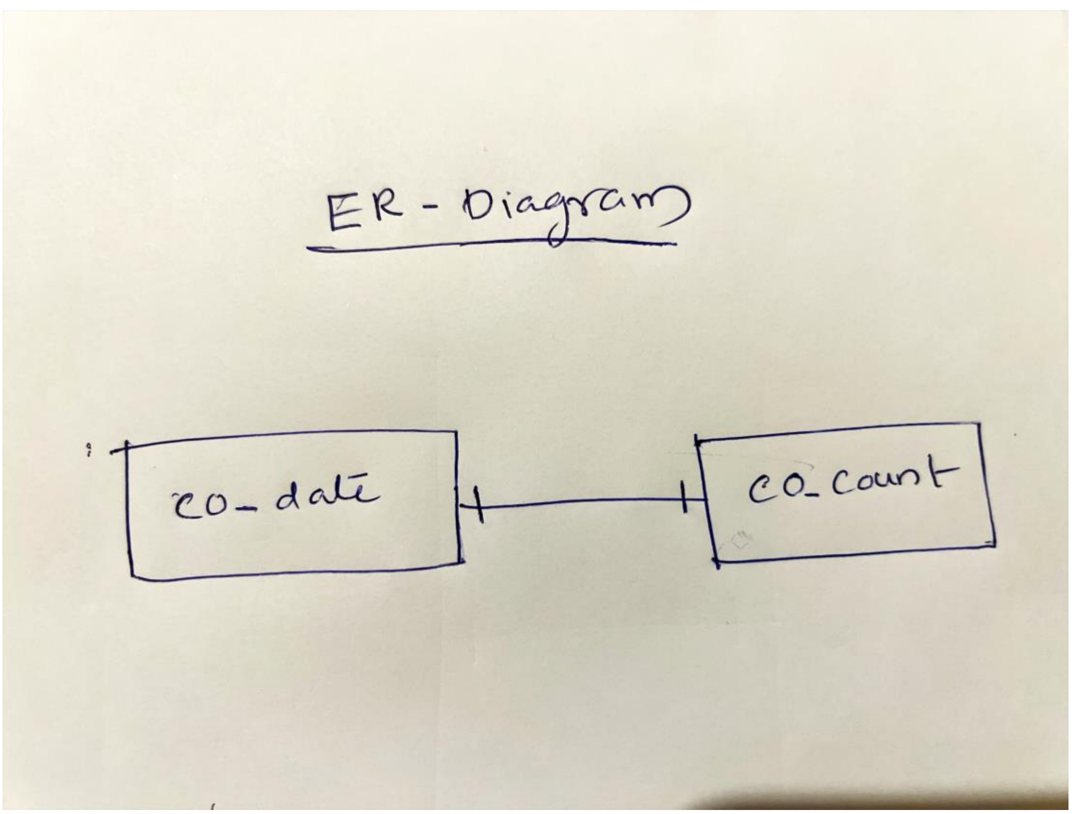
Resampling

In [40]:
```
resample = ts_data.resample('M')
monthly_mean_co = resample.mean()
monthly_mean_co.head(5)
```

Out[40]:
```
date
2000-01-31    1.932794
2000-02-29    1.453488
2000-03-31    0.989344
2000-04-30    0.838000
2000-05-31    0.621348
Freq: M, Name: co_count, dtype: float64
```

Fig 1.6(resampling to convert daily data to monthly)

4. The database schema (e.g. E-R diagram) itself.



From my understanding, the data we used in this project is comparatively smaller compared to previous semester project in DA. The reason being that we have to work with the same time series dataset which we used for time series prediction. It only contains 2 columns. So, the schema is made according to the available dataset for the year of 2000 carbon emission in the US.

5. Connections to/from the database for Time Series Analysis module

So, as we know for time series analysis, we need sequential data and that is what time series is all about. In my study my project was focusing on daily carbon emissions in the US. Database contains same time series data which is taken for doing analysis in psql. Transformation techniques from time series has ben applied to this project making it a part of the time series analysis module.

I have always wondered how to make python techniques we used in time series completely in sql and this assignment gave me chance to look deeper and explore more levels of PostgreSQL.