Project 01: Explore Weather Trends

Accessing data with sql:

- There are three tables named as city_data, city_list and global_data
- These tables are accessed using the following sql commands

    Select * from city_data;

    Select * from city_list;

    Select * from global_data;

- Downloaded these tables into csv files to local machine

    I have found Hyderabad (closest city and India (country)) is the city I live in. So, extracted the relevant data using the below sql query.

    select * from city_data where city ='Hyderabad' and country = 'India';

Importing Files:

These files have been imported to Jupiter notebooks as shown below. Also used head method to view the top 5 entries

Please note, city_filtered is the major data frame for further manipulations

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

city_data = pd.read_csv('D:\Data Analyst - Udacity\Datasets\city_data.csv')
city_list = pd.read_csv('D:\Data Analyst - Udacity\Datasets\city_list.csv')
global_data = pd.read_csv('D:\Data Analyst - Udacity\Datasets\global_data.csv')
city_data.head()
```

]:

	year	city	country	avg_temp
0	1849	Abidjan	Côte D'Ivoire	25.58
1	1850	Abidjan	Côte D'Ivoire	25.52
2	1851	Abidjan	Côte D'Ivoire	25.67
3	1852	Abidjan	Côte D'Ivoire	NaN
4	1853	Abidjan	Côte D'Ivoire	NaN

```
city_list.head()
```

	city	country
0	Abidjan	Côte D'Ivoire
1	Abu Dhabi	United Arab Emirates
2	Abuja	Nigeria
3	Accra	Ghana
4	Adana	Turkey

```
global_data.head()
```

	year	avg_temp
0	1750	8.72
1	1751	7.98
2	1752	5.78
3	1753	8.39
4	1754	8.47

```
city_filtered = pd.read_csv('D:\Data Analyst - Udacity\Datasets\city_filtered.csv')
city_filtered.head()
```

	year	city	country	avg_temp
0	1796	Hyderabad	India	26.53
1	1797	Hyderabad	India	27.48
2	1798	Hyderabad	India	26.20
3	1799	Hyderabad	India	26.84
4	1800	Hyderabad	India	26.88

Merging City and Global data files

City_filtered and global_data data frames have been merged on year as this the unique column in both files. The column names have been modified from '**avg_temp_x', 'avg_temp_y'** to **'city_avg_temp'** and **'global_avg_temp**' respectively for easy understanding

Source for renaming columns: https://www.geeksforgeeks.org/how-to-rename-columns-in-pandas-dataframe/

```
#merge global_data with city_filtered data

city_filtered_merge = pd.merge(city_filtered, global_data, on = 'year', how = 'inner')
print (city_filtered_merge.head())
city_filtered_merge = city_filtered_merge.rename(columns = {'avg_temp_x' : 'city_avg_temp', 'avg_temp_y' : 'global_avg_temp'
print (city_filtered_merge.head())
```

```
   year       city country  avg_temp_x  avg_temp_y
0  1796  Hyderabad   India       26.53        8.27
1  1797  Hyderabad   India       27.48        8.51
2  1798  Hyderabad   India       26.20        8.67
3  1799  Hyderabad   India       26.84        8.51
4  1800  Hyderabad   India       26.88        8.48
   year       city country  city_avg_temp  global_avg_temp
0  1796  Hyderabad   India          26.53             8.27
1  1797  Hyderabad   India          27.48             8.51
2  1798  Hyderabad   India          26.20             8.67
3  1799  Hyderabad   India          26.84             8.51
4  1800  Hyderabad   India          26.88             8.48
```

City_filtered_merge is the final file that I will be using for the moving averages

Assessing the data quality:

I have used .info () method to view any missing values and data quality issues.

There are 7 missing values in **‘city_avg_temp’** column. The missing values can be determined by looking at total entries and any num-null count column in the below table.

```
#checeking the missing values and data types
city_filtered_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 218 entries, 0 to 217
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   year             218 non-null    int64
 1   city             218 non-null    object
 2   country          218 non-null    object
 3   city_avg_temp    211 non-null    float64
 4   global_avg_temp  218 non-null    float64
dtypes: float64(2), int64(1), object(2)
memory usage: 10.2+ KB
```

Imputation:

In this step, I will be imputing the missing values with mean values as my strategy.

Asses the imputation is complete by applying .info () method on the data frame

The missing values have been imputed as you can see the 'city_avg_temp' has got 218 entries.

```
# filling missing values with mean values in city_avg_temp column
city_filtered_merge['city_avg_temp'].fillna(city_filtered_merge['city_avg_temp'].mean(), inplace=True)

#altering the year datatype
#city_filtered_merge['year']= pd.to_datetime(city_filtered_merge['year'])
```

```
city_filtered_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 218 entries, 0 to 217
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   year             218 non-null    int64
 1   city             218 non-null    object
 2   country          218 non-null    object
 3   city_avg_temp    218 non-null    float64
 4   global_avg_temp  218 non-null    float64
dtypes: float64(2), int64(1), object(2)
memory usage: 10.2+ KB
```

Moving Averages:

I have defined a moving_averages function to calculate the moving averages on year column with a dynamic window size (10 is optimal in this case)

```
# defining moving averages function

def moving_averages (data= None, window_size= None):
    mov_avg = data.rolling(window=window_size, center=False, on='year').mean().dropna()

    return(mov_avg)

roll_window_size = 10

mov_avg = moving_averages(data=city_filtered_merge, window_size=roll_window_size)
print(mov_avg.head())
```

```
    year  city_avg_temp  global_avg_temp
9   1805      26.931000            8.551
10  1806      26.967000            8.567
11  1807      26.834000            8.544
12  1808      26.900156            8.440
13  1809      26.902313            8.297
```

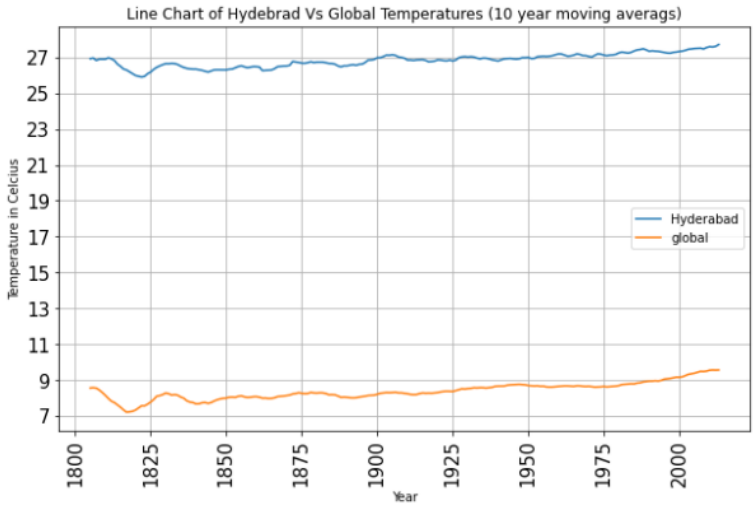
Plotting moving averages:

Matplotlib is the library I have utilized for plotting the below diagram

Source for plotting: https://www.youtube.com/watch?v=-_xM5oKECUg

```
plt.figure(figsize=(10,6))
x1=mov_avg['year']
y1=mov_avg['city_avg_temp']
y2 = mov_avg['global_avg_temp']
plt.xticks(np.arange(1750, 2015, 25),fontsize=15, fontweight=10, rotation='vertical')
plt.yticks(np.arange(1,30,2), fontsize=15, fontweight=10)
plt.grid()
plt.plot(x1, y1, label="Hyderabad")
plt.plot(x1,y2, label='global')
plt.plot
plt.legend(loc='best')
plt.xlabel('Year')
plt.ylabel('Temperature in Celcius')
plt.title("Line Chart of Hydebrad Vs Global Temperatures ({} year moving averags)".format(roll_window_size))
```

```
Text(0.5, 1.0, 'Line Chart of Hydebrad Vs Global Temperatures (10 year moving averags)')
```



Observations:

1. The min and max temperatures are 25.9° C, 0.47 ° C and 27.71 ° C, 9.55 ° C respectively for city and global
2. The mean and standard deviations are 26.83, 8.37 and 0.37 and 0.47 for city and global respectively
3. From the above picture, the global and city temperatures had dropped in early 1820's
4. The city's average temperatures are comparatively higher than the average global temperatures
5. The average city's temperature lies between 26 to 28 degrees overall
6. The average global temperature lies between 7 to 10.5 degrees overall

7. The Hyderabad city temperature have positive correlation with global temperature
8. The temperatures tend to increase from 1990 approximately

```
mov_avg.describe()
```

53]:

	year	city_avg_temp	global_avg_temp
count	209.000000	209.000000	209.000000
mean	1909.000000	26.839856	8.376301
std	60.477268	0.379773	0.472061
min	1805.000000	25.906000	7.203000
25%	1857.000000	26.532000	8.070000
50%	1909.000000	26.865000	8.288000
75%	1961.000000	27.105000	8.658000
max	2013.000000	27.719000	9.556000