Data Analyst Nanodegree

Name: Christopher Charles-Noriega
Due Date: 8/29/2021

# Project 1: Explore Weather Trends

---

## Overview

This project aims to examine local and worldwide meteorological data by comparing air temperatures trends from one specific location to the overall global temperature trends. The goal is to create visualizations illustrating temperature trends and providing a written analysis of the similarities and differences based on local and global results.

## Objectives

- Investigate and extract data from a database using Structured Query Language (SQL).
- Manipulate data using data analysis tools and techniques to compute the moving averages.
- Create a time-series visualization to communicate the trends and comparisons between data.
- Conducts and document multiple observations based on the statistics the data visualization provides.

## Tools

- SQL
- Jupyter Notebook
- Python
- Microsoft Word

---

# Methodology

## Extract Data from Database

Provided Datasets:

- ***city_list*** – table contains a list of countries and cities.
- ***city_data*** – table contains a list of cities and their annual average temperatures (°C).
- ***global_data*** – table contains the annual average temperatures (°C).

### 1. Write SQL statements to investigate data tables in database schema

Since the '***city_list***' table only contains a list of counties and cities, the other two data tables are only needed for this task:

- **SELECT** * **FROM** city_data
- **SELECT** * **FROM** global_data

### 2. Write an SQL Statement to Query Multiple Tables

Rather than downloading two separate CSV files, writing an SQL statement to query both tables to return only the columns we are interested would significantly simplify the data extraction and later analysis processes.

We begin by selecting the '***year***' and '***avg_temp***' columns from the '***global_data***' table, and the '***city***' and '***avg_temp***' columns from the '***city_data***' table. It is important to note that creating an alias for both datasets '***avg_temp***' columns is crucial for distinguishing between the local and global average temperature values during the analysis processes. In addition, creating aliases for both table names would yield a more concise and readable SQL statement. This is done using the '**AS**' keyword.

- **SELECT** global.year, global.avg_temp **AS** global_avg_temp, city.city, city.avg_temp **AS** local_avg_temp
- **FROM** city_data **AS** city

Since both the ***'city data***' and '***global data***' tables have two related columns ('***year***' and '***avg_temp***'), utilizing the **SQL INNER JOIN** keyword would retrieve records with matching values in both tables. In this instance, '**INNER JOIN**' would be the optimal method for executing this task. Since the values in the *'**avg_temp***' column of both tables do not match, the '***year***' column would be the only viable option.

- **INNER JOIN** global_data **AS** global **ON** city.year **=** global.year

We conclude the SQL statement by applying a filter on the targeted city and average temperature values in the '***city_data***' table. The '**WHERE**' clause returns only the specified records.

- **WHERE** city.city = 'San Diego';

### 3. Download Results to CSV file

- Finally, we analyze the output and download the data as a CSV file for further analysis.

## Explore Data using Pythonic Methods

### 1. Start a New Project in Jupyter Notebook and Import Python Libraries (Pandas, Matplotlib)

- **Pandas**: the most widely used Python library used for data science/data analysis and machine learning tasks.

- **Matplotlib**: a multi-platform, data visualization and graphical plotting library and its numerical extension NumPy.

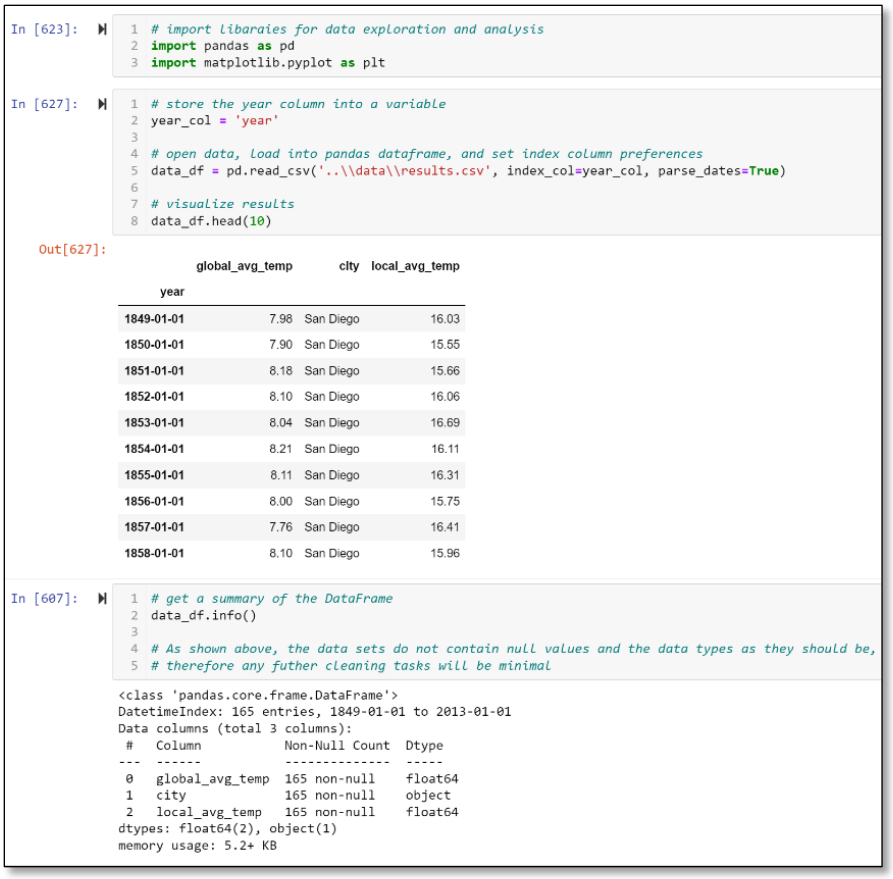
### 2. Open CSV File and Create a DataFrame to Display Data for Inspection

#### Functions

- **.read_csv()**: pandas loads data into a DataFrame.
- **.head()**: pandas outputs the first 5 records unless specified otherwise.
- **.info()**: pandas outputs a summary of DataFrame.

#### Parameters

- **index_col**: pandas immediately sets the index as the date.
- **parse_dates**: pandas parses the index as a date if passed as True.



```
In [623]:
# import libaraies for data exploration and analysis
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [627]:
# store the year column into a variable
year_col = 'year'

# open data, load into pandas dataframe, and set index column preferences
data_df = pd.read_csv('..\\data\\results.csv', index_col=year_col, parse_dates=True)

# visualize results
data_df.head(10)
```

Out[627]:

| year | global_avg_temp | city | local_avg_temp |
|---|---|---|---|
| 1849-01-01 | 7.98 | San Diego | 16.03 |
| 1850-01-01 | 7.90 | San Diego | 15.55 |
| 1851-01-01 | 8.18 | San Diego | 15.66 |
| 1852-01-01 | 8.10 | San Diego | 16.06 |
| 1853-01-01 | 8.04 | San Diego | 16.69 |
| 1854-01-01 | 8.21 | San Diego | 16.11 |
| 1855-01-01 | 8.11 | San Diego | 16.31 |
| 1856-01-01 | 8.00 | San Diego | 15.75 |
| 1857-01-01 | 7.76 | San Diego | 16.41 |
| 1858-01-01 | 8.10 | San Diego | 15.96 |

```
In [607]:
# get a summary of the DataFrame
data_df.info()

# As shown above, the data sets do not contain null values and the data types as they should be,
# therefore any futher cleaning tasks will be minimal
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 165 entries, 1849-01-01 to 2013-01-01
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   global_avg_temp  165 non-null    float64
 1   city             165 non-null    object
 2   local_avg_temp   165 non-null    float64
dtypes: float64(2), object(1)
memory usage: 5.2+ KB
```

During the data extraction phase, the SQL methods reduced the Pythonic steps that would have been required to arrange the Global and Local Average Temperature data if they were opened as two separate CSV files. The image indicates the data is ready to be manipulated and analyzed.

## 3. Convert the Values for Global and Local Average Temperatures from Celsius to Fahrenheit by Deploying Python Lambda Function (Pandas)

### Functions

- **.assign()**: pandas assigns new columns to a DataFrame.
- **lambda**: allows immediate pass without defining a name when used inside another. function while taking any number of arguments in the form of expressions
- **.round()**: if an integer is given, pandas rounds each column to the same number of decimal places.

In [608]:

```
# convert temperature scale from Celsius to Fahrenheit
data_df = data_df.assign(global_avg_temp = lambda x: (9/5)*x['global_avg_temp']+32)
data_df = data_df.assign(local_avg_temp = lambda x: (9/5)*x['local_avg_temp']+32)

# round all values to the nearest tenth decimal place
data_df = data_df.round(1)

# visualize results from the minimum & maximum year
data_df
```

Out[608]:

| year | global_avg_temp | city | local_avg_temp |
|---|---|---|---|
| 1849-01-01 | 46.4 | San Diego | 60.9 |
| 1850-01-01 | 46.2 | San Diego | 60.0 |
| 1851-01-01 | 46.7 | San Diego | 60.2 |
| 1852-01-01 | 46.6 | San Diego | 60.9 |
| 1853-01-01 | 46.5 | San Diego | 62.0 |
| ... | ... | ... | ... |
| 2009-01-01 | 49.1 | San Diego | 62.7 |
| 2010-01-01 | 49.5 | San Diego | 61.1 |
| 2011-01-01 | 49.1 | San Diego | 61.3 |
| 2012-01-01 | 49.1 | San Diego | 63.0 |
| 2013-01-01 | 49.3 | San Diego | 62.9 |

165 rows × 3 columns

I chose to convert the temperature scale from Celsius to Fahrenheit since this scale is most used in the United States.

# Step 3: Data Analysis and Visualization

## 1. Calculate the Global and Local Moving Average Temperatures (Pandas)

### Functions

- **.rolling()**: pandas calculates and returns the rolling window over a specified column
- **.mean()**: pandas calculates the and returns the mean of the values over a specified axis.
- **.rolling().mean()**: pandas calculates and returns the rolling mean of previous periods in a time series if a number is specified.
- **.plot()**: pandas plots a DataFrame.

### Parameters

- **window**: the size of a moving window for a fixed number of observations when an integer is passed.
- **min_periods**: minimum requirement for number of observations in window.

In [638]:

```
# calculate the 30-year global & local moving average w/ minimum period of 1
df_global_range["MA_30"] = df_global_range[global_col].rolling(window=30, min_periods=1).mean()
df_local_range["MA_30"] = df_local_range[local_col].rolling(window=30, min_periods=1).mean()

# round all values to the nearest tenth decimal place
df_global_range = df_global_range.round(1)
df_local_range = df_local_range.round(1)
```

In [642]:

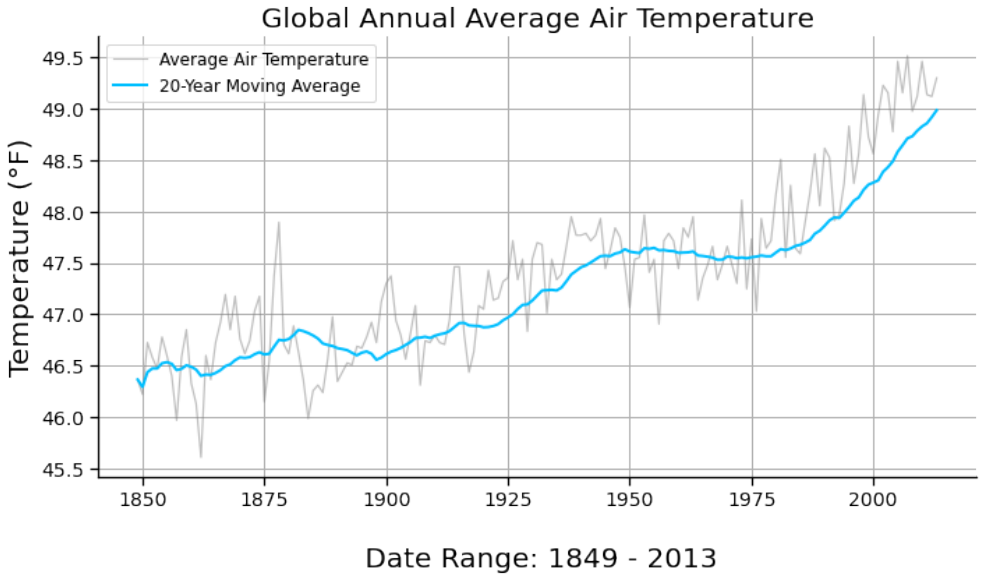
```
# plot temperature data to visualize results

# set the line color preferences
colors = ['silver', 'deepskyblue']

# initiate line plot
df_global_range.plot(color=colors, linewidth=2, figsize=(12,6))

# set line plot visual preferences
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend(labels =['Average Air Temperature', '30-Year Moving Average'], fontsize=14)

# set line plot title and labels
plt.title('Global Annual Average Air Temperature', fontsize=20)
plt.xlabel('\n Date Range: 1849 - 2013', fontsize=16)
plt.ylabel('Temperature (°F)', fontsize=16)
```

In [640]:

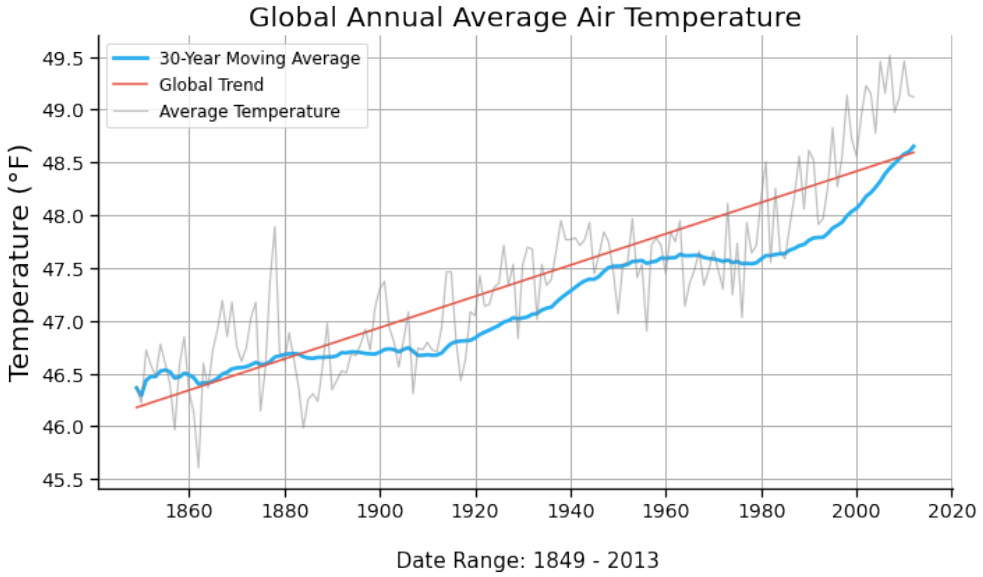
```
# plot temperature data to visualize results

# set the line color preferences
colors = ['silver', 'tomato']

# initiate line plot
df_local_range.plot(color=colors, linewidth=2, figsize=(12,6))

# set line plot visual preferences
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend(labels =['Average Air Temperature', '30-Year Moving Average'], fontsize=14)

# set line plot title and labels
plt.title('San Diego, California \n Annual Average Air Temperature', fontsize=20)
plt.xlabel('\n Date Range: 1849 - 2013', fontsize=16)
plt.ylabel('Temperature (°F)', fontsize=16)
```

I created two new columns for the moving averages by calculating the global and local 30-year moving averages using the '**.rolling().mean()**' function by passing a window size of 30 with a minimum number of periods of 1. Moving averages improve information content in graphs by smoothing out data which helps discover long-term anomalies.

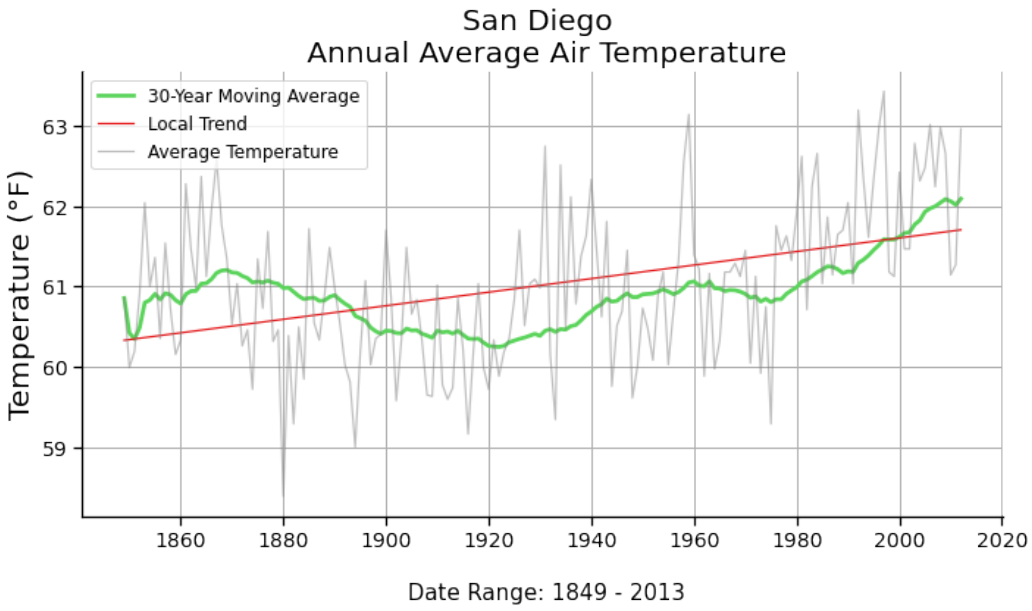
## 2. Create a Time-Series Line Charts Based on the Local and Global Moving Average Temperatures (Pandas, Matplotlib)




I created a line chart by plotting the global average air temperatures over the base period (1849-2013) to visualize peaks and valleys. In addition, I calculated and added the 20-year global moving average to help with identifying variations that have occurred over time.




I modified the moving window size to 30 periods to increase the smoothing, making detecting temperature anomalies easier. Even better, adding a regression line to the chart helps with visualizing how annual temperatures are trending.




I created a line chart by plotting the average local (San Diego) air temperatures over the base period (1849-2013), so significant valleys and spikes in air temperature changes can be observed. In addition, to help visualize how the annual local air temperature is trending, I plotted a smoother line representing the 30-year moving average and a regression line.

# Observations

**Question 1:**

Is your city hotter or cooler on average compared to the global average? Has the difference been consistent over time?

**Answer 1:**

When observing the base period (1984-2013) for global and local average air temperatures, the long-term moving average over 30 years indicates that San Diego *is significantly warmer than the global average*. As of 2013, the global average temperature sits at 48.7°F, whereas the annual average temperature for San Diego is 62.1°F, indicating that, on average, San Diego is 13.4°F warmer than the global average. It is important to note that the records for global temperatures account for polar regions and all the world's sea surfaces, which are significantly cooler than surface air temperatures in regions that exhibit seasonal changes yearly. The line charts exemplify this, as warming substantially differs when you compare the entire world to a

fraction of its landmass. Overall, the local to global variations have been consistent time. However, observing San Diego's line chart, there are several noticeable differences in the city's annual temperatures compared to global records. In the 1880s, San Diego's annual temperature dropped below the global record by 11.8°F. Then again about one hundred years later, in the 1970s, which was even lower, a recorded 11.6°F difference.

**Question 2:**

How do the changes in your city's temperatures over time compare to the changes in the global average?

**Answer 2:**

When comparing the two-line charts, the enormous increases in San Diego's average temperature between the early 1860s and the late 1940s-1950s stand out the most. Surprisingly, these shifts take place roughly 100 years apart. Between 1859 and 1868, the average temperature in San Diego changed substantially. Between 1861 and 1862, the average rose from 60°F to 62.3°F, and it appears that the average continued to rise and fall by 1 degree over the following years, eventually falling to its lowest average temperature on record. Until the outbreak of World War II, San Diego's average temperatures remained well below the trend line.

In comparison, the global average appears to have mostly risen during the years San Diego declined. In 1876, the global average temperature reached 47.9°F, becoming the warmest year of that century. Both local and global averages remained stagnant yet relatively low throughout the following 40 years.

**Question 3:**

**What does the overall trend look like?**

**Answer 3:**

The overall trends in local and global temperatures appear to increase from start to finish. Since the Industrial Revolution, much of the world has become modernized as technological advances and population growth allowed nations to expand way beyond their borders. Rapid global expansion, extraction, greenhouse gas emissions, and wartime had increased global economic growth at the expense of the physical environment leading to an alarming upward trend in global average temperatures over space and time. Both line charts exemplify this, as temperatures have been trending upward since the late 1880s.

**Question 4:**

**Is the world getting hotter or cooler? Has the trend been consistent over the last few hundred years?**

**Answer 4:**

Indeed, the trend has remained consistent over the last one hundred years, but only up until the 21st century. Observing San Diego's line chart, we can see that since 2000, air temperatures have risen significantly, well above the 30-year average. If we compare this to the global line chart, we can see that the 30-year average line also exceeds the trend line about a decade after. Therefore, when we consider our previous observations, we can conclude that the primary reason for the rapid rise in local average temperatures is due to global anthropogenic warming.

**References**


https://www.sqltutorial.org/

https://www.w3schools.com/sql/default.asp

https://www.datacamp.com/workspace/templates/recipe-python-time-series-shifts

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html

https://www.dallasfed.org/research/basics/moving.aspx

https://towardsdatascience.com/moving-averages-in-python-16170e20f6c

https://matplotlib.org/

https://www.tutorialspoint.com/write-a-program-in-python-pandas-to-convert-a-dataframe-celsius-data-column-into-fahrenheit

https://scikit-learn.org/stable/modules/linear_model.html

https://earthobservatory.nasa.gov/world-of-change/global-temperatures