Project1_Exploring_Weather_Trends

February 24, 2021

1 Exploring Weather Trends - Project Instructions

1.0.1 Summary

In this project, you will analyze local and global temperature data and compare the temperature trends where you live to overall global temperature trends.

1.0.2 Instructions

Your goal will be to create a visualization and prepare a write up describing the similarities and differences between global temperature trends and temperature trends in the closest big city to where you live. To do this, you'll follow the steps below:

- Extract the data from the database. There's a workspace in the previous section that is connected to a database. You'll need to export the temperature data for the world as well as for the closest big city to where you live. You can find a list of cities and countries in the `city_list` table. To interact with the database, you'll need to write a SQL query.
  - Write a SQL query to extract the city level data. Export to CSV.
  - Write a SQL query to extract the global data. Export to CSV.
- Open up the CSV in whatever tool you feel most comfortable using. We suggest using Excel or Google sheets, but you are welcome to use another tool, such as Python or R.
- Create a line chart that compares your city's temperatures with the global temperatures. Make sure to plot the *moving average* rather than the yearly averages in order to smooth out the lines, making trends more observable (the last concept in the previous lesson goes over how to do this in a spreadsheet).
- Make observations about the similarities and differences between the world averages and your city's averages, as well as overall trends. Here are some questions to get you started.
  - Is your city hotter or cooler on average compared to the global average? Has the difference been consistent over time?
  - "How do the changes in your city's temperatures over time compare to the changes in the global average?"
  - What does the overall trend look like? Is the world getting hotter or cooler? Has the trend been consistent over the last few hundred years?

1.0.3 Submission

Your submission should be a PDF that includes:

- An outline of steps taken to prepare the data to be visualized in the chart, such as:
  - What tools did you use for each step? (Python, SQL, Excel, etc)

- How did you calculate the moving average?
  - What were your key considerations when deciding how to visualize the trends?
- Line chart with local and global temperature trends
- At least four observations about the similarities and/or differences in the trends

1.0.4 Rubric

A Udacity reviewer will assess your project based on the criteria in the project rubric. Use the rubric as a guide while you complete the project, then give yourself a quick self-assessment before you submit it.

If you're on this page, then you should have completed your exploration of local and global temperature trends. Congratulations! Before you submit your project, make sure to check the following points:

- Please submit your project as a PDF. Your report should include documentation of the steps in your analysis, a line chart depicting the local and global temperature data, and your observations regarding the trends.
- Don't forget to review the rubric! Reviewers will use the rubric to assess your work, so make sure it 'meets specifications' on all points before you submit.

Once you've checked the above, click on the "Submit Project" button below to go to the project submission page. After you submit your project, it can take up to a week for it to be evaluated. Most of the time, it is much faster! In the meantime, you can feel free to continue to other parts of the program to continue your learning!

2 Project 1 - Larijohn Adorable

2.1 Tools Used

- Jupyter Lab, python3, pandas, seaborn, matplotlib, numpy, PostgresSQL

2.2 1. Extract the data using SQL and exporting it to a CSV

- I was able to import the csv's into my own local postgres database
- Use SQL to dump the tables city_data, city_list, global_data into a csv. I also imported it into my own local db in case I needed to do some EDA (Exploratory Data Analysis) from there

[67]:
```
# Extract and Load data into PostgresSQL Database as well as Pandas Dataframes
from sqlalchemy import create_engine
import pandas as pd
import os

db = os.environ.get('LDB')
engine = create_engine(db)

# read csvs into pandas
```

```
cd = pd.read_csv('city_data.csv')
cl = pd.read_csv('city_list.csv')
gd = pd.read_csv('global_data.csv')

# create and insert tables into my postgresdb
#cd.to_sql('city_data', engine)
#cl.to_sql('city_list', engine)
#gd.to_sql('global_data', engine)
```

[11]:
```
# Here is an example of how I could query the database from jupyter lab
city_data_s = %sql $db select * from city_data
city_list_s = %sql $db select * from city_list
global_data_s = %sql $db select * from global_data

cd = city_data_s.DataFrame()
cl = city_list_s.DataFrame()
gl = global_data_s.DataFrame()
```

```
70792 rows affected.
342 rows affected.
266 rows affected.
```

[68]:
```
# Add city_id as a column, this will be merged into the city_data dataframe.␣
↪This was performed because it is easier to reference and group by a city,␣
↪country combination. City names are not unique globally
cl['city_id'] = cl.index
cl = cl.reindex(columns=['city_id', 'city','country'])
print(cl.head(3))
print(cl.shape)
```

```
   city_id        city               country
0        0     Abidjan         Côte D'Ivoire
1        1   Abu Dhabi  United Arab Emirates
2        2       Abuja               Nigeria
(342, 3)
```

[69]:
```
# Afterwards, merge city_id column into city_data (cd) dataframe
cd = pd.merge(cl, cd, left_on=['city','country'],right_on=['city','country'])
```

[70]:
```
# The list below demonstrates how city names are not unique globally.␣
↪Therefore, I needed to add the city_id column
city_value_counts = cl.city.value_counts()
city_value_counts[city_value_counts > 1]
```

[70]:
```
Santiago         3
Hyderabad        2
Santo Domingo    2
```

```
London          2
Barcelona       2
Alexandria      2
Colombo         2
Los Angeles     2
Kingston        2
Birmingham      2
La Paz          2
Valencia        2
Name: city, dtype: int64
```

[71]:
```
# Next I added an avg_temp_f (Fareinheit) column, since I am not familiar with␣
 ↪Celius as an American
# and find it easier to compare temperatures across the world, to a temperature␣
 ↪classification medium I am familiar with
def convertC_to_F(c_temp):
     return c_temp * (9/5)+32

cd['avg_temp_f'] = cd.avg_temp.dropna().apply(lambda x : convertC_to_F(x))
gd['avg_temp_f'] = gd.avg_temp.dropna().apply(lambda x : convertC_to_F(x))
print(cd.head(3))
print(gd.head(3))
```

```
   city_id     city        country  year  avg_temp  avg_temp_f
0        0  Abidjan  Côte D'Ivoire  1849     25.58      78.044
1        0  Abidjan  Côte D'Ivoire  1850     25.52      77.936
2        0  Abidjan  Côte D'Ivoire  1851     25.67      78.206
   year  avg_temp  avg_temp_f
0  1750      8.72      47.696
1  1751      7.98      46.364
2  1752      5.78      42.404
```

2.2.1 Next, in order to calculate moving averages data per city, I performed the steps below.

1. Grouped City Data by City_ID
2. For each city_group, calculate the rolling averages (10 year and 25 year) as set them as new columns in the temporary cg dataframe, which is just the grouped city dataframe
3. Merged the rows of the temporary cg dataframe back into cd using loc, such that I only update the columns which are part of that row/index subset

[72]:
```
# The difference between my solution above and Sertac's solution is that the␣
 ↪transform allows to run an apply on a group of dataframe rows, where as I␣
 ↪was not familiar with the
# transform function.
```

```
## Vectorized solution provided by Sertac Ozker https://knowledge.udacity.com/
↪questions/500117
cd.sort_values(by=['city_id', 'year'], inplace=True)

cd['MA10_C'] = cd.groupby('city_id')['avg_temp'].transform(lambda x: x.
↪rolling(10,1).mean())
cd['MA10_F'] = cd.groupby('city_id')['avg_temp_f'].transform(lambda x: x.
↪rolling(10,1).mean())
cd['MA25_C'] = cd.groupby('city_id')['avg_temp'].transform(lambda x: x.
↪rolling(25,1).mean())
cd['MA25_F'] = cd.groupby('city_id')['avg_temp_f'].transform(lambda x: x.
↪rolling(25,1).mean())

# Perform moving average on global average dataset as well
gd['MA10_C'] = gd.avg_temp.rolling(10,1).mean()
gd['MA25_C'] = gd.avg_temp.rolling(25,1).mean()
gd['MA10_F'] = gd.avg_temp_f.rolling(10,1).mean()
gd['MA25_F'] = gd.avg_temp_f.rolling(25,1).mean()
```

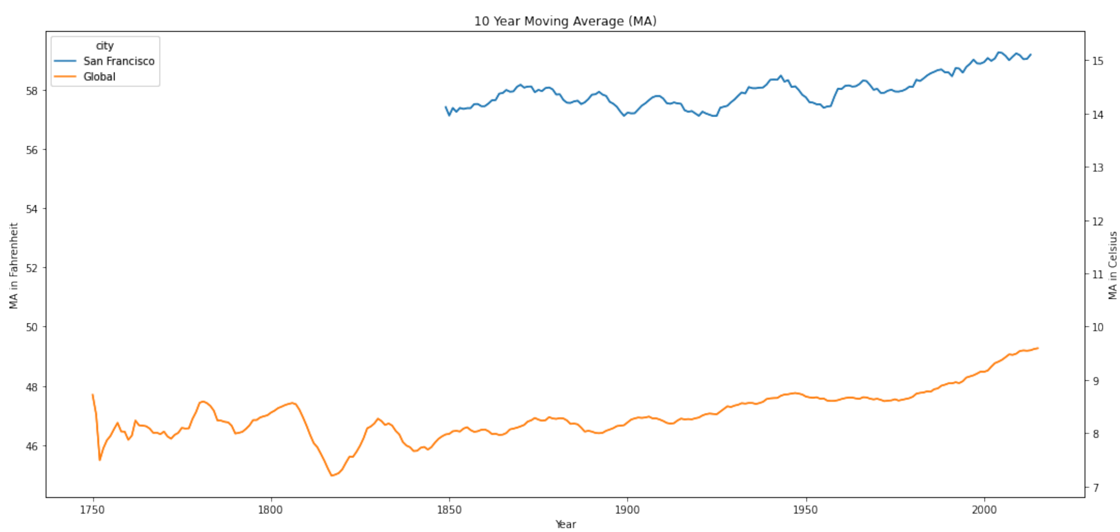
```
# Method below deprecated by cell above, answered by Sertac Ozker, Udacity DA mentor
for index, cg in cdgb:
    cg['MA10_C'] = cg[cg.city_id == cg.city_id].avg_temp.rolling(window=10).mean()
    cg['MA25_C'] = cg[cg.city_id == cg.city_id].avg_temp.rolling(window=25).mean()
    cg['MA10_F'] = cg[cg.city_id == cg.city_id].avg_temp_f.rolling(window=10).mean()
    cg['MA25_F'] = cg[cg.city_id == cg.city_id].avg_temp_f.rolling(window=25).mean()
    cd.loc[cg.index, 'MA10_C'] = cg['MA10_C']
    cd.loc[cg.index, 'MA25_C'] = cg['MA25_C']
    cd.loc[cg.index, 'MA10_F'] = cg['MA10_F']
    cd.loc[cg.index, 'MA25_F'] = cg['MA25_F']

# Calculate Moving Averages for Global Data below
gd['MA10_C'] = gd.avg_temp.rolling(window=10).mean()
gd['MA25_C'] = gd.avg_temp.rolling(window=25).mean()
gd['MA10_F'] = gd.avg_temp_f.rolling(window=10).mean()
gd['MA25_F'] = gd.avg_temp_f.rolling(window=25).mean()
```

3 Graph 1: Plot of Graph San Francisco vs Global Weather Trends

[74]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

a4_dims = (18, 8.27)
fig, ax = plt.subplots(figsize= a4_dims)

# Graph San Francisco vs Global Weather Trends
```

```
city = 'San Francisco'
df = cd[cd.city == 'San Francisco']
mdf = gd
mdf['city'] ='Global'
mdf = pd.concat([df,gd])

plt.title("10 Year Moving Average (MA)")

plt.xlabel('Year')
plt.ylabel('MA in Fahrenheit')
sns.lineplot(x = mdf.year
             , y = mdf.MA10_F
             , data=mdf
             , ci=None
             , color='b'
             , hue='city'
            )
ax2 = plt.twinx()
plt.ylabel('MA in Celsius')
sns.lineplot(x = mdf.year
             , y = mdf.MA10_C
             , data=mdf
             , ci=None
             , color='r'
             , hue='city'
            )
plt.show()
```



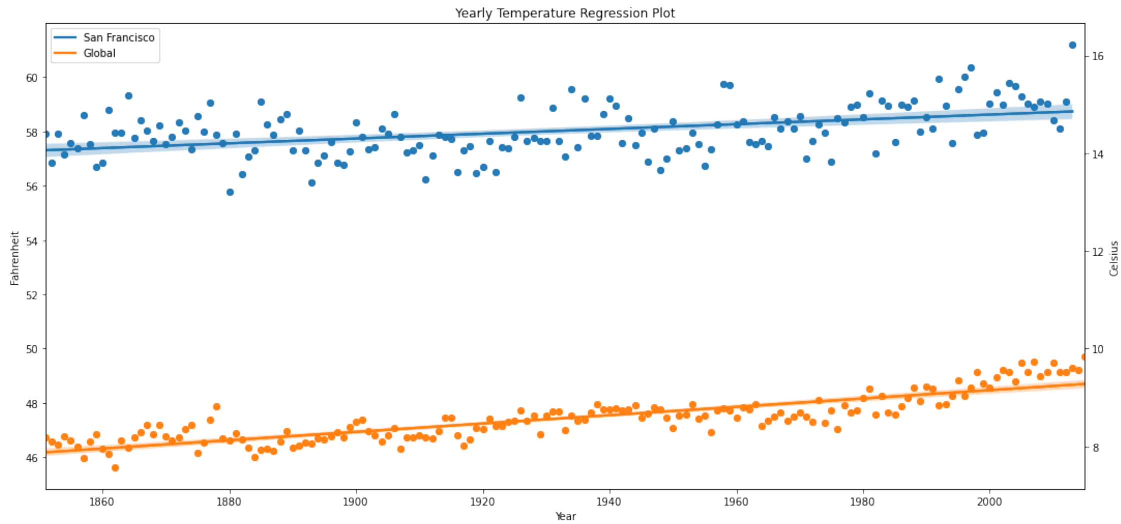
3.0.1 Insights on Graph 1

- My city, San Francisco is warmer than the global average
- Since approximately 1925, both San Francisco and world's temperatures have increased on average year-by-year

To provide further evidence, I decided to use a Seaborn regression plot to fit the data trends onto a line and to provide further confidence that the temperature is increasing

3.1 Graph 2 : Plot a regression plot for San Francisco vs Global Trends

[75]:
```
fig, ax = plt.subplots(figsize= a4_dims)

city = 'San Francisco'

df = cd[(cd.city == 'San Francisco') & (cd.year > 1850)]

mdf['city'] = 'Global'
mdf_gt_1900 = gd[gd.year > 1850]

plt.title("Yearly Temperature Regression Plot")
sns.regplot(x = df.year
             , y = df.avg_temp_f
             , data=df )
sns.regplot(x = mdf_gt_1900.year
             , y = mdf_gt_1900.avg_temp_f
             , data=mdf_gt_1900 )

plt.ylabel('Fahrenheit')
plt.xlabel('Year')
ax2 = ax.twinx()
sns.regplot(x = df.year
             , y = df.avg_temp
             , data=df )
sns.regplot(x = mdf_gt_1900.year
             , y = mdf_gt_1900.avg_temp
             , data=mdf_gt_1900 )

plt.ylabel('Celsius')
plt.legend([city, 'Global'])
plt.show()
```



The plot above clearly shows that both San Francisco and the Global temperature has been rising for the last couple hundred years.

Also, it shows that the differerence between both San Francisco and the Global average temperatures, year-by-year, remain generally the same, however, they tend to slight be more similar in more recent years.

3.1.1 Next, I will get all the 'best cities' in the world from wikipedia to compare the same findings across many of the 'best cities'

Note that the best cities are defined by the following criteria as stated in Wikipedia: Global City

- A variety of international financial services,[10] notably in finance, insurance, real estate, banking, accountancy, and marketing
- Headquarters of several multinational corporations
- The existence of financial headquarters, a stock exchange, and major financial institutions
- Domination of the trade and economy of a large surrounding area
- Major manufacturing centres with port and container facilities
- Considerable decision-making power on a daily basis and at a global level
- Centres of new ideas and innovation in business, economics, culture, and politics
- Centres of media and communications for global networks
- Dominance of the national region with great international significance
- High percentage of residents employed in the services sector and information sector
- High-quality educational institutions, including renowned universities, international student attendance,[11] and research facilities
- Multi-functional infrastructure offering some of the best legal, medical, and entertainment facilities in the country

- High diversity in language, culture, religion, and ideologies

[76]:
```
best_cities = pd.read_html('https://en.wikipedia.org/wiki/
↪Global_city#The_World\'s_Best_Cities')
best_cities[0].City
```

[76]:
```
0          New York
1            London
2             Tokyo
3         Hong Kong
4             Paris
5         Singapore
6       Los Angeles
7          Shanghai
8           Beijing
9             Seoul
10          Chicago
11    San Francisco
12          Toronto
13           Sydney
14           Berlin
15           Boston
16           Moscow
17        Amsterdam
18            Dubai
19         Istanbul
Name: City, dtype: object
```

The list of cities below are the cities found both in the city_list dataset and the world cities list from wikipedia

- I noticed the following cities are missing
- Hong Kong
- Dubai
- Beijing

[77]:
```
best_cities = cl[cl.city.isin(best_cities[0].City.to_list())]
best_cities
```

[77]:
```
     city_id         city         country
15        15    Amsterdam     Netherlands
42        42       Berlin         Germany
48        48       Boston   United States
69        69      Chicago   United States
132      132     Istanbul          Turkey
173      173       London          Canada
174      174       London  United Kingdom
176      176  Los Angeles           Chile
```

```
177       177    Los Angeles   United States
211       211         Moscow          Russia
224       224       New York   United States
237       237          Paris          France
275       275  San Francisco   United States
287       287          Seoul     South Korea
288       288       Shanghai           China
292       292      Singapore       Singapore
300       300         Sydney       Australia
311       311          Tokyo           Japan
312       312        Toronto          Canada
```

[78]:
```
# Since there are two Los Angeles and two London cities in the world, I need to␣
↪remove their duplicates from this list
best_cities = best_cities[~best_cities.city_id.isin([176,173])].
↪sort_values(by='city')
best_cities
```

[78]:
```
     city_id           city         country
15        15      Amsterdam     Netherlands
42        42         Berlin         Germany
48        48         Boston   United States
69        69        Chicago   United States
132      132       Istanbul          Turkey
174      174         London  United Kingdom
177      177    Los Angeles   United States
211      211         Moscow          Russia
224      224       New York   United States
237      237          Paris          France
275      275  San Francisco   United States
287      287          Seoul     South Korea
288      288       Shanghai           China
292      292      Singapore       Singapore
300      300         Sydney       Australia
311      311          Tokyo           Japan
312      312        Toronto          Canada
```

[79]:
```
df = cd[cd.city_id.isin(best_cities.city_id)]
df[['city_id','city','country']].drop_duplicates().sort_values(by=['city'])
```

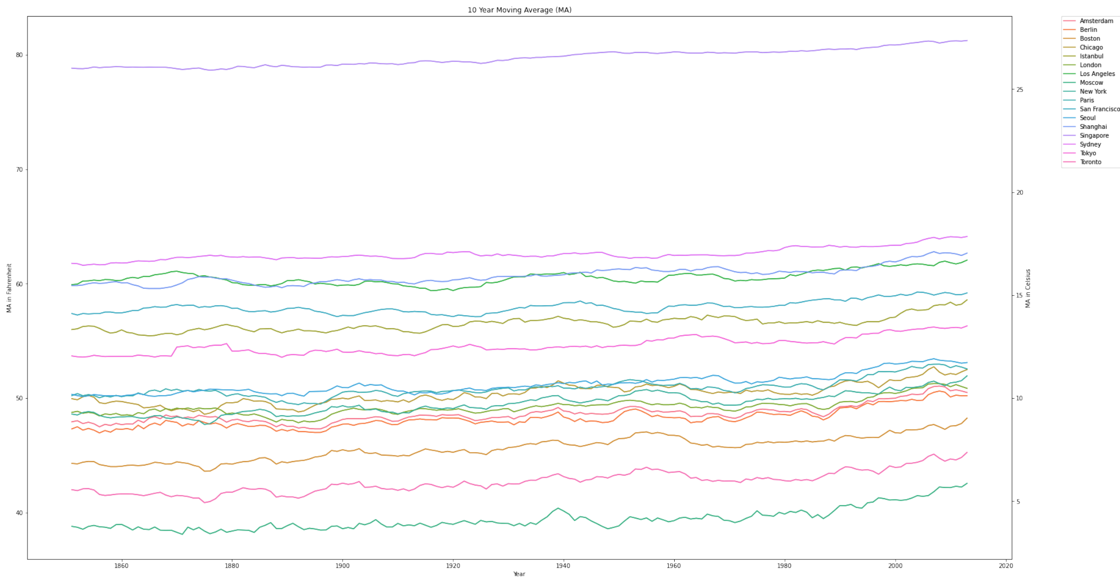
[79]:
```
       city_id           city         country
3050        15      Amsterdam     Netherlands
8990        42         Berlin         Germany
10457       48         Boston   United States
15107       69        Chicago   United States
27633      132       Istanbul          Turkey
36012      174         London  United Kingdom
```

```
36607      177    Los Angeles   United States
43352      211         Moscow          Russia
46168      224       New York   United States
48918      237          Paris          France
56816      275  San Francisco   United States
58999      287          Seoul     South Korea
59174      288       Shanghai           China
59878      292      Singapore       Singapore
61617      300         Sydney       Australia
63890      311          Tokyo           Japan
64059      312        Toronto          Canada
```

4 Graph 3: Plot the 10yr moving averages for all best cities found in city_data

[65]:

```
a4_dims = (30, 17)
fig, ax = plt.subplots(figsize= a4_dims)

df = cd[cd.city_id.isin(best_cities.city_id) &(cd.year > 1850)]

plt.title("10 Year Moving Average (MA)")

plt.xlabel('Year')
plt.ylabel('MA in Fahrenheit')
sns.lineplot(x = df.year
             , y = df.MA10_F
             , data=df
             , ci=None
             , color='b'
             , hue='city'
            )

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
ax2 = plt.twinx()
plt.ylabel('MA in Celsius')
sns.lineplot(x = df.year
             , y = df.MA10_C
             , data=df
             , ci=None
             , color='r'
             , hue='city'
            )
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.show()
```

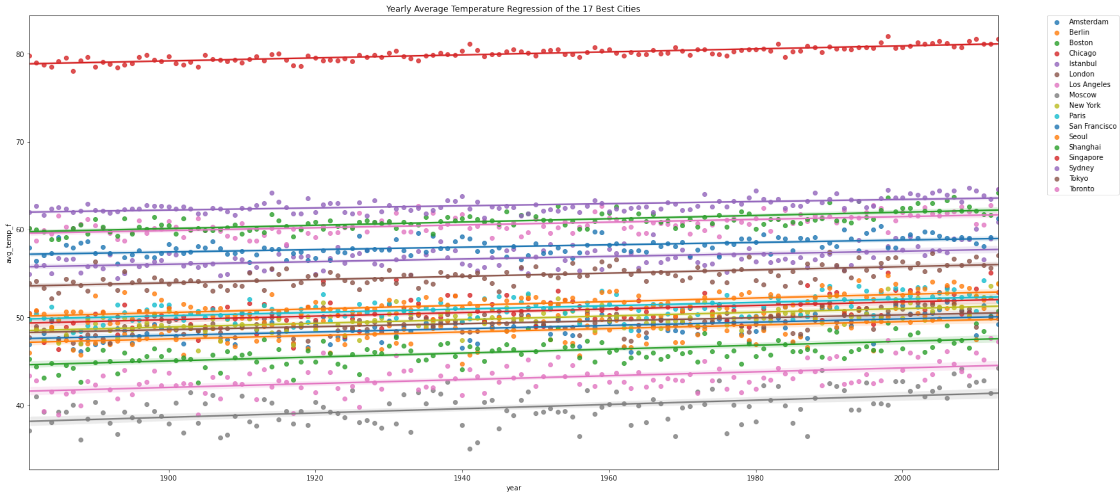


4.0.1 Insights from Graph 3 : Graph the moving averages of all the 'best cities' according to Wikipedia

- All the 17 cities tend to have very similar trending rates throughout time. They all seem to increase progressively. They also tend to not deviate too far from their prior yearly average temps
- Interestingly, the coldest 'best' city is Moscow, Russia, while the warmest is Singapore

5 Graph 4 : Regression Plots for 17 Best Cities

[80]:
```
a4_dims = (25, 12)
fig, ax = plt.subplots(figsize= a4_dims)

df = cd[cd.city_id.isin(best_cities.city_id) &(cd.year > 1880)]
dfgb = df.groupby('city_id')

plt.title('Yearly Average Temperature Regression of the 17 Best Cities')

for city_id, this_df in dfgb:

    city_name = cl[cl.city_id == city_id].city.item()
    this_plot = sns.regplot(x = this_df.year
             , y = this_df.avg_temp_f
             , data=this_df , label=city_name)

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.show()
```



5.0.1 Insights for Graph 4

- The regression plots more clearly show that every 'best city' across the world is increasing in warmth, year after year
- An interesting note is that a large ration of the 'best cities' have temperatures near the Global Average for that given year. Between 46-48 F* / 8-9 C*

The next set of Jupyter Cells call two APIs/Frameworks to generate the following data for additional observations

- geopy; for latitude / longitude information
- pycountry_convert; to get continent info

Since I have generated this previously, we can generate the dataframe from the pre-generated df of city_list_extended.csv

[81]:
```
cl = pd.read_csv('city_list_extended.csv', index_col=0)
```

[40]:
```
# retrieve longitutde and latitudes
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="udacity_p1_ladorable")

for index, city_info in cl.iterrows():
    city = city_info.city
    country = city_info.country
    # city_list country for Congo doesn't retrieve any results from Geopy, so␣
 ↪we must clean/Transform this data
    if country == 'Congo (Democratic Republic Of The)':
        country = 'Congo'
    location = geolocator.geocode("{0},{1}".format(city, country))
```

```
#     print(city,country)
#     print(location.raw['lat'])
#     print(location.raw['lon'])
    cl.loc[index,'lat'] = location.raw['lat']
    cl.loc[index,'lon'] = location.raw['lon']
```

[41]:
```
# retrieve continent info
import pycountry_convert as pc

continents = {
    'NA': 'North America',
    'SA': 'South America',
    'AS': 'Asia',
    'OC': 'Australia',
    'AF': 'Africa',
    'EU': 'Europe'
}

for index, city_info in cl.iterrows():
    # The countries in the if statements below (not else) all have issues, so␣
 ↪they must be interpreted/cleaned
    if(city_info.country == 'Guinea Bissau'):
        country_code = 'GW'
    elif(city_info.country == 'Bosnia And Herzegovina'):
        country_code = 'BA'
    elif(city_info.country == "Côte D'Ivoire"):
        country_code = 'CI'
    elif(city_info.country == "Congo (Democratic Republic Of The)"):
        country_code = pc.country_name_to_country_alpha2('Congo')
    else:
        country_code = pc.country_name_to_country_alpha2(city_info.country)

    continent_name = pc.country_alpha2_to_continent_code(country_code)
    cl.loc[index, 'continent'] = continents[continent_name]
```

[54]:
```
cl.set_index('city_id', inplace=True)
```

[83]:
```
cl.lat = cl.lat.astype('float32')
cl.lon = cl.lon.astype('float32')
```

[56]:
```
# save new city list info
cl.to_csv('city_list_extended.csv')
cl
```

[56]:
```
                     city              country       lat         lon  \
city_id
0                 Abidjan        Côte D'Ivoire  5.320357   -4.016107
```

```
1          Abu Dhabi  United Arab Emirates  24.453835   54.377403
2              Abuja              Nigeria   9.064330    7.489297
3              Accra                Ghana   5.560014   -0.205744
4              Adana               Turkey  36.993618   35.325836
…                  …                    …          …           …
337           Xuzhou                China  34.206657  117.278290
338      Yamoussoukro        Côte D'Ivoire   6.809107   -5.273263
339          Yerevan              Armenia  40.177612   44.512585
340           Zagreb              Croatia  45.813183   15.977178
341          Zapopan               Mexico  20.721121 -103.391365

             continent
city_id
0               Africa
1                 Asia
2               Africa
3               Africa
4                 Asia
…                    …
337               Asia
338             Africa
339               Asia
340             Europe
341      North America

[342 rows x 5 columns]
```

[57]:
```
# Get Average temperatures for ALL contintents and compare it to the global␣
↪average
clcon_gb = cl.groupby('continent')
clcon_gb.groups

contintent_city_ids =  []

for continent, con_df in clcon_gb:
    contintent_dict = {
        'continent' : continent,
        'city_ids' : con_df.index
    }
    contintent_city_ids.append(contintent_dict)

ccdf = pd.DataFrame (contintent_city_ids)
ccdf.set_index('continent', inplace=True)
ccdf.to_csv('continent_city_ids.csv')
ccdf
```

```
[57]:                                                      city_ids
continent
Africa         Int64Index([  0,   2,   3,   9,  11,  18,  20,…
Asia           Int64Index([  1,   4,   6,   7,  12,  13,  14,…
Australia      Int64Index([5, 51, 60, 197, 240, 248, 300, 331…
Europe         Int64Index([ 15,  22,  33,  38,  39,  42,  43,…
North America  Int64Index([  8,  10,  19,  23,  24,  27,  46,…
South America  Int64Index([ 34,  35,  36,  40,  58,  59,  61,…
```

```
[84]: # go through each continent city groups, and generate an average of all their␣
↪(per continent) temperatures by year, save as con_df
ccd_df_list = []
for index, cci in ccdf.iterrows():
    this_df = cd[cd.city_id.isin(cci.city_ids.to_list())]

    con_avg_df = this_df[['avg_temp', 'avg_temp_f', 'MA10_C', 'MA10_F',␣
↪'MA25_F', 'MA25_C', 'year']].groupby('year').mean().reset_index()
    con_avg_df['continent'] = index
    ccd_df_list.append( con_avg_df)
con_df = pd.concat(ccd_df_list)
con_df.to_csv('con_df.csv')
con_df
```

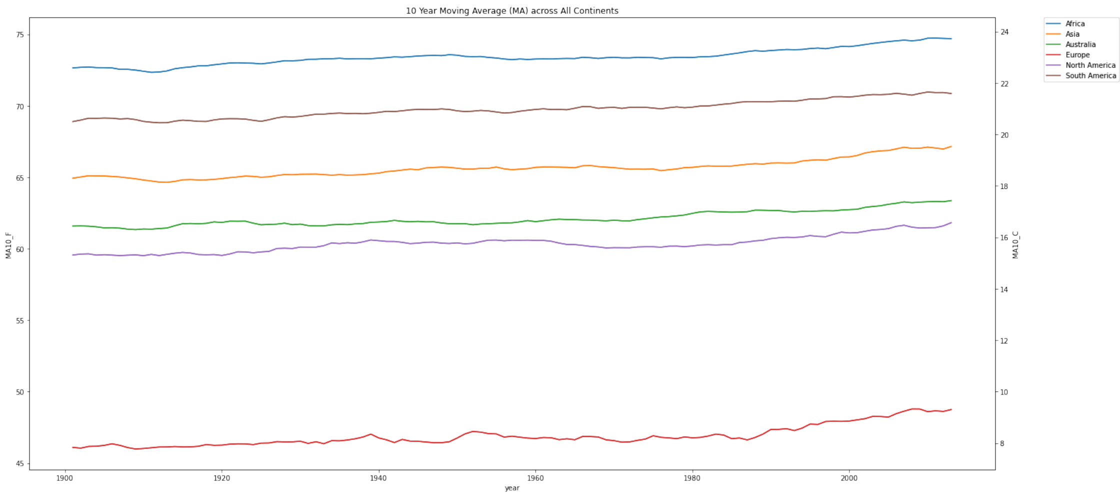
```
[84]:      year   avg_temp  avg_temp_f     MA10_C    MA10_F     MA25_F     MA25_C  \
0     1743  14.720000     58.4960  14.720000  58.49600  58.496000  14.720000
1     1744  19.660000     67.3880  17.190000  62.94200  62.942000  17.190000
2     1745  11.820000     53.2760  15.400000  59.72000  59.720000  15.400000
3     1746        NaN         NaN  15.400000  59.72000  59.720000  15.400000
4     1747        NaN         NaN  15.400000  59.72000  59.720000  15.400000
..     …          …          …         …          …        …          …
185   2009  21.781667     71.2070  21.593800  70.86884  70.656752  21.475973
186   2010  21.739333     71.1308  21.647933  70.96628  70.703696  21.502053
187   2011  21.445667     70.6022  21.628333  70.93100  70.715288  21.508493
188   2012  21.802000     71.2436  21.629433  70.93298  70.724000  21.513333
189   2013  21.332667     70.3988  21.592800  70.86704  70.738928  21.521627

          continent
0            Africa
1            Africa
2            Africa
3            Africa
4            Africa
..              …
185   South America
186   South America
187   South America
188   South America
```

```
189  South America

[1447 rows x 8 columns]
```

6 Graph 5 : Plot Temperature Trends of each continent, for any meaninful observations

[85]:

```
a4_dims = (25, 12)
fig, ax = plt.subplots(figsize= a4_dims)

df = con_df[con_df.year > 1900]

plt.title("10 Year Moving Average (MA) across All Continents")
#plt.xlabel('Year')
#plt.ylabel('MA in Fahrenheit')
sns.lineplot(x = df.year
             , y = df.MA10_F
             , data=df
             , ci=None
             , color='b'
             , hue='continent'
            )
#
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
ax2 = plt.twinx()
#plt.ylabel('MA in Celsius')
sns.lineplot(x = df.year
             , y = df.MA10_C
             , data=df
             , ci=None
             , color='r'
             , hue='continent'
            )
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.show()
```



7 Graph 5 : Insights

- Coldest continent is Europe
- Warmest continent is Africa
- Although there has been a generally upward trend since 1900, the rate has significantly increased after 1980

7.0.1 Hypthoesis: Does latitude near 0 (along the band of the equator) have consistently higher temperatures, than elsewhere?

- For my next and final insight, I hope to figure out the weather it is true that all cities near the equator is hot. In my work after Graph 4 above, I retrieved the latitude and longitude data for every city in the city_data list using external API's and python packages

[86]:
```
# Found cities near the equator by searching for cities near 0 latitude, with a␣
↪4* latitude margin difference to lat == 0
cl_near_equator = cl[(cl.lat < 4)& (cl.lat > -4)]
cl_near_equator
```

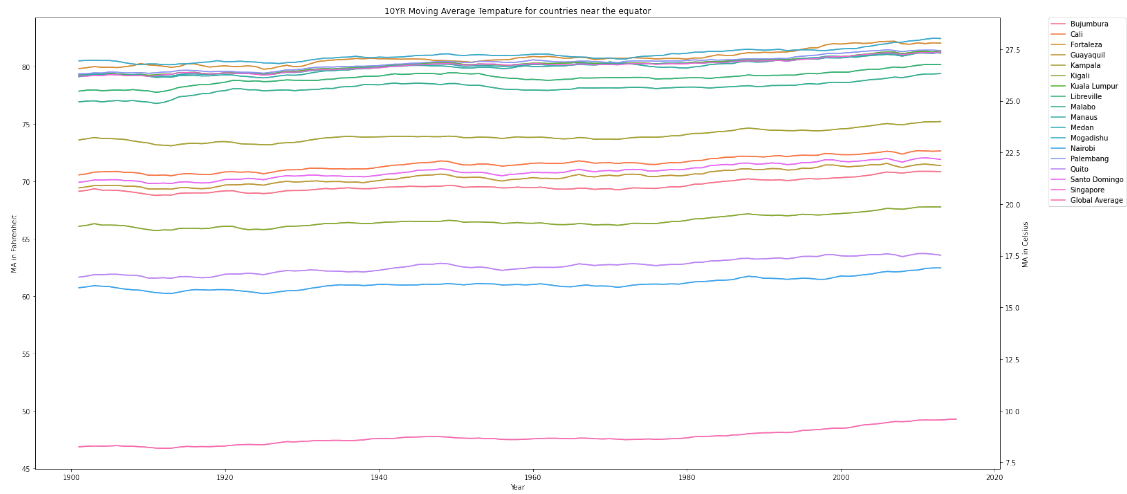
[86]:

city_id	city	country	lat	lon	continent
55	Bujumbura	Burundi	-3.363812	29.367502	Africa
58	Cali	Colombia	3.451792	-76.532494	South America
99	Fortaleza	Brazil	-3.730451	-38.521797	South America
110	Guayaquil	Ecuador	-2.170414	-79.905022	South America
143	Kampala	Uganda	0.317714	32.581352	Africa
154	Kigali	Rwanda	-1.950851	30.061506	Africa
160	Kuala Lumpur	Malaysia	3.151696	101.694237	Asia
168	Libreville	Gabon	0.390002	9.454001	Africa
186	Malabo	Equatorial Guinea	3.752828	8.780061	Africa

```
189          Manaus      Brazil -3.131633  -59.982506  South America
196           Medan   Indonesia  3.589665   98.673828           Asia
206       Mogadishu     Somalia  2.034931   45.341919         Africa
216         Nairobi       Kenya -1.303169   36.826061         Africa
235       Palembang   Indonesia -2.988830  104.756859           Asia
258           Quito     Ecuador -0.220164  -78.512329  South America
283  Santo Domingo     Ecuador -0.340189  -79.171547  South America
292       Singapore   Singapore  1.357107  103.819496           Asia
```

[102]:

```
fig, ax = plt.subplots(figsize= a4_dims)

region = 'Countries Near Equator'
df = cd[cd.city_id.isin(cl_near_equator.index.to_list())]
mdf = gd
mdf['city'] ='Global Average'
mdf = pd.concat([df,gd])
mdf = mdf[mdf.year > 1900]
#city = 'San Francisco'
#df = cd[cd.city == 'San Francisco']
#mdf = gd
#mdf['city'] ='Global'
#mdf = pd.concat([df,gd])

plt.title("10YR Moving Average Tempature for countries near the equator")
plt.xlabel('Year')
plt.ylabel('MA in Fahrenheit')
sns.lineplot(x = mdf.year
             , y = mdf.MA10_F
             , data=mdf
             , ci=None
             , color='b'
             , hue='city'
            )
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
ax2 = plt.twinx()
plt.ylabel('MA in Celsius')
sns.lineplot(x = mdf.year
             , y = mdf.MA10_C
             , data=mdf
             , ci=None
             , color='r'
             , hue='city'
            )
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.show()
```



8 Graph 6 : Insights

* So my original hypthoesis is close to being true. Notice, how the global average is much lower than all the equatorial cities. ##### * However, I found that two cities were not necessary very hot; Quito and Nairobi. This is because I forgot to consider elevation as a factor in determining temperature. Also, humidity, and precipitation was not considered in this dataset. In summary, this generally proves that cities near the equator are generally hot. I could improve the evidence of the claim, if I compare averages of cities across different 'bins' of latitudes. ***

8.0.1 Conclusion

- I found this project very interesting and there are hundreds of more ways the data could be utilized interpretted and combined/juxtaposed with information to provide even more interesting insights.

9 Environment Preperation Steps

[6]:
```
# Set up SQL string
import os
db = os.environ.get('LDB')
```

[7]:
```
# load sql magic
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

[8]:
```

```

```
[ ]:
```