Global Warming Analysis

Analysis by Koome Derrick

In [1]:

```
#importing all the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)
```

In [2]:

```
#bring the dataframe into the jupyter environment
global_temp_country = pd.read_csv('C:/Users/koome/Desktop/Geospatial/Global
global_temp_country.head(5)
```

Out[2]:

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland
4	1744-03-01	NaN	NaN	Åland

Carrying out exploratory data analysis on the data

In [3]:

```
global_temp_country.shape
```

Out[3]:

```
(577462, 4)
```

577,462 rows of data in my dataframe. Interested to know whether there are any missing values in my data

In [4]:

```
global_temp_country.isna().sum()
```

Out[4]:

```
dt                                   0
AverageTemperature               32651
AverageTemperatureUncertainty    31912
Country                              0
dtype: int64
```

Approximately 33,000 rows don't have average temperature. Dropping these rows from my dataframe will not be a big deal since my dataset is still very big.

```
In [5]:  global_temp_country.dropna(axis='index', how='any', subset=['AverageTempera
```

```
In [6]:  global_temp_country.isna().sum()
```

```
Out[6]: dt                               0
        AverageTemperature               0
        AverageTemperatureUncertainty    0
        Country                          0
        dtype: int64
```

Removed all rows with missing values. Now interested to know how many unique countries we have in our dataframe.

```
In [7]:  global_temp_country['Country'].nunique()
```

```
Out[7]: 242
```

Impressive number of countries, but am curious whether some countries have been duplicated.

In [8]:

```
global_temp_country['Country'].unique()
```

```
Out[8]: array(['Åland', 'Afghanistan', 'Africa', 'Albania', 'Algeria',
               'American Samoa', 'Andorra', 'Angola', 'Anguilla',
               'Antigua And Barbuda', 'Argentina', 'Armenia', 'Aruba', 'Asia',
               'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
               'Baker Island', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium',
               'Belize', 'Benin', 'Bhutan', 'Bolivia',
               'Bonaire, Saint Eustatius And Saba', 'Bosnia And Herzegovina',
               'Botswana', 'Brazil', 'British Virgin Islands', 'Bulgaria',
               'Burkina Faso', 'Burma', 'Burundi', "Côte D'Ivoire", 'Cambodia',
               'Cameroon', 'Canada', 'Cape Verde', 'Cayman Islands',
               'Central African Republic', 'Chad', 'Chile', 'China',
               'Christmas Island', 'Colombia', 'Comoros',
               'Congo (Democratic Republic Of The)', 'Congo', 'Costa Rica',
               'Croatia', 'Cuba', 'Curaçao', 'Cyprus', 'Czech Republic',
               'Denmark (Europe)', 'Denmark', 'Djibouti', 'Dominica',
               'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador',
               'Equatorial Guinea', 'Eritrea', 'Estonia', 'Ethiopia', 'Europe',
               'Falkland Islands (Islas Malvinas)', 'Faroe Islands',
               'Federated States Of Micronesia', 'Fiji', 'Finland',
               'France (Europe)', 'France', 'French Guiana', 'French Polynesia',
               'French Southern And Antarctic Lands', 'Gabon', 'Gambia',
               'Gaza Strip', 'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenlan
        d',
               'Grenada', 'Guadeloupe', 'Guam', 'Guatemala', 'Guernsey',
               'Guinea Bissau', 'Guinea', 'Guyana', 'Haiti',
               'Heard Island And Mcdonald Islands', 'Honduras', 'Hong Kong',
               'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq',
               'Ireland', 'Isle Of Man', 'Israel', 'Italy', 'Jamaica', 'Japan',
               'Jersey', 'Jordan', 'Kazakhstan', 'Kenya', 'Kingman Reef',
               'Kiribati', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
               'Lesotho', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania',
               'Luxembourg', 'Macau', 'Macedonia', 'Madagascar', 'Malawi',
               'Malaysia', 'Mali', 'Malta', 'Martinique', 'Mauritania',
               'Mauritius', 'Mayotte', 'Mexico', 'Moldova', 'Monaco', 'Mongolia',
               'Montenegro', 'Montserrat', 'Morocco', 'Mozambique', 'Namibia',
               'Nepal', 'Netherlands (Europe)', 'Netherlands', 'New Caledonia',
               'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'Niue',
               'North America', 'North Korea', 'Northern Mariana Islands',
               'Norway', 'Oceania', 'Oman', 'Pakistan', 'Palau', 'Palestina',
               'Palmyra Atoll', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
               'Philippines', 'Poland', 'Portugal', 'Puerto Rico', 'Qatar',
               'Reunion', 'Romania', 'Russia', 'Rwanda', 'Saint Barthélemy',
               'Saint Kitts And Nevis', 'Saint Lucia', 'Saint Martin',
               'Saint Pierre And Miquelon', 'Saint Vincent And The Grenadines',
               'Samoa', 'San Marino', 'Sao Tome And Principe', 'Saudi Arabia',
               'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore',
               'Sint Maarten', 'Slovakia', 'Slovenia', 'Solomon Islands',
               'Somalia', 'South Africa', 'South America',
               'South Georgia And The South Sandwich Isla', 'South Korea',
               'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
               'Svalbard And Jan Mayen', 'Swaziland', 'Sweden', 'Switzerland',
               'Syria', 'Taiwan', 'Tajikistan', 'Tanzania', 'Thailand',
               'Timor Leste', 'Togo', 'Tonga', 'Trinidad And Tobago', 'Tunisia',
               'Turkey', 'Turkmenistan', 'Turks And Caicas Islands', 'Uganda',
               'Ukraine', 'United Arab Emirates', 'United Kingdom (Europe)',
               'United Kingdom', 'United States', 'Uruguay', 'Uzbekistan',
               'Venezuela', 'Vietnam', 'Virgin Islands', 'Western Sahara',
               'Yemen', 'Zambia', 'Zimbabwe'], dtype=object)
```

It appears that is the case with a few countries e.g Denmark(Europe) and Denmark, France(Europe) and France etc. To replace them, I'll need a dictionary because of the key-value pairs.

In [9]:

```
dict={
    'Congo (Democratic Republic Of The)': 'Congo',
    'Denmark (Europe)':'Denmark',
    'France (Europe)':'France',
    'Netherlands (Europe)':'Netherlands',
    'United Kingdom (Europe)':'United Kingdom'
}
```

In [10]:

```
global_temp_country['Country'].replace(dict, inplace=True)
```

In [11]:

```
global_temp_country['Country'].unique()
```

```
Out[11]: array(['Åland', 'Afghanistan', 'Africa', 'Albania', 'Algeria',
                'American Samoa', 'Andorra', 'Angola', 'Anguilla',
                'Antigua And Barbuda', 'Argentina', 'Armenia', 'Aruba', 'Asia',
                'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
                'Baker Island', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium',
                'Belize', 'Benin', 'Bhutan', 'Bolivia',
                'Bonaire, Saint Eustatius And Saba', 'Bosnia And Herzegovina',
                'Botswana', 'Brazil', 'British Virgin Islands', 'Bulgaria',
                'Burkina Faso', 'Burma', 'Burundi', "Côte D'Ivoire", 'Cambodia',
                'Cameroon', 'Canada', 'Cape Verde', 'Cayman Islands',
                'Central African Republic', 'Chad', 'Chile', 'China',
                'Christmas Island', 'Colombia', 'Comoros', 'Congo', 'Costa Rica',
                'Croatia', 'Cuba', 'Curaçao', 'Cyprus', 'Czech Republic',
                'Denmark', 'Djibouti', 'Dominica', 'Dominican Republic', 'Ecuado
         r',
                'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
                'Ethiopia', 'Europe', 'Falkland Islands (Islas Malvinas)',
                'Faroe Islands', 'Federated States Of Micronesia', 'Fiji',
                'Finland', 'France', 'French Guiana', 'French Polynesia',
                'French Southern And Antarctic Lands', 'Gabon', 'Gambia',
                'Gaza Strip', 'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenlan
         d',
                'Grenada', 'Guadeloupe', 'Guam', 'Guatemala', 'Guernsey',
                'Guinea Bissau', 'Guinea', 'Guyana', 'Haiti',
                'Heard Island And Mcdonald Islands', 'Honduras', 'Hong Kong',
                'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq',
                'Ireland', 'Isle Of Man', 'Israel', 'Italy', 'Jamaica', 'Japan',
                'Jersey', 'Jordan', 'Kazakhstan', 'Kenya', 'Kingman Reef',
                'Kiribati', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                'Lesotho', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania',
                'Luxembourg', 'Macau', 'Macedonia', 'Madagascar', 'Malawi',
                'Malaysia', 'Mali', 'Malta', 'Martinique', 'Mauritania',
                'Mauritius', 'Mayotte', 'Mexico', 'Moldova', 'Monaco', 'Mongolia',
                'Montenegro', 'Montserrat', 'Morocco', 'Mozambique', 'Namibia',
                'Nepal', 'Netherlands', 'New Caledonia', 'New Zealand',
                'Nicaragua', 'Niger', 'Nigeria', 'Niue', 'North America',
                'North Korea', 'Northern Mariana Islands', 'Norway', 'Oceania',
                'Oman', 'Pakistan', 'Palau', 'Palestina', 'Palmyra Atoll',
                'Panama', 'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines',
                'Poland', 'Portugal', 'Puerto Rico', 'Qatar', 'Reunion', 'Romani
         a',
                'Russia', 'Rwanda', 'Saint Barthélemy', 'Saint Kitts And Nevis',
                'Saint Lucia', 'Saint Martin', 'Saint Pierre And Miquelon',
                'Saint Vincent And The Grenadines', 'Samoa', 'San Marino',
                'Sao Tome And Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
                'Seychelles', 'Sierra Leone', 'Singapore', 'Sint Maarten',
                'Slovakia', 'Slovenia', 'Solomon Islands', 'Somalia',
                'South Africa', 'South America',
                'South Georgia And The South Sandwich Isla', 'South Korea',
                'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                'Svalbard And Jan Mayen', 'Swaziland', 'Sweden', 'Switzerland',
                'Syria', 'Taiwan', 'Tajikistan', 'Tanzania', 'Thailand',
                'Timor Leste', 'Togo', 'Tonga', 'Trinidad And Tobago', 'Tunisia',
                'Turkey', 'Turkmenistan', 'Turks And Caicas Islands', 'Uganda',
                'Ukraine', 'United Arab Emirates', 'United Kingdom',
                'United States', 'Uruguay', 'Uzbekistan', 'Venezuela', 'Vietnam',
                'Virgin Islands', 'Western Sahara', 'Yemen', 'Zambia', 'Zimbabw
         e'],
               dtype=object)
```

Very nice. Now my data is looking good. Let's now calculate the average temperature of each Country.

In [12]:
```
avg_temp = global_temp_country.groupby(['Country'])['AverageTemperature'].m
avg_temp
```

Out[12]:

	Country	AverageTemperature
0	Afghanistan	14.045007
1	Africa	24.074203
2	Albania	12.610646
3	Algeria	22.985112
4	American Samoa	26.611965
...	...	...
232	Western Sahara	22.319818
233	Yemen	26.253597
234	Zambia	21.282956
235	Zimbabwe	21.117547
236	Åland	5.291383

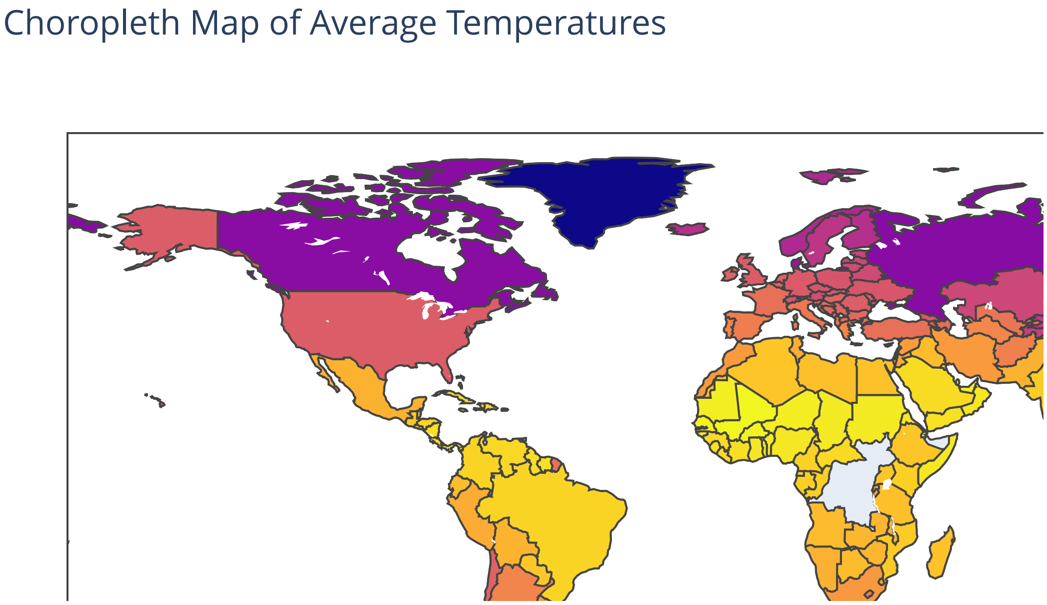
237 rows × 2 columns

Spatial Analysis on the Avg_temp Dataframe

Since we are looking at average global temperatures, how about we see a visual of that in a choropleth map

In [13]:

```
choro_fig = px.choropleth(data_frame=avg_temp, locations='Country', locatio
choro_fig.update_layout(title='Choropleth Map of Average Temperatures')
choro_fig.show()
```



Beautiful! By pointing my cursor on specific countries, I am able to get the average temperature of each country.

Problem statement: Where is the evidence for global warming?

In [14]:

```
#importing the relevant data to demonstrate
global_temp = pd.read_csv('C:/Users/koome/Desktop/Geospatial/GlobalTempData
global_temp.head()
```

Out[14]:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature
0	1750-01-01	3.034	3.574	NaN
1	1750-02-01	3.083	3.702	NaN
2	1750-03-01	5.626	3.076	NaN
3	1750-04-01	8.490	2.451	NaN
4	1750-05-01	11.573	2.072	NaN

In [15]:

```
global_temp.shape
```

Out[15]:

```
(3192, 9)
```

In [16]:

```
global_temp.isna().sum()
```

Out[16]:

```
dt                                              0
LandAverageTemperature                         12
LandAverageTemperatureUncertainty              12
LandMaxTemperature                           1200
LandMaxTemperatureUncertainty                1200
LandMinTemperature                           1200
LandMinTemperatureUncertainty                1200
LandAndOceanAverageTemperature               1200
LandAndOceanAverageTemperatureUncertainty    1200
dtype: int64
```

A lot of missing values on some columns but I'm mostly interested in 'LandAverageTemperature' and 'LandAverageTemperatureUncertainty' which have few missing values which can be removed

In [17]:

```
global_temp.dropna(axis='index', subset='LandAverageTemperature', inplace=T
```

In [18]:

```
global_temp.isna().sum()
```

Out[18]:

```
dt                                           0
LandAverageTemperature                       0
LandAverageTemperatureUncertainty            0
LandMaxTemperature                        1188
LandMaxTemperatureUncertainty             1188
LandMinTemperature                        1188
LandMinTemperatureUncertainty             1188
LandAndOceanAverageTemperature            1188
LandAndOceanAverageTemperatureUncertainty 1188
dtype: int64
```

Since I need to group the data by years, I need to parse the date string so that I retrieve the year only. The dataframe is indexed by monthly temperatures since 1750.

In [19]:

```
def fetch_year(date):
    return date.split('-')[0]
```

In [20]:

```
global_temp['years']=global_temp['dt'].apply(fetch_year)
global_temp.head()
```

Out[20]:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature
0	1750-01-01	3.034	3.574	NaN
1	1750-02-01	3.083	3.702	NaN
2	1750-03-01	5.626	3.076	NaN
3	1750-04-01	8.490	2.451	NaN
4	1750-05-01	11.573	2.072	NaN

Created a new column called years to store the parsed 'year'. Now I need to carry out some aggregation functions on the two aforementioned columns by grouping the data through the newly created column.

In [21]:

```
data = global_temp.groupby('years').agg({'LandAverageTemperature':'mean', '
data.head()
```

Out[21]:

	years	LandAverageTemperature	LandAverageTemperatureUncertainty
0	1750	8.719364	2.637818
1	1751	7.976143	2.781143
2	1752	5.779833	2.977000
3	1753	8.388083	3.176000
4	1754	8.469333	3.494250

To showcase evidence of global warming, the above columns are not enough. I need two new columns; 'uncertainty Top' and 'Uncertainty Bottom' which are derived from adding and subtracting the above columns respectively.

In [22]:

```
data['UncertaintyTop'] = data['LandAverageTemperature'] + data['LandAverage
data['UncertaintyBottom'] = data['LandAverageTemperature'] - data['LandAver
data.head()
```

Out[22]:

	years	LandAverageTemperature	LandAverageTemperatureUncertainty	UncertaintyTop	Unc
0	1750	8.719364	2.637818	11.357182	
1	1751	7.976143	2.781143	10.757286	
2	1752	5.779833	2.977000	8.756833	
3	1753	8.388083	3.176000	11.564083	
4	1754	8.469333	3.494250	11.963583	

Now that I have good plotting data, about time I plotted the same with a line graph for good visualization

In [23]:

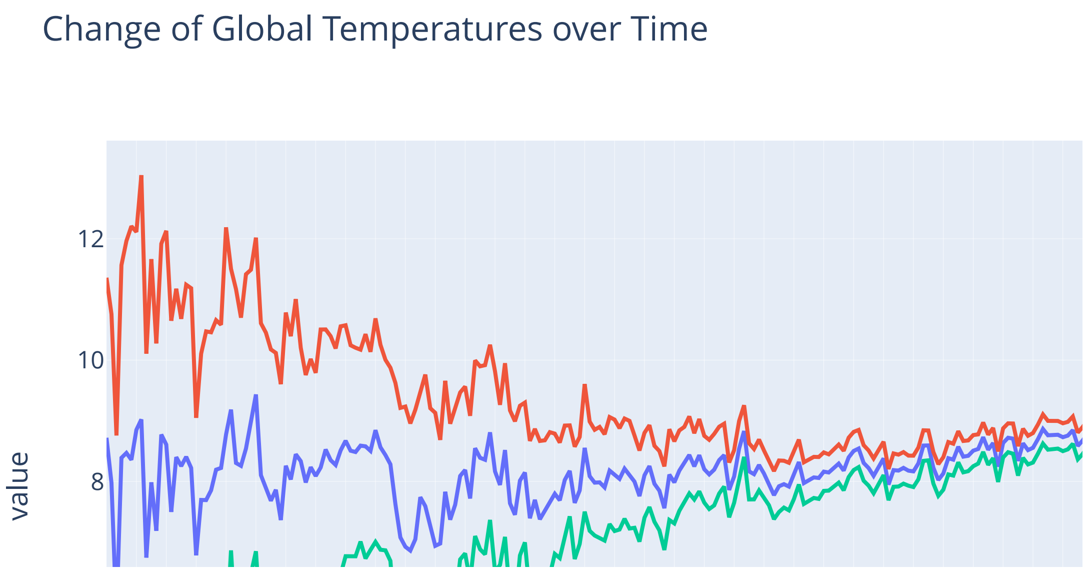
```
data.columns
```

Out[23]:

```
Index(['years', 'LandAverageTemperature', 'LandAverageTemperatureUncertai
nty',
       'UncertaintyTop', 'UncertaintyBottom'],
      dtype='object')
```

In [24]:

```
fig = px.line(data, x='years',y=['LandAverageTemperature','UncertaintyTop',
fig.show()
```



Conclusion: From the line-chart one can see that there has been a steady rise in global temperatures with very minimal temperature uncertainty. If there was no global phenomena, the three lines would have remained equidistant over the period. The convergence of the Uncertainty Top and Uncertainty Bottom with the Average Land Temperature over the period coincides with the 1st industrial revolution that took place from 1740-1870 and an acceleration from 1870-1914 which coincides with the 2nd industrial revolution.

Type *Markdown* and LaTeX: α^2

Problem statement: Analyze average temperature in each season.

```
In [25]: global_temp.head()
```

Out[25]:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature
0	1750-01-01	3.034	3.574	NaN
1	1750-02-01	3.083	3.702	NaN
2	1750-03-01	5.626	3.076	NaN
3	1750-04-01	8.490	2.451	NaN
4	1750-05-01	11.573	2.072	NaN

```
In [26]: global_temp['dt'].dtype
```

```
Out[26]: dtype('O')
```

My dataframe does not have any season column but I have a date column which is of string type. So I have to extract the seasons myself from the date column after converting it into a datetime type.

```
In [27]: global_temp['dt'] = pd.to_datetime(global_temp['dt'])
```

```
In [28]: global_temp['dt'].dtype
```

```
Out[28]: dtype('<M8[ns]')
```

Great. Now the date column is of a datetime type, but I am more interested in extracting the months so that I can match them to a season.

```
In [29]: global_temp['month'] = global_temp['dt'].dt.month
```

In [30]:

```
global_temp.head()
```

Out[30]:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature
0	1750-01-01	3.034	3.574	NaN
1	1750-02-01	3.083	3.702	NaN
2	1750-03-01	5.626	3.076	NaN
3	1750-04-01	8.490	2.451	NaN
4	1750-05-01	11.573	2.072	NaN

Now that I have extracted the exact date months, I'll need a function that can assign all the months to a season and generate another season column.

In [31]:

```
def fetch_season(month):
    if 3<=month<=5:
        return 'spring'
    elif 6<=month<=8:
        return 'summer'
    elif 9<=month<=11:
        return 'autumn'
    else:
        return 'winter'
```

In [32]:

```
global_temp['season']=global_temp['month'].apply(fetch_season)
global_temp.head(10)
```

Out[32]:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature
0	1750-01-01	3.034	3.574	NaN
1	1750-02-01	3.083	3.702	NaN
2	1750-03-01	5.626	3.076	NaN
3	1750-04-01	8.490	2.451	NaN
4	1750-05-01	11.573	2.072	NaN
5	1750-06-01	12.937	1.724	NaN
6	1750-07-01	15.868	1.911	NaN
7	1750-08-01	14.750	2.231	NaN
8	1750-09-01	11.413	2.637	NaN
9	1750-10-01	6.367	2.668	NaN

Great. Now I have a season column. Time to find a solution to the problem statement.

In [33]:

```
years=global_temp['years'].unique()
```

In [34]:

```
spring_temps=[]
summer_temps=[]
autumn_temps=[]
winter_temps=[]
for year in years:
    current_df=global_temp[global_temp['years']==year]
    spring_temps.append(current_df[current_df['season']=='spring']['LandAve
    summer_temps.append(current_df[current_df['season']=='summer']['LandAve
    autumn_temps.append(current_df[current_df['season']=='autumn']['LandAve
    winter_temps.append(current_df[current_df['season']=='winter']['LandAve
```

In [35]:
```
spring_temps
```

Out[35]:
```
[8.563,
 6.734999999999999,
 7.035499999999999,
 8.627333333333334,
 9.074333333333334,
 8.583666666666666,
 9.466,
 8.604666666666667,
 6.896666666666666,
 6.897333333333333,
 6.653666666666666,
 8.916,
 7.809333333333332,
 6.716,
 8.192,
 8.868666666666668,
 8.432333333333332,
 7.831,
 6.144000000000001,
```

Having extracted the average temperatures for each season over the entire period, let's do some data visualization. First, I need to create a new dataframe to store the data generated in the season lists.

In [36]:
```
seasons=pd.DataFrame()
```

In [37]:
```
seasons['years']=years
seasons['spring_temps']=spring_temps
seasons['summer_temps']=summer_temps
seasons['autumn_temps']=autumn_temps
seasons['winter_temps']=winter_temps
```

In [38]:
```
seasons.head()
```

Out[38]:

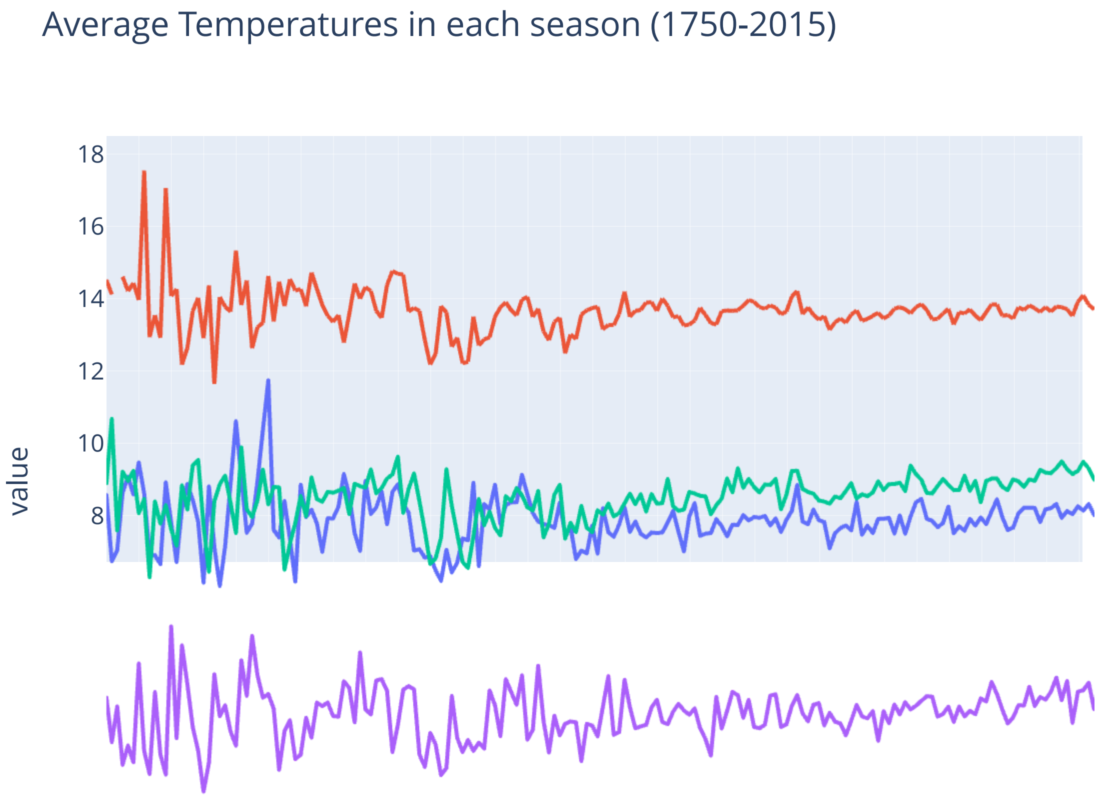
	years	spring_temps	summer_temps	autumn_temps	winter_temps
0	1750	8.563000	14.518333	8.890000	2.963000
1	1751	6.735000	14.116000	10.673000	1.729000
2	1752	7.035500	NaN	7.587000	2.717000
3	1753	8.627333	14.608333	9.212333	1.104333
4	1754	9.074333	14.208333	8.957333	1.637333

In [39]:
```
seasons.columns
```

Out[39]:
```
Index(['years', 'spring_temps', 'summer_temps', 'autumn_temps',
       'winter_temps'],
      dtype='object')
```

In [40]:

```
fig=px.line(seasons,x='years',y=['spring_temps', 'summer_temps', 'autumn_te
        'winter_temps'], title='Average Temperatures in each season (1750-20
fig.show()
```



Conclusion: From the line-chart, it is clear that the average temperatures across all the seasons are on a steady rise from around the year 1978. So we can deduce from it that there is a global warming occurring.

Problem statement 3: Analyze Average Temperatures of US States

In [41]:

```
#importing the relevant data for analysis
state_temp=pd.read_csv('C:/Users/koome/Desktop/Geospatial/GlobalTempData/Gl
state_temp.head()
```

Out[41]:

	dt	AverageTemperature	AverageTemperatureUncertainty	State	Country
0	1855-05-01	25.544	1.171	Acre	Brazil
1	1855-06-01	24.228	1.103	Acre	Brazil
2	1855-07-01	24.371	1.044	Acre	Brazil
3	1855-08-01	25.427	1.073	Acre	Brazil
4	1855-09-01	25.675	1.014	Acre	Brazil

Brazil is not a US state so we have to find a way to filter the data to only get US states

In [ ]:

```

```

In [42]:

```
filter = state_temp['Country']=='United States'
USA_temp=state_temp[filter]
USA_temp.head()
```

Out[42]:

	dt	AverageTemperature	AverageTemperatureUncertainty	State	Country
7458	1743-11-01	10.722	2.898	Alabama	United States
7459	1743-12-01	NaN	NaN	Alabama	United States
7460	1744-01-01	NaN	NaN	Alabama	United States
7461	1744-02-01	NaN	NaN	Alabama	United States
7462	1744-03-01	NaN	NaN	Alabama	United States

In [43]:

```
USA_temp.shape
```

Out[43]:

```
(149745, 5)
```

Initial indications are that there is too much missing data. We have to drop those NA values

In [44]:

```
USA_temp.isna().sum()
```

Out[44]:

```
dt                                  0
AverageTemperature               7815
AverageTemperatureUncertainty    7815
State                               0
Country                             0
dtype: int64
```

The missing values represent only 5% of the data. Dropping them will not affect our sample size for final analysis.

In [45]:

```
USA_temp.dropna(axis='index', how='any', subset='AverageTemperature',inplac
USA_temp.isna().sum()
```

```
C:\Users\koome\AppData\Local\Temp\ipykernel_13632\4260614531.py:1: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
```

Out[45]:

```
dt                               0
AverageTemperature               0
AverageTemperatureUncertainty    0
State                            0
Country                          0
dtype: int64
```

In [46]:

```
USA_temp['State'].unique()
```

Out[46]:

```
array(['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
       'Colorado', 'Connecticut', 'Delaware', 'District Of Columbia',
       'Florida', 'Georgia (State)', 'Hawaii', 'Idaho', 'Illinois',
       'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine',
       'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
       'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
       'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',
       'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington',
       'West Virginia', 'Wisconsin', 'Wyoming'], dtype=object)
```

In [47]:

```
USA_temp['State'].nunique()
```

Out[47]:

```
51
```

For the sake of our analysis we'll only need the AverageTemperature and State coulumns

In [48]:
```
States=USA_temp[['AverageTemperature','State']]
States.head()
```

Out[48]:

	AverageTemperature	State
7458	10.722	Alabama
7463	19.075	Alabama
7464	21.197	Alabama
7465	25.290	Alabama
7466	26.420	Alabama

Saved the extracted dataframe under a new name 'States'. There is now need to group the dataframe by state and get the mean for each state.

In [49]:
```
States_temp=States.groupby('State')['AverageTemperature'].mean().reset_inde
States_temp.head()
```

Out[49]:

	State	AverageTemperature
0	Alabama	17.066138
1	Alaska	-4.890738
2	Arizona	15.381526
3	Arkansas	15.573963
4	California	14.327677

Now we have sufficient data to generate a heatmap, but before that we need to generate position data, latitude and longitudes, of the states. Opencage is a python library that can help us do that.

In [50]:

```
pip install opencage
```

```
Requirement already satisfied: opencage in c:\users\koome\anaconda3\lib\s
ite-packages (2.4.0)Note: you may need to restart the kernel to use updat
ed packages.

Requirement already satisfied: Requests>=2.31.0 in c:\users\koome\anacond
a3\lib\site-packages (from opencage) (2.31.0)
Requirement already satisfied: backoff>=2.2.1 in c:\users\koome\anaconda3
\lib\site-packages (from opencage) (2.2.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\koome
\anaconda3\lib\site-packages (from Requests>=2.31.0->opencage) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\koome\anaconda3\l
ib\site-packages (from Requests>=2.31.0->opencage) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\koome\anaco
nda3\lib\site-packages (from Requests>=2.31.0->opencage) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\koome\anaco
nda3\lib\site-packages (from Requests>=2.31.0->opencage) (2024.2.2)
```

In [51]:

```
from opencage.geocoder import OpenCageGeocode
```

There is need to establish a connection with OpenGageGeocode with the use of an API key which can be obtained from the Opencage documentation website.

In [52]:

```
key='66d1389c65094b54b1f0caf77fd5bcee'
```

In [53]:

```
geocoder=OpenCageGeocode(key)
```

Now that the geocoder is setup, let's test it with my location in Nairobi.

In [54]:

```
location='Nairobi, Kenya'
nai=geocoder.geocode(location)
nai
```

```
  'bounds': {'northeast': {'lat': -1.1606749, 'lng': 37.1048735},
   'southwest': {'lat': -1.4448822, 'lng': 36.6647016}},
  'components': {'ISO_3166-1_alpha-2': 'KE',
   'ISO_3166-1_alpha-3': 'KEN',
   'ISO_3166-2': ['KE-30'],
   '_category': 'place',
   '_normalized_city': 'Nairobi',
   '_type': 'city',
   'city': 'Nairobi',
   'continent': 'Africa',
   'country': 'Kenya',
   'country_code': 'ke',
   'state': 'Nairobi County'},
  'confidence': 2,
  'formatted': 'Nairobi, Nairobi County, Kenya',
  'geometry': {'lat': -1.2832533, 'lng': 36.8172449}},
 {'annotations': {'DMS': {'lat': "1° 18' 9.41328'' S",
    'lng': "36° 49' 43.83120'' E"},
   'MGRS': '37MBU5840855917',
   'Maidenhead': 'KI88iq97li',
```

Great information regarding Nairobi but we'll only need the 'geometry' field which will give us our latitude and longitude.

In [56]:

```
nai[0]['geometry']
```

Out[56]:

```
{'lat': -1.2832533, 'lng': 36.8172449}
```

The test has passed, so back to our united states data. We'll need a for loop to iterate over all the states in our States_temp dataframe

In [58]:

```
list_lat =[]
list_long=[]

for state in States_temp['State']:
    results=geocoder.geocode(state)
    lat=results[0]['geometry']['lat']
    long=results[0]['geometry']['lng']

    list_lat.append(lat)
    list_long.append(long)
```

Took a while to fish out that info but its time to update our dataframe with two new columns for latitude and longitude.

In [62]:

```
States_temp['Latitude']=list_lat
States_temp['Longitude']=list_long
States_temp.drop(['latitude','longitude'],axis=1, inplace=True)
States_temp.head()
```

Out[62]:

	State	AverageTemperature	Latitude	Longitude
0	Alabama	17.066138	33.258882	-86.829534
1	Alaska	-4.890738	64.445961	-149.680909
2	Arizona	15.381526	34.395342	-111.763275
3	Arkansas	15.573963	35.204888	-92.447911
4	California	14.327677	36.701463	-118.755997

Now the data is ready for some spatial analysis. Am thinking a heatmap would be a good data visualization for this data. Let's call upon Folium to send us a Hail Mary throughpass.

In [64]:

```
pip install folium
```

```
Collecting foliumNote: you may need to restart the kernel to use updated
packages.

  Obtaining dependency information for folium from https://files.pythonho
sted.org/packages/b9/98/9ba4b9d2d07dd32765ddb4e4c189dcbdd7dca4d5a735e2e4e
a756f40c36b/folium-0.16.0-py2.py3-none-any.whl.metadata (https://files.py
thonhosted.org/packages/b9/98/9ba4b9d2d07dd32765ddb4e4c189dcbdd7dca4d5a73
5e2e4ea756f40c36b/folium-0.16.0-py2.py3-none-any.whl.metadata)
  Downloading folium-0.16.0-py2.py3-none-any.whl.metadata (3.6 kB)
Collecting branca>=0.6.0 (from folium)
  Obtaining dependency information for branca>=0.6.0 from https://files.p
ythonhosted.org/packages/17/ce/14166d0e273d12065516625fb02426350298e7b4ba
59198b5fe454b46202/branca-0.7.1-py3-none-any.whl.metadata (https://files.
pythonhosted.org/packages/17/ce/14166d0e273d12065516625fb02426350298e7b4b
a59198b5fe454b46202/branca-0.7.1-py3-none-any.whl.metadata)
  Downloading branca-0.7.1-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: jinja2>=2.9 in c:\users\koome\anaconda3\li
b\site-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in c:\users\koome\anaconda3\lib\site
-packages (from folium) (1.24.3)
Requirement already satisfied: requests in c:\users\koome\anaconda3\lib\s
ite-packages (from folium) (2.31.0)
Requirement already satisfied: xyzservices in c:\users\koome\anaconda3\li
b\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\koome\anaconda
3\lib\site-packages (from jinja2>=2.9->folium) (2.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\koome
\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\koome\anaconda3\l
ib\site-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\koome\anaco
nda3\lib\site-packages (from requests->folium) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\koome\anaco
nda3\lib\site-packages (from requests->folium) (2024.2.2)
Downloading folium-0.16.0-py2.py3-none-any.whl (100 kB)
   ---------------------------------------- 0.0/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   ------------------------------------ --- 92.2/100.0 kB ? eta -:--:--
   -------------------------------------- 100.0/100.0 kB 261.0 kB/s eta
0:00:00
Downloading branca-0.7.1-py3-none-any.whl (25 kB)
Installing collected packages: branca, folium
Successfully installed branca-0.7.1 folium-0.16.0
```

In [75]:
```
import folium
from folium.plugins import HeatMap
```

In [78]:
```
basemap=folium.Map()
basemap
```

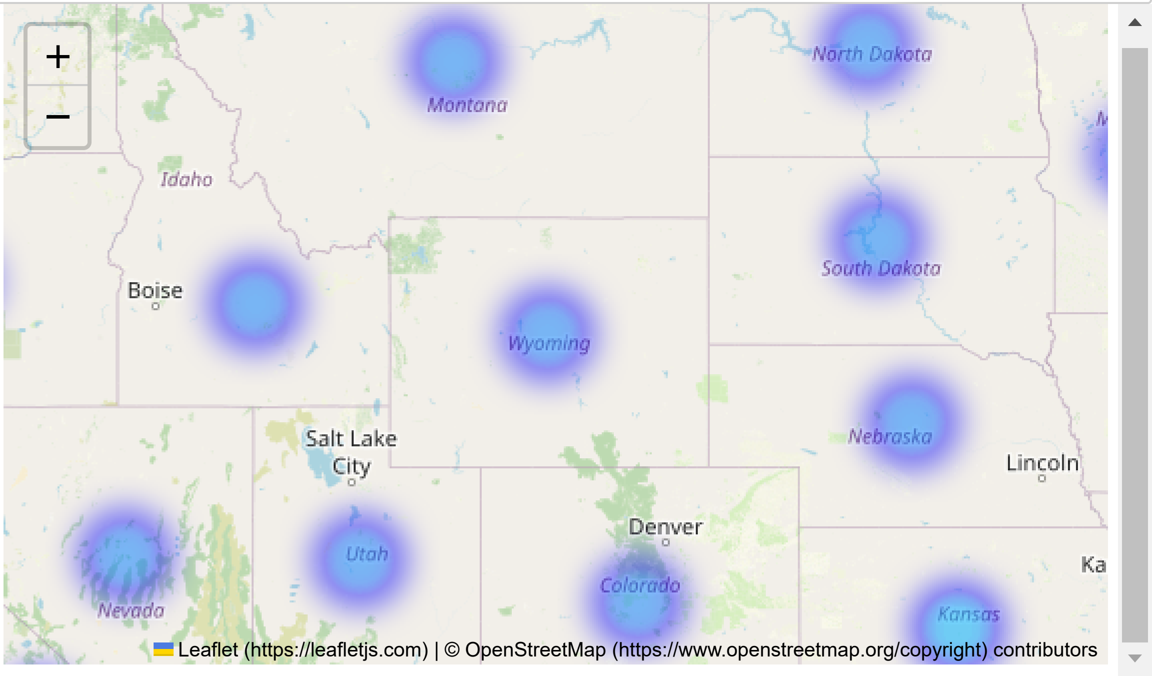
Out[78]: Make this Notebook Trusted to load map: File -> Trust Notebook

+
–
Leaflet (https://leafletjs.com) | © OpenStreetMap (https://www.openstreetmap.org/copyright) contributors

Excellent view of the basemap. Now we add just three columns of our dataframe to the heatmap which we'll subsequently add to the basemap

In [79]:
```
HeatMap(States_temp[['Latitude','Longitude','AverageTemperature']]).add_to(
basemap
```

Out[79]:



Problem Statement: Analyze Average Temperatures of Major Kenyan Cities by month

In [81]:

```
cities=pd.read_csv('C:/Users/koome/Desktop/Geospatial/GlobalTempData/Global
cities.head()
```

Out[81]:

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Lo
0	1743-11-01	6.068	1.737	Århus	Denmark	57.05N	
1	1743-12-01	NaN	NaN	Århus	Denmark	57.05N	
2	1744-01-01	NaN	NaN	Århus	Denmark	57.05N	
3	1744-02-01	NaN	NaN	Århus	Denmark	57.05N	
4	1744-03-01	NaN	NaN	Århus	Denmark	57.05N	

Need to filter out the dataframe to get data from Kenya

In [86]:

```
kenya=cities[cities['Country']=='Kenya']
kenya.head()
```

Out[86]:

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitu
2201688	1850-01-01	20.504	1.453	Eldoret	Kenya	0.8
2201689	1850-02-01	21.904	1.485	Eldoret	Kenya	0.8
2201690	1850-03-01	21.474	2.222	Eldoret	Kenya	0.8
2201691	1850-04-01	20.195	1.580	Eldoret	Kenya	0.8
2201692	1850-05-01	19.298	1.006	Eldoret	Kenya	0.8

In [87]:

```
kenya.shape
```

Out[87]:

```
(11790, 7)
```

In [89]:

```
kenya['City'].unique()
```

Out[89]:

```
array(['Eldoret', 'Kisumu', 'Mombasa', 'Nairobi', 'Nakuru', 'Ruiru'],
      dtype=object)
```

The Latitude and Longitude columns have a suffix 'n' and 'e' respectively which will need to be stripped before any analysis

In [105]:

```
kenya['Latitude']=kenya['Latitude'].str.strip('N')
kenya['Longitude']=kenya['Longitude'].str.strip('E')
kenya.head()
import warnings
warnings.filterwarnings('ignore')#I was getting too many warnings
```

In [99]:

```
kenya['dt'].dtypes
```

Out[99]:

```
dtype('O')
```

The problem statement seeks to know the average temperature by month and our 'dt' column is of string type. Therefore we need to convert it into a datetime object, extract the month and add it as another column in the dataframe.

In [106]:

```
kenya['dt']=pd.to_datetime(kenya['dt'])
```

In [107]:

```
kenya['Month']=kenya['dt'].dt.month
kenya.head()
```

Out[107]:

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitu
2201688	1850-01-01	20.504	1.453	Eldoret	Kenya	0.
2201689	1850-02-01	21.904	1.485	Eldoret	Kenya	0.
2201690	1850-03-01	21.474	2.222	Eldoret	Kenya	0.
2201691	1850-04-01	20.195	1.580	Eldoret	Kenya	0.
2201692	1850-05-01	19.298	1.006	Eldoret	Kenya	0.

Will now group the dataframe by Month and Cities and then carry out an agg function

In [121]:
```
kenya_temps=kenya.groupby(['Month','City'])['AverageTemperature'].mean().to
kenya_temps.columns=['month','City','Mean_temp']
kenya_temps.head()
```

Out[121]:

	month	City	Mean_temp
0	1	Eldoret	21.572388
1	1	Kisumu	21.984561
2	1	Mombasa	26.733417
3	1	Nairobi	16.726799
4	1	Nakuru	16.726799

In [126]:
```
df=kenya_temps.merge(kenya,on='City')
# df.drop_duplicates(subset=['Month_x','City'])
df.head()
```

Out[126]:

	month	City	Mean_temp	dt	AverageTemperature	AverageTemperatureUncertainty	(
0	1	Eldoret	21.572388	1850-01-01	20.504	1.453	
1	1	Eldoret	21.572388	1850-02-01	21.904	1.485	
2	1	Eldoret	21.572388	1850-03-01	21.474	2.222	
3	1	Eldoret	21.572388	1850-04-01	20.195	1.580	
4	1	Eldoret	21.572388	1850-05-01	19.298	1.006	

In [125]:
```
k1=df.drop_duplicates(subset=['month','City'])
k1.head()
```

Out[125]:

	month	City	Mean_temp	dt	AverageTemperature	AverageTemperatureUncertaint
0	1	Eldoret	21.572388	1850-01-01	20.504	1.45
1965	2	Eldoret	21.974321	1850-01-01	20.504	1.45
3930	3	Eldoret	21.959429	1850-01-01	20.504	1.45
5895	4	Eldoret	21.131107	1850-01-01	20.504	1.45
7860	5	Eldoret	20.764057	1850-01-01	20.504	1.45

In [133]:
```
k2=k1[['month','City','Mean_temp','Country','Latitude','Longitude']]
k2.head()
```

Out[133]:

	month	City	Mean_temp	Country	Latitude	Longitude
0	1	Eldoret	21.572388	Kenya	0.80	34.55
1965	2	Eldoret	21.974321	Kenya	0.80	34.55
3930	3	Eldoret	21.959429	Kenya	0.80	34.55
5895	4	Eldoret	21.131107	Kenya	0.80	34.55
7860	5	Eldoret	20.764057	Kenya	0.80	34.55

In [132]:
```
k2.shape
```

Out[132]:
```
(72, 6)
```

With this data a HeatMap can be visualized

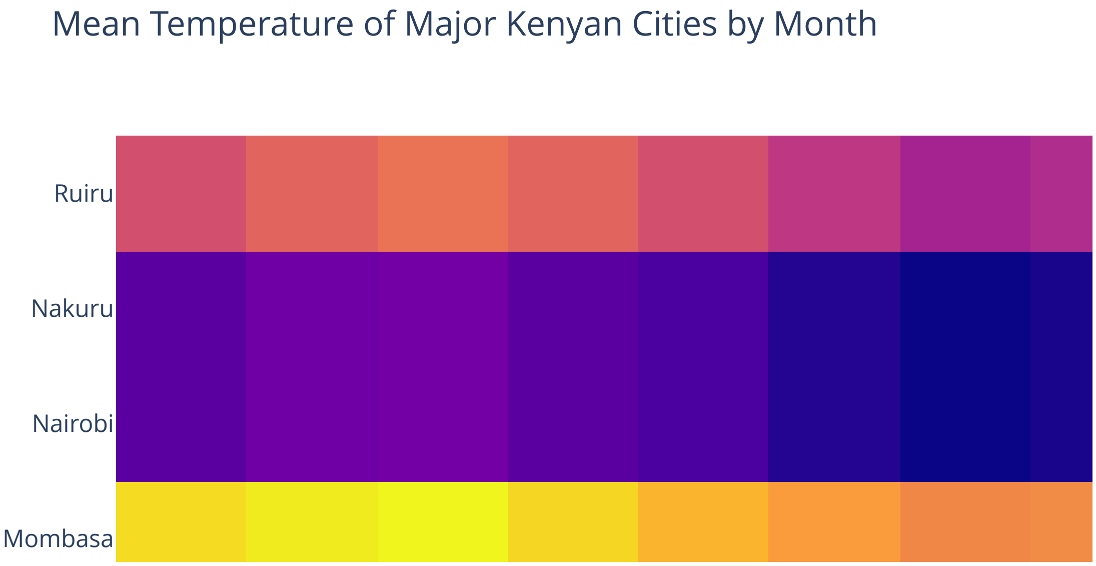
In [134]:
```
import plotly.graph_objs as go
```

In [135]:
```
data=[go.Heatmap(x=k2['month'],y=k2['City'],z=k2['Mean_temp'])]
```

In [136]:
```
layout=go.Layout(title='Mean Temperature of Major Kenyan Cities by Month')
```

In [137]:

```
fig=go.Figure(data,layout)
fig.show()
```



In [ ]:

In [ ]: