This project is part of CS245 (Principles of Data Science) in Computer Science, Faculty of Science and Technology, Thammasat University which I am responsible for all coding. We will use the Environment Temperature Change dataset that contains the yearly temperature for many countries from 1961 to 2019.

Import all required packages and load the data set.

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


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df = pd.read_csv('/content/drive/MyDrive/환경/climate_change/Environment_Temperature_change_E_All_Data_NOFLAG.csv', encoding='latin-1')
```

Summary Information About Data Set

```
df.loc[:, ~df.columns.isin(['Area Code', 'Months Code', 'Element Code'])].describe()
```

	Y1961	Y1962	Y1963	Y1964	Y1965	Y1966	Y1967	Y1968	Y1969	Y1970
count	8287.000000	8322.000000	8294.000000	8252.000000	8281.000000	8364.000000	8347.000000	8345.00000	8326.000000	8308.000000
mean	0.402433	0.315527	0.317393	0.269382	0.217839	0.376419	0.263239	0.24487	0.382172	0.365322
std	0.701567	0.713777	0.853133	0.749216	0.739418	0.737370	0.725421	0.75490	0.725313	0.662412
min	-4.018000	-5.391000	-8.483000	-7.309000	-4.728000	-8.147000	-6.531000	-8.40700	-6.784000	-5.847000
25%	0.057000	-0.033000	0.030250	-0.102500	-0.214000	0.055000	-0.169000	-0.16400	0.171000	0.094000
50%	0.366000	0.333000	0.355000	0.326000	0.303000	0.360000	0.313000	0.31200	0.385000	0.367000
75%	0.676500	0.627000	0.647750	0.609000	0.584000	0.660250	0.601000	0.59500	0.677000	0.642000
max	5.771000	4.373000	4.666000	5.233000	5.144000	5.771000	4.768000	4.37300	4.411000	4.373000

8 rows × 59 columns

```
df.info()
```

```
 10  Y1964          8252 non-null   float64
 11  Y1965          8281 non-null   float64
 12  Y1966          8364 non-null   float64
 13  Y1967          8347 non-null   float64
 14  Y1968          8345 non-null   float64
 15  Y1969          8326 non-null   float64
 16  Y1970          8308 non-null   float64
 17  Y1971          8303 non-null   float64
 18  Y1972          8323 non-null   float64
 19  Y1973          8394 non-null   float64
 20  Y1974          8374 non-null   float64
 21  Y1975          8280 non-null   float64
 22  Y1976          8209 non-null   float64
 23  Y1977          8257 non-null   float64
 24  Y1978          8327 non-null   float64
 25  Y1979          8290 non-null   float64
 26  Y1980          8283 non-null   float64
 27  Y1981          8276 non-null   float64
```

```
 43  Y1997          8369 non-null   float64
 44  Y1998          8370 non-null   float64
 45  Y1999          8324 non-null   float64
 46  Y2000          8342 non-null   float64
 47  Y2001          8241 non-null   float64
 48  Y2002          8312 non-null   float64
 49  Y2003          8390 non-null   float64
 50  Y2004          8415 non-null   float64
 51  Y2005          8424 non-null   float64
 52  Y2006          8503 non-null   float64
 53  Y2007          8534 non-null   float64
 54  Y2008          8475 non-null   float64
 55  Y2009          8419 non-null   float64
 56  Y2010          8435 non-null   float64
 57  Y2011          8437 non-null   float64
 58  Y2012          8350 non-null   float64
 59  Y2013          8427 non-null   float64
 60  Y2014          8377 non-null   float64
 61  Y2015          8361 non-null   float64
 62  Y2016          8348 non-null   float64
 63  Y2017          8366 non-null   float64
 64  Y2018          8349 non-null   float64
 65  Y2019          8365 non-null   float64
dtypes: float64(59), int64(3), object(4)
memory usage: 4.9+ MB
```

Data Cleansing / Data Validity

```
print("Number of NaN values before filling :₩n", df.isnull().sum().sort_values(ascending=False))

# Select only numeric columns before calculating the mean
numeric_df = df.select_dtypes(include=['number'])

# Fill NaN values in numeric columns with their respective means
df[numeric_df.columns] = numeric_df.fillna(numeric_df.mean())

print("₩nNumber of NaN values after filling :₩n", df.isnull().sum().sort_values(ascending=False))
```

```
Number of NaN values before filling :
 Area Code    0
Y2003         0
Y1989         0
Y1990         0
Y1991         0
             ..
Y1981         0
Y1982         0
Y1983         0
Y1984         0
Y2019         0
Length: 66, dtype: int64

Number of NaN values after filling :
 Area Code    0
Y2003         0
Y1989         0
Y1990         0
Y1991         0
             ..
Y1981         0
Y1982         0
Y1983         0
Y1984         0
Y2019         0
Length: 66, dtype: int64
```

```
print("Number of rows with all NaNs =", len(df[df.isnull().all(axis=1)]))
```

```
Number of rows with all NaNs = 0
```

```
print("Number of duplicate rows =", len(df[df.duplicated()]))
```

```
Number of duplicate rows = 0
```

Data Wrangling

```
df = df.loc[df['Element'] == 'Temperature change']
```

```
df.drop(['Area Code', 'Months Code', 'Element Code', 'Element', 'Unit'], axis=1, inplace=True)
```

```
<ipython-input-13-7d3a9c089d25>:1: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(['Area Code', 'Months Code', 'Element Code', 'Element', 'Unit'], axis=1, inplace=True)
```

```
df.replace(to_replace=r'DecWx96JanWx96Feb', value='Winter', regex=True, inplace=True)
df.replace(to_replace=r'MarWx96AprWx96May', value='Spring', regex=True, inplace=True)
df.replace(to_replace=r'JunWx96JulWx96Aug', value='Summer', regex=True, inplace=True)
df.replace(to_replace=r'SepWx960ctWx96Nov', value='Fall', regex=True, inplace=True)
```

```
<ipython-input-14-39a0927abe6b>:1: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.replace(to_replace=r'DecWx96JanWx96Feb', value='Winter', regex=True, inplace=True)
<ipython-input-14-39a0927abe6b>:2: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.replace(to_replace=r'MarWx96AprWx96May', value='Spring', regex=True, inplace=True)
<ipython-input-14-39a0927abe6b>:3: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.replace(to_replace=r'JunWx96JulWx96Aug', value='Summer', regex=True, inplace=True)
<ipython-input-14-39a0927abe6b>:4: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.replace(to_replace=r'SepWx960ctWx96Nov', value='Fall', regex=True, inplace=True)
```

```
df.rename(columns = {'Area':'country_name', 'Months':'months'}, inplace=True)
```

```
<ipython-input-15-a1c0c1e3e204>:1: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.rename(columns = {'Area':'country_name', 'Months':'months'}, inplace=True)
```

```
df = pd.melt(df, id_vars=['country_name', 'months'], var_name='year', value_name='temp_change')
df['year'] = [i.split('Y')[-1] for i in df.year]
```

```
df
```

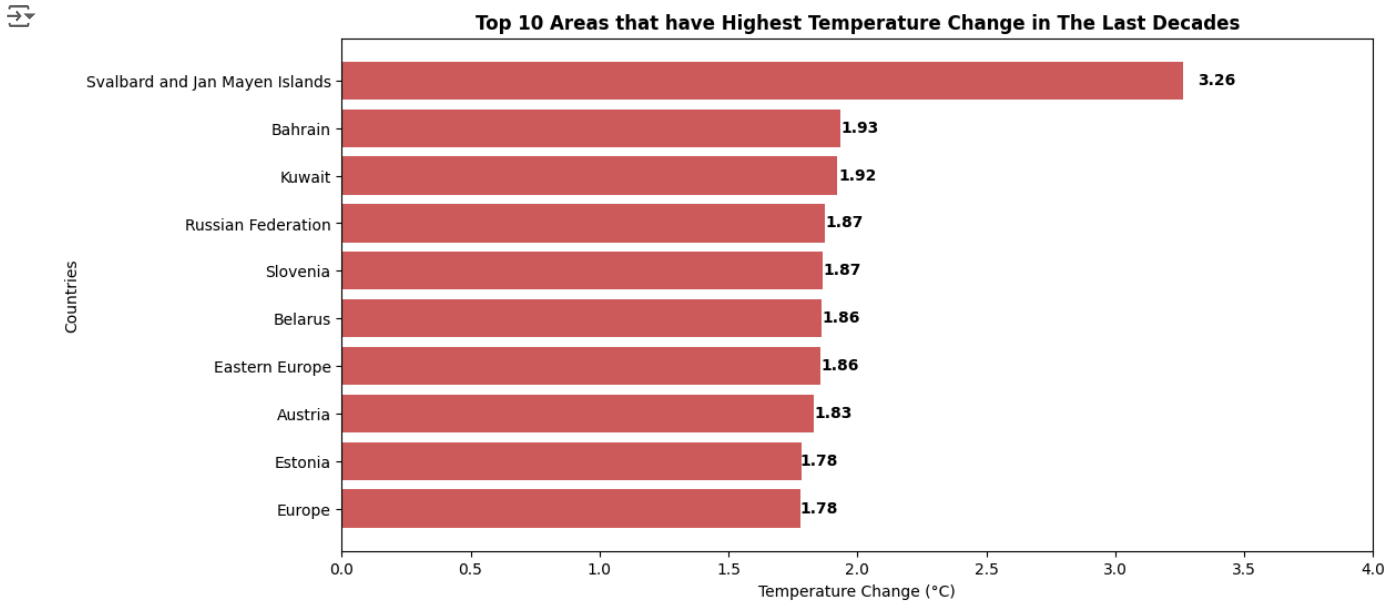
	country_name	months	year	temp_change
0	Afghanistan	January	1961	0.777
1	Afghanistan	February	1961	-1.743
2	Afghanistan	March	1961	0.516
3	Afghanistan	April	1961	-1.709
4	Afghanistan	May	1961	1.412
...	...	...	...	...
284847	OECD	Winter	2019	1.527
284848	OECD	Spring	2019	1.352
284849	OECD	Summer	2019	1.078
284850	OECD	Fall	2019	1.233
284851	OECD	Meteorological year	2019	1.297

284852 rows × 4 columns

Data Visualization

Top 10 Areas with The Highest Temperature Change

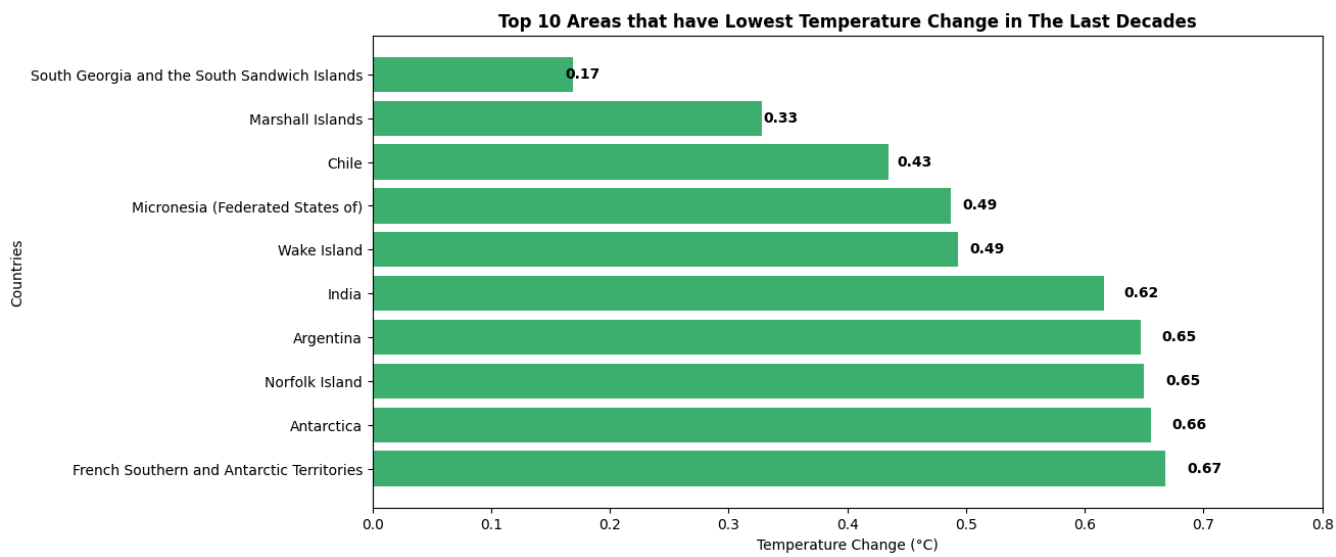
```
df1 = df.copy()
df1.set_index('year', inplace=True)
df1 = df1.loc[['2010','2011','2012','2013','2014','2015','2016','2017','2018','2019']]
df1.reset_index(inplace=True)

df1 = df1.groupby(['country_name',]).agg({'temp_change':'mean'})
df1.reset_index(inplace=True)
df1 = df1.sort_values(by=['temp_change'], ascending=False).head(10)

fig, ax = plt.subplots(figsize=(12,6))

bar1 = plt.barh(df1['country_name'], df1['temp_change'], color='indianred')

temp = round(df1['temp_change'], 2).to_list()

i = 0
for p in bar1:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    plt.text(x + width * 1.04, y + height * 0.6, str(temp[i]), ha='center', weight='bold')
    i+=1

ax.set_xlim([0, 4.0])
ax.invert_yaxis()
plt.title("Top 10 Areas that have Highest Temperature Change in The Last Decades", weight='bold')
plt.xlabel("Temperature Change (° C)")
plt.ylabel("Countries")
plt.show()
```



The bar chart shows the average temperature change values of 10 areas that have the highest temperature change in the last decade. All countries on the list are industrialized countries, excluding Svalbard and Jan Mayen Islands. This area is near Europe and Russia and it is the arctic area that is greatly affected by climate change.

Top 10 Areas with The Lowest Temperature Change

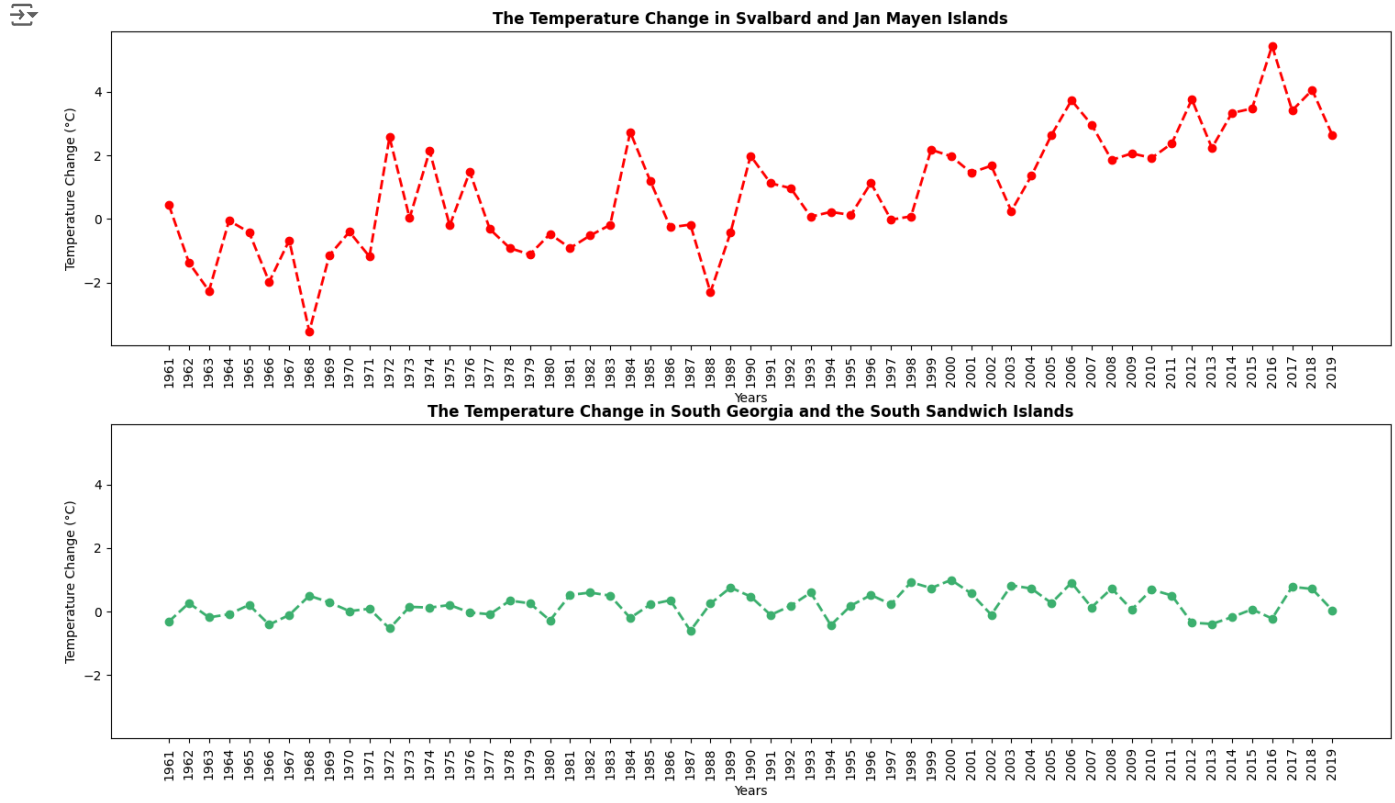
```
df2 = df.copy()
df2.set_index('year', inplace=True)
df2 = df2.loc[['2010','2011','2012','2013','2014','2015','2016','2017','2018','2019']]
df2.reset_index(inplace=True)

df2 = df2.groupby(['country_name',]).agg({'temp_change':'mean'})
df2.reset_index(inplace=True)
df2 = df2.sort_values(by=['temp_change'], ascending=True).head(10)

fig, ax = plt.subplots(figsize=(12, 6))

bar2 = plt.barh(df2['country_name'], df2['temp_change'], color='mediumseagreen')

temp = round(df2['temp_change'], 2).to_list()

j = 0
for p in bar2:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    plt.text(x + width * 1.05, y + height * 0.6, str(temp[j]), ha='center', weight='bold')
    j+=1

ax.set_xlim([0, 0.8])
ax.invert_yaxis()
plt.title("Top 10 Areas that have Lowest Temperature Change in The Last Decades", weight='bold')
plt.xlabel("Temperature Change (° C)")
plt.ylabel("Countries")
plt.show()
```



The bar chart shows the average temperature change values of 10 areas that have the lowest temperature change in the last decade. As we can see, there is no developed country on this list. Moreover, India is on this list. Even though India is a developing country and has lots of industrial activities. So, these activities may not have much effect on temperature changes.

The Temperature Change in Svalbard and Jan Mayen Islands and South Georgia and the South Sandwich Islands

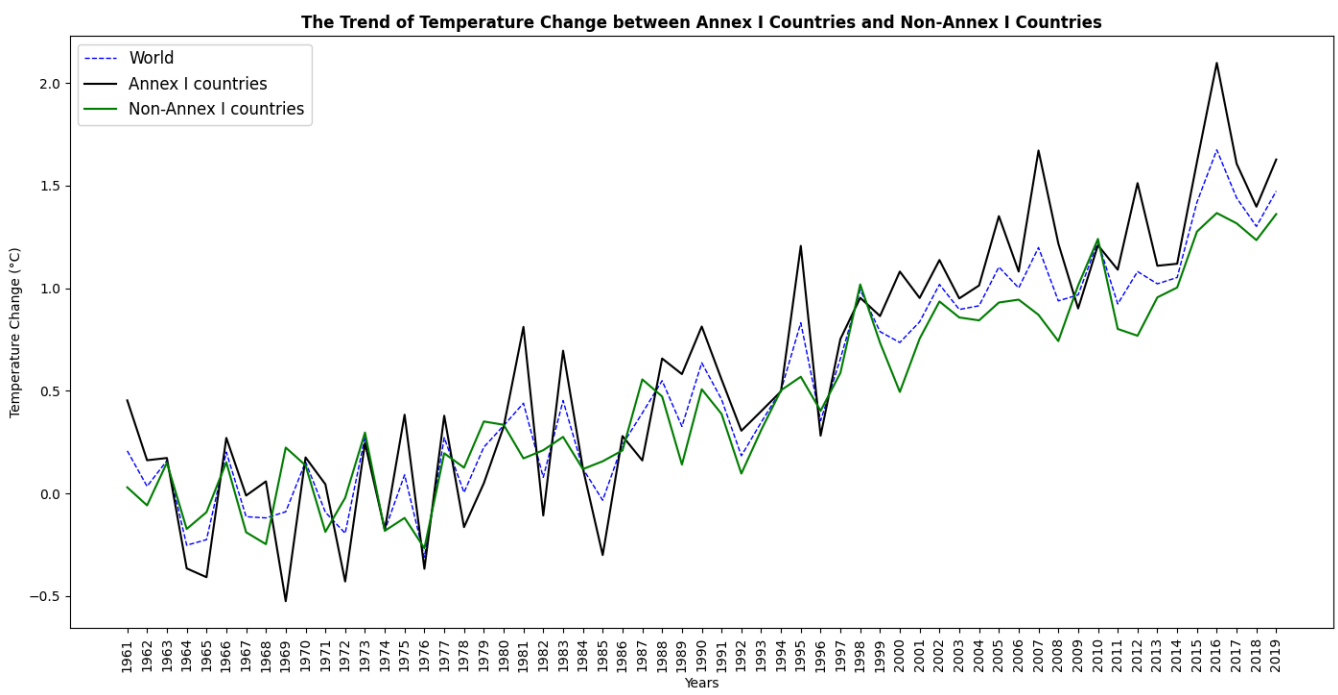
```
df3 = df[df['country_name'] == 'Svalbard and Jan Mayen Islands']
country1 = df3.groupby(['year',]).agg({'temp_change':'mean'})
country1.reset_index(inplace=True)
df4 = df[df['country_name'] == 'South Georgia and the South Sandwich Islands']
country2 = df4.groupby(['year',]).agg({'temp_change':'mean'})
country2.reset_index(inplace=True)

fig = plt.figure(figsize=(18, 10))
plt.subplots_adjust(hspace=0.25)

ax1 = fig.add_subplot(211)
plt.xticks(rotation=90)
plt.plot(country1['year'], country1['temp_change'], 'o--', color='red', linewidth=2)
plt.title("The Temperature Change in Svalbard and Jan Mayen Islands", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")

ax2 = fig.add_subplot(212, sharey=ax1)
plt.xticks(rotation=90)
plt.plot(country2['year'], country2['temp_change'], 'o--', color='mediumseagreen', linewidth=2)
plt.title("The Temperature Change in South Georgia and the South Sandwich Islands", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")

plt.show()
```



The graph shows the trend of average temperature changing each year in Svalbard and Jan Mayen Islands and South Georgia and the South Sandwich Islands. The temperature change in Svalbard and Jan Mayen Islands fluctuation and tend to increase after 1990. On the other hand, the temperature change in South Georgia and the South Sandwich Islands is quite stable. There is no sharp increase or decrease.

The Trend of Temperature Change between Annex I Countries and Non-Annex I Countries

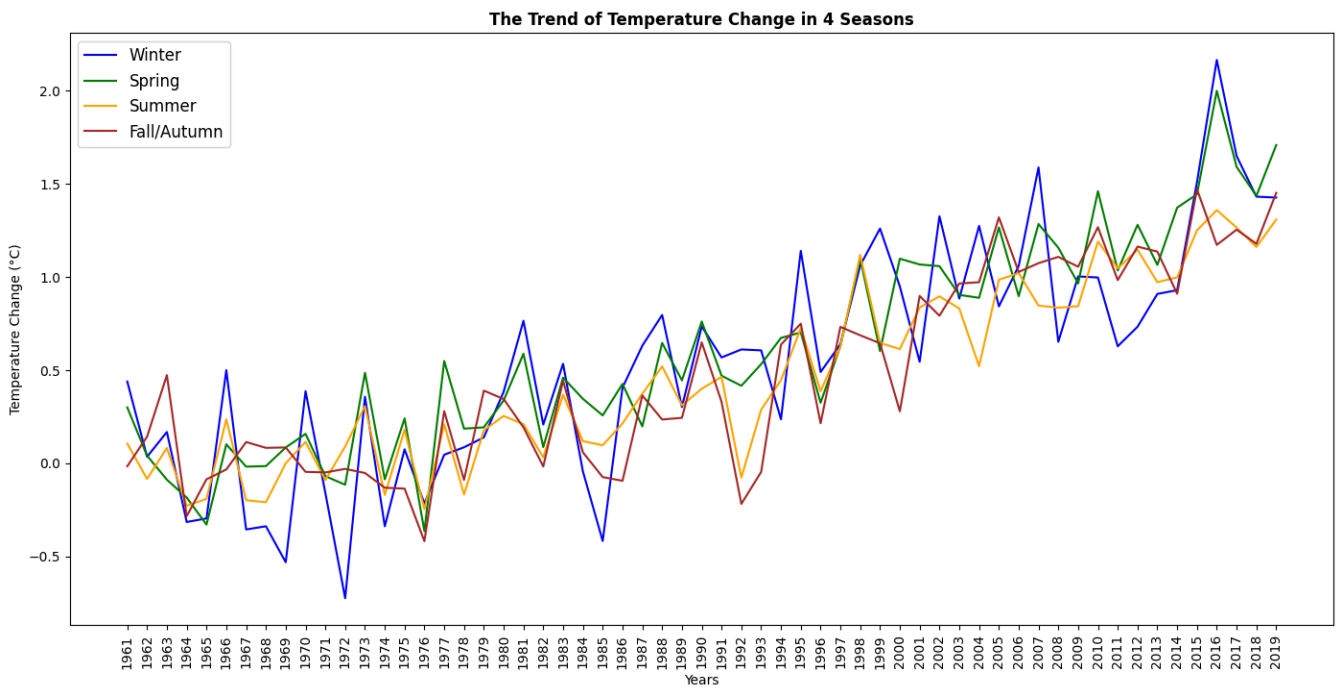
```
df5 = df[df['months'] == 'Meteorological year']
world = df5[df5['country_name'] == 'World']
annexI = df5[df5['country_name'] == 'Annex I countries']
non_annexI = df5[df5['country_name'] == 'Non-Annex I countries']

fig, ax = plt.subplots(figsize=(17, 8))

plt.plot(world['year'], world['temp_change'], '--', color='blue', label='World', linewidth=1)
plt.plot(annexI['year'], annexI['temp_change'], '-', color='black', label='Annex I countries')
plt.plot(non_annexI['year'], non_annexI['temp_change'], '-', color='green', label='Non-Annex I countries')

plt.xticks(rotation=90)
plt.legend(fontsize='large')
plt.title("The Trend of Temperature Change between Annex I Countries and Non-Annex I Countries", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")
plt.show()
```



As we can see, the average temperature in Annex I Countries (developing countries and underdeveloped countries) each year has relatively stable changes. But the average temperature in Non-Annex I Countries (developed countries or industrialized countries) has fluctuated changes.

The Trend of Temperature Change in 4 Seasons

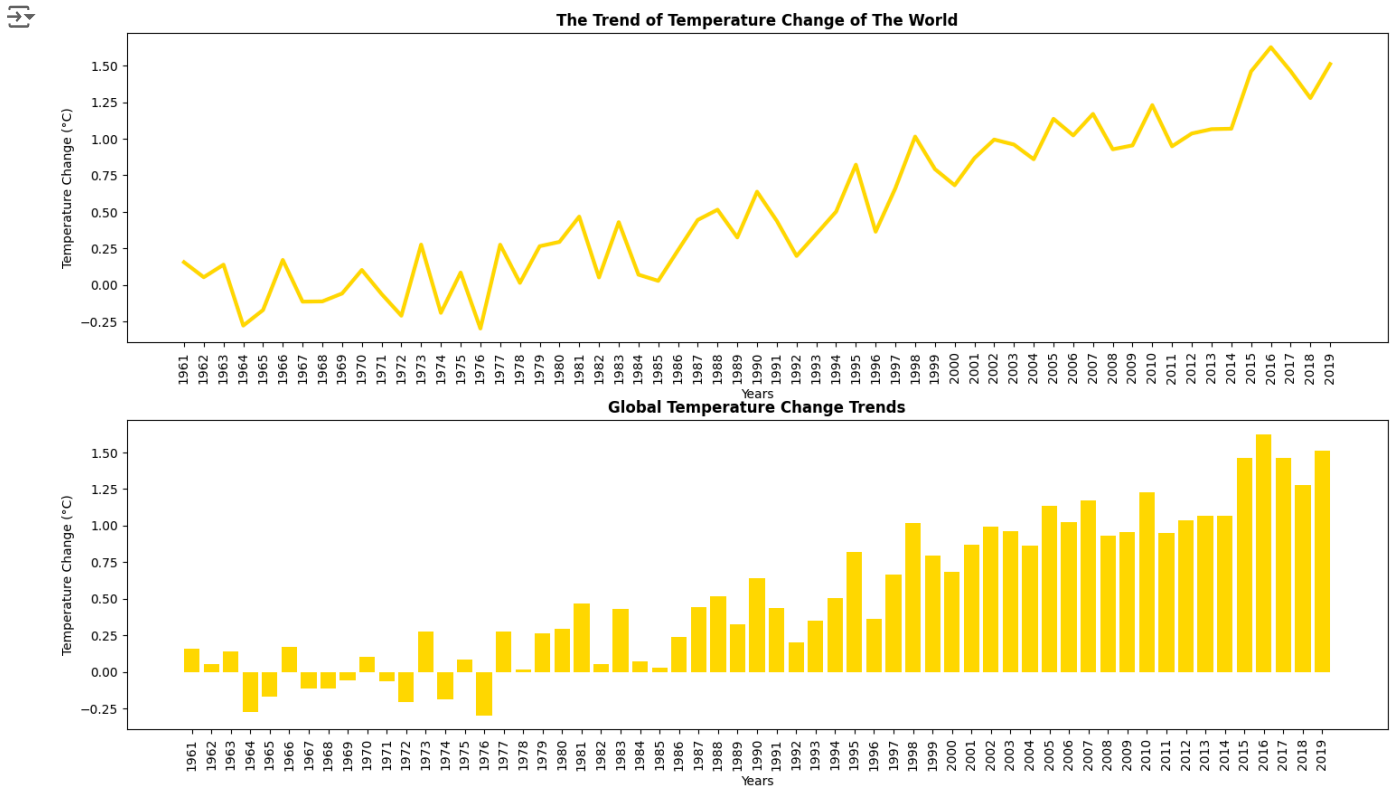
```
df6 = df[df['country_name'] == 'World']
winter = df6[df6['months'] == 'Winter']
spring = df6[df6['months'] == 'Spring']
summer = df6[df6['months'] == 'Summer']
fall = df6[df6['months'] == 'Fall']

fig, ax = plt.subplots(figsize=(17, 8))

plt.plot(winter['year'], winter['temp_change'], '-', color='blue', label='Winter')
plt.plot(spring['year'], spring['temp_change'], '-', color='green', label='Spring')
plt.plot(summer['year'], summer['temp_change'], '-', color='orange', label='Summer')
plt.plot(fall['year'], fall['temp_change'], '-', color='brown', label='Fall/Autumn')

plt.xticks(rotation=90)
plt.legend(fontsize='large')
plt.title("The Trend of Temperature Change in 4 Seasons", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")
plt.show()
```



This line chart compares the trend of temperature changes each year of 4 seasons: winter, spring, summer, and autumn. Summer has a relatively stable change same as Autumn, but Autumn fluctuation in the late 1980s until the early 1990s. Spring has a relatively volatile change. Winter has the most volatile change and a lot of peaks.

Global Temperature Change Trends

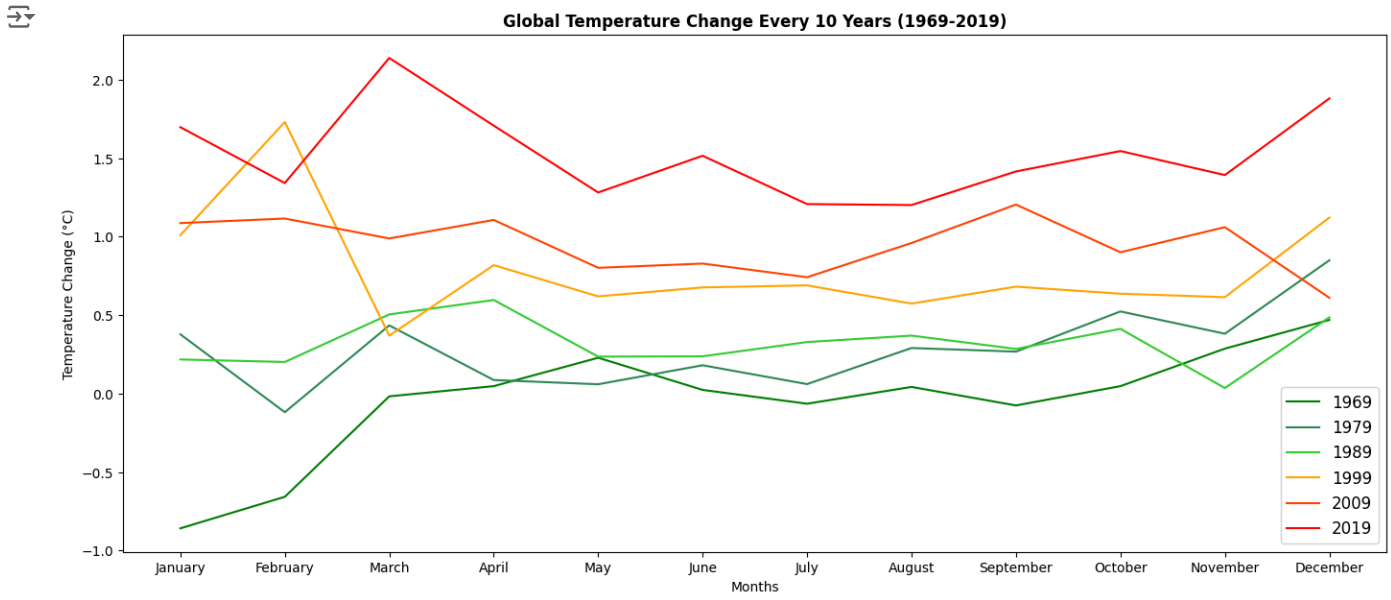
```
df7 = df[df['country_name'] == 'World']
# Select only the relevant columns before grouping
df7 = df7[['year', 'temp_change']].groupby(['year'], as_index=False).mean()

fig = plt.figure(figsize=(18, 10))
plt.subplots_adjust(hspace=0.25)

ax1 = fig.add_subplot(211)
plt.plot(df7['year'], df7['temp_change'], '-', color='gold', linewidth=3)
plt.xticks(rotation=90)
plt.title("The Trend of Temperature Change of The World", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")

ax2 = fig.add_subplot(212)
plt.bar(df7['year'], df7['temp_change'], color='gold')
plt.xticks(rotation=90)
plt.title("Global Temperature Change Trends", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")

plt.show()
```



Both charts show the trend of global temperature change from 1961 to 2019. According to the chart, global temperature changes from 1961 to 1976 are very volatile. There is an unstable increase and decrease. From 1977 onwards, the average temperature does not decrease below 0°C and tends to increase significantly. In addition, 2016 has the highest temperature.

Global Temperature Change Every 10 Years (1969-2019)

```
df8 = df[df['country_name'] == 'World']
y1969 = df8[df8['year'] == '1969'].head(12)
y1979 = df8[df8['year'] == '1979'].head(12)
y1989 = df8[df8['year'] == '1989'].head(12)
y1999 = df8[df8['year'] == '1999'].head(12)
y2009 = df8[df8['year'] == '2009'].head(12)
y2019 = df8[df8['year'] == '2019'].head(12)

fig, ax = plt.subplots(figsize=(17, 7))

plt.plot(y1969['months'], y1969['temp_change'], '-', color='green', label='1969')
plt.plot(y1979['months'], y1979['temp_change'], '-', color='seagreen', label='1979')
plt.plot(y1989['months'], y1989['temp_change'], '-', color='limegreen', label='1989')
plt.plot(y1999['months'], y1999['temp_change'], '-', color='orange', label='1999')
plt.plot(y2009['months'], y2009['temp_change'], '-', color='orangered', label='2009')
plt.plot(y2019['months'], y2019['temp_change'], '-', color='red', label='2019')

plt.legend(fontsize='large')
plt.title("Global Temperature Change Every 10 Years (1969-2019)", weight='bold')
plt.xlabel("Months")
plt.ylabel("Temperature Change (° C)")
plt.show()
```

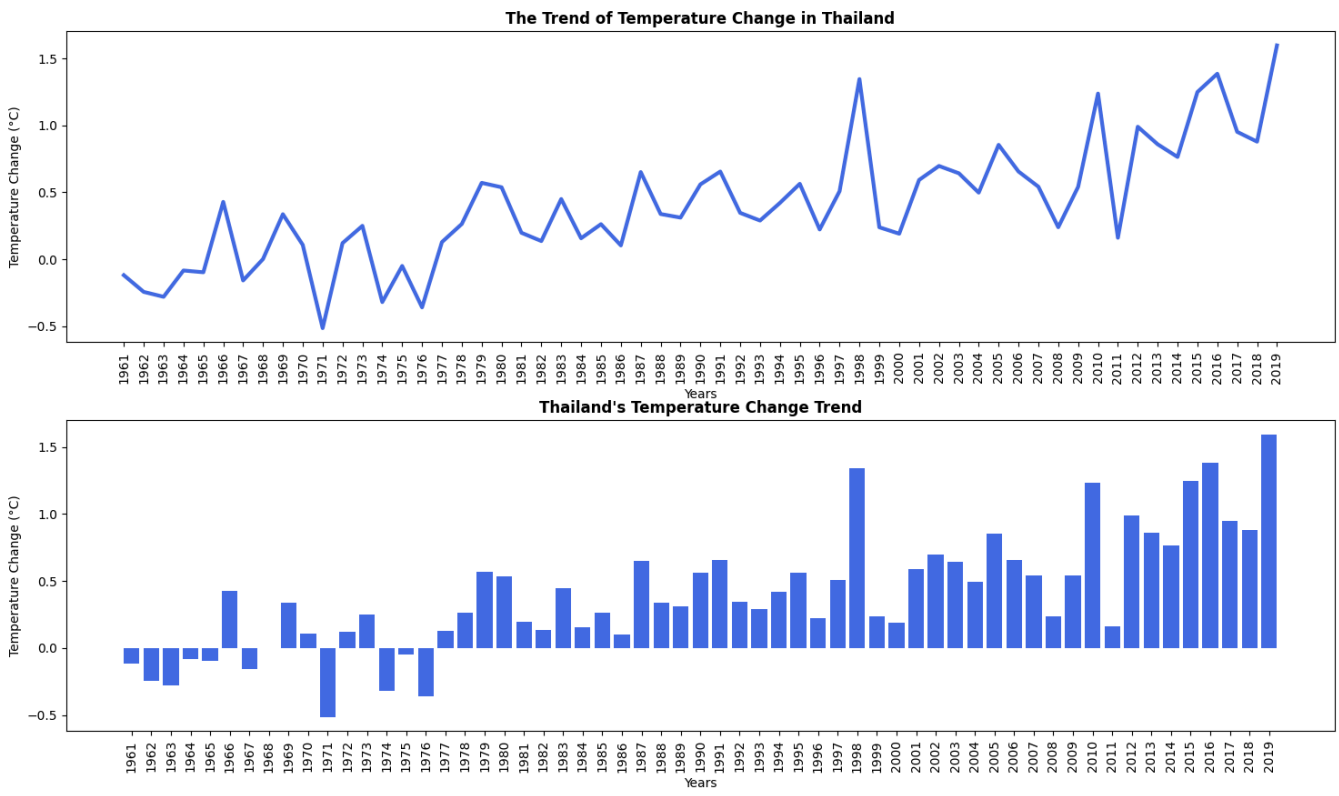


This line chart compares the change in global temperature every 10 years. From the chart, the line of the temperature change each year gradually increases significantly.

Thailand's Temperature Change Trend

```
df9 = df[df['country_name'] == 'Thailand']
df9 = df9.groupby(['year'], as_index=False)['temp_change'].mean() # Select the 'temp_change' column for aggregation

fig = plt.figure(figsize=(18, 10))
plt.subplots_adjust(hspace=0.25)

ax1 = fig.add_subplot(211)
plt.plot(df9['year'], df9['temp_change'], '-', color='royalblue', linewidth=3)
plt.xticks(rotation=90)
plt.title("The Trend of Temperature Change in Thailand", weight='bold')
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")

ax2 = fig.add_subplot(212)
plt.bar(df9['year'], df9['temp_change'], color='royalblue')
plt.xticks(rotation=90)
plt.title("Thailand's Temperature Change Trend", weight='bold')
```

```
plt.xlabel("Years")
plt.ylabel("Temperature Change (° C)")

plt.show()
```



Both charts represent the trend of Thailand's temperature change from 1961 to 2019. From both charts, it can be seen that Thailand's temperature changes are very volatile. There is an unstable increase and decrease. From 1977 to 2019, the average temperature has not dropped below 0°C and there is a noticeable increase in trend. In addition, 2019 has the highest temperature.

Thailand's Temperature Change Every 10 Years (1969-2019)

```
df10 = df[df['country_name'] == 'Thailand']
y1969 = df10[df10['year'] == '1969'].head(12)
y1979 = df10[df10['year'] == '1979'].head(12)
y1989 = df10[df10['year'] == '1989'].head(12)
y1999 = df10[df10['year'] == '1999'].head(12)
y2009 = df10[df10['year'] == '2009'].head(12)
y2019 = df10[df10['year'] == '2019'].head(12)

fig, ax = plt.subplots(figsize=(17, 8))
```