In [1]:
```python
import pandas as pd

# Load the datasets using your actual file paths
temperature_data = pd.read_csv(r'C:\Users\Sharon\Desktop\Docs\Carbon Emissions\t
co2_data = pd.read_csv(r'C:\Users\Sharon\Desktop\Docs\Carbon Emissions\carbon_em

# Preview the first few rows of each dataset
temperature_data_preview = temperature_data.head()
co2_data_preview = co2_data.head()

temperature_data_preview, co2_data_preview
```

Out[1]:
```
(   ObjectId                       Country ISO2 ISO3  F1961  F1962  F1963  \
 0         1  Afghanistan, Islamic Rep. of   AF  AFG -0.113 -0.164  0.847
 1         2                       Albania   AL  ALB  0.627  0.326  0.075
 2         3                       Algeria   DZ  DZA  0.164  0.114  0.077
 3         4                American Samoa   AS  ASM  0.079 -0.042  0.169
 4         5      Andorra, Principality of   AD  AND  0.736  0.112 -0.752

    F1964  F1965  F1966  ...  F2013  F2014  F2015  F2016  F2017  F2018  F2019
\
 0 -0.764 -0.244  0.226  ...  1.281  0.456  1.093  1.555  1.540  1.544  0.910
 1 -0.166 -0.388  0.559  ...  1.333  1.198  1.569  1.464  1.121  2.028  1.675
 2  0.250 -0.100  0.433  ...  1.192  1.690  1.121  1.757  1.512  1.210  1.115
 3 -0.140 -0.562  0.181  ...  1.257  1.170  1.009  1.539  1.435  1.189  1.539
 4  0.308 -0.490  0.415  ...  0.831  1.946  1.690  1.990  1.925  1.919  1.964

    F2020  F2021  F2022
 0  0.498  1.327  2.012
 1  1.498  1.536  1.518
 2  1.926  2.330  1.688
 3  1.430  1.268  1.256
 4  2.562  1.533  3.243

 [5 rows x 66 columns],
    ObjectId Country     Date   Value
 0         1   World  1958M03  315.70
 1         2   World  1958M04  317.45
 2         3   World  1958M05  317.51
 3         4   World  1958M06  317.24
 4         5   World  1958M07  315.86)
```

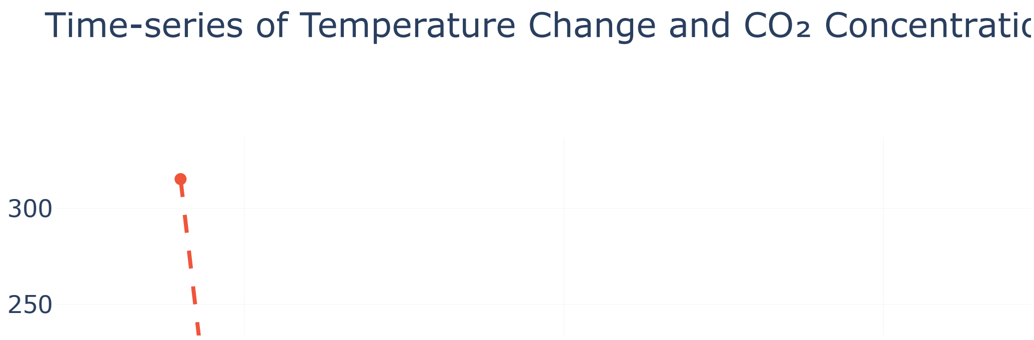
In [2]:
```python
# selecting and computing statistics for temperature changes
temperature_values = temperature_data.filter(regex='^F').stack()  # extracting a
temperature_stats = {
    "Mean": temperature_values.mean(),
    "Median": temperature_values.median(),
    "Variance": temperature_values.var()
}

# computing statistics for CO2 concentrations
co2_values = co2_data["Value"]  # extracting the Value column
co2_stats = {
    "Mean": co2_values.mean(),
    "Median": co2_values.median(),
    "Variance": co2_values.var()
}
```

```python
temperature_stats, co2_stats
```

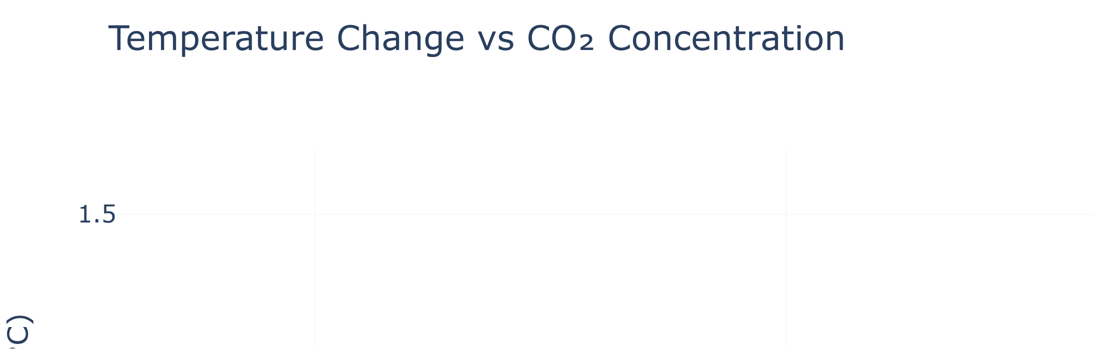
Out[2]:
```
({'Mean': 0.5377713483146068, 'Median': 0.47, 'Variance': 0.4294524831504378},
 {'Mean': 180.71615286624203,
  'Median': 313.835,
  'Variance': 32600.00200469294})
```

In [3]:
```python
import plotly.graph_objects as go
import plotly.express as px

# extracting time-series data for plotting
# temperature: averaging across countries for each year
temperature_years = temperature_data.filter(regex='^F').mean(axis=0)
temperature_years.index = temperature_years.index.str.replace('F', '').astype(in

# CO2: parsing year and averaging monthly data
co2_data['Year'] = co2_data['Date'].str[:4].astype(int)
co2_yearly = co2_data.groupby('Year')['Value'].mean()

# time-series plot for temperature and CO2 levels
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=temperature_years.index, y=temperature_years.values,
    mode='lines+markers', name="Temperature Change (°C)"
))
fig.add_trace(go.Scatter(
    x=co2_yearly.index, y=co2_yearly.values,
    mode='lines+markers', name="CO₂ Concentration (ppm)", line=dict(dash='dash')
))
fig.update_layout(
    title="Time-series of Temperature Change and CO₂ Concentrations",
    xaxis_title="Year",
    yaxis_title="Values",
    template="plotly_white",
    legend_title="Metrics"
)
fig.show()

# correlation heatmap
merged_data = pd.DataFrame({
    "Temperature Change": temperature_years,
    "CO₂ Concentration": co2_yearly
}).dropna()

heatmap_fig = px.imshow(
    merged_data.corr(),
    text_auto=".2f",
    color_continuous_scale="RdBu",  # diverging colormap similar to coolwarm
    title="Correlation Heatmap"
)
heatmap_fig.update_layout(
    template="plotly_white"
)
heatmap_fig.show()

# scatter plot: temperature vs CO2 concentrations
scatter_fig = px.scatter(
    merged_data,
    x="CO₂ Concentration", y="Temperature Change",
```

```
    labels={"CO₂ Concentration": "CO₂ Concentration (ppm)", "Temperature Change"
    title="Temperature Change vs CO₂ Concentration",
    template="plotly_white"
)
scatter_fig.update_traces(marker=dict(size=10, opacity=0.7))
scatter_fig.show()
```



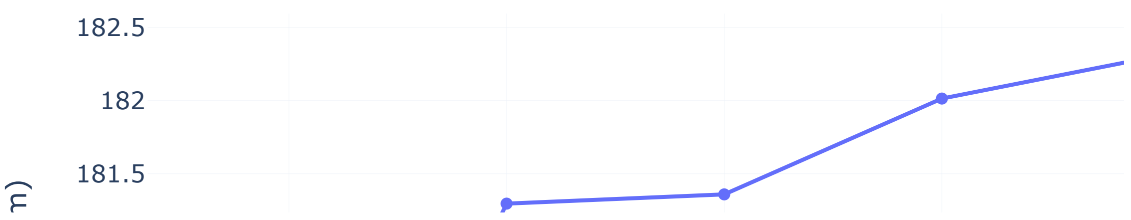
Correlation Heatmap



In [4]:
```
from scipy.stats import linregress

# temperature trend
temp_trend = linregress(temperature_years.index, temperature_years.values)
temp_trend_line = temp_trend.slope * temperature_years.index + temp_trend.interc

# CO2 trend
co2_trend = linregress(co2_yearly.index, co2_yearly.values)
co2_trend_line = co2_trend.slope * co2_yearly.index + co2_trend.intercept

fig_trends = go.Figure()

fig_trends.add_trace(go.Scatter(
    x=temperature_years.index, y=temperature_years.values,
    mode='lines+markers', name="Temperature Change (°C)"
))
fig_trends.add_trace(go.Scatter(
    x=temperature_years.index, y=temp_trend_line,
    mode='lines', name=f"Temperature Trend (Slope: {temp_trend.slope:.2f})", lin
))
fig_trends.add_trace(go.Scatter(
    x=co2_yearly.index, y=co2_yearly.values,
    mode='lines+markers', name="CO₂ Concentration (ppm)"
))
fig_trends.add_trace(go.Scatter(
    x=co2_yearly.index, y=co2_trend_line,
    mode='lines', name=f"CO₂ Trend (Slope: {co2_trend.slope:.2f})", line=dict(da
))
```

```python
fig_trends.update_layout(
    title="Trends in Temperature Change and CO₂ Concentrations",
    xaxis_title="Year",
    yaxis_title="Values",
    template="plotly_white",
    legend_title="Metrics"
)
fig_trends.show()

# seasonal variations in CO2 concentrations
co2_data['Month'] = co2_data['Date'].str[-2:].astype(int)
co2_monthly = co2_data.groupby('Month')['Value'].mean()

fig_seasonal = px.line(
    co2_monthly,
    x=co2_monthly.index,
    y=co2_monthly.values,
    labels={"x": "Month", "y": "CO₂ Concentration (ppm)"},
    title="Seasonal Variations in CO₂ Concentrations",
    markers=True
)
fig_seasonal.update_layout(
    xaxis=dict(tickmode="array", tickvals=list(range(1, 13))),
    template="plotly_white"
)
fig_seasonal.show()
```

Seasonal Variations in CO_2 Concentrations



In [5]:
```
from scipy.stats import pearsonr, spearmanr
from statsmodels.tsa.stattools import grangercausalitytests

# pearson and spearman correlation coefficients
pearson_corr, _ = pearsonr(merged_data["CO₂ Concentration"], merged_data["Temper
spearman_corr, _ = spearmanr(merged_data["CO₂ Concentration"], merged_data["Temp

# granger causality test
granger_data = merged_data.diff().dropna()  # first differencing to make data st
granger_results = grangercausalitytests(granger_data, maxlag=3, verbose=False)

# extracting p-values for causality
granger_p_values = {f"Lag {lag}": round(results[0]['ssr_chi2test'][1], 4)
                    for lag, results in granger_results.items()}

pearson_corr, spearman_corr, granger_p_values
```

```
C:\Users\Sharon\anaconda3\Lib\site-packages\statsmodels\tsa\stattools.py:1488: Fu
tureWarning:

verbose is deprecated since functions should not print results
```

Out[5]:
```
(0.9554282559257312,
 0.9379013371609882,
 {'Lag 1': 0.0617, 'Lag 2': 0.6754, 'Lag 3': 0.2994})
```

In [6]:
```
import statsmodels.api as sm

# creating lagged CO2 data to investigate lagged effects
merged_data['CO₂ Lag 1'] = merged_data["CO₂ Concentration"].shift(1)
merged_data['CO₂ Lag 2'] = merged_data["CO₂ Concentration"].shift(2)
merged_data['CO₂ Lag 3'] = merged_data["CO₂ Concentration"].shift(3)

# dropping rows with NaN due to lags
lagged_data = merged_data.dropna()

X = lagged_data[['CO₂ Concentration', 'CO₂ Lag 1', 'CO₂ Lag 2', 'CO₂ Lag 3']]
y = lagged_data['Temperature Change']
X = sm.add_constant(X)  # adding a constant for intercept

model = sm.OLS(y, X).fit()

model_summary = model.summary()
model_summary
```

Out[6]: OLS Regression Results

Dep. Variable:	Temperature Change	**R-squared:**	0.949
Model:	OLS	**Adj. R-squared:**	0.945
Method:	Least Squares	**F-statistic:**	252.5
Date:	Fri, 27 Dec 2024	**Prob (F-statistic):**	2.97e-34
Time:	16:36:48	**Log-Likelihood:**	45.098
No. Observations:	59	**AIC:**	-80.20
Df Residuals:	54	**BIC:**	-69.81
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-4.7980	0.317	-15.137	0.000	-5.434	-4.163
CO_2 Concentration	0.3245	0.055	5.942	0.000	0.215	0.434
CO_2 Lag 1	-0.2962	0.068	-4.361	0.000	-0.432	-0.160
CO_2 Lag 2	0.0104	0.068	0.153	0.879	-0.126	0.146
CO_2 Lag 3	-0.0107	0.056	-0.191	0.849	-0.123	0.101

Omnibus:	2.369	**Durbin-Watson:**	1.554
Prob(Omnibus):	0.306	**Jarque-Bera (JB):**	2.077
Skew:	-0.457	**Prob(JB):**	0.354
Kurtosis:	2.902	**Cond. No.**	7.54e+03

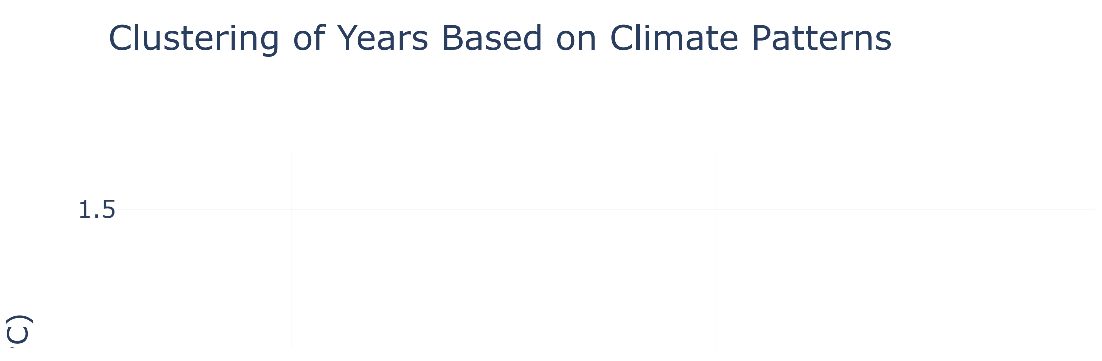
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.54e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [7]:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import numpy as np

# preparing the data for clustering
clustering_data = merged_data[["Temperature Change", "CO₂ Concentration"]].dropn

scaler = StandardScaler()
scaled_data = scaler.fit_transform(clustering_data)

# applying K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)  # assuming 3 clusters for simpli
clustering_data['Cluster'] = kmeans.fit_predict(scaled_data)
```

```python
# adding labels for periods with similar climate patterns
clustering_data['Label'] = clustering_data['Cluster'].map({
    0: 'Moderate Temp & CO₂',
    1: 'High Temp & CO₂',
    2: 'Low Temp & CO₂'
})

import plotly.express as px

fig_clusters = px.scatter(
    clustering_data,
    x="CO₂ Concentration",
    y="Temperature Change",
    color="Label",
    color_discrete_sequence=px.colors.qualitative.Set2,
    labels={
        "CO₂ Concentration": "CO₂ Concentration (ppm)",
        "Temperature Change": "Temperature Change (°C)",
        "Label": "Climate Pattern"
    },
    title="Clustering of Years Based on Climate Patterns"
)

fig_clusters.update_layout(
    template="plotly_white",
    legend_title="Climate Pattern"
)

fig_clusters.show()
```

```
C:\Users\Sharon\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: Futur
eWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning

C:\Users\Sharon\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: User
Warning:

KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
```



In [8]:

```
# setting up a simple predictive model using linear regression
from sklearn.linear_model import LinearRegression

# Preparing data
X = merged_data[["CO₂ Concentration"]].values  # CO₂ concentration as input
y = merged_data["Temperature Change"].values   # temperature change as target

model = LinearRegression()
model.fit(X, y)

# function to simulate "what-if" scenarios
def simulate_temperature_change(co2_percentage_change):
    # Calculate new CO2 concentrations
    current_mean_co2 = merged_data["CO₂ Concentration"].mean()
    new_co2 = current_mean_co2 * (1 + co2_percentage_change / 100)

    # predict temperature change
    predicted_temp = model.predict([[new_co2]])
    return predicted_temp[0]

# simulating scenarios
scenarios = {
    "Increase CO₂ by 10%": simulate_temperature_change(10),
    "Decrease CO₂ by 10%": simulate_temperature_change(-10),
    "Increase CO₂ by 20%": simulate_temperature_change(20),
    "Decrease CO₂ by 20%": simulate_temperature_change(-20),
}
```

```
scenarios
```

Out[8]:

```
{'Increase CO₂ by 10%': 1.0866445037958155,
 'Decrease CO₂ by 10%': -0.059993041237237144,
 'Increase CO₂ by 20%': 1.6599632763123413,
 'Decrease CO₂ by 20%': -0.633311813753763}
```

In [ ]:

```
```