```
import numpy as np # For Linear Algebra
import pandas as pd # To Work With Data
# for visualizations
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from datetime import datetime # Time Series analysis.
```

```
df = pd.read_csv("/content/Weather.csv")
```

```
df.head()
```

	Unnamed: 0	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
0	0	1901	17.99	19.43	23.49	26.41	28.28	28.60	27.49	26.98	26.26	25.08	21.73	18.95
1	1	1902	19.00	20.39	24.10	26.54	28.68	28.44	27.29	27.05	25.95	24.37	21.33	18.78
2	2	1903	18.32	19.79	22.46	26.03	27.93	28.41	28.04	26.63	26.34	24.57	20.96	18.29
3	3	1904	17.77	19.39	22.95	26.73	27.83	27.85	26.84	26.73	25.84	24.36	21.07	18.84

```
df = pd.read_csv("/content/Weather.csv", index_col=0)
```

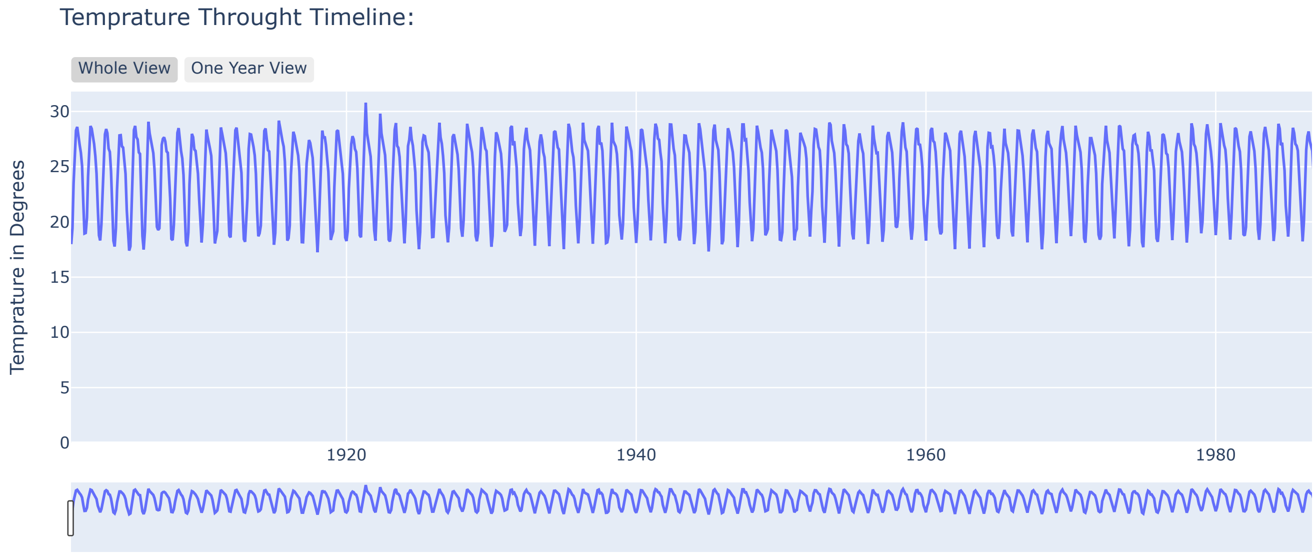
```
df1 = pd.melt(df, id_vars='YEAR', value_vars=df.columns[1:])
df1.head()
```

	YEAR	variable	value
0	1901	JAN	17.99
1	1902	JAN	19.00
2	1903	JAN	18.32
3	1904	JAN	17.77

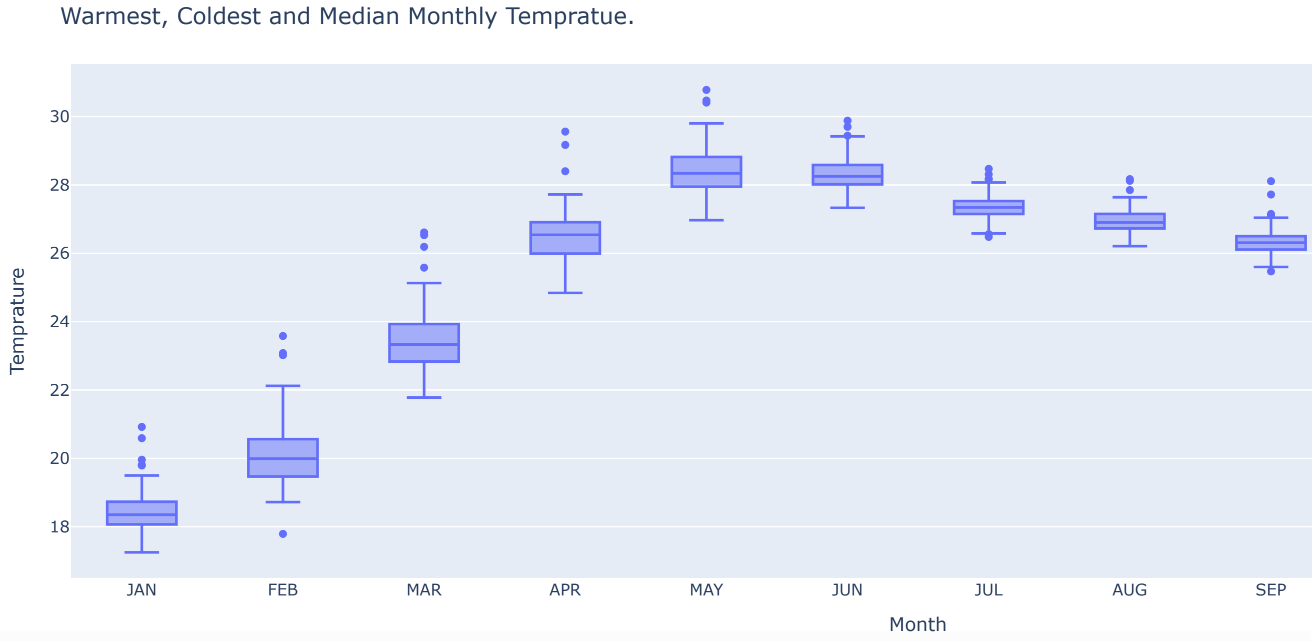
```
df1['Date'] = df1['variable'] + ' ' + df1['YEAR'].astype(str)
df1.loc[:,'Date'] = df1['Date'].apply(lambda x : datetime.strptime(x, '%b %Y')) ## Converting String to datetime object
df1.head()
```

	YEAR	variable	value	Date
0	1901	JAN	17.99	1901-01-01 00:00:00
1	1902	JAN	19.00	1902-01-01 00:00:00
2	1903	JAN	18.32	1903-01-01 00:00:00
3	1904	JAN	17.77	1904-01-01 00:00:00
4	1905	JAN	17.40	1905-01-01 00:00:00

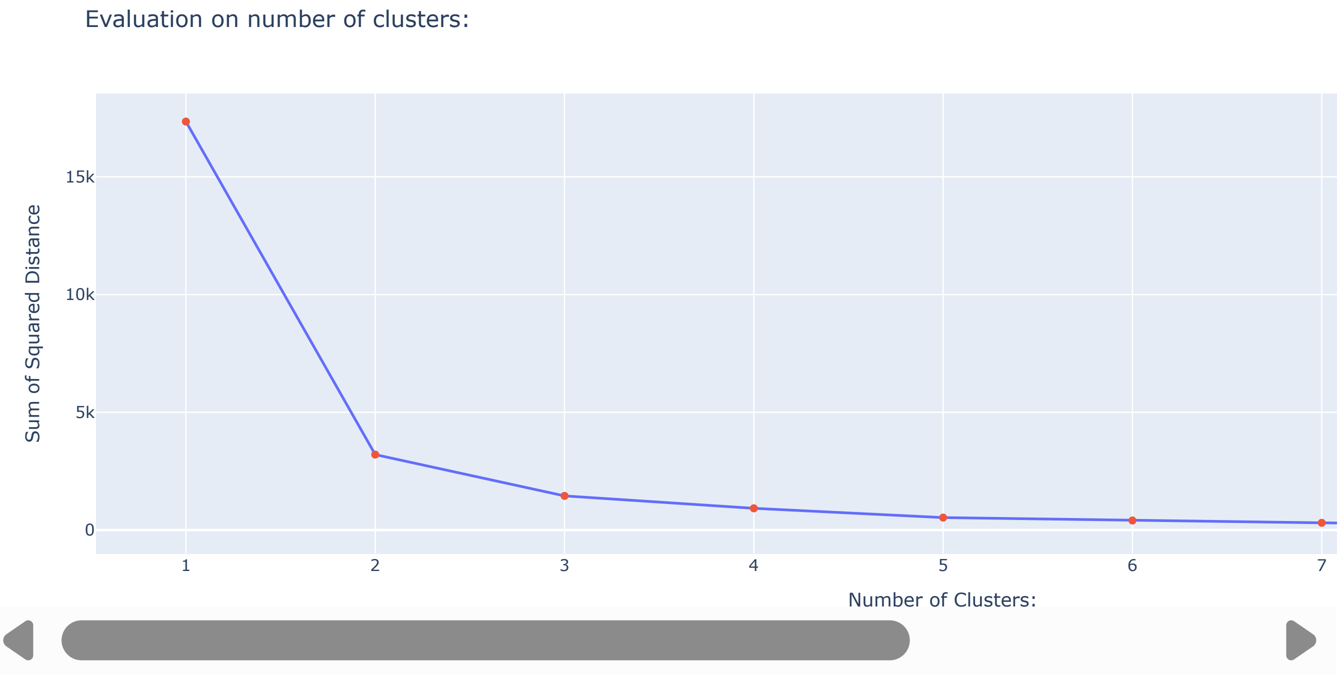
```
df1.columns=['Year', 'Month', 'Temprature', 'Date']
df1.sort_values(by='Date', inplace=True) ## To get the time series right.
fig = go.Figure(layout = go.Layout(yaxis=dict(range=[0, df1['Temprature'].max()+1])))
fig.add_trace(go.Scatter(x=df1['Date'], y=df1['Temprature']), )
fig.update_layout(title='Temprature Throught Timeline:',
                 xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.update_layout(xaxis=go.layout.XAxis(
    rangeselector=dict(
        buttons=list([dict(label="Whole View", step="all"),
                      dict(count=1,label="One Year View",step="year",stepmode="todate")
                     ])),
        rangeslider=dict(visible=True),type="date")
)
fig.show()
```



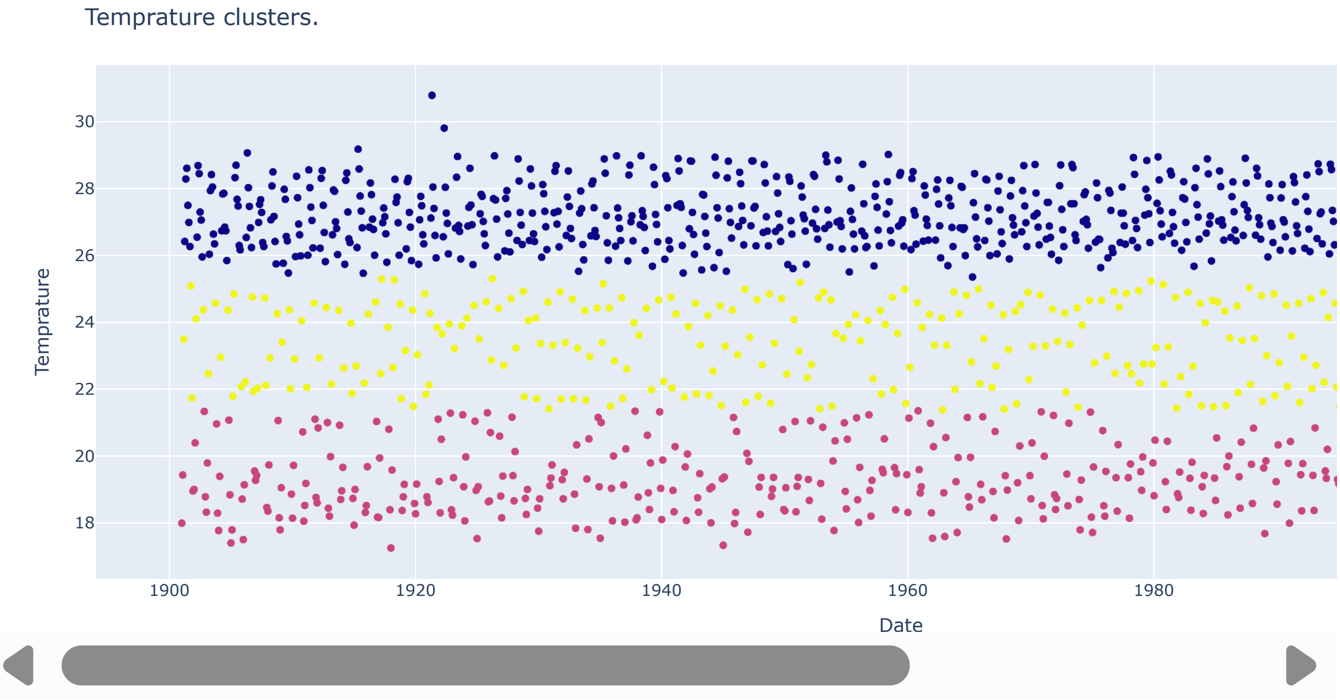
```
fig = px.box(df1, 'Month', 'Temprature')
fig.update_layout(title='Warmest, Coldest and Median Monthly Tempratue.')
fig.show()
```



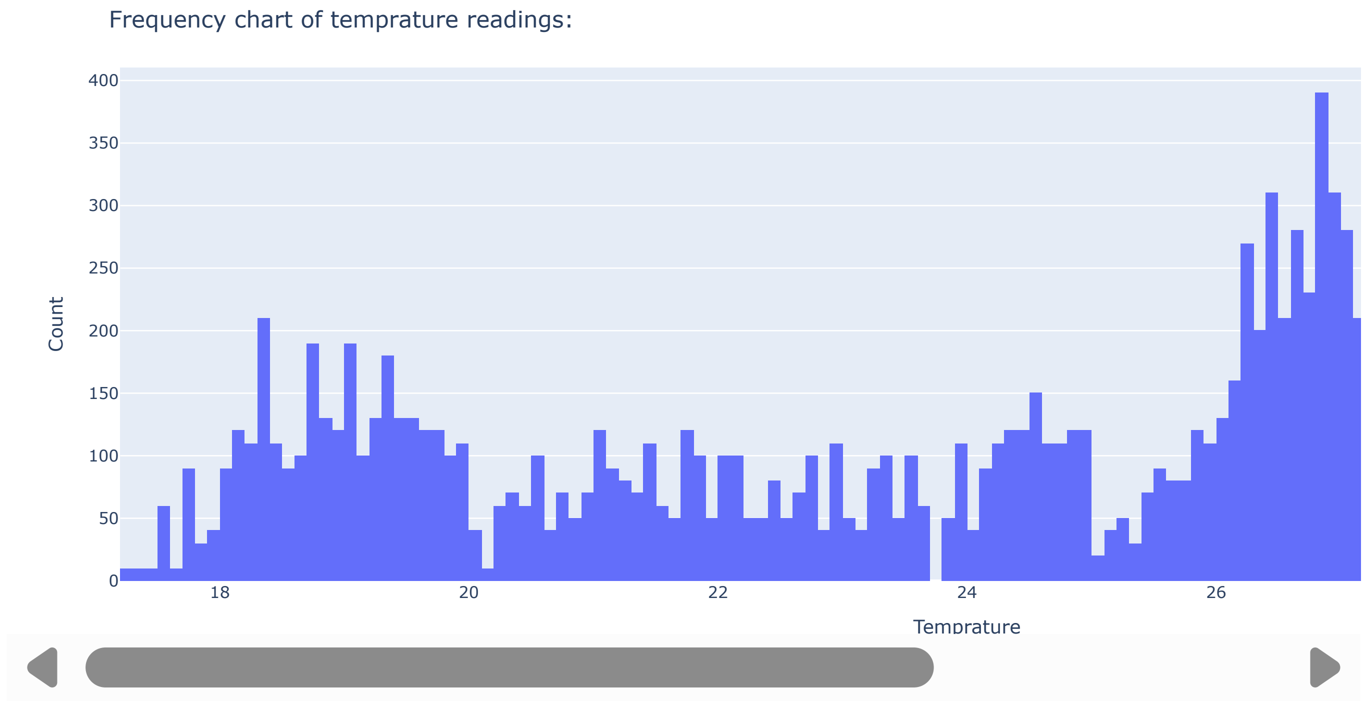
```
from sklearn.cluster import KMeans
sse = []
target = df1['Temprature'].to_numpy().reshape(-1,1)
num_clusters = list(range(1, 10))

for k in num_clusters:
    km = KMeans(n_clusters=k)
    km.fit(target)
    sse.append(km.inertia_)

fig = go.Figure(data=[
    go.Scatter(x = num_clusters, y=sse, mode='lines'),
    go.Scatter(x = num_clusters, y=sse, mode='markers')
])

fig.update_layout(title="Evaluation on number of clusters:",
                 xaxis_title = "Number of Clusters:",
                 yaxis_title = "Sum of Squared Distance",
                 showlegend=False)
fig.show()
```



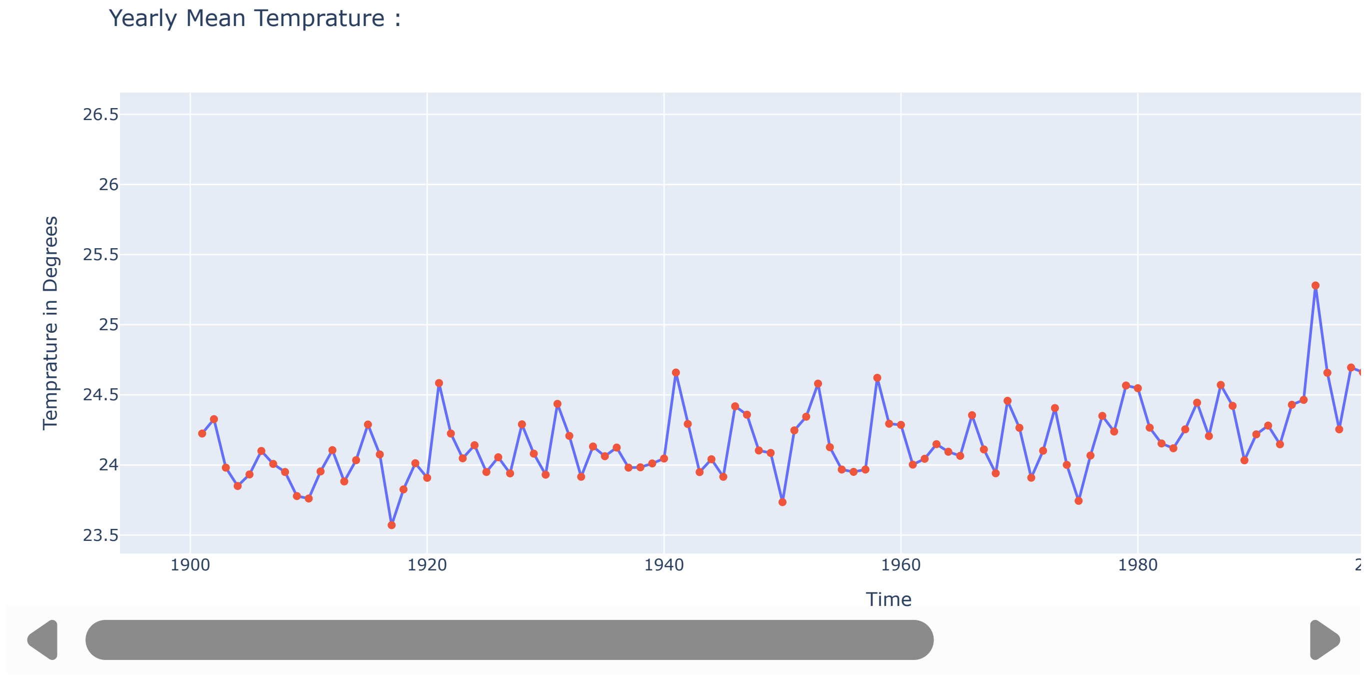
```
km = KMeans(3)
km.fit(df1['Temprature'].to_numpy().reshape(-1,1))
df1.loc[:,'Temp Labels'] = km.labels_
fig = px.scatter(df1, 'Date', 'Temprature', color='Temp Labels')
fig.update_layout(title = "Temprature clusters.",
                 xaxis_title="Date", yaxis_title="Temprature")
fig.show()
```



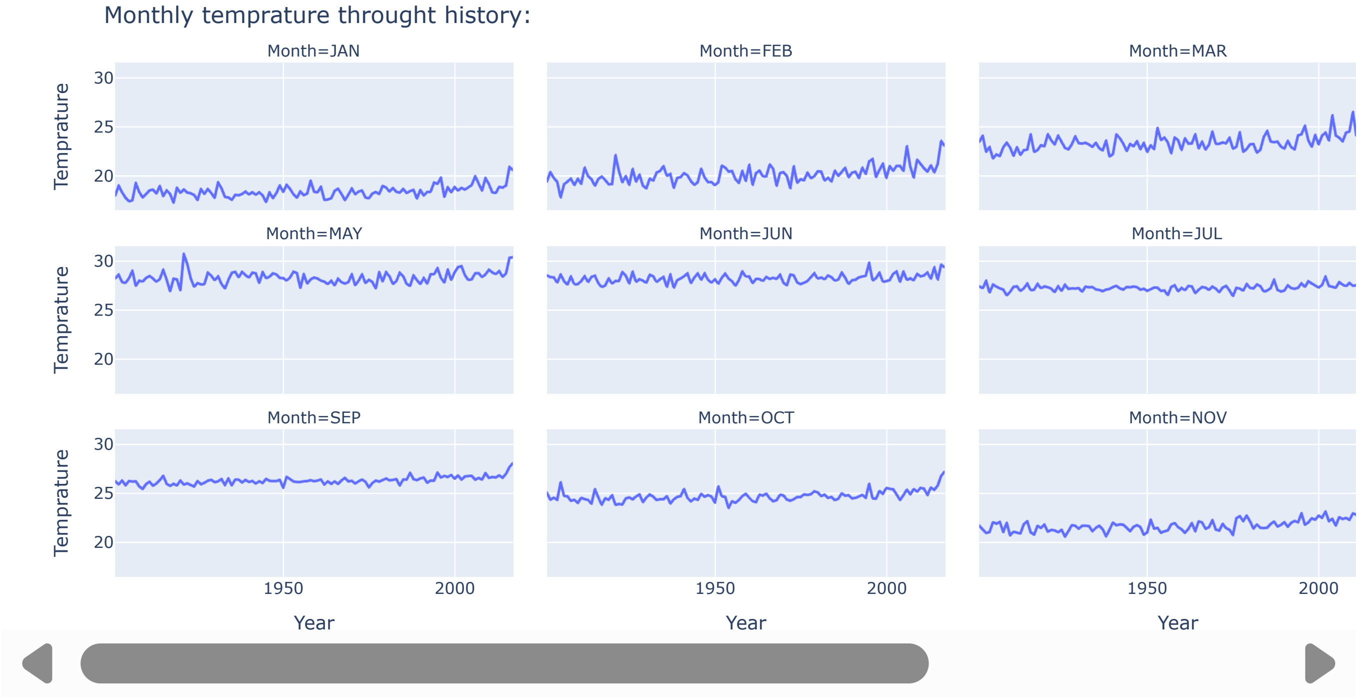
```
fig = px.histogram(x=df1['Temprature'], nbins=200, histnorm='density')
fig.update_layout(title='Frequency chart of temprature readings:',
                 xaxis_title='Temprature', yaxis_title='Count')
```



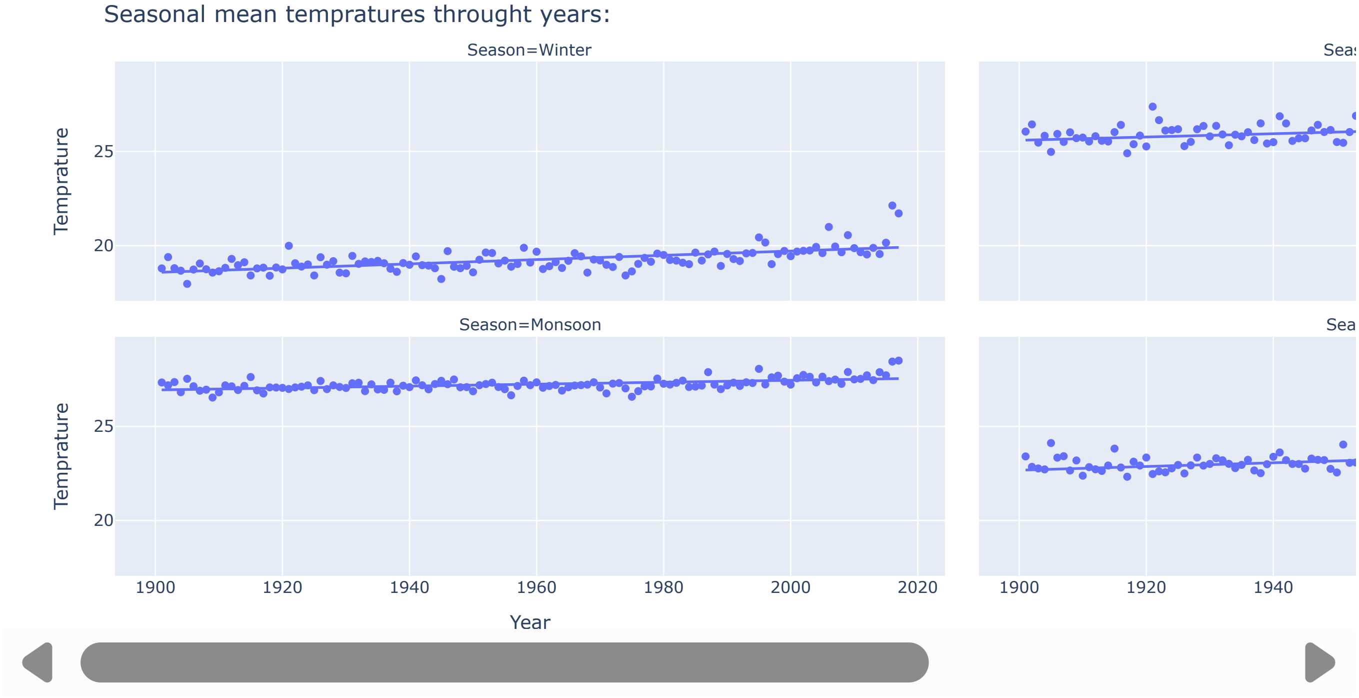
```
df['Yearly Mean'] = df.iloc[:,1:].mean(axis=1) ## Axis 1 for row wise and axis 0 for columns.
fig = go.Figure(data=[
    go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly Mean'], mode='lines'),
    go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly Mean'], mode='markers')
])
fig.update_layout(title='Yearly Mean Temprature :',
                 xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.show()
```



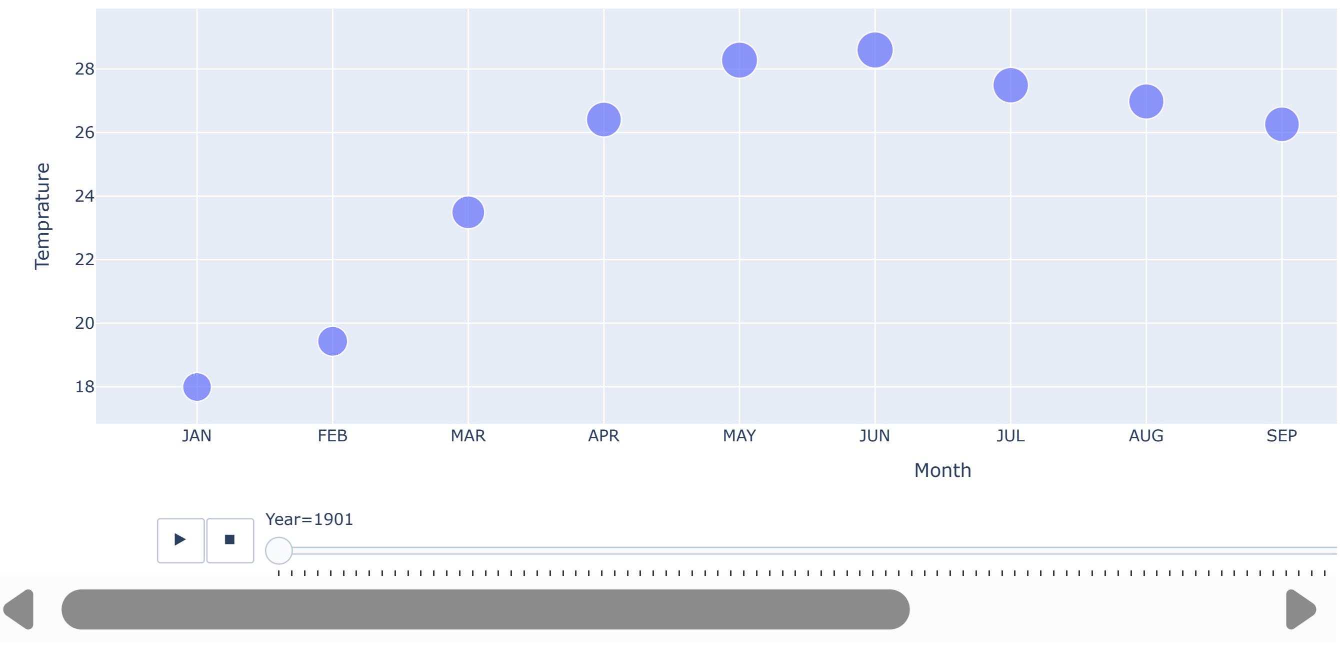
```
fig = px.line(df1, 'Year', 'Temprature', facet_col='Month', facet_col_wrap=4)
fig.update_layout(title='Monthly temprature throught history:')
fig.show()
```



```
df['Winter'] = df[['DEC', 'JAN', 'FEB']].mean(axis=1)
df['Summer'] = df[['MAR', 'APR', 'MAY']].mean(axis=1)
df['Monsoon'] = df[['JUN', 'JUL', 'AUG', 'SEP']].mean(axis=1)
df['Autumn'] = df[['OCT', 'NOV']].mean(axis=1)
seasonal_df = df[['YEAR', 'Winter', 'Summer', 'Monsoon', 'Autumn']]
seasonal_df = pd.melt(seasonal_df, id_vars='YEAR', value_vars=seasonal_df.columns[1:])
seasonal_df.columns=['Year', 'Season', 'Temprature']

fig = px.scatter(seasonal_df, 'Year', 'Temprature', facet_col='Season', facet_col_wrap=2, trendline='ols')
fig.update_layout(title='Seasonal mean tempratures throught years:')
fig.show()
```



```
px.scatter(df1, 'Month', 'Temprature', size='Temprature', animation_frame='Year')
```



```
# I am using decision tree regressor for prediction as the data does not actually have a linear trend.
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

df2 = df1[['Year', 'Month', 'Temprature']].copy()
df2 = pd.get_dummies(df2)
y = df2[['Temprature']]
x = df2.drop(columns='Temprature')

dtr = DecisionTreeRegressor()
train_x, test_x, train_y, test_y = train_test_split(x,y,test_size=0.3)
dtr.fit(train_x, train_y)
pred = dtr.predict(test_x)
r2_score(test_y, pred)
```

```
0.9655452131725264
```

```
next_Year = df1[df1['Year']==2017][['Year', 'Month']]
next_Year.Year.replace(2017,2018, inplace=True)
next_Year= pd.get_dummies(next_Year)
temp_2018 = dtr.predict(next_Year)

temp_2018 = {'Month':df1['Month'].unique(), 'Temprature':temp_2018}
temp_2018=pd.DataFrame(temp_2018)
temp_2018['Year'] = 2018
temp_2018
```

```
<ipython-input-17-2838019938>:2: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col
```

```
forecasted_temp = pd.concat([df1,temp_2018], sort=False).groupby(by='Year')['Temprature'].mean().reset_index()
fig = go.Figure(data=[
    go.Scatter(name='Yearly Mean Temprature', x=forecasted_temp['Year'], y=forecasted_temp['Temprature'], mode='lines'),
    go.Scatter(name='Yearly Mean Temprature', x=forecasted_temp ['Year'], y=forecasted_temp['Temprature'], mode='markers')
])
fig.update_layout(title='Forecasted Temprature:',
                 xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.show()
```