Climate Change Analysis & Data Visualisation & forecasting

In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# importing dataset
temp_by_country = pd.read_csv('GlobalLandTemperaturesByCountry.csv'
temp_by_country.head()
```

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

In [3]:
```
temp_by_country
```

Out[3]:

	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
...	...	...	...	...
577457	2013-05-01	19.059	1.022	Zimbabwe
577458	2013-06-01	17.613	0.473	Zimbabwe
577459	2013-07-01	17.000	0.453	Zimbabwe
577460	2013-08-01	19.759	0.717	Zimbabwe
577461	2013-09-01	NaN	NaN	Zimbabwe

577462 rows × 4 columns

In [4]:
```
#datatypes of data
temp_by_country.dtypes
```

Out[4]:
```
dt                               object
AverageTemperature              float64
AverageTemperatureUncertainty   float64
Country                          object
dtype: object
```

In [5]:
```
temp_by_country.shape
```

Out[5]:
```
(577462, 4)
```

In [6]:
```
#information about the dataset
temp_by_country.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 577462 entries, 0 to 577461
Data columns (total 4 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   dt                             577462 non-null  object 
 1   AverageTemperature             544811 non-null  float64
 2   AverageTemperatureUncertainty  545550 non-null  float64
 3   Country                        577462 non-null  object 
dtypes: float64(2), object(2)
memory usage: 17.6+ MB
```

In [7]:
```
#description about the dataset
temp_by_country.describe()
```

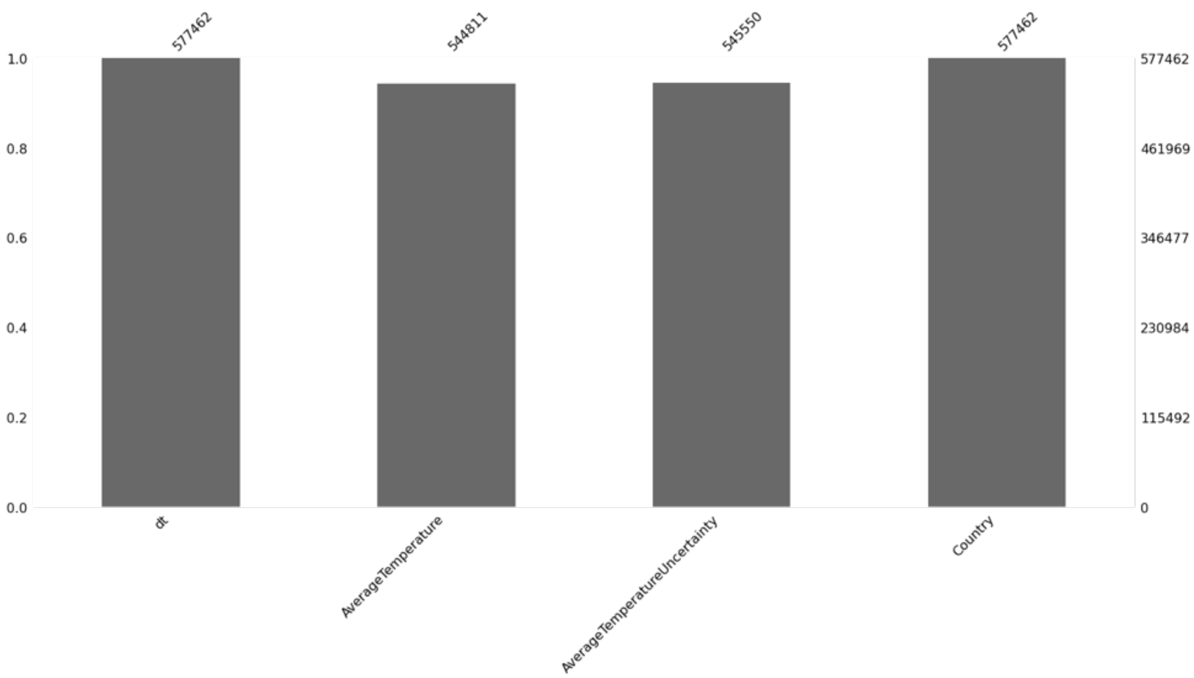
Out[7]:

	AverageTemperature	AverageTemperatureUncertainty
count	544811.000000	545550.000000
mean	17.193354	1.019057
std	10.953966	1.201930
min	-37.658000	0.052000
25%	10.025000	0.323000
50%	20.901000	0.571000
75%	25.814000	1.206000
max	38.842000	15.003000

Checking the null values

In [8]:
```
import missingno as msno
msno.bar(temp_by_country)
```

Out[8]:
```
<AxesSubplot:>
```

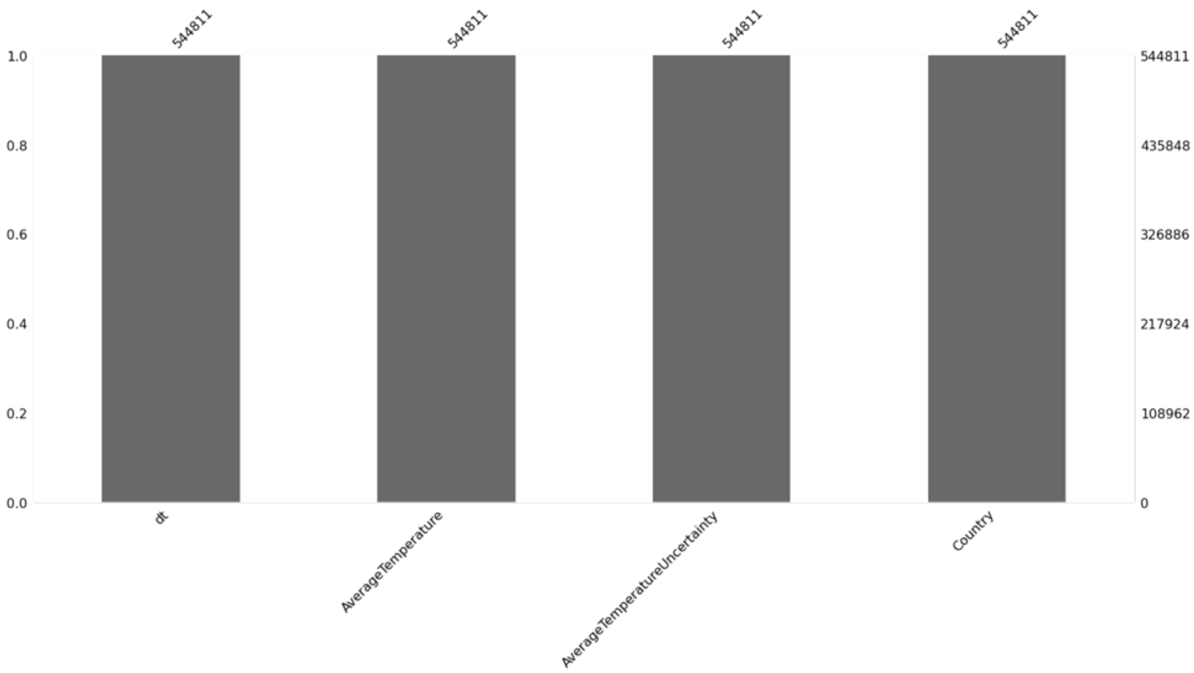


In [9]:
```
temp_by_country.isnull().sum()
temp_by_country = temp_by_country.dropna(how='any' ,axis=0)
temp_by_country.shape
```

Out[9]:
```
(544811, 4)
```

In [10]:
```
import missingno as msno
msno.bar(temp_by_country)
#no missing data, after dropping 'nan'
```

Out[10]: <AxesSubplot:>



Exploratory Data Analysis

Is there any global warming ?

Firstly,we seprate the year from the date column

In [11]:
```
temp_by_country['dt'][0].split('-')[0]
```

Out[11]: '1743'

In [12]:
```
def fetch_year(date):
    return date.split('-')[0]
```

In [13]:
```
temp_by_country['years']=temp_by_country['dt'].apply(fetch_year)
```

In [14]:
```
temp_by_country.head()
```

Out[14]:

	dt	AverageTemperature	AverageTemperatureUncertainty	Country	years
0	1743-11-01	4.384	2.294	Åland	1743
5	1744-04-01	1.530	4.680	Åland	1744
6	1744-05-01	6.702	1.789	Åland	1744
7	1744-06-01	11.609	1.577	Åland	1744
8	1744-07-01	15.342	1.410	Åland	1744

In [15]:
```
by_country=temp_by_country[['Country','AverageTemperature']].groupb
byx=pd.concat([by_country[20:23],by_country[72:75]],axis=0)
```

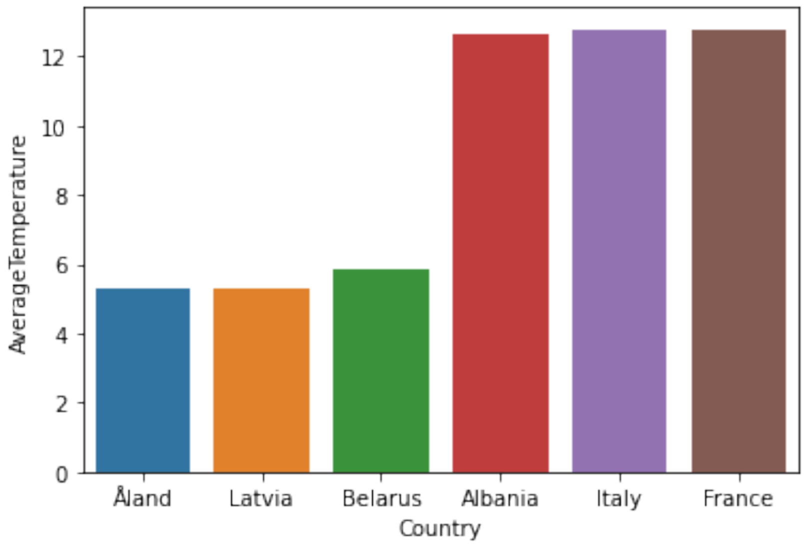
In [16]:
```
byx
```

Out[16]:

Country	AverageTemperature
Åland	5.291383
Latvia	5.320545
Belarus	5.819288
Albania	12.610646
Italy	12.737122
France	12.772446

In [17]:
```
sns.barplot(x=byx.index, y=byx['AverageTemperature'][:])
#average temperature by country
```

Out[17]: <AxesSubplot:xlabel='Country', ylabel='AverageTemperature'>



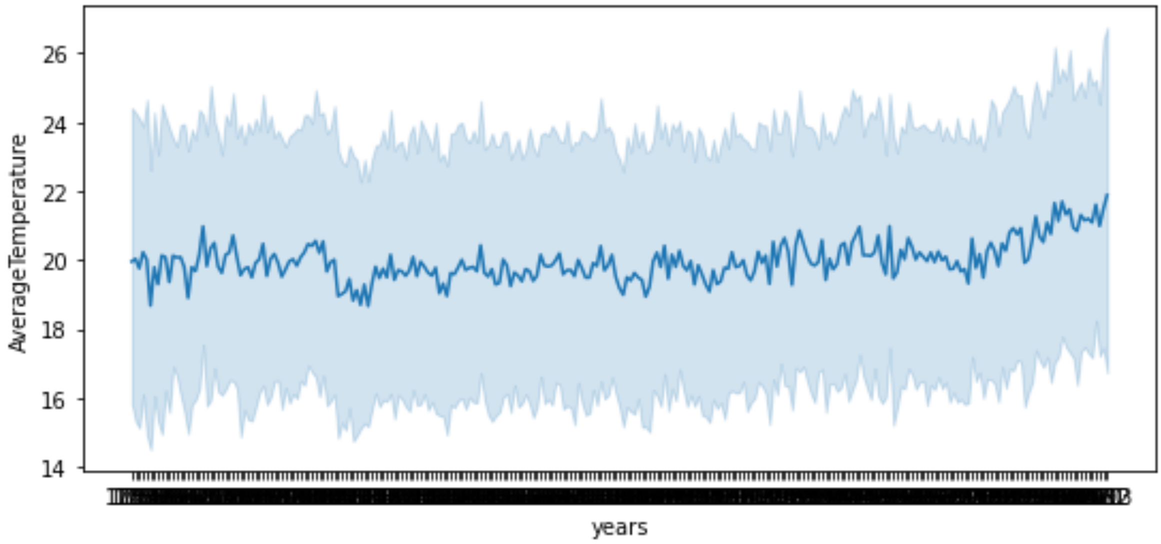
Analysing Tunisia Climate Change

In [18]:
```
temp_Tunisia=temp_by_country.loc[(temp_by_country.Country == 'Tunis
temp_Tunisia['dt'] = pd.to_datetime(temp_Tunisia['dt'])
temp_Tunisia.set_index('dt',inplace = True)
temp_Tunisia['AverageTemperature'].mean()
```

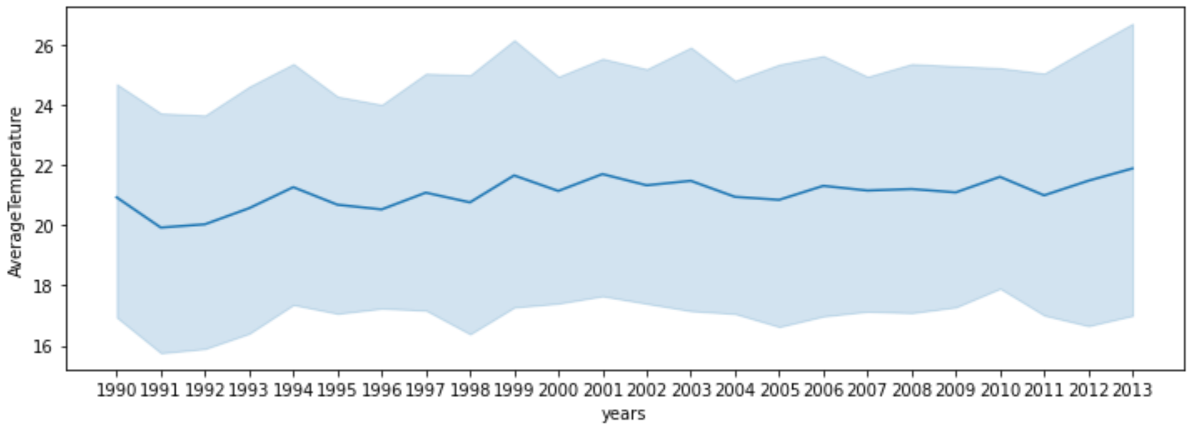
Out[18]: 19.97059047314578

Data Visualisation

In [19]:
```
plt.figure(figsize=(9,4))
sns.lineplot(x = "years" , y = "AverageTemperature",data=temp_Tunis
plt.show()
```



In [20]:
```
temp_Tunisia_latest=temp_Tunisia.loc['1990':'2013']
plt.figure(figsize=(12,4))
sns.lineplot(x = "years" , y = "AverageTemperature",data=temp_Tunis
plt.show()
```



In [21]:
```
temp_Tunisia.index
```

Out[21]:
```
DatetimeIndex(['1753-01-01', '1753-02-01', '1753-03-01', '1753-04-
01',
               '1753-05-01', '1753-06-01', '1753-07-01', '1753-08-
01',
               '1753-09-01', '1753-10-01',
               ...
               '2012-11-01', '2012-12-01', '2013-01-01', '2013-02-
01',
               '2013-03-01', '2013-04-01', '2013-05-01', '2013-06-
01',
               '2013-07-01', '2013-08-01'],
              dtype='datetime64[ns]', name='dt', length=3128, freq
=None)
```

In [22]:
```
temp_Tunisia['month']=temp_Tunisia.index.month
temp_Tunisia
```

Out[22]:

	AverageTemperature	AverageTemperatureUncertainty	Country	years	month
dt					
1753-01-01	8.754	5.363	Tunisia	1753	1
1753-02-01	10.597	3.183	Tunisia	1753	2
1753-03-01	16.105	2.805	Tunisia	1753	3
1753-04-01	18.181	5.257	Tunisia	1753	4
1753-05-01	23.571	2.230	Tunisia	1753	5
...	...	...	...	...	...
2013-04-01	20.383	0.746	Tunisia	2013	4
2013-05-01	24.268	0.336	Tunisia	2013	5
2013-06-01	27.488	0.947	Tunisia	2013	6
2013-07-01	31.156	0.753	Tunisia	2013	7
2013-08-01	30.399	0.795	Tunisia	2013	8

3128 rows × 5 columns

In [23]:

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

In [24]:

```
temp_Tunisia['season']=temp_Tunisia['month'].apply(get_season)
years=temp_Tunisia['years'].unique()
temp_Tunisia.head()
```

Out[24]:

	AverageTemperature	AverageTemperatureUncertainty	Country	years	month	seasor
dt						
1753-01-01	8.754	5.363	Tunisia	1753	1	winte
1753-02-01	10.597	3.183	Tunisia	1753	2	winte
1753-03-01	16.105	2.805	Tunisia	1753	3	spring
1753-04-01	18.181	5.257	Tunisia	1753	4	spring
1753-05-01	23.571	2.230	Tunisia	1753	5	spring

In [25]:
```
spring_temp = []
summer_temp = []
autumn_temp = []
winter_temp = []
for year in years:
    current_df=temp_Tunisia[temp_Tunisia['years'] == year]
    spring_temp.append(current_df[current_df['season'] == 'spring']
    summer_temp.append(current_df[current_df['season'] == 'summer']
    autumn_temp.append(current_df[current_df['season'] == 'autumn']
    winter_temp.append(current_df[current_df['season'] == 'winter']
```

In [26]:
```
len(spring_temp)
```

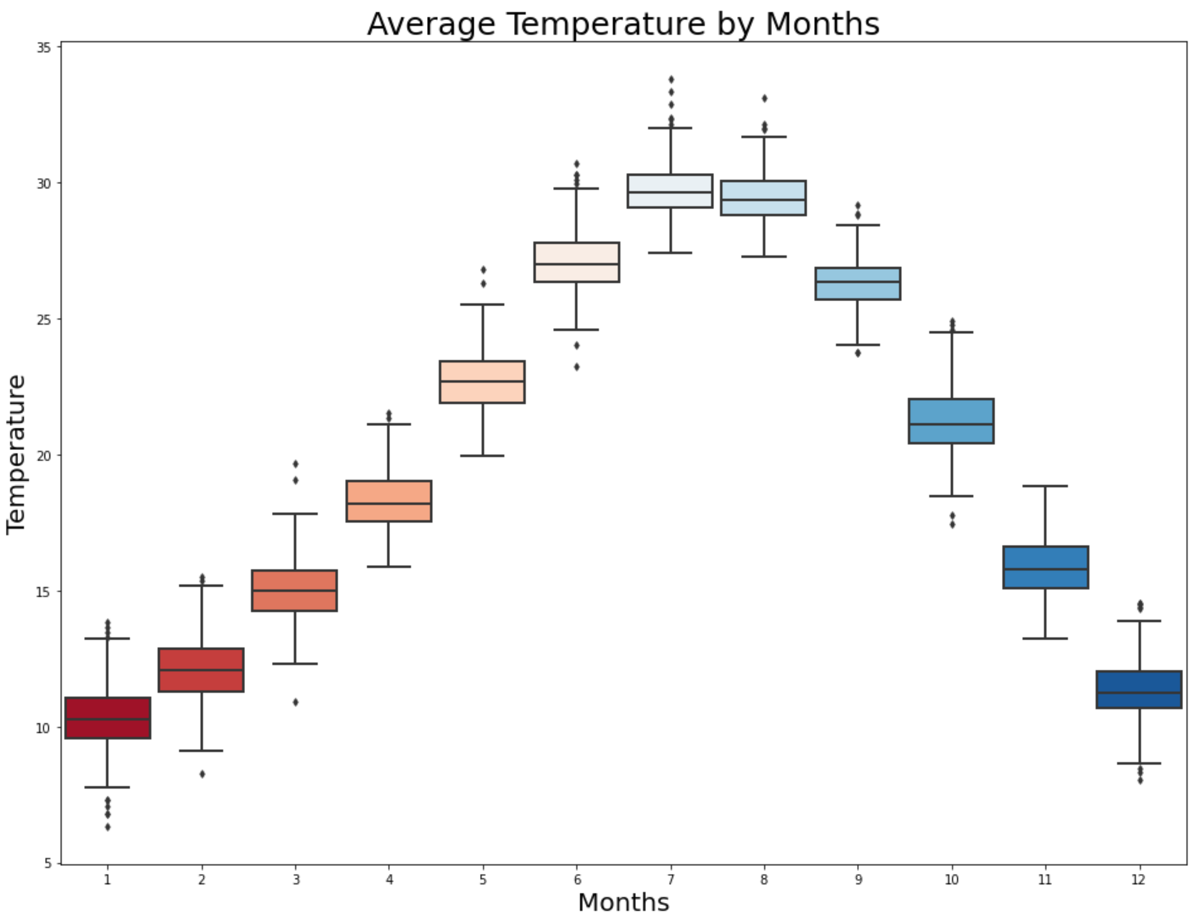
Out[26]:
```
261
```

Type *Markdown* and LaTeX: α^2

Monthly Analysis

In [27]:
```
GlobalTemp=temp_Tunisia
GlobalTemp.loc[GlobalTemp['month']=='1','month'] = 'January'
GlobalTemp.loc[GlobalTemp['month']=='2','month'] = 'February'
GlobalTemp.loc[GlobalTemp['month']=='3','month'] = 'March'
GlobalTemp.loc[GlobalTemp['month']=='4','month'] = 'April'
GlobalTemp.loc[GlobalTemp['month']=='5','month'] = 'May'
GlobalTemp.loc[GlobalTemp['month']=='6','month'] = 'June'
GlobalTemp.loc[GlobalTemp['month']=='7','month'] = 'July'
GlobalTemp.loc[GlobalTemp['month']=='8','month'] = 'August'
GlobalTemp.loc[GlobalTemp['month']=='9','month'] = 'September'
GlobalTemp.loc[GlobalTemp['month']=='10','month'] = 'October'
GlobalTemp.loc[GlobalTemp['month']=='11','month'] = 'November'
GlobalTemp.loc[GlobalTemp['month']=='12','month'] = 'December'
year_month = GlobalTemp.groupby(by = ['years','month']).mean().reset
# Figure size
plt.figure(figsize=(16,12))

# The plot
sns.boxplot(x = 'month', y = 'AverageTemperature', data = year_month

# Make pretty
plt.title('Average Temperature by Months', fontsize = 25)
plt.xlabel('Months', fontsize = 20)
plt.ylabel('Temperature', fontsize = 20)
```

Out[27]: Text(0, 0.5, 'Temperature')



In [28]:
```
year_month #1 2..12 pour chaque annee
```

Out[28]:

	years	month	AverageTemperature	AverageTemperatureUncertainty
0	1753	1	8.754	5.363
1	1753	2	10.597	3.183
2	1753	3	16.105	2.805
3	1753	4	18.181	5.257
4	1753	5	23.571	2.230
...	...	...	...	...
3123	2013	4	20.383	0.746
3124	2013	5	24.268	0.336
3125	2013	6	27.488	0.947
3126	2013	7	31.156	0.753
3127	2013	8	30.399	0.795

3128 rows × 4 columns

In [29]:
```
year_season = GlobalTemp.groupby(by = ['years','season']).mean().re
winter = year_season.loc[year_season['season'] == 'winter',:]
spring = year_season.loc[year_season['season'] == 'spring',:]
summer = year_season.loc[year_season['season'] == 'summer',:]
autumn = year_season.loc[year_season['season'] == 'autumn',:]
```

In [30]:
```
year_season
```

Out[30]:

	years	season	AverageTemperature	AverageTemperatureUncertainty
0	1753	autumn	20.836000	2.106000
1	1753	spring	19.285667	3.430667
2	1753	summer	29.951667	2.897333
3	1753	winter	9.771667	3.750667
4	1754	autumn	21.784000	2.413000
...	...	...	...	...
1038	2012	summer	31.498667	0.575333
1039	2012	winter	11.016333	0.307667
1040	2013	spring	20.744333	0.544333
1041	2013	summer	29.681000	0.831667
1042	2013	winter	11.910500	0.335500

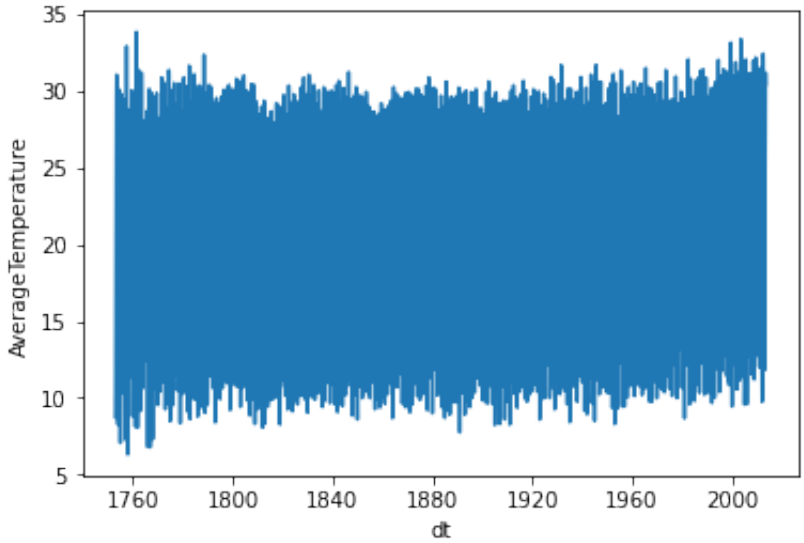
1043 rows × 4 columns

In [31]:

```
import plotly.graph_objects as go
fig2 = go.Figure()
for template in ["plotly_white"]:
    fig2.add_trace(go.Scatter(x=winter['years'], y=winter['AverageT
                    mode='lines',
                    name='winter',
                    marker_color='#838B8B'))
    fig2.add_trace(go.Scatter(x=spring['years'], y=spring['AverageT
                    mode='lines',
                    name='spring',
                    marker_color='#FFB5C5'))
    fig2.add_trace(go.Scatter(x=summer['years'], y=summer['AverageT
                    mode='lines',
                    name='summer',
                    marker_color='#87CEFF'))
    fig2.add_trace(go.Scatter(x=autumn['years'], y=autumn['AverageT
                    mode='lines',
                    name='autumn',
                    marker_color='#FF8000'))
    fig2.update_layout(
    height=800,
    xaxis_title="Years",
    yaxis_title='Temperature in degree',
    title_text='Average Temperature seasonwise over the years',
    template=template)



fig2.show()
```

In [32]:
```
sns.lineplot(x=temp_Tunisia.index,y=temp_Tunisia['AverageTemperatur
```

Out[32]: <AxesSubplot:xlabel='dt', ylabel='AverageTemperature'>



Whether it is stationary or not:

Conditions:

1.Time series should have a constant mean.

2.Time series should have a constant standard deviation.

3.Time series's auto-covariance should not depend on time.

In [33]:
```
from statsmodels.tsa.stattools import adfuller
test_result=adfuller(temp_Tunisia['AverageTemperature'])
```

In [34]:
```
test_result
```

Out[34]:
```
(-4.435854163177158,
 0.0002563408873216847,
 25,
 3102,
 {'1%': -3.4324598386855327,
  '5%': -2.8624721950635,
  '10%': -2.5672662300273403},
 9295.642439062402)
```

In [35]:
```
def adfuller_test(temp):
    result=adfuller(temp)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number o
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), rej
    else:
        print("weak evidence against null hypothesis, time series h
```

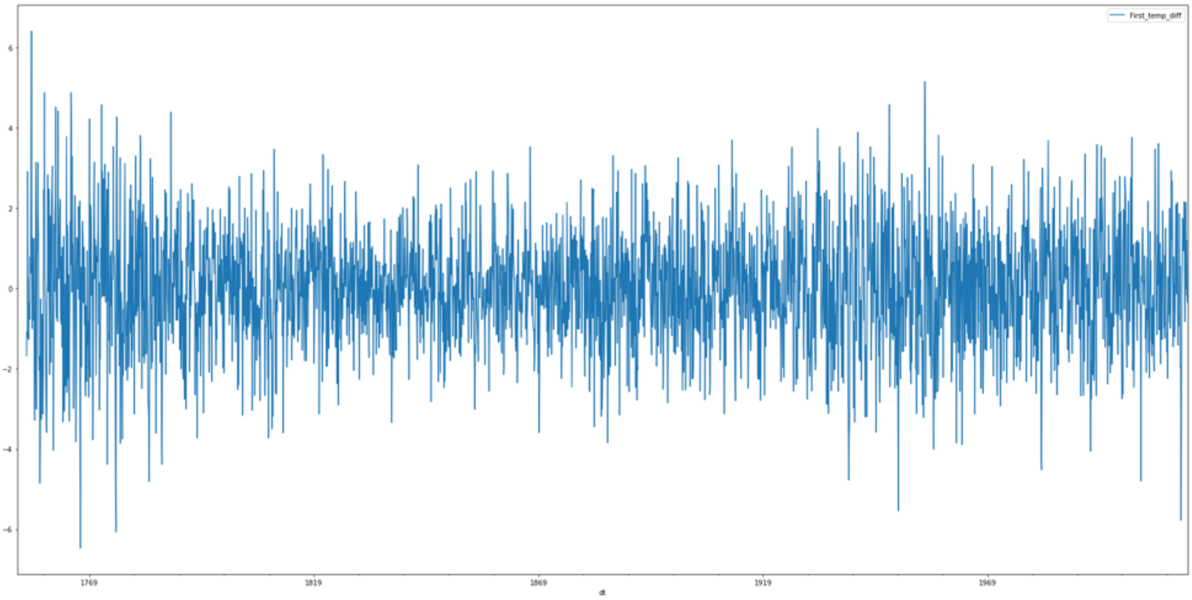
In [36]:
```
adfuller_test(temp_Tunisia['AverageTemperature'])
```

```
ADF Test Statistic : -4.435854163177158
p-value : 0.0002563408873216847
#Lags Used : 25
Number of Observations Used : 3102
strong evidence against the null hypothesis(Ho), reject the null h
ypothesis. Data has no unit root and is stationary
```

In [37]:
```
df=temp_Tunisia.copy()

df.head()

df['First_temp_diff']=df['AverageTemperature']-df['AverageTemperatu

adfuller_test(df['First_temp_diff'].dropna())

df[['First_temp_diff']].plot(figsize=(30,15))
```

```
ADF Test Statistic : -17.013761633012393
p-value : 8.554149174109644e-30
#Lags Used : 24
Number of Observations Used : 3079
strong evidence against the null hypothesis(Ho), reject the null h
ypothesis. Data has no unit root and is stationary
```

Out[37]: <AxesSubplot:xlabel='dt'>



Examine if there is a seasonality factor in data or not?

In [38]:
```
df
```

Out[38]:

	AverageTemperature	AverageTemperatureUncertainty	Country	years	month	seaso
dt						
1753-01-01	8.754	5.363	Tunisia	1753	1	winte
1753-02-01	10.597	3.183	Tunisia	1753	2	winte
1753-03-01	16.105	2.805	Tunisia	1753	3	sprin
1753-04-01	18.181	5.257	Tunisia	1753	4	sprin
1753-05-01	23.571	2.230	Tunisia	1753	5	sprin
...	...	...	...	...	...	..
2013-04-01	20.383	0.746	Tunisia	2013	4	sprin
2013-05-01	24.268	0.336	Tunisia	2013	5	sprin
2013-06-01	27.488	0.947	Tunisia	2013	6	summe
2013-07-01	31.156	0.753	Tunisia	2013	7	summe
2013-08-01	30.399	0.795	Tunisia	2013	8	summe

3128 rows × 7 columns

In [39]:
```
pivot = temp_Tunisia.pivot_table(values='AverageTemperature',index=
```

In [40]:
```
pivot
```

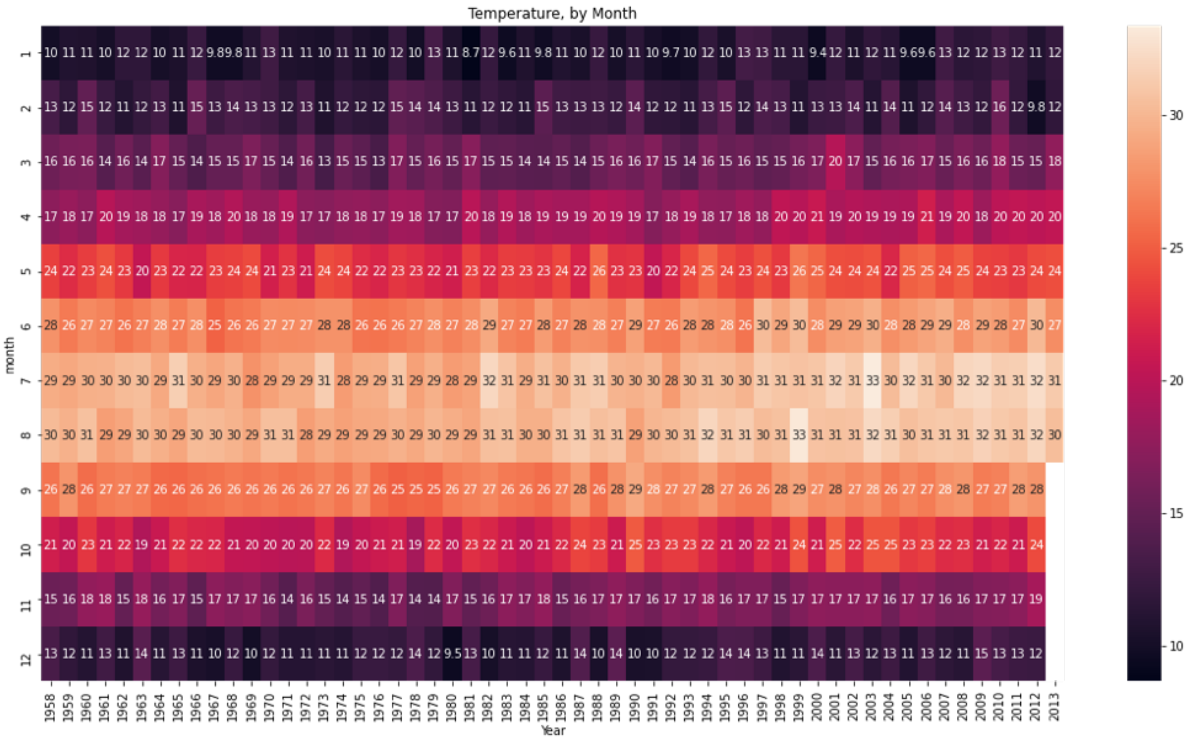
Out[40]:

years	1753	1754	1755	1756	1757	1758	1759	1760	1761	1762	...
month											
1	8.754	8.841	7.074	11.171	7.282	6.321	12.163	8.797	8.130	13.218	...
2	10.597	8.274	9.515	14.686	12.659	11.002	13.148	11.149	12.841	13.321	...
3	16.105	13.388	14.866	15.225	14.956	14.961	14.822	12.819	16.423	13.474	...
4	18.181	20.666	21.094	19.667	18.089	17.489	16.852	19.301	16.786	19.546	...
5	23.571	23.589	23.898	24.847	24.757	21.589	22.517	22.171	24.486	22.768	...
6	29.551	28.542	29.766	28.714	30.710	27.632	27.204	25.794	29.421	27.755	...
7	31.022	30.058	29.745	29.396	32.879	28.760	29.290	29.099	33.811	31.318	...
8	29.282	29.772	28.699	28.982	30.504	27.522	30.004	29.686	29.980	29.155	...
9	26.413	25.927	26.048	27.170	27.372	24.042	27.544	26.500	26.809	25.690	...
10	21.278	22.511	20.515	19.561	18.774	17.460	21.606	20.509	20.886	19.617	...
11	14.817	16.914	15.606	13.628	13.573	16.078	13.495	14.857	13.985	15.651	...
12	9.964	11.978	10.349	9.702	8.654	11.368	8.877	10.971	8.060	9.510	...

12 rows × 261 columns

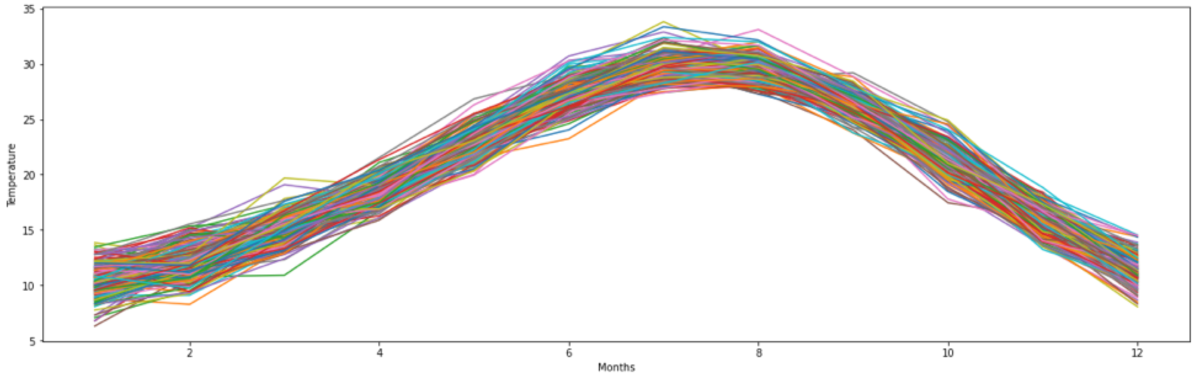
In [41]:
```
# Set the width and height of the figure
plt.figure(figsize=(19,10))

# Add title
plt.title("Temperature, by Month")

# Heatmap showing average arrival delay for each airline by month
sns.heatmap(data=pivot.loc[:,years[205:]], annot=True)

# Add label for horizontal axis
plt.xlabel("Year")
```

Out[41]:
```
Text(0.5, 69.0, 'Year')
```



In [42]:
```
pivot.plot(figsize=(20,6))
plt.legend().remove()
plt.xlabel('Months')
plt.ylabel('Temperature')
```

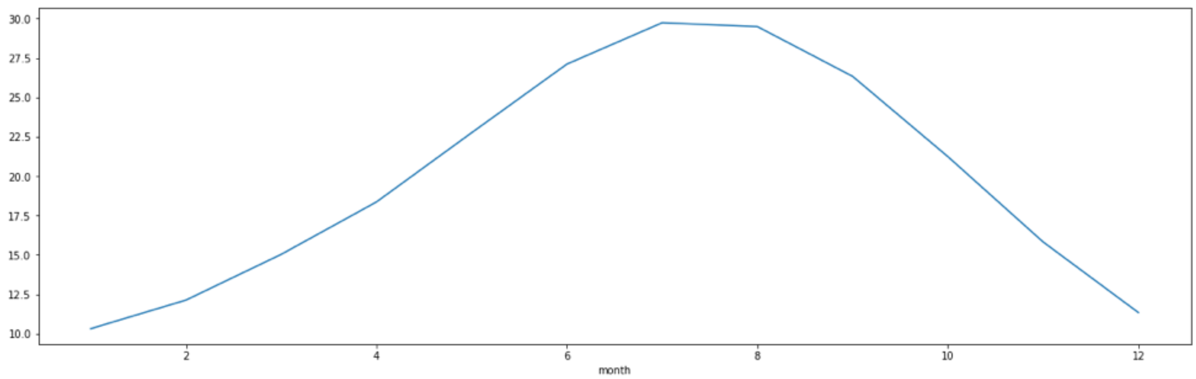
Out[42]:
```
Text(0, 0.5, 'Temperature')
```



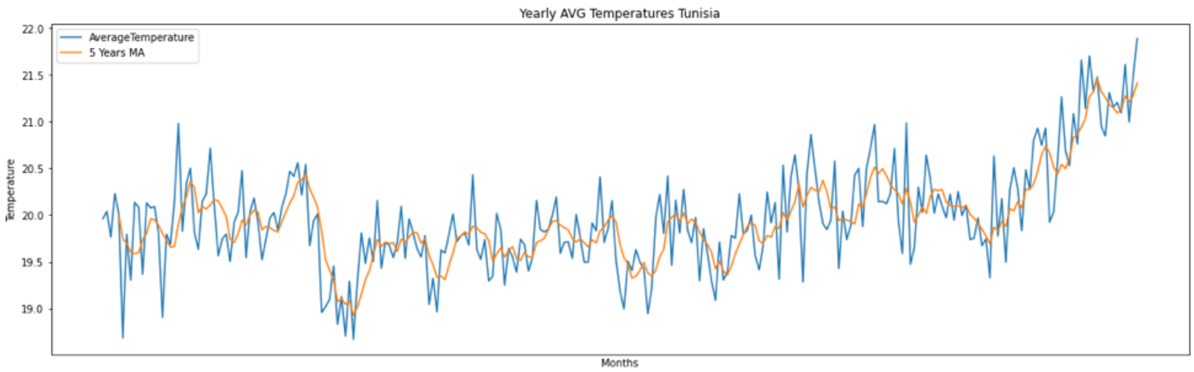
We found some kind of seasonality in the data.

In [43]:
```
monthly_seasonality=pivot.mean(axis=1)
monthly_seasonality.plot(figsize=(20,6))
```

Out[43]:
```
<AxesSubplot:xlabel='month'>
```



In [44]:
```
year_avg = pd.pivot_table(temp_Tunisia, values='AverageTemperature'
year_avg['5 Years MA'] = year_avg['AverageTemperature'].rolling(5).
year_avg[['AverageTemperature','5 Years MA']].plot(figsize=(20,6))
plt.title('Yearly AVG Temperatures Tunisia')
plt.xlabel('Months')
plt.ylabel('Temperature')
plt.xticks([x for x in range(2007,2012,12)])
plt.show()
```



In [45]:
```
year_avg
```

Out[45]:

	AverageTemperature	5 Years MA
years		
1753	19.961250	NaN
1754	20.038333	NaN
1755	19.764583	NaN
1756	20.229083	NaN
1757	20.017417	20.002133
...	...	...
2009	21.092500	21.121500
2010	21.609750	21.274300
2011	20.994917	21.211283
2012	21.479167	21.276183
2013	21.887125	21.412692

261 rows × 2 columns

Build time series

Moving average

In [46]:
```
df
```

Out[46]:

	AverageTemperature	AverageTemperatureUncertainty	Country	years	month	seasoı
dt						
1753-01-01	8.754	5.363	Tunisia	1753	1	winte
1753-02-01	10.597	3.183	Tunisia	1753	2	winte
1753-03-01	16.105	2.805	Tunisia	1753	3	sprin
1753-04-01	18.181	5.257	Tunisia	1753	4	sprin
1753-05-01	23.571	2.230	Tunisia	1753	5	sprin
...	...	...	...	...	...	..
2013-04-01	20.383	0.746	Tunisia	2013	4	sprin
2013-05-01	24.268	0.336	Tunisia	2013	5	sprin
2013-06-01	27.488	0.947	Tunisia	2013	6	summe
2013-07-01	31.156	0.753	Tunisia	2013	7	summe
2013-08-01	30.399	0.795	Tunisia	2013	8	summe

3128 rows × 7 columns

In [47]:
```
df=df[['First_temp_diff']]
df.dropna(inplace=True)
```

In [48]:
```
df.head()
```

Out[48]:

	First_temp_diff
dt	
1755-01-01	-1.680
1755-02-01	-1.082
1755-03-01	-1.239
1755-04-01	2.913
1755-05-01	0.327

In [49]:
```
df['First_temp_diff'].rolling(window=12).mean()
```

Out[49]:
```
dt
1755-01-01         NaN
1755-02-01         NaN
1755-03-01         NaN
1755-04-01         NaN
1755-05-01         NaN
                 ...
2013-04-01    0.875333
2013-05-01    0.924417
2013-06-01    0.787750
2013-07-01    0.675583
2013-08-01    0.568083
Name: First_temp_diff, Length: 3104, dtype: float64
```

In [50]:
```

value=pd.DataFrame(df['First_temp_diff'])

temp_df=pd.concat([value,df['First_temp_diff'].rolling(window=12).m

temp_df.columns=['actual_temp','forecast_temp']
temp_df.head(12)
```

Out[50]:

dt	actual_temp	forecast_temp
1755-01-01	-1.680	NaN
1755-02-01	-1.082	NaN
1755-03-01	-1.239	NaN
1755-04-01	2.913	NaN
1755-05-01	0.327	NaN
1755-06-01	0.215	NaN
1755-07-01	-1.277	NaN
1755-08-01	-0.583	NaN
1755-09-01	-0.365	NaN
1755-10-01	-0.763	NaN
1755-11-01	0.789	NaN
1755-12-01	0.385	-0.196667

In [51]:
```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(temp_df['forecast_temp'][11:],temp_df['a
```

Out[51]:
```
1.3815385869100967
```

In [52]:
```
!pip install scikit-learn==0.24
```

```
Collecting scikit-learn==0.24
  Downloading scikit_learn-0.24.0-cp37-cp37m-manylinux2010_x86_64.
whl (22.3 MB)
     |████████████████████████████████| 22.3 MB 894 kB/s
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/pyt
hon3.7/site-packages (from scikit-learn==0.24) (1.19.5)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/pyth
on3.7/site-packages (from scikit-learn==0.24) (1.0.1)
Requirement already satisfied: scipy>=0.19.1 in /opt/conda/lib/pyt
hon3.7/site-packages (from scikit-learn==0.24) (1.5.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/
lib/python3.7/site-packages (from scikit-learn==0.24) (2.1.0)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 0.24.1
    Uninstalling scikit-learn-0.24.1:
      Successfully uninstalled scikit-learn-0.24.1
ERROR: pip's dependency resolver does not currently take into acco
unt all the packages that are installed. This behaviour is the sou
rce of the following dependency conflicts.
pyldavis 3.3.1 requires numpy>=1.20.0, but you have numpy 1.19.5 w
hich is incompatible.
pyldavis 3.3.1 requires pandas>=1.2.0, but you have pandas 1.1.5 w
hich is incompatible.
pdpbox 0.2.1 requires matplotlib==3.1.1, but you have matplotlib 3
.4.1 which is incompatible.
Successfully installed scikit-learn-0.24.0
```

Using ARIMA

ARIMA Model has three parameters: p: it is the number of autoregressive lags.¶ d: it is the order of differencing required to make the series stationary. q: it is the number of moving average lags.

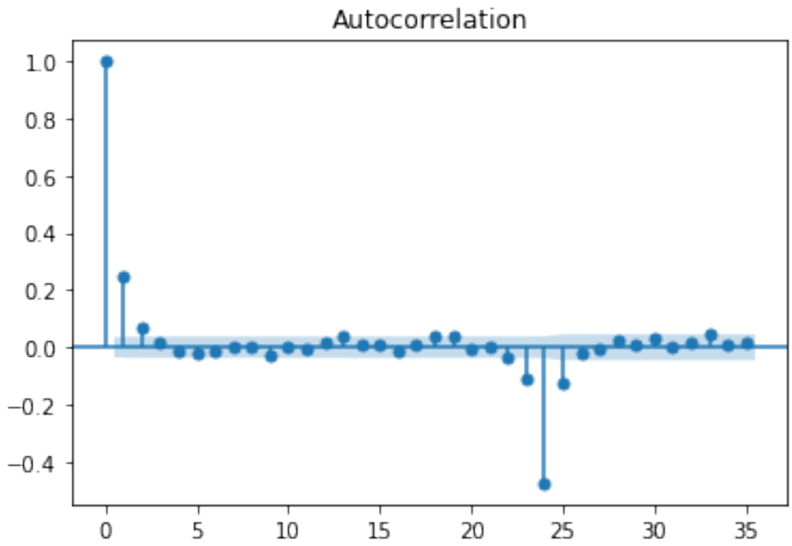
ARIMA stands for Autoregressive Integrated Moving Average. It is a combination of two models which are autoregressive and moving average.

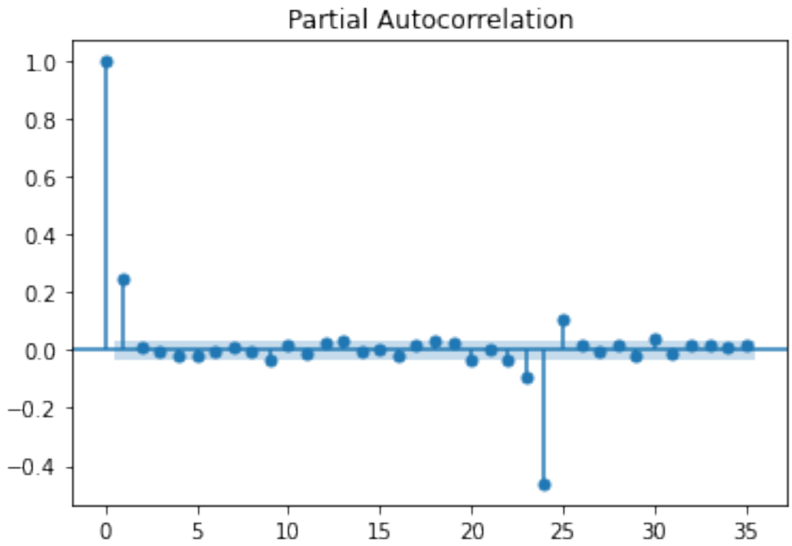
In [53]:

```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

plot_acf(df['First_temp_diff'].dropna())

plot_pacf(df['First_temp_diff'].dropna())

df.isna().sum()
```

Out[53]:

```
First_temp_diff    0
dtype: int64
```





In [54]:
```
df
```

Out[54]:

	First_temp_diff
dt	
1755-01-01	-1.680
1755-02-01	-1.082
1755-03-01	-1.239
1755-04-01	2.913
1755-05-01	0.327
...	...
2013-04-01	-0.009
2013-05-01	1.190
2013-06-01	0.107
2013-07-01	-0.322
2013-08-01	-0.351

3104 rows × 1 columns

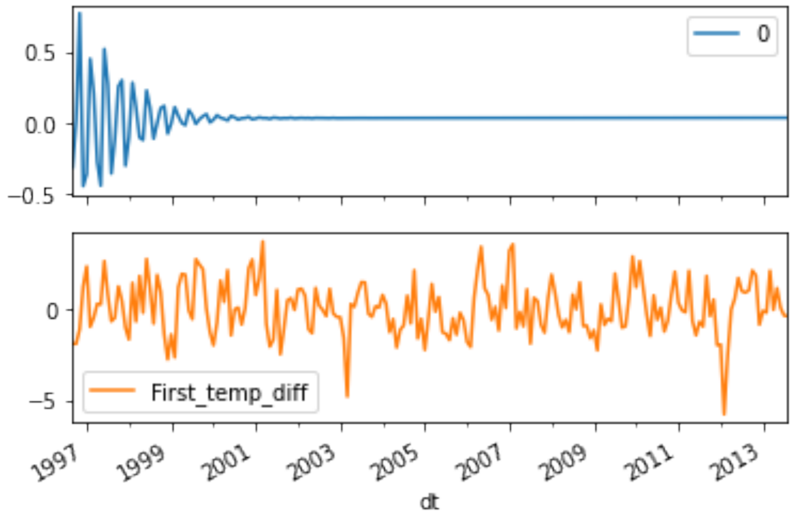
In [55]:
```
training_data=df[0:2900]
test_data = df[2900:]

from statsmodels.tsa.arima_model import ARIMA

arima = ARIMA(training_data,order=(12,1,5))
```

```
/opt/conda/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_mo
del.py:527: ValueWarning:

No frequency information was provided, so inferred frequency MS wi
ll be used.

/opt/conda/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_mo
del.py:527: ValueWarning:

No frequency information was provided, so inferred frequency MS wi
ll be used.
```

In [56]:
```
#fit the model
model= arima.fit()

#predictions
predictions=model.forecast(steps=len(test_data))[0]
```

```
/opt/conda/lib/python3.7/site-packages/statsmodels/base/model.py:5
48: HessianInversionWarning:

Inverting hessian failed, no bse or cov_params available

/opt/conda/lib/python3.7/site-packages/statsmodels/base/model.py:5
68: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retv
als
```

In [57]:
```
pd.Series(predictions).to_frame().set_index(test_data.index).join(t
plt.show()
```



In [58]:
```
np.sqrt(mean_squared_error(test_data,predictions))
```

Out[58]: 1.4654342876047328

In [59]:
```
pd.Series(predictions).to_frame().set_index(test_data.index)
```

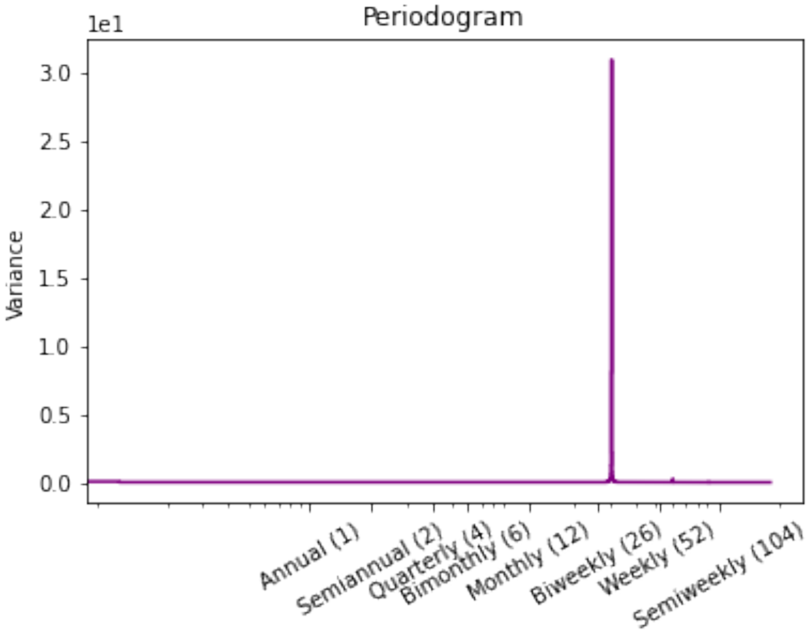
Out[59]:

	0
dt	
1996-09-01	-0.308589
1996-10-01	0.016643
1996-11-01	0.768611
1996-12-01	-0.440571
1997-01-01	-0.356028
...	...
2013-04-01	0.037082
2013-05-01	0.037100
2013-06-01	0.037118
2013-07-01	0.037136
2013-08-01	0.037155

204 rows × 1 columns

In [60]:
```
def plot_periodogram(ts, detrend='linear', ax=None):
    from scipy.signal import periodogram
    fs = pd.Timedelta("1Y") / pd.Timedelta("1D")
    freqencies, spectrum = periodogram(
        ts,
        fs=fs,
        detrend=detrend,
        window="boxcar",
        scaling='spectrum',
    )
    if ax is None:
        _, ax = plt.subplots()
    ax.step(freqencies, spectrum, color="purple")
    ax.set_xscale("log")
    ax.set_xticks([1, 2, 4, 6, 12, 26, 52, 104])
    ax.set_xticklabels(
        [
            "Annual (1)",
            "Semiannual (2)",
            "Quarterly (4)",
            "Bimonthly (6)",
            "Monthly (12)",
            "Biweekly (26)",
            "Weekly (52)",
            "Semiweekly (104)",
        ],
        rotation=30,
    )
    ax.ticklabel_format(axis="y", style="sci", scilimits=(0, 0))
    ax.set_ylabel("Variance")
    ax.set_title("Periodogram")
    return ax
```

In [61]:
```
plot_periodogram(temp_Tunisia["AverageTemperature"]);
```



Seasonal Arima

In [62]:
```
train = temp_Tunisia[:2000].copy()
val = temp_Tunisia[2000:2700].copy()
test = temp_Tunisia[2700:].copy()
```

In [63]:
```
train
```

Out[63]:

dt	AverageTemperature	AverageTemperatureUncertainty	Country	years	month	seaso
1753-01-01	8.754	5.363	Tunisia	1753	1	winte
1753-02-01	10.597	3.183	Tunisia	1753	2	winte
1753-03-01	16.105	2.805	Tunisia	1753	3	sprin
1753-04-01	18.181	5.257	Tunisia	1753	4	sprin
1753-05-01	23.571	2.230	Tunisia	1753	5	sprin
...	...	...	...	...	...	..
1919-04-01	18.060	0.889	Tunisia	1919	4	sprin
1919-05-01	20.506	1.087	Tunisia	1919	5	sprin
1919-06-01	26.548	0.534	Tunisia	1919	6	summe
1919-07-01	29.574	0.559	Tunisia	1919	7	summe
1919-08-01	28.562	0.647	Tunisia	1919	8	summe

2000 rows × 6 columns

In [64]:
```
baseline = val['AverageTemperature'].shift()
baseline.dropna(inplace=True)
baseline.head()
```

Out[64]:
```
dt
1919-10-01    26.497
1919-11-01    20.781
1919-12-01    16.450
1920-01-01    10.453
1920-02-01    10.879
Name: AverageTemperature, dtype: float64
```

In [65]:
```
val.iloc[1:,0]
```

Out[65]:
```
dt
1919-10-01    20.781
1919-11-01    16.450
1919-12-01    10.453
1920-01-01    10.879
1920-02-01    12.369
               ...
1977-08-01    29.623
1977-09-01    25.125
1977-10-01    21.320
1977-11-01    16.631
1977-12-01    12.140
Name: AverageTemperature, Length: 699, dtype: float64
```

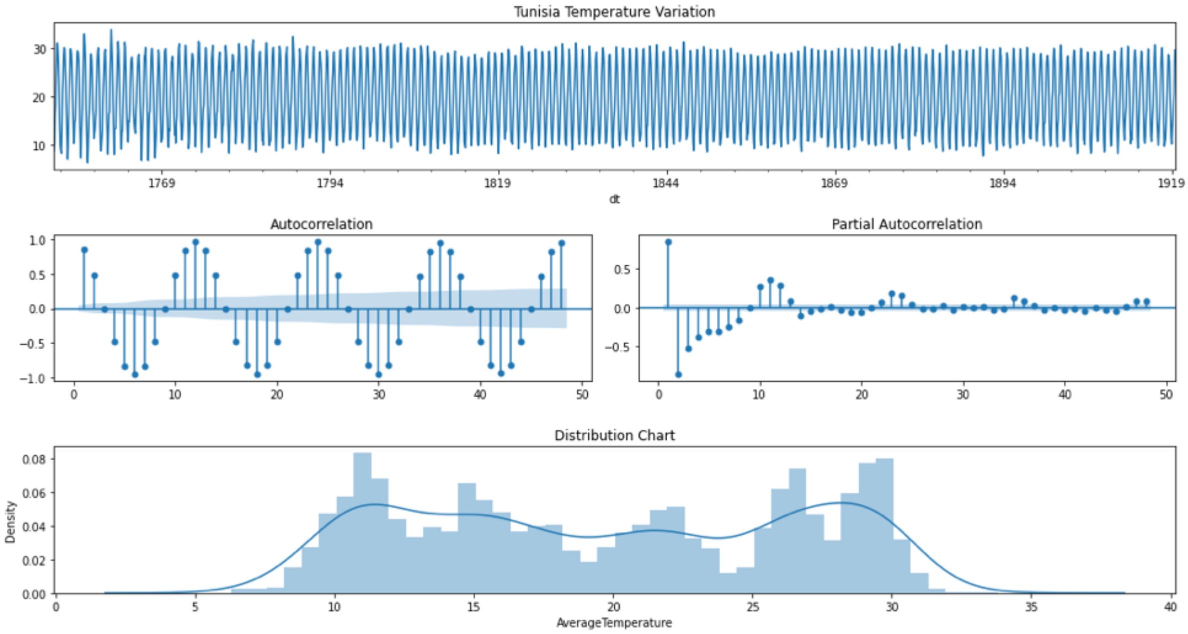
In [66]:
```
import math
def measure_rmse(y_true, y_pred):
    return math.sqrt(mean_squared_error(y_true,y_pred))

# Using the function with the baseline values
rmse_base = measure_rmse(val.iloc[1:,0],baseline)
print(f'The RMSE of the baseline that we will try to diminish is {r
```

```
The RMSE of the baseline that we will try to diminish is 3.7778 ce
lsius degrees
```

In [67]:
```
def check_stationarity(y, lags_plots=48, figsize=(22,8)):
    "Use Series as parameter"

    # Creating plots of the DF
    y = pd.Series(y)
    fig = plt.figure()

    ax1 = plt.subplot2grid((3, 3), (0, 0), colspan=2)
    ax2 = plt.subplot2grid((3, 3), (1, 0))
    ax3 = plt.subplot2grid((3, 3), (1, 1))
    ax4 = plt.subplot2grid((3, 3), (2, 0), colspan=2)

    y.plot(ax=ax1, figsize=figsize)
    ax1.set_title('Tunisia Temperature Variation')
    plot_acf(y, lags=lags_plots, zero=False, ax=ax2);
    plot_pacf(y, lags=lags_plots, zero=False, ax=ax3);
    sns.distplot(y, bins=int(math.sqrt(len(y))), ax=ax4)
    ax4.set_title('Distribution Chart')

    plt.tight_layout()

    print('Results of Dickey-Fuller Test:')
    adfinput = adfuller(y)
    adftest = pd.Series(adfinput[0:4], index=['Test Statistic','p-v
    adftest = round(adftest,4)

    for key, value in adfinput[4].items():
        adftest["Critical Value (%s)"%key] = value.round(4)

    print(adftest)

    if adftest[0].round(2) < adftest[5].round(2):
        print('\nThe Test Statistics is lower than the Critical Val
    else:
        print("\nThe Test Statistics is higher than the Critical Va
```

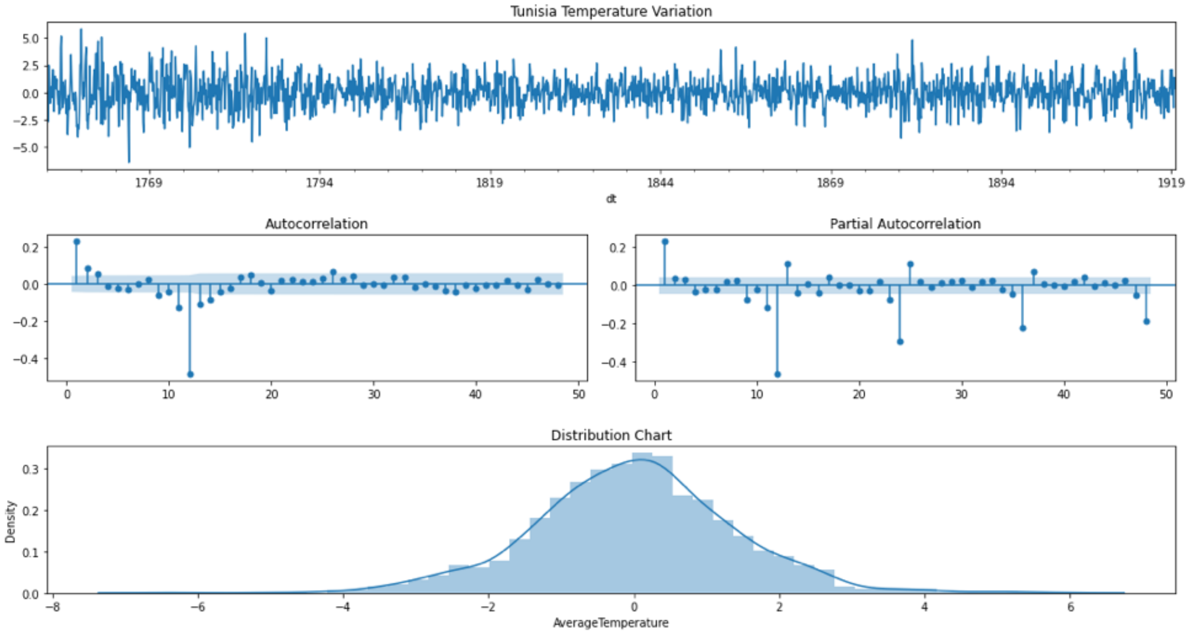
In [68]:
```
check_stationarity(train['AverageTemperature'])
```

```
Results of Dickey-Fuller Test:
Test Statistic                   -5.1282
p-value                           0.0000
Lags Used                        24.0000
Number of Observations Used    1975.0000
Critical Value (1%)              -3.4337
Critical Value (5%)              -2.8630
Critical Value (10%)             -2.5675
dtype: float64

The Test Statistics is lower than the Critical Value of 5%.
The serie seems to be stationary
```



In [69]:
```
check_stationarity(train['AverageTemperature'].diff(12).dropna())
```

```
Results of Dickey-Fuller Test:
Test Statistic                   -13.5195
p-value                            0.0000
Lags Used                         24.0000
Number of Observations Used     1963.0000
Critical Value (1%)               -3.4337
Critical Value (5%)               -2.8630
Critical Value (10%)              -2.5676
dtype: float64

The Test Statistics is lower than the Critical Value of 5%.
The serie seems to be stationary
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: