India's Weather Analysis

import Library

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
```

Reading the data

In [3]:
```
df=pd.read_csv(r'C:\Users\admin\Downloads\Weather Data in India from 1901 to
df
```

Out[3]:

	Unnamed: 0	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV
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
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
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
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
4	4	1905	17.40	17.79	21.78	24.84	28.32	28.69	27.67	27.47	26.29	26.16	22.07
...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	112	2013	18.88	21.07	24.53	26.97	29.06	28.24	27.50	27.22	26.87	25.63	22.18
113	113	2014	18.81	20.35	23.34	26.91	28.45	29.42	28.07	27.42	26.61	25.38	22.53
114	114	2015	19.02	21.23	23.52	26.52	28.82	28.15	28.03	27.64	27.04	25.82	22.95
115	115	2016	20.92	23.58	26.61	29.56	30.41	29.70	28.18	28.17	27.72	26.81	23.90
116	116	2017	20.59	23.08	25.58	29.17	30.47	29.44	28.31	28.12	28.11	27.24	23.92

117 rows × 14 columns

Data Cleaning

drop the data

In [4]:
```
df=df.drop(columns=['Unnamed: 0'])
df
```

Out[4]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
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	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	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	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
4	1905	17.40	17.79	21.78	24.84	28.32	28.69	27.67	27.47	26.29	26.16	22.07	18.71
...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	2013	18.88	21.07	24.53	26.97	29.06	28.24	27.50	27.22	26.87	25.63	22.18	19.69
113	2014	18.81	20.35	23.34	26.91	28.45	29.42	28.07	27.42	26.61	25.38	22.53	19.50
114	2015	19.02	21.23	23.52	26.52	28.82	28.15	28.03	27.64	27.04	25.82	22.95	20.21
115	2016	20.92	23.58	26.61	29.56	30.41	29.70	28.18	28.17	27.72	26.81	23.90	21.89
116	2017	20.59	23.08	25.58	29.17	30.47	29.44	28.31	28.12	28.11	27.24	23.92	21.47

117 rows × 13 columns

To view all the data

In [5]:
```
df
```

Out[5]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
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	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	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	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
4	1905	17.40	17.79	21.78	24.84	28.32	28.69	27.67	27.47	26.29	26.16	22.07	18.71
...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	2013	18.88	21.07	24.53	26.97	29.06	28.24	27.50	27.22	26.87	25.63	22.18	19.69
113	2014	18.81	20.35	23.34	26.91	28.45	29.42	28.07	27.42	26.61	25.38	22.53	19.50
114	2015	19.02	21.23	23.52	26.52	28.82	28.15	28.03	27.64	27.04	25.82	22.95	20.21
115	2016	20.92	23.58	26.61	29.56	30.41	29.70	28.18	28.17	27.72	26.81	23.90	21.89
116	2017	20.59	23.08	25.58	29.17	30.47	29.44	28.31	28.12	28.11	27.24	23.92	21.47

117 rows × 13 columns

First let's create another dataframe to make the

data easier to visualize. We are going to organize the data into a time series.

In [6]:
```
dates = {}

i = 0
for y in df['YEAR']:
    for m in df.columns[1:]:

        dat = str(m) + '/' + str(y)
        dates[dat] = df[m][i]
    i += 1

dates = pd.DataFrame(pd.Series(dates).reset_index())
dates.columns = ['date', 'temp']

dates['date'] = pd.to_datetime(dates['date'], format= '%b/%Y')
dates['year'] = dates['date'].dt.year
dates['month'] = dates['date'].dt.month_name()

dates.head()
```

Out[6]:

	date	temp	year	month
0	1901-01-01	17.99	1901	January
1	1901-02-01	19.43	1901	February
2	1901-03-01	23.49	1901	March
3	1901-04-01	26.41	1901	April
4	1901-05-01	28.28	1901	May

Check the datatype of the dataset

In [7]:
```
dates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1404 entries, 0 to 1403
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    1404 non-null   datetime64[ns]
 1   temp    1404 non-null   float64       
 2   year    1404 non-null   int64         
 3   month   1404 non-null   object        
dtypes: datetime64[ns](1), float64(1), int64(1), object(1)
memory usage: 44.0+ KB
```

To check the missing values

In [8]:
```
1 dates.isnull().sum()
```

Out[8]:
```
date     0
temp     0
year     0
month    0
dtype: int64
```

Top 5 records

In [9]:
```
1 data=dates.head(5)
2 data
```

Out[9]:

	date	temp	year	month
0	1901-01-01	17.99	1901	January
1	1901-02-01	19.43	1901	February
2	1901-03-01	23.49	1901	March
3	1901-04-01	26.41	1901	April
4	1901-05-01	28.28	1901	May

Top 5 bottom report

In [10]:
```
1 dates.tail(5)
```

Out[10]:

	date	temp	year	month
1399	2017-08-01	28.12	2017	August
1400	2017-09-01	28.11	2017	September
1401	2017-10-01	27.24	2017	October
1402	2017-11-01	23.92	2017	November
1403	2017-12-01	21.47	2017	December

To print field name

In [11]:
```
1 dates.columns
```

Out[11]:
```
Index(['date', 'temp', 'year', 'month'], dtype='object')
```

print information of a dataset dates

To print the Statistical information

In [14]:
```
1 dates.describe(include='all')
```

Out[14]:

	date	temp	year	month
count	1404	1404.000000	1404.000000	1404
unique	1404	NaN	NaN	12
top	1918-10-01 00:00:00	NaN	NaN	April
freq	1	NaN	NaN	117
first	1901-01-01 00:00:00	NaN	NaN	NaN
last	2017-12-01 00:00:00	NaN	NaN	NaN
mean	NaN	24.294160	1959.000000	NaN
std	NaN	3.516446	33.785791	NaN
min	NaN	17.250000	1901.000000	NaN
25%	NaN	21.067500	1930.000000	NaN
50%	NaN	25.570000	1959.000000	NaN
75%	NaN	27.240000	1988.000000	NaN
max	NaN	30.780000	2017.000000	NaN

To check is there any duplicate value

In [15]:
```
1 dates.duplicated().sum()
```

Out[15]:
```
0
```

To Count number of categories

In [16]:
```
1 dates['month'].value_counts()
```

Out[16]:
```
April        117
November     117
October      117
March        117
September    117
May          117
January      117
August       117
July         117
June         117
February     117
December     117
Name: month, dtype: int64
```

rename the columns

In [17]:
```
dates.rename(columns={'date':'Date','temp':'Temp','year':'Year','month':'mon
```

In [18]:
```
dates
```

Out[18]:

	Date	Temp	Year	months
0	1901-01-01	17.99	1901	January
1	1901-02-01	19.43	1901	February
2	1901-03-01	23.49	1901	March
3	1901-04-01	26.41	1901	April
4	1901-05-01	28.28	1901	May
...	...	...	...	...
1399	2017-08-01	28.12	2017	August
1400	2017-09-01	28.11	2017	September
1401	2017-10-01	27.24	2017	October
1402	2017-11-01	23.92	2017	November
1403	2017-12-01	21.47	2017	December

1404 rows × 4 columns

max Temprature

In [19]:
```
d5=dates.Temp.max()
d5
```

Out[19]:
```
30.78
```

min Temprature

In [20]:
```
dates.Temp.min()
```

Out[20]:
```
17.25
```

To check the unique Temprature

In [21]:
```
1 dates.Temp.unique()
```

Out[21]:
```
array([17.99, 19.43, 23.49, 26.41, 28.28, 28.6 , 27.49, 26.98, 26.26,
       25.08, 21.73, 18.95, 19.  , 20.39, 24.1 , 26.54, 28.68, 28.44,
       27.29, 27.05, 25.95, 24.37, 21.33, 18.78, 18.32, 19.79, 22.46,
       26.03, 27.93, 28.41, 28.04, 26.63, 26.34, 24.57, 20.96, 18.29,
       17.77, 19.39, 22.95, 26.73, 27.83, 27.85, 26.84, 25.84, 24.36,
       21.07, 18.84, 17.4 , 17.79, 21.78, 24.84, 28.32, 28.69, 27.67,
       27.47, 26.29, 26.16, 22.07, 18.71, 17.5 , 19.14, 22.21, 26.53,
       29.06, 28.02, 27.46, 26.82, 26.23, 24.75, 21.93, 19.55, 19.27,
       19.42, 22.03, 27.52, 27.66, 27.28, 26.38, 24.72, 22.11, 18.46,
       18.35, 19.73, 22.93, 27.06, 28.07, 28.49, 27.16, 25.74, 24.25,
       21.06, 18.15, 19.05, 23.4 , 25.76, 27.97, 26.56, 26.43, 25.47,
       22.01, 18.86, 18.14, 19.72, 22.9 , 25.96, 28.36, 27.72, 26.93,
       26.61, 25.98, 24.04, 20.72, 18.05, 18.52, 19.18, 22.05, 26.  ,
       28.55, 27.44, 27.04, 26.22, 21.1 , 18.76, 18.6 , 20.84, 26.21,
       28.3 , 28.53, 26.68, 25.81, 24.44, 21.  , 18.44, 18.2 , 19.98,
       22.15, 27.95, 27.91, 27.  , 26.8 , 26.02, 24.35, 20.92, 18.7 ,
       18.96, 19.66, 22.63, 25.73, 28.24, 28.46, 26.49, 23.97, 21.87,
       18.73, 17.93, 22.69, 29.17, 28.58, 27.77, 27.32, 25.46, 22.18,
       18.31, 19.68, 24.24, 28.16, 27.81, 27.08, 26.77, 24.6 , 21.03,
       18.17, 18.16, 19.94, 25.29, 26.97, 27.41, 27.15, 26.64, 25.79,
       23.85, 20.8 , 18.39, 17.25, 19.58, 22.64, 25.26, 28.27, 27.58,
       27.74, 24.54, 21.71, 18.37, 19.15, 23.15, 26.19, 28.2 , 21.48,
       18.58, 18.27, 19.16, 23.02, 27.76, 27.48, 24.85, 21.84, 18.61,
       22.12, 24.26, 27.11, 30.78, 27.4 , 26.59, 25.92, 23.84, 19.24,
       18.3 , 20.5 , 23.65, 26.55, 29.8 , 28.03, 27.26, 26.95, 26.04,
       23.94, 21.28, 18.23, 19.35, 23.21, 26.81, 28.33, 28.95, 26.89,
       26.71, 25.89, 23.89, 21.23, 19.08, 18.06, 19.97, 24.12, 26.87,
       27.43, 27.5 , 26.91, 25.72, 24.5 , 21.04, 18.97, 17.53, 23.5 ,
       27.24, 27.82, 27.01, 24.61, 21.29, 18.63, 18.65, 20.7 , 22.87,
       25.3 , 27.69, 28.97, 27.65, 24.41, 20.59, 18.8 , 19.4 , 22.71,
       26.13, 27.7 , 27.23, 26.66, 26.1 , 24.7 , 21.16, 19.41, 18.66,
       20.13, 23.23, 26.44, 28.88, 28.22, 26.9 , 26.32, 24.92, 21.77,
       18.25, 24.05, 17.75, 18.72, 23.36, 25.94, 28.11, 27.84, 27.31,
       26.85, 26.17, 21.41, 19.11, 19.34, 23.31, 27.27, 28.51, 26.25,
       24.91, 21.7 , 19.29, 19.51, 23.39, 28.52, 26.5 , 24.69, 17.84,
       20.33, 23.22, 25.52, 27.92, 27.39, 25.86, 21.67, 19.31, 17.8 ,
       20.51, 22.97, 26.58, 28.13, 27.42, 26.74, 24.42, 21.15, 17.54,
       25.15, 28.45, 27.18, 26.37, 25.85, 21.49, 19.03, 20.  , 22.84,
       26.27, 27.12, 24.73, 21.72, 19.13, 18.02, 20.21, 22.6 , 25.83,
       28.4 , 23.98, 21.34, 18.09, 18.77, 23.61, 27.33, 26.83, 26.14,
       20.62, 18.9 , 18.4 , 21.98, 25.67, 28.63, 27.22, 26.92, 26.4 ,
       24.66, 21.32, 18.1 , 19.88, 22.23, 25.88, 28.38, 28.29, 24.74,
       18.33, 20.28, 28.89, 27.54, 21.76, 19.67, 18.07, 20.06, 23.87,
       26.79, 28.82, 28.81, 26.62, 24.56, 21.85, 18.75, 19.47, 25.56,
       27.8 , 24.2 , 21.81, 19.02, 18.  , 19.07, 22.54, 25.63, 28.93,
       28.39, 26.08, 24.49, 21.51, 19.32, 17.33, 19.37, 23.28, 28.31,
       26.51, 17.98, 20.73, 26.86, 28.48, 28.14, 26.31, 24.98, 21.6 ,
       20.08, 17.72, 19.84, 23.55, 26.88, 26.28, 24.67, 19.36, 22.72,
       28.71, 26.72, 21.58, 18.79, 19.01, 23.37, 28.35, 27.86, 20.79,
       18.36, 22.45, 28.34, 28.21, 27.03, 25.6 , 24.07, 19.09, 23.13,
       25.19, 27.2 , 27.09, 22.34, 19.3 , 18.67, 21.05, 22.73, 26.96,
       27.37, 26.48, 18.11, 20.86, 24.9 , 28.99, 28.79, 27.35, 26.24,
       19.85, 20.45, 23.66, 26.99, 28.84, 23.52, 20.99, 18.94, 18.42,
       23.93, 25.5 , 28.01, 27.07, 24.22, 21.14, 18.69, 18.01, 23.44,
       28.72, 22.31, 25.68, 26.75, 19.6 , 19.5 , 29.01, 27.6 , 19.65,
```

```
28.47, 21.57, 19.44, 21.13, 22.65, 28.5 , 26.33, 24.58, 21.35,
19.59, 18.89, 26.42, 26.45, 24.23, 20.98, 28.25, 27.53, 21.38,
17.59, 20.55, 25.69, 27.71, 21.99, 19.23, 17.71, 19.95, 28.06,
25.99, 24.8 , 18.47, 19.96, 22.81, 25.35, 27.57, 27.14, 24.99,
22.16, 21.17, 23.51, 27.02, 24.51, 27.36, 26.7 , 26.35, 21.4 ,
17.52, 18.98, 23.18, 24.32, 21.56, 19.2 , 20.3 , 24.52, 24.89,
22.29, 27.17, 27.25, 24.81, 18.12, 19.99, 23.29, 24.39, 21.21,
23.43, 28.08, 28.7 , 26.76, 24.27, 21.91, 19.46, 18.51, 23.33,
28.62, 24.43, 21.46, 19.28, 23.91, 27.89, 24.65, 21.31, 22.78,
26.39, 26.47, 20.76, 19.54, 22.98, 27.94, 27.34, 22.47, 20.34,
24.45, 24.86, 22.7 , 19.76, 28.92, 28.42, 24.94, 19.53, 22.75,
28.83, 25.23, 18.81, 20.47, 23.24, 27.55, 28.94, 25.12, 20.44,
23.26, 26.65, 26.36, 21.43, 18.87, 19.52, 22.37, 26.15, 27.68,
19.33, 18.38, 19.81, 22.68, 27.51, 21.5 , 18.28, 28.43, 26.94,
21.47, 20.54, 28.05, 24.33, 18.24, 23.53, 27.75, 28.19, 27.3 ,
24.48, 21.89, 19.38, 20.41, 23.45, 28.9 , 27.13, 25.03, 22.14,
19.75, 20.83, 26.6 , 28.37, 24.79, 21.63, 19.64, 17.68, 23.  ,
21.8 , 18.56, 22.79, 22.08, 19.78, 20.43, 23.58, 28.17, 27.59,
19.77, 26.78, 26.11, 28.73, 24.88, 22.2 , 20.2 , 24.14, 28.56,
22.06, 29.34, 29.88, 22.99, 20.65, 25.13, 26.67, 17.86, 23.64,
25.55, 20.6 , 29.18, 27.78, 25.27, 22.48, 19.21, 21.26, 24.18,
24.97, 22.33, 19.57, 25.58, 18.5 , 29.46, 27.63, 27.62, 25.49,
22.51, 20.52, 29.56, 28.77, 25.44, 23.17, 19.87, 23.62, 28.64,
22.17, 18.83, 27.38, 28.12, 24.34, 22.43, 19.93, 25.41, 22.59,
20.97, 22.4 , 18.49, 19.83, 28.1 , 21.66, 24.55, 25.2 , 20.22,
29.19, 19.22, 24.11, 25.51, 28.91, 27.98, 22.26, 19.91, 18.88,
24.53, 19.69, 20.35, 23.34, 29.42, 25.38, 22.53, 26.52, 28.15,
27.64, 25.82, 30.41, 29.7 , 28.18, 23.9 , 23.08, 30.47, 29.44,
23.92])
```

Sort the dataset

In [22]:
```
1 sort=dates.sort_values('Temp',ascending=True).head(5)
2 sort
```

Out[22]:

	Date	Temp	Year	months
204	1918-01-01	17.25	1918	January
528	1945-01-01	17.33	1945	January
48	1905-01-01	17.40	1905	January
60	1906-01-01	17.50	1906	January
804	1968-01-01	17.52	1968	January

In [23]:
```
1 dates.sort_values('Temp',ascending=False).head(5)
```

Out[23]:

	Date	Temp	Year	months
244	1921-05-01	30.78	1921	May
1396	2017-05-01	30.47	2017	May
1384	2016-05-01	30.41	2016	May
1133	1995-06-01	29.88	1995	June
256	1922-05-01	29.80	1922	May

In [24]:
```
1 pd.crosstab(dates['months'],dates['Temp'])
```

Out[24]:

Temp	17.25	17.33	17.40	17.50	17.52	17.53	17.54	17.59	17.68	17.71	...	29.42	29.44
months													
April	0	0	0	0	0	0	0	0	0	0	...	0	0
August	0	0	0	0	0	0	0	0	0	0	...	0	0
December	0	0	0	0	0	0	0	0	0	0	...	0	0
February	0	0	0	0	0	0	0	0	0	0	...	0	0
January	1	1	1	1	1	1	2	1	1	2	...	0	0
July	0	0	0	0	0	0	0	0	0	0	...	0	0
June	0	0	0	0	0	0	0	0	0	0	...	1	1
March	0	0	0	0	0	0	0	0	0	0	...	0	0
May	0	0	0	0	0	0	0	0	0	0	...	0	0
November	0	0	0	0	0	0	0	0	0	0	...	0	0
October	0	0	0	0	0	0	0	0	0	0	...	0	0
September	0	0	0	0	0	0	0	0	0	0	...	0	0

12 rows × 748 columns

min temprature in january

In [25]:
```
df=dates[dates['months']=='January']
df.head(10)
d2=df.sort_values('months',ascending=True)
d2
```

Out[25]:

	Date	Temp	Year	months
0	1901-01-01	17.99	1901	January
1008	1985-01-01	18.67	1985	January
996	1984-01-01	18.28	1984	January
984	1983-01-01	18.38	1983	January
972	1982-01-01	18.76	1982	January
...	...	...	...	...
384	1933-01-01	17.84	1933	January
372	1932-01-01	18.72	1932	January
360	1931-01-01	19.34	1931	January
504	1943-01-01	18.32	1943	January
1392	2017-01-01	20.59	2017	January

117 rows × 4 columns

Group by months and aggregate Temperature

In [26]:
```
d4=dates.groupby("months").Temp.sum().sort_values(ascending=False)
d4.head(5)
```

Out[26]:
```
months
May       3321.21
June      3311.21
July      3202.20
August    3151.99
April     3102.15
Name: Temp, dtype: float64
```

Temperature grater than 30

In [27]:
```
High_Temp=dates.loc[dates['Temp']>30]
High_Temp
```

Out[27]:

	Date	Temp	Year	months
244	1921-05-01	30.78	1921	May
1384	2016-05-01	30.41	2016	May
1396	2017-05-01	30.47	2017	May

Temperature less than 20

In [28]:
```
Low_Temp=dates.loc[dates['Temp']<20]
Low_Temp
```

Out[28]:

	Date	Temp	Year	months
0	1901-01-01	17.99	1901	January
1	1901-02-01	19.43	1901	February
11	1901-12-01	18.95	1901	December
12	1902-01-01	19.00	1902	January
23	1902-12-01	18.78	1902	December
...	...	...	...	...
1344	2013-01-01	18.88	2013	January
1355	2013-12-01	19.69	2013	December
1356	2014-01-01	18.81	2014	January
1367	2014-12-01	19.50	2014	December
1368	2015-01-01	19.02	2015	January

284 rows × 4 columns

In [29]:
```
#loc is rows and colmuns
#iloc: indexing the row and colu
```

copy

In [30]:
```
d1=dates.copy()
d1=d1.pivot_table('Temp',columns='months',aggfunc='sum')
d1
```

Out[30]:

months	April	August	December	February	January	July	June	March	May	Novemb
Temp	3102.15	3151.99	2243.28	2356.14	2155.52	3202.2	3311.21	2741.8	3321.21	2546

In [31]:
```
d2=dates.copy()
d2=d2.pivot_table('Temp',columns='months',aggfunc='min')
d2
```

Out[31]:

months	April	August	December	February	January	July	June	March	May	November	Oct
Temp	24.84	26.21	17.98	17.79	17.25	26.48	27.33	21.78	26.97	20.59	

In [32]:
```
d2=dates.copy()
d2=d2.pivot_table('Temp',columns='months',aggfunc='max')
d2
```

Out[32]:

months	April	August	December	February	January	July	June	March	May	November	Oct
Temp	29.56	28.17	21.89	23.58	20.92	28.47	29.88	26.61	30.78	23.92	

Conclusion:-

* I'm read the data in the variable df and then for visualising the data more clear I have convert the India weather data into time series data. And then I read my data into the variable dates and drop the unused column(unnamed) and apart from that I rename the columns of data as 'date':'Date','temp':'Temp','year':'Year','month':'mo

* There is not any null value shown in the datset.

* The name of the field available in the dataset are date','temp','year','month

* The datatype of the fields are date: datetime,temp:float,year: int,and month: object

* There are 141 value counts and unique values in Dates field are available, 141 value counts are available for the Temperature.

*** Hence by analysing the data of weather of India between the time period 1901-2017. We say that the highest Temp was 30.78 degree C in the year 1921 in the month may. And the lowest Temp. in India was 17.98 in the year January,1918.**

*** Apart from that, After the grouping months with aggregate temperature, we analysed that the Top hot month are may which is showing the highest aggregate temperature which is 3321.21, the second hot month is June: 3311.21, and third top hoest month is July: 3202.20**

*** After loc the dates columns with the temperature we are cheking the temp. greater than 30, So the output showing that may is the only month in which the temperature is more than 30. And the months in which temp lowers than 20 are the some winter months namely January, December, and February**

*** I'm copying the data into the another variable d1 and cso sove pivot table analysis of the categorical variable namely months and the numeric variable Temp and check the aggfunc by max and min function**

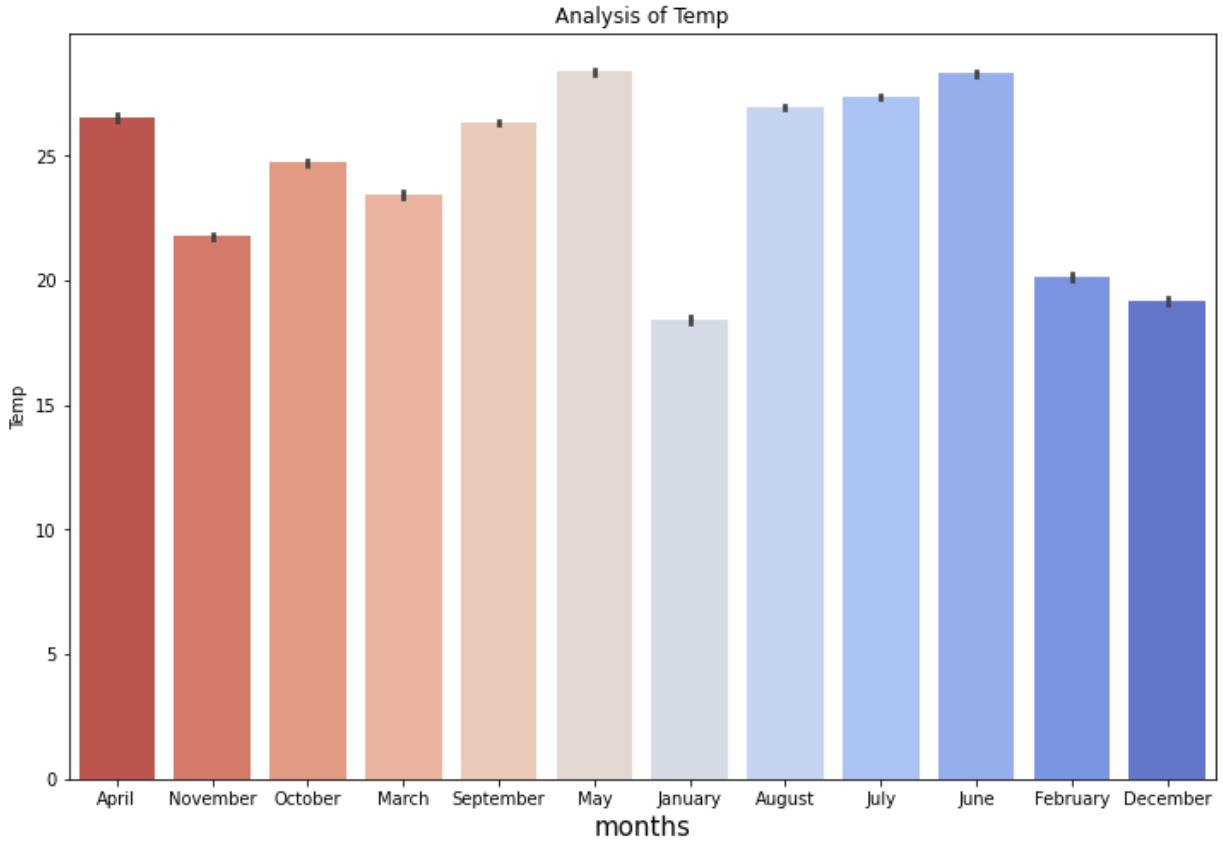
Data Visualization

In [33]:
```
1 import seaborn as sns
```

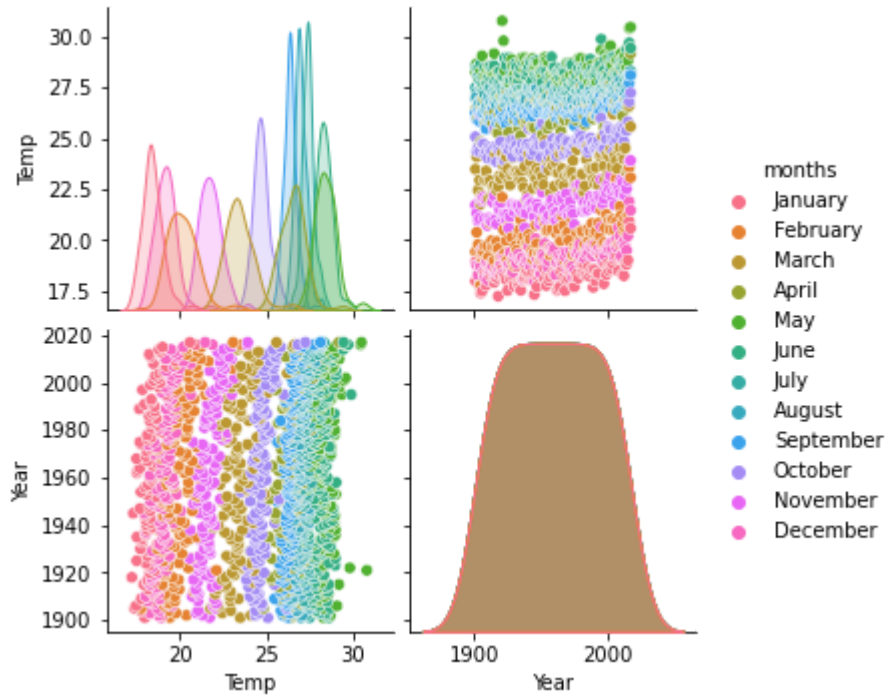
Bar Plot

In [34]:
```
plt.figure(figsize=(12,8))
plt.title("Analysis of Temp")
sns.barplot(x='months',y='Temp',data=dates,order=dates['months'].value_count
plt.xlabel('months',fontsize=15)
plt.show()

```

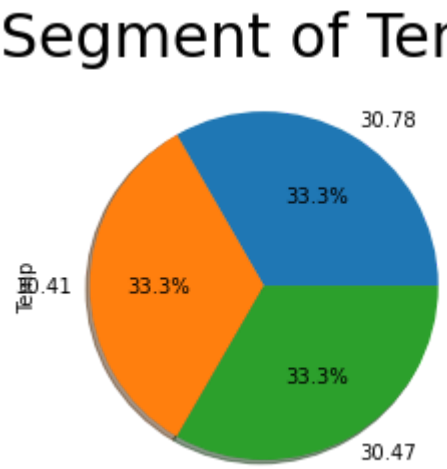


In [35]:
```
sns.pairplot(dates,hue='months')
plt.show()
```



Pie chart of max Temp

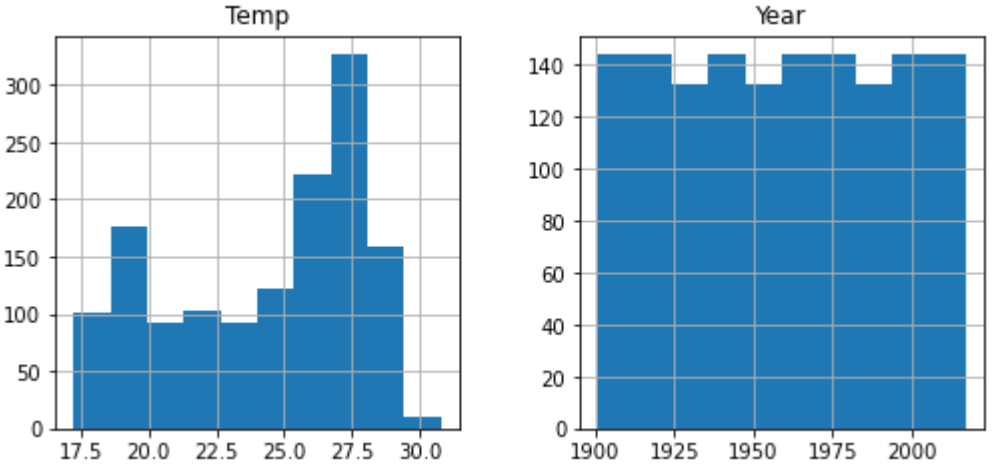
In [36]:
```
plt.title('Segment of Temp',fontsize=30)
High_Temp['Temp'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```



Hist Plot

In [37]:
```
dates.hist(figsize=(13,13),layout=(3,3),sharex=False)
```

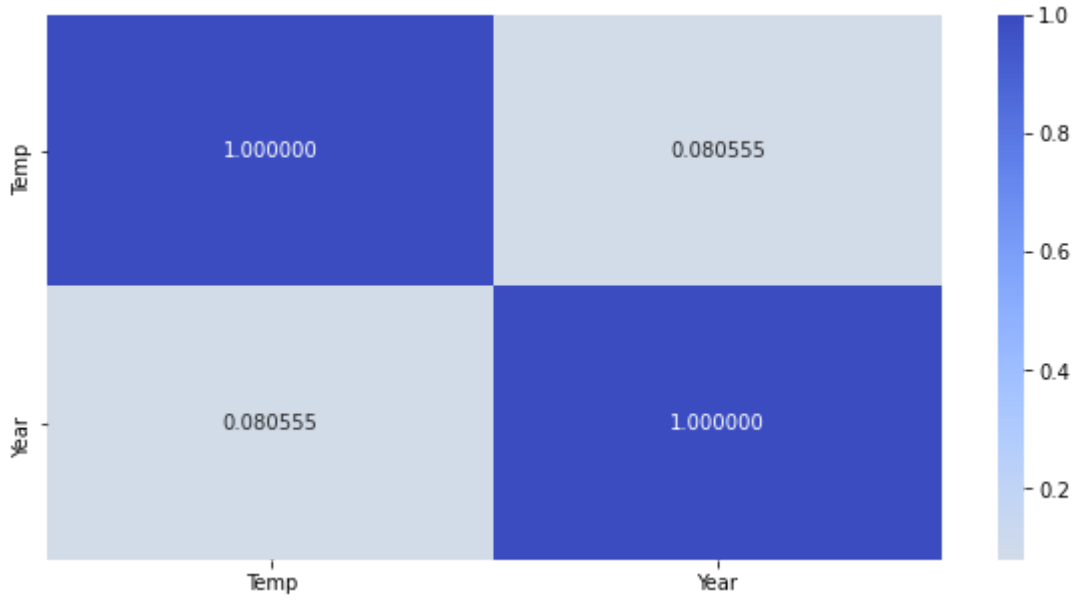
Out[37]:
```
array([[<AxesSubplot:title={'center':'Temp'}>,
        <AxesSubplot:title={'center':'Year'}>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



In [38]:
```
#Heatmap
```

In [39]:
```
plt.figure(figsize=(10,5))
sns.heatmap(dates.corr(),annot=True,fmt='4f',cmap='coolwarm_r',center=0)
```

Out[39]: <AxesSubplot:>



#conclusion

*** By analysing the bar plot we analyse that may and june are the top hot months in India.**

*****

*** By analsying the top max temp by using the pie chart, we say that 33.3% in the dataset has the temperature 30.78, 33.3% data in the dataset have the temp 30.45**

*** By using Histogrm we count the number of temperatures that shows in the datasets and apredict that temp 26 to 32 is the most frequent temperature that shows in the datset**

Heatmap tell us there is highly correlation between tha temp and year.