Importing the required libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Reading the csv file in data variable

In [2]:
```
A =pd.read_csv(r"C:\Users\1pooj\Downloads\archive\temperatures.csv")
```

Display first five rows of dataset

In [3]:
```
A.head()
```

Out[3]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNU
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21	30.39	30.47	29.97	27.31	24.49	28
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92	30.73	29.80	29.12	26.31	24.04	29
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34	29.98	29.85	29.04	26.08	23.65	28
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36	30.09	30.04	29.20	26.36	23.63	28
4	1905	22.00	22.83	26.68	30.01	33.32	33.25	31.44	30.68	30.12	30.67	27.52	23.82	28

Display last five rows of dataset

In [4]:
```
A.tail()
```

Out[4]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANI
112	2013	24.56	26.59	30.62	32.66	34.46	32.44	31.07	30.76	31.04	30.27	27.83	25.37	
113	2014	23.83	25.97	28.95	32.74	33.77	34.15	31.85	31.32	30.68	30.29	28.05	25.08	
114	2015	24.58	26.89	29.07	31.87	34.09	32.48	31.88	31.52	31.55	31.04	28.10	25.67	
115	2016	26.94	29.72	32.62	35.38	35.72	34.03	31.64	31.79	31.66	31.98	30.11	28.01	
116	2017	26.45	29.46	31.60	34.95	35.84	33.82	31.88	31.72	32.22	32.29	29.60	27.18	

Describes the dataset

In [5]:
```
A.describe()
```

Out[5]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	
count	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.(
mean	1959.000000	23.687436	25.597863	29.085983	31.975812	33.565299	32.774274	31.(
std	33.919021	0.834588	1.150757	1.068451	0.889478	0.724905	0.633132	0.4
min	1901.000000	22.000000	22.830000	26.680000	30.010000	31.930000	31.100000	29.7
25%	1930.000000	23.100000	24.780000	28.370000	31.460000	33.110000	32.340000	30.7
50%	1959.000000	23.680000	25.480000	29.040000	31.950000	33.510000	32.730000	31.(
75%	1988.000000	24.180000	26.310000	29.610000	32.420000	34.030000	33.180000	31.3
max	2017.000000	26.940000	29.720000	32.620000	35.380000	35.840000	34.480000	32.7

Display the shape of the dataset i.e (row,columns)

In [6]:
```
A.shape
```

Out[6]:
```
(117, 18)
```

Find the missing values in dataset

In [7]:
```
A.isnull()
```

Out[7]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	False	False	False	False	False	False	False	False	False	False	False	False	False	False
113	False	False	False	False	False	False	False	False	False	False	False	False	False	False
114	False	False	False	False	False	False	False	False	False	False	False	False	False	False
115	False	False	False	False	False	False	False	False	False	False	False	False	False	False
116	False	False	False	False	False	False	False	False	False	False	False	False	False	False

117 rows × 18 columns

Display the data types of the dataset

In [8]:
```
A.dtypes
```

Out[8]:
```
YEAR         int64
JAN        float64
FEB        float64
MAR        float64
APR        float64
MAY        float64
JUN        float64
JUL        float64
AUG        float64
SEP        float64
OCT        float64
NOV        float64
DEC        float64
ANNUAL     float64
JAN-FEB    float64
MAR-MAY    float64
JUN-SEP    float64
OCT-DEC    float64
dtype: object
```

Display the names of the columns in the dataset

In [9]:
```
A.columns
```

Out[9]:
```
Index(['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP',
       'OCT', 'NOV', 'DEC', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP',
       'OCT-DEC'],
      dtype='object')
```

x value is 'year' and y is 'feb'

In [10]:
```
from sklearn.model_selection import train_test_split
x=A[['YEAR']]
y=A[['FEB']]
```

Spliting the training and testing data into xtrain, xtest, ytrain, ytest

In [ ]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=30)
```

Importing the required libraries from sklearn.linear_model

In [11]:
```
from sklearn.linear_model import LinearRegression
```

Read the LinearRegression function in reg variable

In [ ]:
```
reg= LinearRegression()
model =reg.fit(xtrain,ytrain)
```

Print the intercept value and coeff of the model

In [12]:
```
print(model.intercept_)
print(model.coef_)
```

```
[-21.98480901]
[[0.02430294]]
```

store the predicted data in ypredict

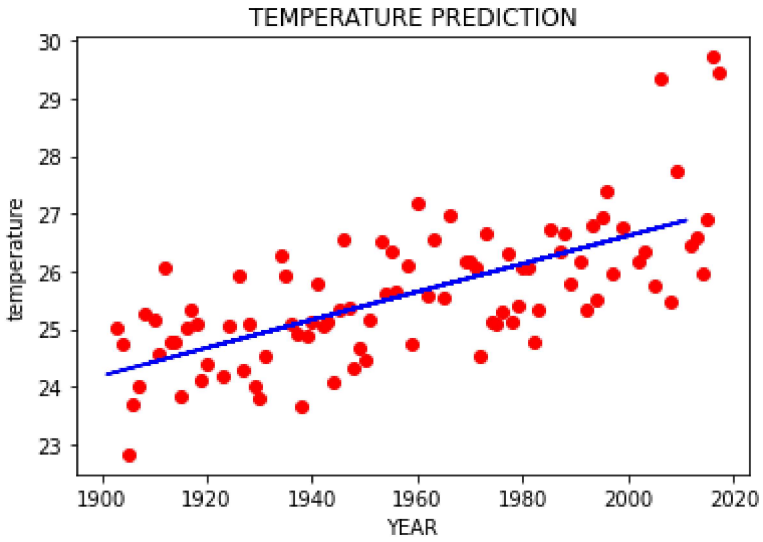
In [13]:
```
ypredict=model.predict(xtest)
print("prediction",ypredict)
```

```
prediction [[26.37804672]
 [24.23938778]
 [24.21508483]
 [26.62107615]
 [24.40950838]
 [26.88840852]
 [24.992779  ]
 [26.86410558]
 [26.71828792]
 [25.84338199]
 [25.45453491]
 [25.67326139]
 [25.57604962]
 [24.72544663]
 [26.79119675]
 [24.70114369]
 [26.57247026]
 [24.96847605]
 [25.81907904]
 [26.64537909]
 [25.74617022]
 [24.79835546]
 [26.23222907]
 [26.28083495]]
```

Give the title, xlabel, ylabel to the graph and plot the line using plt

In [14]:
```
plt.title("TEMPERATURE PREDICTION")
plt.xlabel("YEAR")
plt.ylabel("temperature")
plt.scatter(xtrain,ytrain,color="red")
plt.plot(xtest,reg.predict(xtest),color="blue")
```

Out[14]: [<matplotlib.lines.Line2D at 0x1a84e3c19c0>]



Find the MSE, RMSE, MAE, R2_SCORE metrics values of the given graph

In [15]:
```
from sklearn.metrics import mean_squared_error
import math
mse=mean_squared_error(ytest,ypredict)
rmse=math.sqrt(mse)
print("MSE: ",mse)
print("RMSE: ",rmse)
```

```
MSE:  0.8541305992757143
RMSE:  0.9241918628053994
```

In [16]:
```
from sklearn import metrics
```

In [17]:
```
metrics.mean_absolute_error(ytest,ypredict)
```

Out[17]: 0.7323622543764404

In [19]:
```
R2=metrics.r2_score(ytest,ypredict)
print("R2_score",R2)
```

```
R2_score 0.1421371013771522
```

x is 'year' and y is 'jul'

In [20]:
```
from sklearn.model_selection import train_test_split
x=A[['YEAR']]
y=A[['JUL']]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=30)
```

In [21]:
```
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
model =reg.fit(xtrain,ytrain)
```

In [22]:
```
print(model.intercept_)
print(model.coef_)
```

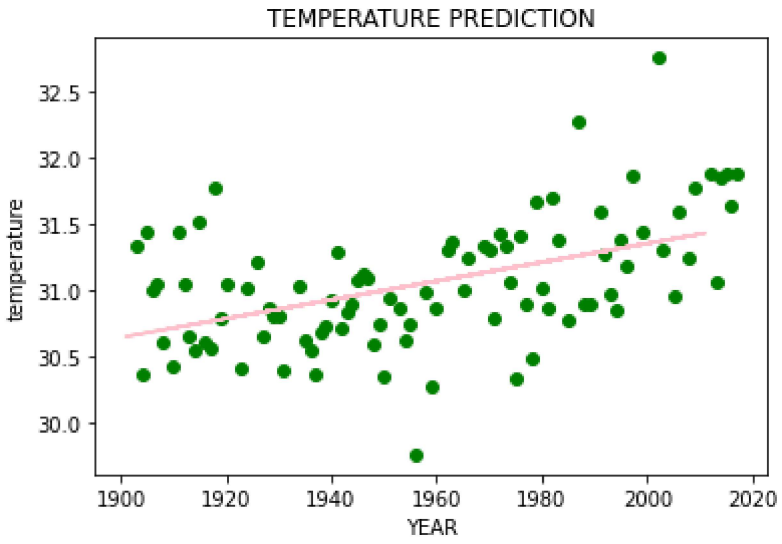
```
[17.14932707]
[[0.00710322]]
```

In [23]:
```
ypredict=model.predict(xtest)
print("prediction",ypredict)
```

```
prediction [[31.28473954]
 [30.65965597]
 [30.65255275]
 [31.35577176]
 [30.70937853]
 [31.4339072 ]
 [30.87985586]
 [31.42680398]
 [31.38418465]
 [31.12846864]
 [31.01481709]
 [31.07874609]
 [31.0503332 ]
 [30.80172042]
 [31.40549432]
 [30.79461719]
 [31.34156531]
 [30.87275264]
 [31.12136542]
 [31.36287498]
 [31.10005575]
 [30.82303008]
 [31.2421202 ]
 [31.25632665]]
```

In [27]:
```
plt.title("TEMPERATURE PREDICTION")
plt.xlabel("YEAR")
plt.ylabel("temperature")
plt.scatter(xtrain,ytrain,color="GREEN")
plt.plot(xtest,reg.predict(xtest),color="PINK")
```

Out[27]:
```
[<matplotlib.lines.Line2D at 0x1a8505f2c80>]
```



In [28]:
```
from sklearn.metrics import mean_squared_error
import math
mse=mean_squared_error(ytest,ypredict)
rmse=math.sqrt(mse)
print("MSE: ",mse)
print("RMSE: ",rmse)
from sklearn import metrics
```

```
MAE=metrics.mean_absolute_error(ytest,ypredict)
print("MAE:",MAE)
R2=metrics.r2_score(ytest,ypredict)
print("R2_score",R2)
```

```
MSE:  0.1407531338918451
RMSE:  0.3751708062894088
MAE: 0.28177828396273386
R2_score 0.09630317486551376
```

x is 'year' and y is 'annual'

In [29]:
```
from sklearn.model_selection import train_test_split
x=A[['YEAR']]
y=A[['ANNUAL']]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=30)
```

In [30]:
```
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
model =reg.fit(xtrain,ytrain)
```

In [31]:
```
print(model.intercept_)
print(model.coef_)
```

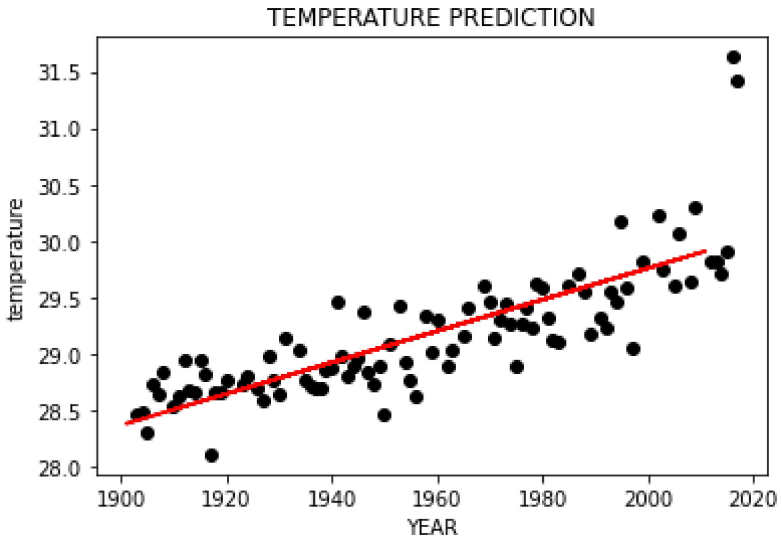
```
[1.96474555]
[[0.01389603]]
```

In [32]:
```
ypredict=model.predict(xtest)
print("prediction",ypredict)
```

```
prediction [[29.61783862]
 [28.39498827]
 [28.38109224]
 [29.75679888]
 [28.49226046]
 [29.90965518]
 [28.8257651 ]
 [29.89575915]
 [29.81238299]
 [29.31212603]
 [29.0897896 ]
 [29.21485384]
 [29.15926974]
 [28.6729088 ]
 [29.85407107]
 [28.65901278]
 [29.72900683]
 [28.81186907]
 [29.29823   ]
 [29.77069491]
 [29.25654192]
 [28.71459688]
 [29.53446246]
 [29.56225451]]
```

In [34]:
```
plt.title("TEMPERATURE PREDICTION")
plt.xlabel("YEAR")
plt.ylabel("temperature")
plt.scatter(xtrain,ytrain,color="BLACK")
plt.plot(xtest,reg.predict(xtest),color="RED")
```

Out[34]:
```
[<matplotlib.lines.Line2D at 0x1a8506daa10>]
```



In [35]:
```
from sklearn.metrics import mean_squared_error
import math
mse=mean_squared_error(ytest,ypredict)
rmse=math.sqrt(mse)
print("MSE: ",mse)
print("RMSE: ",rmse)
from sklearn import metrics
MAE=metrics.mean_absolute_error(ytest,ypredict)
print("MAE:",MAE)
R2=metrics.r2_score(ytest,ypredict)
print("R2_score",R2)
```

```
MSE:  0.09864470576722391
RMSE:  0.31407754737838856
MAE: 0.23891665295417694
R2_score 0.5764036503216072
```

x is 'year' y is 'jan-feb'

In [36]:
```
from sklearn.model_selection import train_test_split
x=A[['YEAR']]
y=A[['JAN-FEB']]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=30)
```

In [37]:
```
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
model =reg.fit(xtrain,ytrain)
```

In [38]:
```
print(model.intercept_)
print(model.coef_)
```

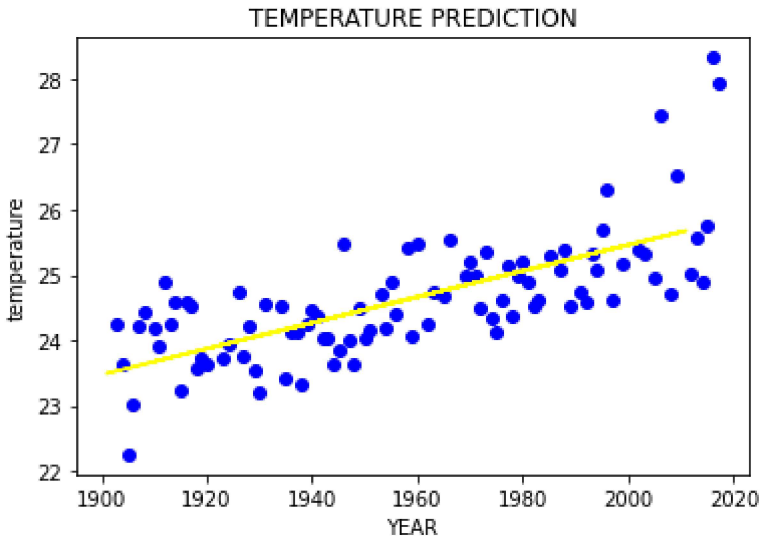
```
[-14.26520785]
[[0.01986202]]
```

In [39]:
```
ypredict=model.predict(xtest)
print("prediction",ypredict)
```

```
prediction [[25.26021427]
 [23.51235641]
 [23.49249438]
 [25.45883448]
 [23.65139055]
 [25.67731671]
 [24.12807906]
 [25.65745469]
 [25.53828256]
 [24.8232498 ]
 [24.50545746]
 [24.68421565]
 [24.60476757]
 [23.90959683]
 [25.59786863]
 [23.88973481]
 [25.41911044]
 [24.10821704]
 [24.80338778]
 [25.4786965 ]
 [24.74380172]
 [23.96918289]
 [25.14104214]
 [25.18076618]]
```

In [40]:
```
plt.title("TEMPERATURE PREDICTION")
plt.xlabel("YEAR")
plt.ylabel("temperature")
plt.scatter(xtrain,ytrain,color="BLUE")
plt.plot(xtest,reg.predict(xtest),color="YELLOW")
```

Out[40]:
```
[<matplotlib.lines.Line2D at 0x1a85073ed10>]
```



In [41]:
```
from sklearn.metrics import mean_squared_error
import math
mse=mean_squared_error(ytest,ypredict)
rmse=math.sqrt(mse)
print("MSE: ",mse)
print("RMSE: ",rmse)
from sklearn import metrics
MAE=metrics.mean_absolute_error(ytest,ypredict)
print("MAE:",MAE)
R2=metrics.r2_score(ytest,ypredict)
print("R2_score",R2)
```

```
MSE:  0.46285748596785686
RMSE:  0.6803363035792349
MAE: 0.5051591729534501
R2_score 0.2652276216543984
```

x is 'year' and y is'oct-dec'

In [42]:
```
from sklearn.model_selection import train_test_split
x=A[['YEAR']]
y=A[['OCT-DEC']]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=30)
```

In [43]:
```
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
model =reg.fit(xtrain,ytrain)
```

In [44]:
```
print(model.intercept_)
print(model.coef_)
```

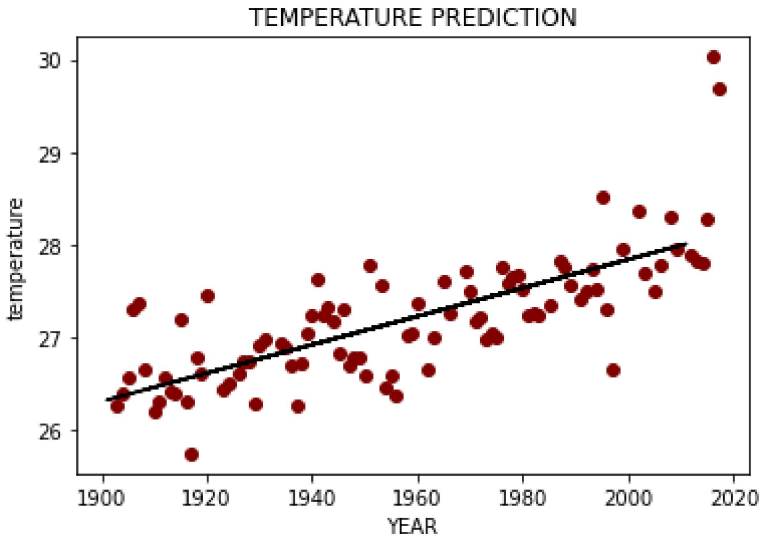
```
[-2.85848612]
[[0.0153481]]
```

In [45]:
```
ypredict=model.predict(xtest)
print("prediction",ypredict)
```

```
prediction [[27.68424266]
 [26.33360943]
 [26.31826133]
 [27.83772371]
 [26.44104617]
 [28.00655287]
 [26.80940068]
 [27.99120476]
 [27.89911613]
 [27.34658436]
 [27.10101468]
 [27.23914762]
 [27.1777552 ]
 [26.64057153]
 [27.94516045]
 [26.62522343]
 [27.8070275 ]
 [26.79405258]
 [27.33123625]
 [27.85307182]
 [27.28519194]
 [26.68661585]
 [27.59215404]
 [27.62285025]]
```

In [46]:
```
plt.title("TEMPERATURE PREDICTION")
plt.xlabel("YEAR")
plt.ylabel("temperature")
plt.scatter(xtrain,ytrain,color="MAROON")
plt.plot(xtest,reg.predict(xtest),color="BLACK")
```

Out[46]:
```
[<matplotlib.lines.Line2D at 0x1a8507b6c20>]
```



In [47]:
```
from sklearn.metrics import mean_squared_error
import math
mse=mean_squared_error(ytest,ypredict)
rmse=math.sqrt(mse)
print("MSE: ",mse)
print("RMSE: ",rmse)
from sklearn import metrics
MAE=metrics.mean_absolute_error(ytest,ypredict)
print("MAE:",MAE)
R2=metrics.r2_score(ytest,ypredict)
print("R2_score",R2)
```

```
MSE:  0.1431725061405512
RMSE:  0.378381429434045
MAE: 0.2918738663196154
R2_score 0.6092081245941505
```

In [ ]: