In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [2]:

```
trainData = pd.read_csv("D:/Users/Janhavi/TE/TE1/ML/temperatures.csv")
```

In [3]:

```
trainData.head(n=10)  #Returns the first 10 rows of the dataframe
```

Out[3]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
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
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
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
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
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
5	1906	22.28	23.69	27.31	31.93	34.11	32.19	31.01	30.30	29.92	29.55	27.60	24.72
6	1907	24.46	24.01	27.04	31.79	32.68	31.92	31.05	29.58	30.67	29.87	27.78	24.44
7	1908	23.57	25.26	28.86	32.42	33.02	33.12	30.61	29.55	29.59	29.35	26.88	23.73
8	1909	22.67	24.36	29.22	30.79	33.06	31.70	29.81	29.81	30.06	29.25	27.69	23.69
9	1910	23.24	25.16	28.48	31.42	33.51	31.84	30.42	29.86	29.82	28.91	26.32	23.37

In [4]:

```
trainData.dtypes  #it is used to find datatype
trainData.columns  # to print column name
```

Out[4]:

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

In [5]:

```
trainData.describe()
# It is used to view some basic statistical details like percentile, mean, std, etc. of
a DataFrame
```

Out[5]:

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

In [6]:

```
trainData.isnull().sum()  # return the number of missing values
```

Out[6]:

```
YEAR       0
JAN        0
FEB        0
MAR        0
APR        0
MAY        0
JUN        0
JUL        0
AUG        0
SEP        0
OCT        0
NOV        0
DEC        0
ANNUAL     0
JAN-FEB    0
MAR-MAY    0
JUN-SEP    0
OCT-DEC    0
dtype: int64
```

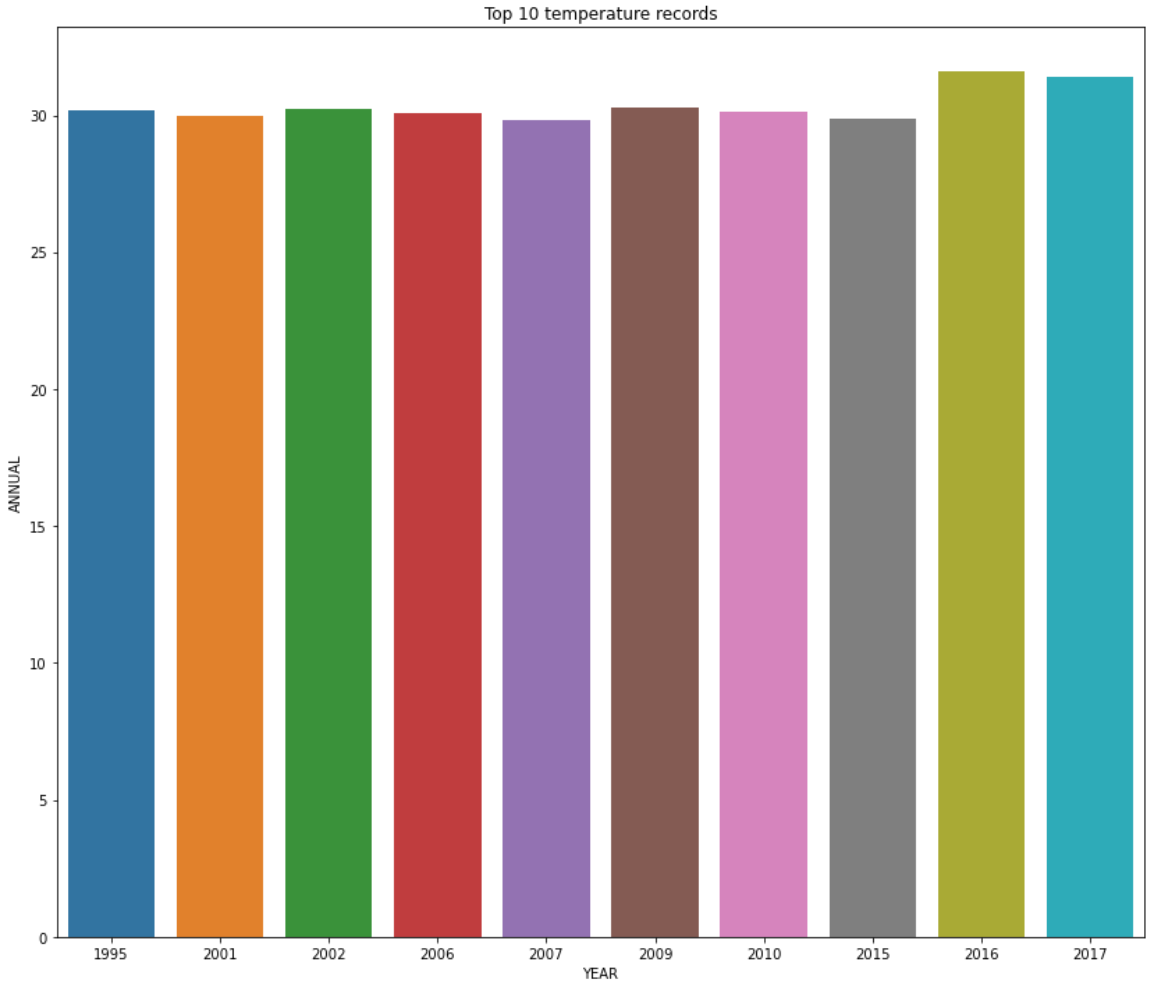
In [7]:

```
# print top 10 temperature records

top_10_data = trainData.nlargest(10, "ANNUAL")
plt.figure(figsize=(14,12))
plt.title("Top 10 temperature records")
sns.barplot(x=top_10_data.YEAR, y=top_10_data.ANNUAL)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20f90a1aa90>
```



In [8]:

```
from sklearn import linear_model, metrics
# Linear_model is a class of the sklearn module if contain different functions for perf
orming machine learning with linear models.
```

In [9]:

```
trainData.columns #Print columns
```

Out[9]:

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

In [10]:

```
# Store values for training and testing
X=trainData[["YEAR"]]
Y=trainData[["JAN"]]
```

In [11]:

```
#Creating training and testing model
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1
)
```

In [12]:

```
len(X_train)
```

Out[12]:

93

In [13]:

```
len(X_test)
```

Out[13]:

24

In [14]:

```
trainData.shape
```

Out[14]:

(117, 18)

In [15]:

```
# linear_model . LinearRegression. Ordinary least squares Linear Regression.
# LinearRegression fits a linear model with coefficients w = (w1, …, wp) to minimize th
e residual sum of
# squares between the observed targets in the dataset, and the targets predicted by the
linear approximation.
```

In [16]:

```
reg = linear_model.LinearRegression()
```

In [17]:

```
print(X_train)
```

```
     YEAR
56   1957
94   1995
35   1936
38   1939
93   1994
..    ...
9    1910
72   1973
12   1913
107  2008
37   1938

[93 rows x 1 columns]
```

In [18]:

```
model = reg.fit(X_train, Y_train)
```

In [19]:

```
r_sq = reg.score(X_train, Y_train)
# reg.score: Returns the coefficient of determination (R^2). A measure of how well obse
rved outcomes are replicated by the model, as the proportion of total variation of outc
omes explained by the model.
```

In [20]:

```
print("determination coefficient:", r_sq)
```

```
determination coefficient: 0.35480458491221156
```

In [21]:

```
print("intercept:", model.intercept_)
```

```
intercept: [-5.35338281]
```

In [22]:

```
print('slope:', model.coef_)
```

```
slope: [[0.01486008]]
```

In [23]:

```
Y_pred = model.predict(X_test)
print('predicted response:', Y_pred, sep='\n')
```

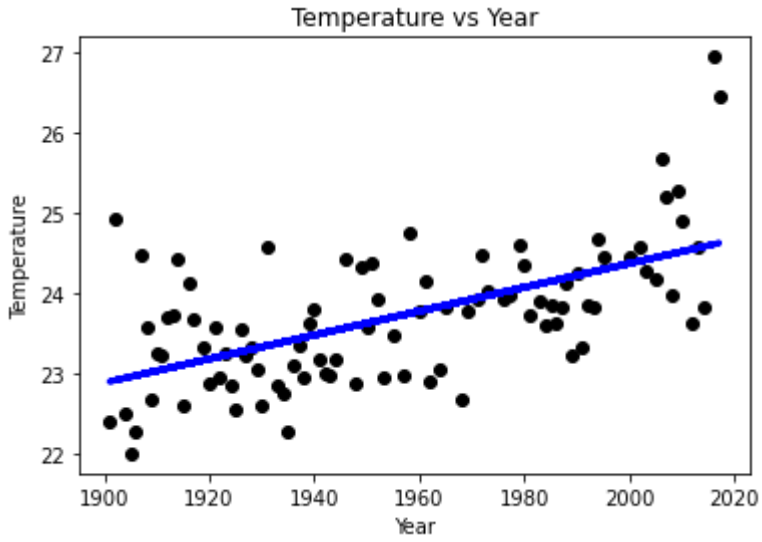
```
predicted response:
[[23.92097555]
 [23.5791937 ]
 [23.75751466]
 [24.58967916]
 [23.98041587]
 [24.35191788]
 [23.35629249]
 [23.68321426]
 [23.86153523]
 [24.32219772]
 [24.30733764]
 [24.3370578 ]
 [22.92535016]
 [23.81695498]
 [24.53023884]
 [23.71293442]
 [24.42621828]
 [24.38163804]
 [23.87639531]
 [23.54947354]
 [24.03985619]
 [23.14825137]
 [24.09929651]
 [23.99527595]]
```

In [24]:

```
# Visualise simple regression model.
```

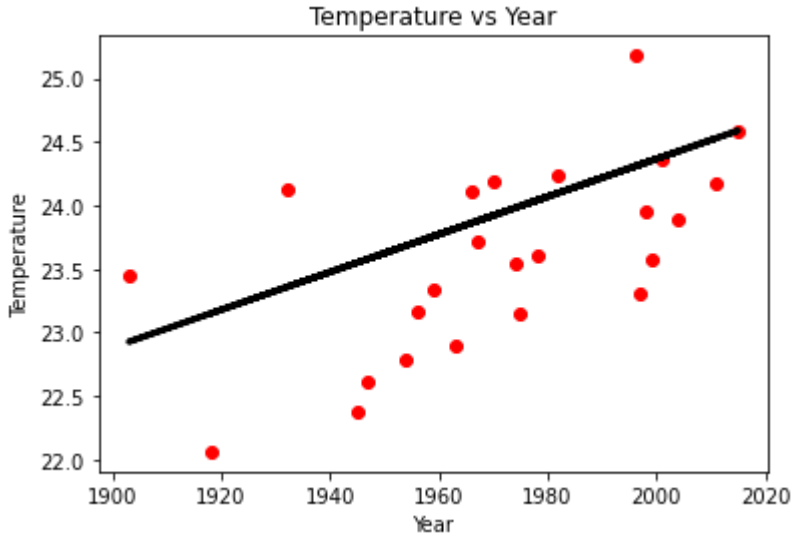
In [25]:

```
plt.scatter(X_train, Y_train, color='black')
plt.plot(X_train, reg.predict(X_train), color='blue', linewidth=3)
plt.title("Temperature vs Year")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.show()
```



In [26]:

```
plt.scatter(X_test, Y_test, color='red')
plt.plot(X_test, reg.predict(X_test), color='black', linewidth=3)
plt.title("Temperature vs Year")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.show()
```



In [27]:

```
print(Y_test)
```

```
       JAN
69   24.19
46   22.61
58   23.33
114  24.58
73   23.54
98   23.57
31   24.13
53   22.79
65   24.11
96   23.30
95   25.18
97   23.95
2    23.44
62   22.90
110  24.18
55   23.16
103  23.89
100  24.36
66   23.72
44   22.38
77   23.60
17   22.06
81   24.23
74   23.15
```

In [28]:

```
# Assess the performance of regression models using MSE, MAE and R-Square metrics
```

In [29]:

```
from sklearn.metrics import mean_absolute_error
print("MAE",mean_absolute_error(Y_test,Y_pred))
```

```
MAE 0.575735016242341
```

In [30]:

```
from sklearn.metrics import mean_squared_error
print("MSE",mean_squared_error(Y_test,Y_pred))
```

```
MSE 0.4474127538283697
```

In [31]:

```
from sklearn.metrics import r2_score
r2=r2_score(Y_test,Y_pred)
print(r2)
```

```
0.13772507149336943
```

In [ ]:

```

```