In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import plotly.express as px
plt.style.use('Solarize_Light2')
plt.style.context('grayscale')
%matplotlib inline

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

In [2]:
```
df = pd.read_csv("Summary of weather.csv")
```

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

Out[3]:

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather	YR	...	FB	FTI	ITH	PGT	TSHDSBRSGF	SD3	RHX	RH
0	10001	1942-7-1	1.016	NaN	25.555556	22.222222	23.888889	0.0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	10001	1942-7-2	0	NaN	28.888889	21.666667	25.555556	0.0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	10001	1942-7-3	2.54	NaN	26.111111	22.222222	24.444444	0.0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	10001	1942-7-4	2.54	NaN	26.666667	22.222222	24.444444	0.0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	10001	1942-7-5	0	NaN	26.666667	21.666667	24.444444	0.0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 31 columns

In [4]:
```
df.dropna(axis = 1,inplace = True)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119040 entries, 0 to 119039
Data columns (total 9 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   STA       119040 non-null  int64
 1   Date      119040 non-null  object
 2   Precip    119040 non-null  object
 3   MaxTemp   119040 non-null  float64
 4   MinTemp   119040 non-null  float64
 5   MeanTemp  119040 non-null  float64
 6   YR        119040 non-null  int64
 7   MO        119040 non-null  int64
 8   DA        119040 non-null  int64
dtypes: float64(3), int64(4), object(2)
memory usage: 8.2+ MB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
STA         0
Date        0
Precip      0
MaxTemp     0
MinTemp     0
MeanTemp    0
YR          0
MO          0
DA          0
dtype: int64
```

In [7]:
```
df.describe()
```

Out[7]:

	STA	MaxTemp	MinTemp	MeanTemp	YR	MO	DA
count	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000	119040.000000
mean	29659.435795	27.045111	17.789511	22.411631	43.805284	6.726016	15.797530
std	20953.209402	8.717817	8.334572	8.297982	1.136718	3.425561	8.794541
min	10001.000000	-33.333333	-38.333333	-35.555556	40.000000	1.000000	1.000000
25%	11801.000000	25.555556	15.000000	20.555556	43.000000	4.000000	8.000000
50%	22508.000000	29.444444	21.111111	25.555556	44.000000	7.000000	16.000000
75%	33501.000000	31.666667	23.333333	27.222222	45.000000	10.000000	23.000000
max	82506.000000	50.000000	34.444444	40.000000	45.000000	12.000000	31.000000

In [8]:
```
df.corr()
```

Out[8]:

	STA	MaxTemp	MinTemp	MeanTemp	YR	MO	DA
STA	1.000000	0.092371	0.059319	0.078112	0.121408	-0.008592	0.000903
MaxTemp	0.092371	1.000000	0.878384	0.969048	0.039585	0.031346	-0.005130
MinTemp	0.059319	0.878384	1.000000	0.965425	-0.020733	0.069078	-0.002576
MeanTemp	0.078112	0.969048	0.965425	1.000000	0.010681	0.050769	-0.004153
YR	0.121408	0.039585	-0.020733	0.010681	1.000000	-0.144360	-0.011196
MO	-0.008592	0.031346	0.069078	0.050769	-0.144360	1.000000	0.006563
DA	0.000903	-0.005130	-0.002576	-0.004153	-0.011196	0.006563	1.000000

In [9]:
```
df[['year','months','day']] = df['Date'].str.split('-',expand = True)
```

In [10]:
```
df[['year','months','day']] = df[['year','months','day']].astype('int')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119040 entries, 0 to 119039
Data columns (total 12 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   STA       119040 non-null  int64  
 1   Date      119040 non-null  object 
 2   Precip    119040 non-null  object 
 3   MaxTemp   119040 non-null  float64
 4   MinTemp   119040 non-null  float64
 5   MeanTemp  119040 non-null  float64
 6   YR        119040 non-null  int64  
 7   MO        119040 non-null  int64  
 8   DA        119040 non-null  int64  
 9   year      119040 non-null  int32  
 10  months    119040 non-null  int32  
 11  day       119040 non-null  int32  
dtypes: float64(3), int32(3), int64(4), object(2)
memory usage: 9.5+ MB
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['STA', 'Date', 'Precip', 'MaxTemp', 'MinTemp', 'MeanTemp', 'YR', 'MO',
       'DA', 'year', 'months', 'day'],
      dtype='object')
```

In [12]:
```
x = df[['STA', 'MinTemp', 'MeanTemp', 'YR', 'MO',
       'DA','year']]
y = df['MaxTemp']
```

Using Linear Regression

In [13]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [14]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size = 0.7,random_state = True)
```

In [15]:
```
model = LinearRegression()
model.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
pred = model.predict(x_test)
```

In [18]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,pred))
print("Mean Square Error:",metrics.mean_squared_error(y_test,pred))
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
Mean Absolute Error: 0.395549962844446
Mean Square Error: 1.1083848633780486
Root Mean Square Error: 1.052798586329811
```

Using KNN

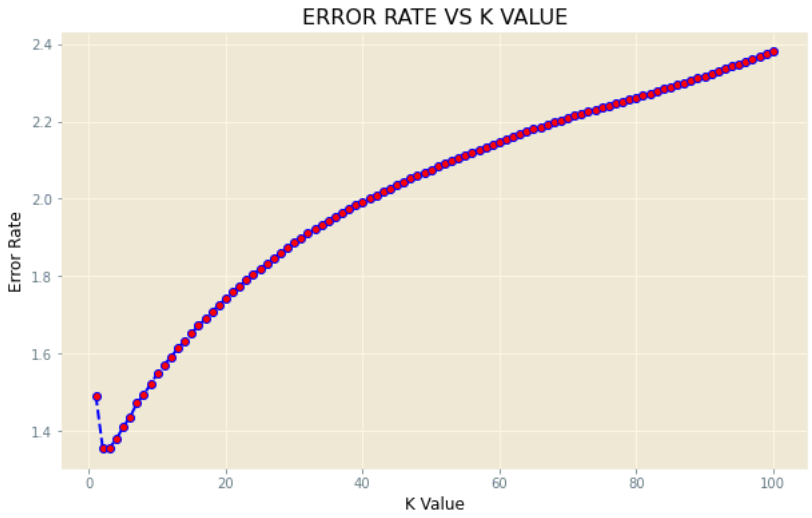
In [24]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [30]:
```
error_rate = []
for i in range(1,101):
    knn = KNeighborsRegressor(n_neighbors = i)
    knn.fit(x_train,y_train)
    pred_i = knn.predict(x_test)
    error = np.sqrt(metrics.mean_squared_error(pred_i,y_test))
    error_rate.append(error)
error_rate
```

Out[30]: [1.4911252954899155,
1.3554417036158122,
1.3567427922130584,
1.3786767776523698,
1.410583255773618,
1.4355133407200436,
1.4719530291453806,
1.493729626134978,
1.5208476531204707,
1.5486230540377732,
1.5712407384691125,
1.5917140850060947,
1.613286773976652,
1.6325727055154557,
1.6532899868369295,
1.6720532495259457,
1.689618506402281,
1.7085629365513646,
1.7264882724060227,
1.7420494505030635,
1.7582785221566273,
1.7745824588404455,
1.7893066950298264,
1.8041638338929589,
1.819571206699218,
1.8335388572885776,
1.847397895011957,
1.8597788347517459,
1.8727429875695767,
1.8864091076272942,
1.898361427668976,
1.9104255184383634,
1.9215052587566128,
1.9324425844510549,
1.9439790964135357,
1.9538517650328886,
1.964056237335846,
1.9746305406767894,
1.983898293371329,
1.9927427671071452,
2.0013748011676094,
2.0100743238556382,
2.018432363376531,
2.0275624897088096,
2.0360619809573066,
2.043899928714871,
2.0521833709307775,
2.0606246600574805,
2.0682319086625687,
2.076012190151337,
2.083789171969893,
2.0911066863307437,
2.0986505082151923,
2.1063705735157368,
2.1138625840473 06,
2.1208371553139767,
2.1276340592717853,
2.1343144536083964,
2.141516444703985,
2.1475193306261358,
2.1540038665648154,
2.1604965141895045,
2.167054529747243,
2.173563070114832,
2.1799468953864474,
2.1860639578563483,
2.191399818663122,
2.19757785708828,
2.203325100576369,
2.209222590084846,
2.2144635506521144,
2.21953378696538,
2.225296481467355,
2.230564400715235,
2.2355648287009764,
2.240442287847749,
2.2458546332142033,
2.2512478056083602,
2.2563953861361017,
2.2617415678196116,
2.267351729000716,
2.2726698671842342,
2.278260859825988,
2.2837063122728853,
2.288814314425682,
2.2943278553562956,

```
2.3001342782387426,
2.305622581169418,
2.311371436715658,
2.3174135874890207,
2.3234558726989514,
2.3294641725792844,
2.3358893234954214,
2.342239857665017,
2.3484639622472048,
2.3546108077958383,
2.360909463834886,
2.3672069981686557,
2.3740540596037905,
2.38031730042987]
```

In [37]:
```
plt.figure(figsize = (10,6))
plt.plot(range(1,101),error_rate,linestyle = 'dashed',color = 'blue',marker = 'o',markerfacecolor = 'red')
plt.xlabel('K Value',color = 'black')
plt.ylabel('Error Rate',color = 'black')
plt.title("ERROR RATE VS K VALUE")
plt.show()
```



In [32]:
```
knn2 = KNeighborsRegressor(n_neighbors = 2)
knn2.fit(x_train,y_train)
pred1 = knn2.predict(x_test)
```

In [33]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,pred1))
print("Mean Square Error:",metrics.mean_squared_error(y_test,pred1))
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,pred1)))
```

```
Mean Absolute Error: 0.9397168088019012
Mean Square Error: 1.8372222119009354
Root Mean Square Error: 1.3554417036158122
```

In [52]:
```
print('Enter STA, MinTemp, MeanTemp, YR, MO, DA, year ')
a,b,c,d,e,f,g = map(float,input().split())
print(f"Max Temprature according to Logistic Regression: {model.predict([[a,b,c,d,e,f,g]])}")
print(f"Max Temprature according to KNN: {knn2.predict([[a,b,c,d,e,f,g]])}")
```

```
Enter STA, MinTemp, MeanTemp, YR, MO, DA, year
10001 22.222 23.88889 42 7 1 1942
Max Temprature according to Logistic Regression: [25.81476985]
Max Temprature according to KNN: [25.83333333]
```

In [ ]: