ASSIGNMENT 2

In [4]:
```
import pandas as pd
df=pd.read_csv(r'temperatures.csv')
print(df)
```

```
     YEAR    JAN    FEB    MAR    APR    MAY    JUN    JUL    AUG    SEP  \
0    1901  22.40  24.14  29.07  31.91  33.41  33.18  31.21  30.39  30.47
1    1902  24.93  26.58  29.77  31.78  33.73  32.91  30.92  30.73  29.80
2    1903  23.44  25.03  27.83  31.39  32.91  33.00  31.34  29.98  29.85
3    1904  22.50  24.73  28.21  32.02  32.64  32.07  30.36  30.09  30.04
4    1905  22.00  22.83  26.68  30.01  33.32  33.25  31.44  30.68  30.12
..    ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
112  2013  24.56  26.59  30.62  32.66  34.46  32.44  31.07  30.76  31.04
113  2014  23.83  25.97  28.95  32.74  33.77  34.15  31.85  31.32  30.68
114  2015  24.58  26.89  29.07  31.87  34.09  32.48  31.88  31.52  31.55
115  2016  26.94  29.72  32.62  35.38  35.72  34.03  31.64  31.79  31.66
116  2017  26.45  29.46  31.60  34.95  35.84  33.82  31.88  31.72  32.22

       OCT    NOV    DEC  ANNUAL  JAN-FEB  MAR-MAY  JUN-SEP  OCT-DEC
0    29.97  27.31  24.49   28.96    23.27    31.46    31.27    27.25
1    29.12  26.31  24.04   29.22    25.75    31.76    31.09    26.49
2    29.04  26.08  23.65   28.47    24.24    30.71    30.92    26.26
3    29.20  26.36  23.63   28.49    23.62    30.95    30.66    26.40
4    30.67  27.52  23.82   28.30    22.25    30.00    31.33    26.57
..     ...    ...    ...     ...      ...      ...      ...      ...
112  30.27  27.83  25.37   29.81    25.58    32.58    31.33    27.83
113  30.29  28.05  25.08   29.72    24.90    31.82    32.00    27.81
114  31.04  28.10  25.67   29.90    25.74    31.68    31.87    28.27
115  31.98  30.11  28.01   31.63    28.33    34.57    32.28    30.03
116  32.29  29.60  27.18   31.42    27.95    34.13    32.41    29.69

[117 rows x 18 columns]
```

In [5]:
```
shape=df.shape
print(f'Shape of data:{shape}')
```

```
Shape of data:(117, 18)
```

In [6]:
```
type=df.dtypes
print(f'Datatypes:{type}')
```

```
Datatypes: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
```

In [7]:
```
zeros=(df==0).sum()
print(f'No. of zeros:{zeros}')
```

```
No. of zeros: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 [8]:
```
missing=df.isnull().sum()
print(f'No. of missing values:{missing}')
```

```
No. of missing values: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
```

a. Apply Linear Regression and predict the month-wise temperature

Here JAN month is chosen

In [9]:
```
from sklearn.model_selection import train_test_split
import numpy as np

X = df['YEAR']
y = df['JAN']

X_reshaped = np.array(X).reshape(-1, 1)

X_train, X_test, y_train, y_test = train_test_split(X_reshaped, y, test_size=0.25, random_state=42)

print("X_train")
print(X_train)
print("\nX_test")
print(X_test)

print("\ny_train")
print(y_train)
print("\ny_test")
print(y_test)
```

```
X_train
[[2012]
 [1966]
 [1981]
 [1931]
 [1974]
 [1948]
 [1910]
 [1934]
 [2011]
 [1968]
 [1929]
 [1946]
 [1984]
 [1906]
 [1985]
 [1967]
 [1940]
 [1936]
 [1917]
 [1965]
 [1935]
 [1956]
 [1908]
 [1944]
 [1971]
 [1978]
 [1928]
 [1920]
 [1994]
 [1999]
 [1926]
 [1909]
 [1997]
 [1950]
 [1914]
 [2008]
 [1904]
 [1918]
 [1939]
 [1991]
 [1907]
 [1989]
 [2001]
 [2015]
 [1955]
 [1951]
 [1979]
 [1947]
 [1982]
 [1962]
 [2014]
 [1980]
 [1992]
 [1942]
 [1959]
 [1949]
 [2009]
 [1958]
 [1976]
 [1933]
 [2010]
 [1960]
 [1964]
 [1998]
 [1938]
 [1930]
 [2004]
 [1902]
 [1953]
 [1922]
 [1903]
 [1924]
 [2000]
 [1988]
 [2006]
 [1975]
 [1987]
 [1983]
 [2017]
```

```
 [1921]
 [1961]
 [1972]
 [2007]
 [1915]
 [1993]
 [1952]
 [2003]]

X_test
[[1945]
 [1905]
 [1954]
 [1943]
 [1911]
 [1986]
 [1973]
 [1995]
 [1937]
 [1912]
 [1941]
 [1927]
 [1996]
 [2005]
 [1919]
 [1901]
 [1963]
 [2013]
 [1977]
 [2002]
 [1969]
 [1990]
 [2016]
 [1913]
 [1916]
 [1970]
 [1932]
 [1925]
 [1957]
 [1923]]

y_train
111    23.61
65     24.11
80     23.73
30     24.57
73     23.54
       ...
106    25.19
14     22.60
92     23.82
51     23.91
102    24.27
Name: JAN, Length: 87, dtype: float64

y_test
44     22.38
4      22.00
53     22.79
42     22.97
10     23.22
85     23.61
72     24.02
94     24.44
36     23.34
11     23.70
40     23.18
26     23.23
95     25.18
104    24.18
18     23.32
0      22.40
62     22.90
112    24.56
76     23.98
101    24.56
68     23.78
89     24.24
115    26.94
12     23.71
```

```
15     24.13
69     24.19
31     24.13
24     22.56
56     22.98
22     23.25
Name: JAN, dtype: float64
```

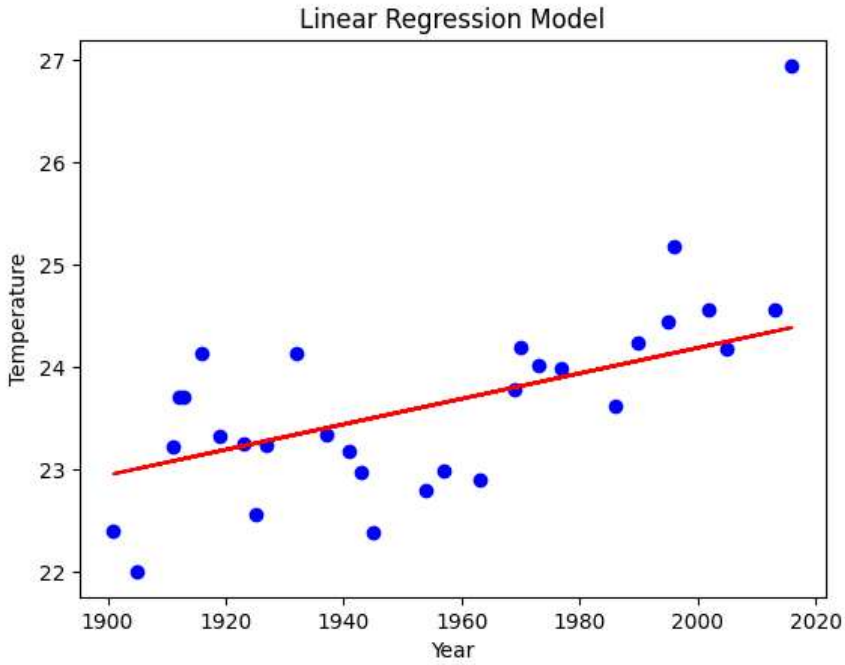
b. Assess the performance of regression models using MSE, MAE and R-Square metrics

In [10]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)

mae=mean_absolute_error(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'R-squared: {r2}')
```

```
MAE: 0.5220295907600104
MSE: 0.5285626287719306
R-squared: 0.41999343079141704
```

c. Visualizing model

In [11]:
```
import matplotlib.pyplot as plt

plt.scatter(X_test,y_test,color="blue",label="Actual values")
plt.plot(X_test,y_pred,color="red",label="Predicted values")
plt.title("Linear Regression Model")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.show()
```



In [ ]: