In [1]:
```
# importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.linear_model import LinearRegression
```

In [2]:
```
# Reading the dataset
data = pd.read_csv("C:\\Users\\hm\\Desktop\\CENTRAL DATASET\\Linear Regression - Sheet1.csv"
```

In [3]:
```
data
```

Out[3]:

	X	Y
0	1	3.888889
1	2	4.555556
2	3	5.222222
3	4	5.888889
4	5	6.555556
...	...	...
295	296	200.555556
296	297	201.222222
297	298	201.888889
298	299	1.888889
299	300	1.888889

300 rows × 2 columns

In [51]:
```
X = np.array(data["X"]).reshape(-1, 1)
Y = np.array(data["Y"])
```

In [52]:
```
X
```

Out[52]:
```
array([[  1],
       [  2],
       [  3],
       [  4],
       [  5],
       [  6],
       [  7],
       [  8],
       [  9],
       [ 10],
       [ 11],
       [ 12],
       [ 13],
       [ 14],
       [ 15],
       [ 16],
       [ 17],
       [ 18],
       [ 19],
```

In [53]:
```
Y
```

```
Out[53]: array([  3.88888889,   4.55555556,   5.22222222,   5.88888889,
                  6.55555556,   7.22222222,   7.88888889,   8.55555556,
                  9.22222222,   9.88888889,  10.55555556,  11.22222222,
                 11.88888889,  12.55555556,  13.22222222,  13.88888889,
                 14.55555556,  15.22222222,  15.88888889,  16.55555556,
                 17.22222222,  17.88888889,  18.55555556,  19.22222222,
                 19.88888889,  20.55555556,  21.22222222,  21.88888889,
                 22.55555556,  23.22222222,  23.88888889,  24.55555556,
                 25.22222222,  25.88888889,  26.55555556,  27.22222222,
                 27.88888889,  28.55555556,  29.22222222,  29.88888889,
                 30.55555556,  31.22222222,  31.88888889,  32.55555556,
                 33.22222222,  33.88888889,  34.55555556,  35.22222222,
                 35.88888889,  36.55555556,  37.22222222,  37.88888889,
                 38.55555556,  39.22222222,  39.88888889,  40.55555556,
                 41.22222222,  41.88888889,  42.55555556,  43.22222222,
                 43.88888889,  44.55555556,  45.22222222,  45.88888889,
                 46.55555556,  47.22222222,  47.88888889,  48.55555556,
                 49.22222222,  49.88888889,  50.55555556,  51.22222222,
                 51.88888889,  52.55555556,  53.22222222,  53.88888889,
                 54.55555556,  55.22222222,  55.88888889,  56.55555556,
                 57.22222222,  57.88888889,  58.55555556,  59.22222222,
                 59.88888889,  60.55555556,  61.22222222,  61.88888889,
                 62.55555556,  63.22222222,  63.88888889,  64.55555556,
                 65.22222222,  65.88888889,  66.55555556,  67.22222222,
                 67.88888889,  68.55555556,  69.22222222,  69.88888889,
                 70.55555556,  71.22222222,  71.88888889,  72.55555556,
                 73.22222222,  73.88888889,  74.55555556,  75.22222222,
                 75.88888889,  76.55555556,  77.22222222,  77.88888889,
                 78.55555556,  79.22222222,  79.88888889,  80.55555556,
                 81.22222222,  81.88888889,  82.55555556,  83.22222222,
                 83.88888889,  84.55555556,  85.22222222,  85.88888889,
                 86.55555556,  87.22222222,  87.88888889,  88.55555556,
                 89.22222222,  89.88888889,  90.55555556,  91.22222222,
                 91.88888889,  92.55555556,  93.22222222,  93.88888889,
                 94.55555556,  95.22222222,  95.88888889,  96.55555556,
                 97.22222222,  97.88888889,  98.55555556,  99.22222222,
                 99.88888889, 100.5555556 , 101.2222222 , 101.8888889 ,
                102.5555556 , 103.2222222 , 103.8888889 , 104.5555556 ,
                105.2222222 , 105.8888889 , 106.5555556 , 107.2222222 ,
                107.8888889 , 108.5555556 , 109.2222222 , 109.8888889 ,
                110.5555556 , 111.2222222 , 111.8888889 , 112.5555556 ,
                113.2222222 , 113.8888889 , 114.5555556 , 115.2222222 ,
                115.8888889 , 116.5555556 , 117.2222222 , 117.8888889 ,
                118.5555556 , 119.2222222 , 119.8888889 , 120.5555556 ,
                121.2222222 , 121.8888889 , 122.5555556 , 123.2222222 ,
                123.8888889 , 124.5555556 , 125.2222222 , 125.8888889 ,
                126.5555556 , 127.2222222 , 127.8888889 , 128.5555556 ,
                129.2222222 , 129.8888889 , 130.5555556 , 131.2222222 ,
                131.8888889 , 132.5555556 , 133.2222222 , 133.8888889 ,
                134.5555556 , 135.2222222 , 135.8888889 , 136.5555556 ,
                137.2222222 , 137.8888889 , 138.5555556 , 139.2222222 ,
                139.8888889 , 140.5555556 , 141.2222222 , 141.8888889 ,
                142.5555556 , 143.2222222 , 143.8888889 , 144.5555556 ,
                145.2222222 , 145.8888889 , 146.5555556 , 147.2222222 ,
                147.8888889 , 148.5555556 , 149.2222222 , 149.8888889 ,
                150.5555556 , 151.2222222 , 151.8888889 , 152.5555556 ,
                153.2222222 , 153.8888889 , 154.5555556 , 155.2222222 ,
                155.8888889 , 156.5555556 , 157.2222222 , 157.8888889 ,
                158.5555556 , 159.2222222 , 159.8888889 , 160.5555556 ,
                161.2222222 , 161.8888889 , 162.5555556 , 163.2222222 ,
                163.8888889 , 164.5555556 , 165.2222222 , 165.8888889 ,
                166.5555556 , 167.2222222 , 167.8888889 , 168.5555556 ,
                169.2222222 , 169.8888889 , 170.5555556 , 171.2222222 ,
                171.8888889 , 172.5555556 , 173.2222222 , 173.8888889 ,
```

```
       174.5555556 , 175.2222222 , 175.8888889 , 176.5555556 ,
       177.2222222 , 177.8888889 , 178.5555556 , 179.2222222 ,
       179.8888889 , 180.5555556 , 181.2222222 , 181.8888889 ,
       182.5555556 , 183.2222222 , 183.8888889 , 184.5555556 ,
       185.2222222 , 185.8888889 , 186.5555556 , 187.2222222 ,
       187.8888889 , 188.5555556 , 189.2222222 , 189.8888889 ,
       190.5555556 , 191.2222222 , 191.8888889 , 192.5555556 ,
       193.2222222 , 193.8888889 , 194.5555556 , 195.2222222 ,
       195.8888889 , 196.5555556 , 197.2222222 , 197.8888889 ,
       198.5555556 , 199.2222222 , 199.8888889 , 200.5555556 ,
       201.2222222 , 201.8888889 ,   1.88888889,   1.88888889])
```

In [54]:

```
data.isnull().sum()
```

Out[54]:

```
X    0
Y    0
dtype: int64
```

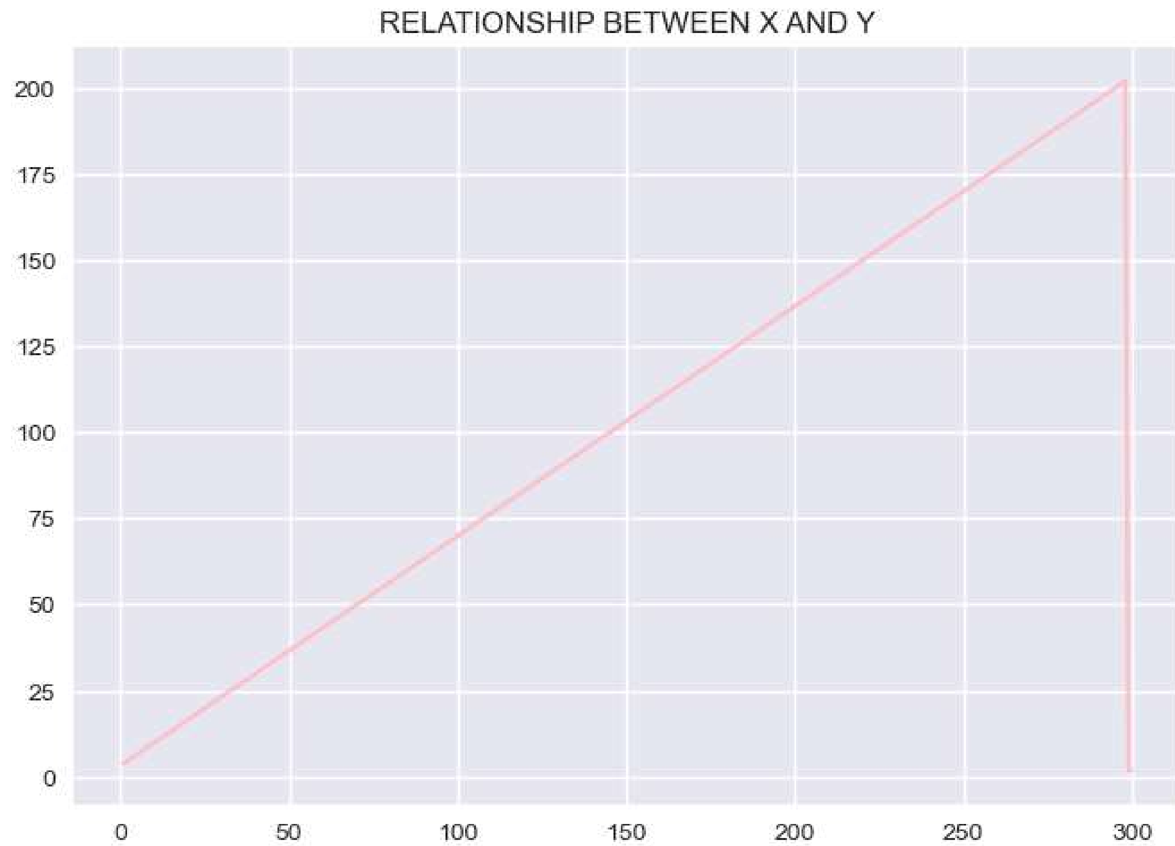
In [55]:

```
# visualizing the relationship between X and Y
import matplotlib.pyplot as plt
from matplotlib import style
```

In [56]:
```
style.use("seaborn")
plt.plot(X,Y, label= "datapoints", color="pink")
plt.title("RELATIONSHIP BETWEEN X AND Y")
```

```
C:\Users\hm\AppData\Local\Temp\ipykernel_9408\2974903442.py:1: MatplotlibDeprecationWarnin
g: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer cor
respond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v
0_8-<style>'. Alternatively, directly use the seaborn API instead.
  style.use("seaborn")
```

Out[56]:
```
Text(0.5, 1.0, 'RELATIONSHIP BETWEEN X AND Y')
```



In [57]:
```
# spliting the data into trainig and testing
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, random_state =43)
```

In [58]:
```
# BUILDING THE LINEAR REGRESSION MODEL
model = LinearRegression()
model.fit(X_train, Y_train)
```

Out[58]:
```
▾ LinearRegression
LinearRegression()
```

In [59]:
```
Y_pred = model.predict(X_test)
```

In [60]:
```
Y_pred
```

Out[60]:
```
array([183.12268535,  49.7306632 ,  80.31322437, 172.0609079 ,
        53.63481994, 125.21102695, 140.17696114,  72.50491088,
       162.30051604,  52.33343436, 102.43677927,  41.27165691,
       107.64232159,  15.24394527, 103.08747206, 192.23238442,
       137.57418998,  82.91599554,  32.16195784,  34.764729  ,
        30.20987947,  47.77858482, 111.54647834, 149.28666022,
        24.35364435,  56.23759111,  25.00433714,  58.84036227,
       199.39000512,  50.38135599, 101.13539368, 159.04705208,
       175.31437186,  49.07997041, 109.59439997, 148.63596742,
        88.77223066, 168.80744395, 129.1151837 , 128.4644909 ,
       116.10132788, 150.5880458 , 129.76587649,   7.43563178,
        28.25780109,  73.80629646,  84.21738112,  71.85421809,
        77.05976042,  54.93620552, 118.05340625,  43.87442808,
        12.64117411,  47.12789203, 180.51991418,  92.02569461,
        97.88192973, 195.48584838, 121.95756299,  65.99798297])
```

In [61]:
```
# Evaluating the model
r2 = r2_score(Y_test, Y_pred)
MAE = mean_absolute_error(Y_test,Y_pred)
MSE = mean_squared_error(Y_test, Y_pred)
```

In [62]:
```
r2
```

Out[62]:
```
0.7711925561576223
```

In [63]:
```
MAE
```

Out[63]:
```
4.397492003518091
```

In [64]:
```
MSE
```

Out[64]:
```
651.9849469663601
```

MODEL OPTIMAZITION

In [65]:
```
# importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.linear_model import LinearRegression
```

In [66]:
```
# MODEL OPTIMIZATION

model = LinearRegression()
```

In [67]:
```
# using parameter
param_grid= {
    "fit_intercept": [True,False ],

    "positive": [True,False]

}
```

```
grid_search = GridSearchCV(model,param_grid, cv=5, scoring="neg_mean_squared_error"(
```

In [69]:
```
grid_search = GridSearchCV(model, param_grid, cv=5, scoring="neg_mean_squared_error")

grid_search.fit(X_train, Y_train)
```

Out[69]:
```
GridSearchCV
estimator: LinearRegression
LinearRegression
```

In [74]:
```
best_param = grid_search.best_params_
# evaluate the model from the test data
best_model = LinearRegression(**best_param)
best_model.fit(X_train, Y_train)
Y_pred = best_model.predict(X_test)
r2 = r2_score(Y_test, Y_pred)
print("best parameters:",best_param)
print("r2:",r2)
```

```
best parameters: {'fit_intercept': True, 'positive': True}
r2: 0.7711925561576224
```

In [ ]: