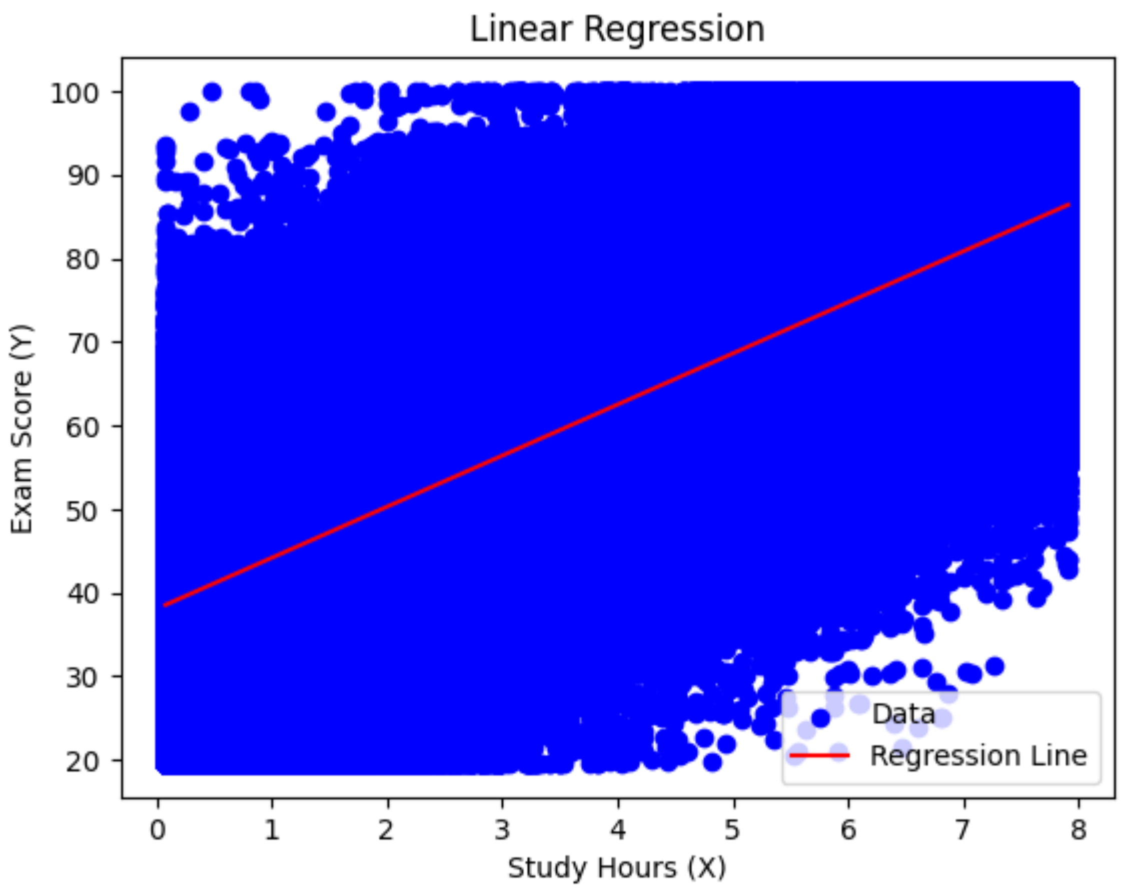
In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score
import random
from sklearn.linear_model import LinearRegression
```

In [3]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
id_1 = train["id"]
id_t = test["id"]
Xt = test["study_hours"]
```

In [4]:
```
X = train["study_hours"].values.reshape(-1, 1)
Y = train["exam_score"]

model = LinearRegression()
model.fit(X, Y)

Y_pred_1 = model.predict(X)

sorted_idx = np.argsort(X.flatten())
X_sorted = X.flatten()[sorted_idx]
Y_pred_sorted = Y_pred_1[sorted_idx]

plt.scatter(X, Y, color='blue', label='Data')
plt.plot(X_sorted, Y_pred_sorted, color='red', label='Regression Line')

plt.title("Linear Regression")
plt.xlabel("Study Hours (X)")
plt.ylabel("Exam Score (Y)")
plt.legend()
plt.show()
```

```
/home/avveer-singh-khurana/anscombe-project/venv/lib/python3.12/site-packa
ges/IPython/core/pylabtools.py:170: UserWarning: Creating legend with loc=
"best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [5]:
```
pred_df = np.column_stack((X.flatten(), Y_pred_1))
```

In [6]:
```
def predict_grade(study_hours):
    study_hours = np.array(study_hours).reshape(-1, 1)
    y_pred = model.predict(study_hours)
    y_pred = np.round(y_pred, 1)
    return float(y_pred[0])

Y_pred_2 = []
for i in range(len(id_t)):
    j = Xt[i]
    Y_pred_2.append(predict_grade(j))
```

In [7]:
```
print(Y_pred_2[:20])
```

```
[79.9, 78.4, 78.4, 56.6, 50.5, 76.1, 60.0, 57.2, 77.7, 85.2, 49.2, 47.9, 4
0.7, 65.8, 40.4, 78.3, 73.1, 64.8, 73.4, 66.5]
```

In [10]:
```
for i in range(len(id_t)):
    if Xt[i] <= 2:
        r3 = random.randint(0,1)
        if r3 == 1:
            r1 = random.uniform(0.1, 40)
            yv = Y_pred_2[i] + r1
            Y_pred_2[i] = yv
        else:
            r1 = random.uniform(0.1, 15)
            yv = Y_pred_2[i] - r1
            Y_pred_2[i] = yv
```

In [12]:
```
for i in range(len(id_t)):
    if 2 < Xt[i] <= 5.5:
        r3 = random.randint(0,1)
        if r3 == 1:
            r1 = random.uniform(0.1, 35)
            yv = Y_pred_2[i] + r1
            Y_pred_2[i] = yv
        else:
            r1 = random.uniform(0.1, 27)
            yv = Y_pred_2[i] - r1
            Y_pred_2[i] = yv
```

In [16]:
```
for i in range(len(id_t)):
    if 5.5 < Xt[i]:
        r3 = random.randint(0,1)
        if r3 == 1:
            r1 = random.uniform(0.1, (100 - Y_pred_2[i]))
            yv = Y_pred_2[i] + r1
            Y_pred_2[i] = yv
        else:
            r1 = random.uniform(0.1, 40)
            yv = Y_pred_2[i] - r1
            Y_pred_2[i] = yv

Y_pred_2 = np.round(Y_pred_2, 1)
```

In [17]:
```
print(Y_pred_2[:20])
```

```
[86.6 22.4 93.4 35.8 90.  93.1 65.7 65.9  7.6 51.2 29.8 42.8 62.5 40.4
 16.7 48.4 78.1 69.4 37.  35.4]
```

In [23]:
```
locking_in_submission = pd.DataFrame({
    'id': id_t,
    'exam_score': Y_pred_2
})

locking_in_submission.to_csv('locking_in_submission.csv', index=False)
locking_in_submission.head
```

Out[23]:
```
<bound method NDFrame.head of             id  exam_score
0       630000        86.6
1       630001        22.4
2       630002        93.4
3       630003        35.8
4       630004        90.0
...        ...         ...
269995  899995        54.1
269996  899996        37.1
269997  899997        69.0
269998  899998        47.4
269999  899999        86.0

[270000 rows x 2 columns]>
```