In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df = pd.read_csv('height-weight.csv')
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Weight  23 non-null     int64
 1   Height  23 non-null     int64
dtypes: int64(2)
memory usage: 496.0 bytes
```

In [4]:

```
df.head()
```

Out[4]:

	Weight	Height
0	45	120
1	58	135
2	48	123
3	60	145
4	70	160

In [7]:

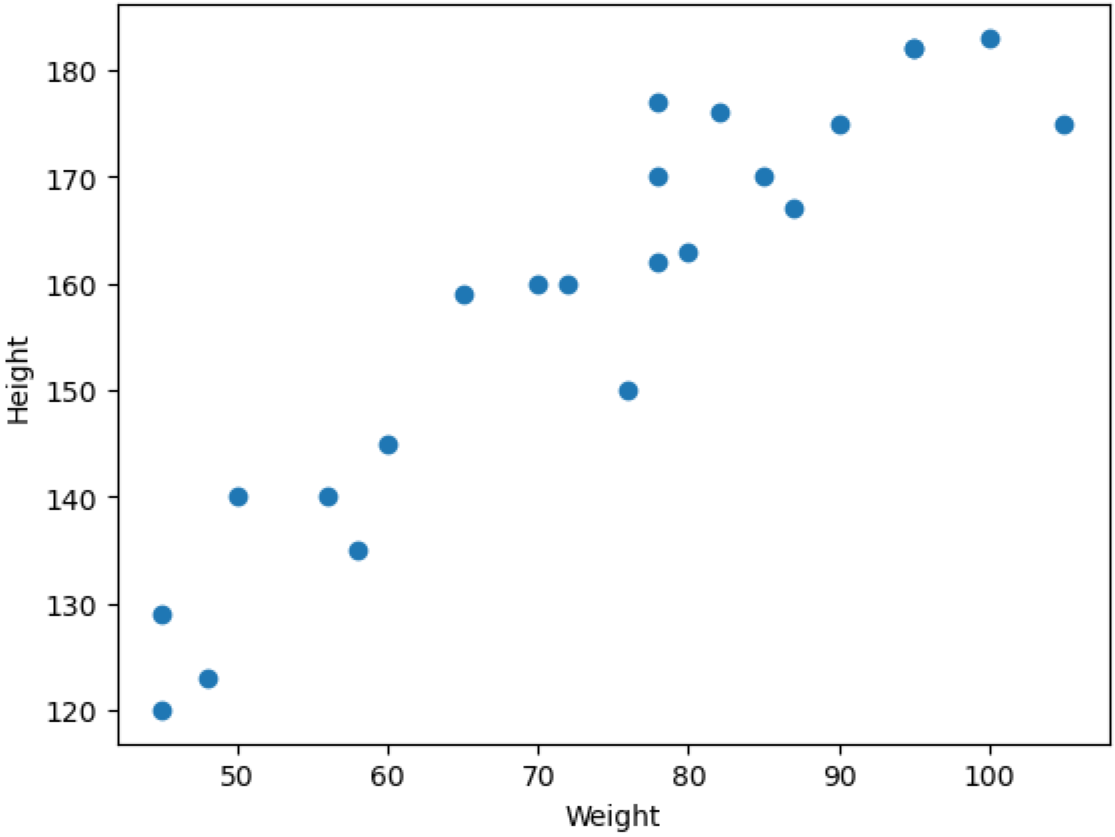
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [6]:

```
plt.scatter(df.Weight,df.Height)#the data is following a linear regression
plt.xlabel('Weight')
plt.ylabel('Height')
```

Out[6]:

```
Text(0, 0.5, 'Height')
```



In [8]:

```
df.corr()#weight and height are positively and strongly correlated
```

Out[8]:

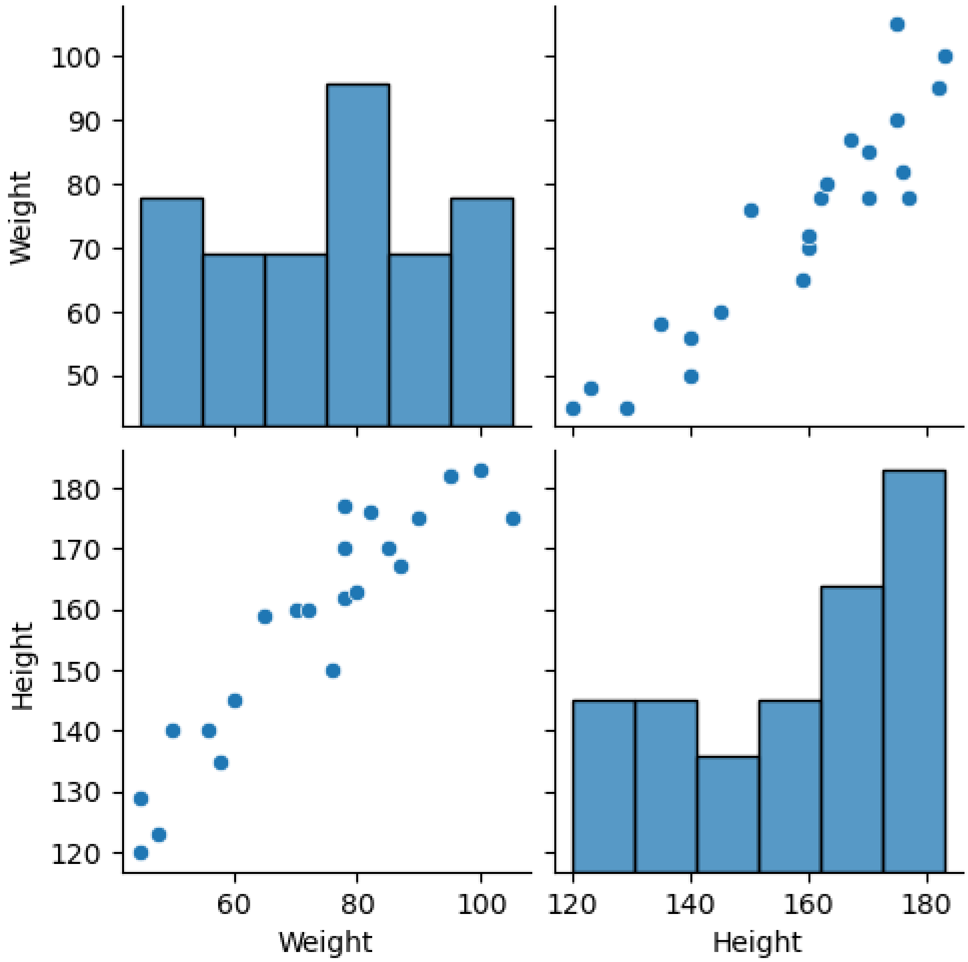
	Weight	Height
Weight	1.000000	0.931142
Height	0.931142	1.000000

In [12]:

```
sns.pairplot(df)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x7fa32ac54fa0>
```



In [13]:
```
X = df[['Weight']]#assigning independant variable to X
#since X should be a dataframe we have used [[]] two brackets
```

In [18]:
```
X.shape
```

Out[18]:
```
(23, 1)
```

In [19]:
```
Y = df['Height']#assigning dependat variable to Y. Here we have used [] as Y can be a se
```

In [20]:
```
Y.shape
```

Out[20]:
```
(23,)
```

In [22]:
```
from sklearn.model_selection import train_test_split
```

In [23]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size= 0.3, random_state=
```

In [24]:
```
from sklearn.preprocessing import StandardScaler
```

In [26]:
```
scaler = StandardScaler()
```

In [28]:
```
X_train = scaler.fit_transform(X_train)
```

In [29]:
```
X_test = scaler.transform(X_test)#using tranform only instead of fit_transform to avoid
```

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [31]:
```
regression = LinearRegression()
```

In [32]:
```
model = regression.fit(X_train, y_train) #building regression model
```

In [33]:
```
model.coef_
```

Out[33]:
```
array([17.03207732])
```

In [34]:
```
print('Coefficient or Slope:', model.coef_)
```

```
Coefficient or Slope: [17.03207732]
```

In [35]:
```
model.intercept_
```

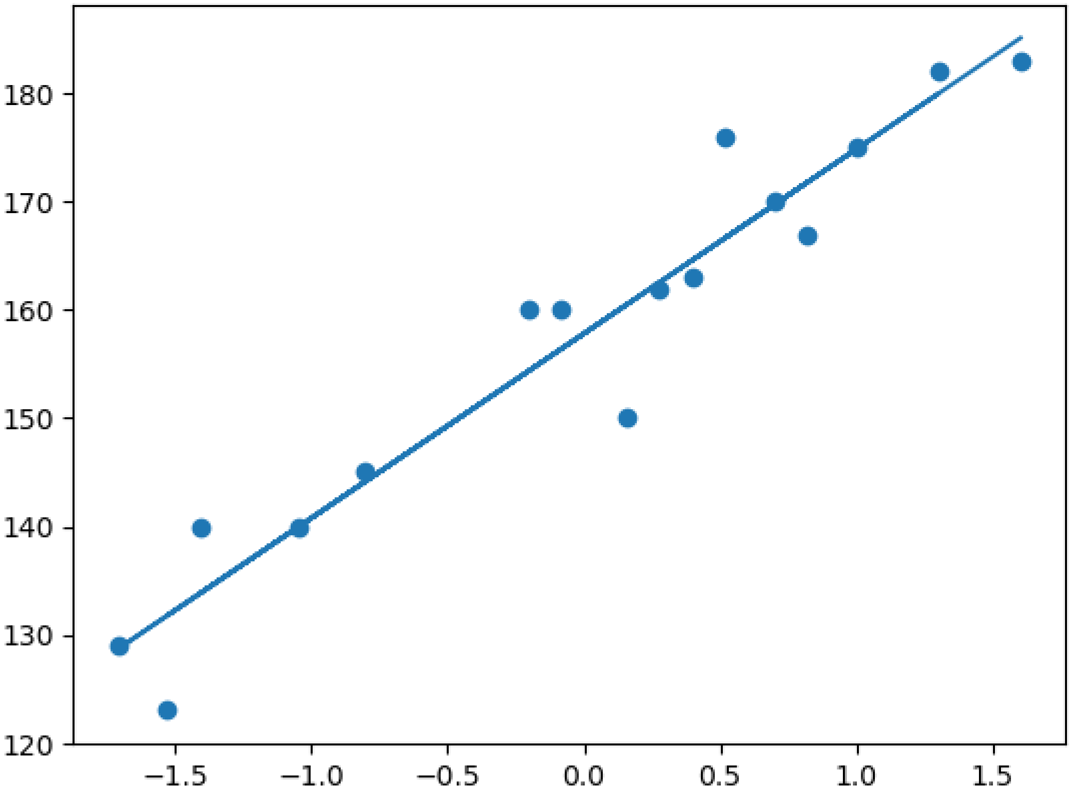
Out[35]:
```
157.8125
```

In [36]:
```
print('Intercept:', model.intercept_)
```

```
Intercept: 157.8125
```

In [37]:
```
plt.scatter(X_train, y_train)
plt.plot(X_train, model.predict(X_train)) #plotting best fit lineb
```

Out[37]:
```
[<matplotlib.lines.Line2D at 0x7fa30ef2f040>]
```



In [38]:
```
#performing prediction
y_pred = model.predict(X_test)
```

In [39]:
```
y_pred
```

Out[39]:
```
array([162.55745791, 162.55745791, 128.70154204, 179.99838426,
       149.22027893, 190.25775271, 142.03872102])
```

In [40]:
```
#finding performance
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [41]:
```
mse = mean_squared_error(y_test, y_pred)
```

In [42]:
```
mse
```
Out[42]:
```
103.09818711844574
```
In [43]:
```
mae = mean_absolute_error(y_test, y_pred)
mae
```
Out[43]:
```
9.237776679921925
```
In [44]:
```
rmse = np.sqrt(mse)
rmse
```
Out[44]:
```
10.153727744944009
```
In [45]:
```
from sklearn.metrics import r2_score
```
In [47]:
```
r2score = r2_score(y_test, y_pred)
r2score #shows the model performs at 78%
```
Out[47]:
```
0.7828485570493535
```
In [50]:
```
#prediction for new data that is predicting height by entering weight
model.predict(scaler.transform([[72]]))#here the height is 156cm if the weight is 72kg
```
```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:465: UserWa
rning: X does not have valid feature names, but StandardScaler was fitted with feature n
ames
  warnings.warn(
```
Out[50]:
```
array([156.40183684])
```
In [51]:
```
#performing regression via OLS
import statsmodels.api as sm
```
In [52]:
```
olsmodel = sm.OLS(y_train, X_train).fit()
```
In [53]:
```
olsmodel.summary()# the coefficient here is almost same as the coefficient arrvied earli
```
```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py:17
69: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=16
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```
Out[53]:

OLS Regression Results

Dep. Variable:	Height	**R-squared (uncentered):**	0.012
Model:	OLS	**Adj. R-squared (uncentered):**	-0.054
Method:	Least Squares	**F-statistic:**	0.1745
Date:	Sat, 18 Nov 2023	**Prob (F-statistic):**	0.682
Time:	14:06:19	**Log-Likelihood:**	-103.69
No. Observations:	16	**AIC:**	209.4
Df Residuals:	15	**BIC:**	210.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	17.0321	40.767	0.418	0.682	-69.861	103.925

Omnibus:	0.957	**Durbin-Watson:**	0.003
Prob(Omnibus):	0.620	**Jarque-Bera (JB):**	0.320
Skew:	-0.347	**Prob(JB):**	0.852
Kurtosis:	2.998	**Cond. No.**	1.00

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.