LINEAR REGRESSION headbrain.csv (MAHWISH KHALID)(BDA-COURSE)

In [46]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

READ THE DATA

In [47]:
```
# Reading Data
df = pd.read_csv("headbrain.csv")
print(df.shape)
df.head()

```

```
(237, 4)
```

Out[47]:

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	4512	1530
1	1	1	3738	1297
2	1	1	4261	1335
3	1	1	3777	1282
4	1	1	4177	1590

ASSIGNING IDEPENDENT AND DEPENDENT VARIABLE

In [48]:
```
#hence we can say head and brain are linear in relationship so we can use linear regression for the predicti
# Collecting X and Y
X = df["Head Size(cm^3)"].values

Y = df["Brain Weight(grams)"].values

```

FIND MEAN VALUES OF INDEPENDENT AND DEPENDENT VARIABLES

In [49]:
```
#In order to find the value of m and c,
#calculate the mean value of X(independent variable) and Y (dependent variable)
# Mean X and Y
mean_x = np.mean(X)
mean_y = np.mean(Y)

print(mean_x)
print(mean_y)
```

```
3633.9915611814345
1282.873417721519
```

FOR SLOPE AND Y-INTERCEPT

In [50]:
```
# find the value of m and c, or b1 and b0==here m or b1 is the slope wile c  or b0 is the y-intercept


# Total number of values
n = len(X)#it shows  number of rows
print(n)
# Using the formula to calculate m and c so we use for loop over the block of code until a sequence is over
#iterate).
#formula for slope (X-mean_x)*(Y-mean_y)/(X-mean_x)(X-mean_x)
numer = 0
denom = 0
for i in range(n):
    numer += (X[i] - mean_x) * (Y[i] - mean_y)
    denom += (X[i] - mean_x) ** 2
m = numer / denom
c = mean_y - (m * mean_x)

# Print coefficients
print(m, c)
```

```
237
0.26342933948939945 325.57342104944223
```

```
The value of m and c from above will be added to this equation
BrainWeight = c + m * HeadSize
```

Plotting Linear Regression Line

Now that we have the equation of the line. So for each actual value of x, we will find the predicted values of y. Once we get the points we can plot them over and create the Linear Regression Line

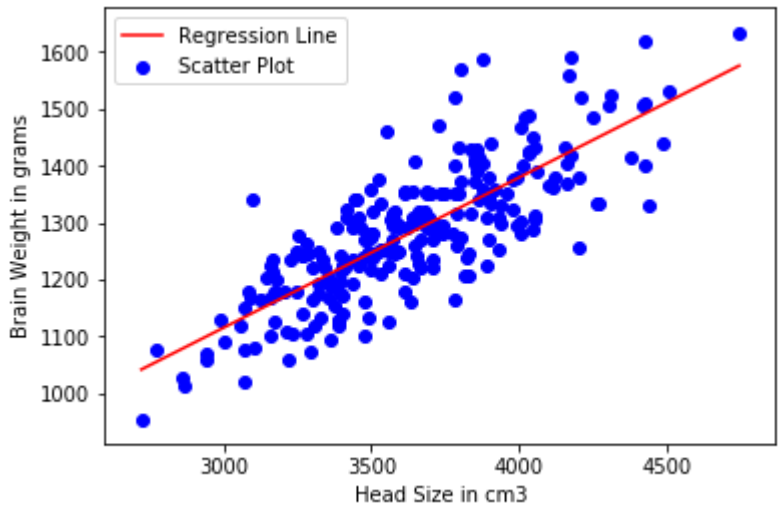
In [51]:
```
# Plotting Values and Regression Line
max_x = np.max(X)
min_x = np.min(X)
print(max_x)
print(min_x)
# Calculating line values x and y
x = np.linspace(start=min_x,stop= max_x,num=1000)

y_pred = c + m * x

# Ploting Line
plt.plot(x, y_pred, color='Red', label='Regression Line')
# Ploting Scatter Points
plt.scatter(X, Y, c='blue', label='Scatter Plot')

plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
```

```
4747
2720
```




```
This is showing you all predicted values of y or we can say it is Yp.when
m = 0.26342933948939945
```

```
c=325.57342104944223
x=given headsize point and we put them in equation  of a line.
```

R Square Method – Goodness of Fit

R–squared value is the statistical measure to show how close the data are to the fitted regression line

```
we know R^2= 1-((predicted value -mean value)^2/(actual value -mean value)^2))
Here,
predicted value =Yp
mean value-mean_y
actual value=Y
```

R-squared does not indicate whether a regression model is adequate. You can have a low R-squared value for a good model, or a high R-squared value for a model that does not fit the data!

R square – Implementation using Python

In [52]:
```
#ss_t is the total sum of squares and ss_r is the total sum of squares of residuals(relate them to the formu
numerator = 0
denomnator = 0
for i in range(n):
    y_pred = c + m * X[i]
    denomnator += (Y[i] - mean_y) ** 2
    numerator += (y_pred - mean_y) ** 2
r2 = (numerator/denomnator)
print(r2)
```

```
0.6393117199570001
```

Linear Regression – Implementation using scikit learn

we have done Linear Regression Algorithm using Least Square Method. Now its time that I tell you about how you can simplify things and implement the same model using a Machine Learning Library called scikit-learn

In [57]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Cannot use Rank 1 matrix in scikit learn
X = X.reshape((n, 1))
# Creating Model
reg = LinearRegression()
# Fitting training data
reg = reg.fit(X, Y)
# Y Prediction
Y_pred = reg.predict(X)

# Calculating R2 Score
r2_score = reg.score(X, Y)


print(r2_score)
```

```
0.639311719957
```

In [ ]:
```

```