In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import linear_model
%matplotlib inline
```

In [2]:

```
!wget -O FuelConsumption.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
cloud/IBMDeveloperSkillsNetwork-ML0101EN-Coursera/labs/Data_files/FuelConsumptionCo2.csv
```

```
--2021-03-22 05:11:21--  https://cf-courses-data.s3.us.cloud-object-storag
e.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-Coursera/labs/Data_fil
es/FuelConsumptionCo2.csv
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-co
urses-data.s3.us.cloud-object-storage.appdomain.cloud)... 198.23.119.245
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (c
f-courses-data.s3.us.cloud-object-storage.appdomain.cloud)|198.23.119.245|:
443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 72629 (71K) [text/csv]
Saving to: 'FuelConsumption.csv'

FuelConsumption.csv 100%[===================>]  70.93K  --.-KB/s    in 0.06
s

2021-03-22 05:11:22 (1.09 MB/s) - 'FuelConsumption.csv' saved [72629/72629]
```

In [3]:

```
data = pd.read_csv("FuelConsumption.csv")
data.head()
```

Out[3]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5
1	2014	ACURA	ILX	COMPACT	2.4	4	M6
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6

In [4]:

```
data.describe()
```

Out[4]:

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_I
count	1067.0	1067.000000	1067.000000	1067.000000	1067.00
mean	2014.0	3.346298	5.794752	13.296532	9.47
std	0.0	1.415895	1.797447	4.101253	2.79
min	2014.0	1.000000	3.000000	4.600000	4.90
25%	2014.0	2.000000	4.000000	10.250000	7.50
50%	2014.0	3.400000	6.000000	12.600000	8.80
75%	2014.0	4.300000	8.000000	15.550000	10.85
max	2014.0	8.400000	12.000000	30.200000	20.50

In [5]:

```
#Filter the data set to just "CYLINDERS", "ENGINESIZE, "FUELCONSUMPTION_COMB" and "CO2EMIS
SIONS".

cdf = data[["ENGINESIZE", "CYLINDERS", "FUELCONSUMPTION_COMB", "CO2EMISSIONS"]]

cdf
```

Out[5]:

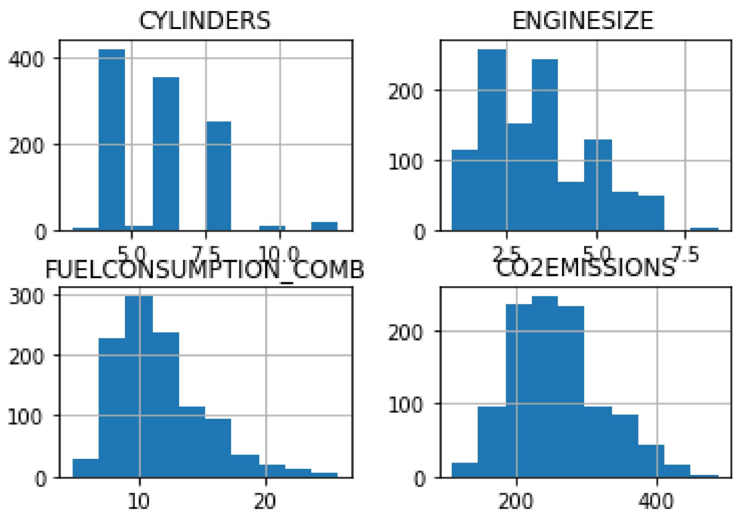
	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
...	...	...	...	...
1062	3.0	6	11.8	271
1063	3.2	6	11.5	264
1064	3.0	6	11.8	271
1065	3.2	6	11.3	260
1066	3.2	6	12.8	294

1067 rows × 4 columns

Plot each of these features saparately:

In [6]:

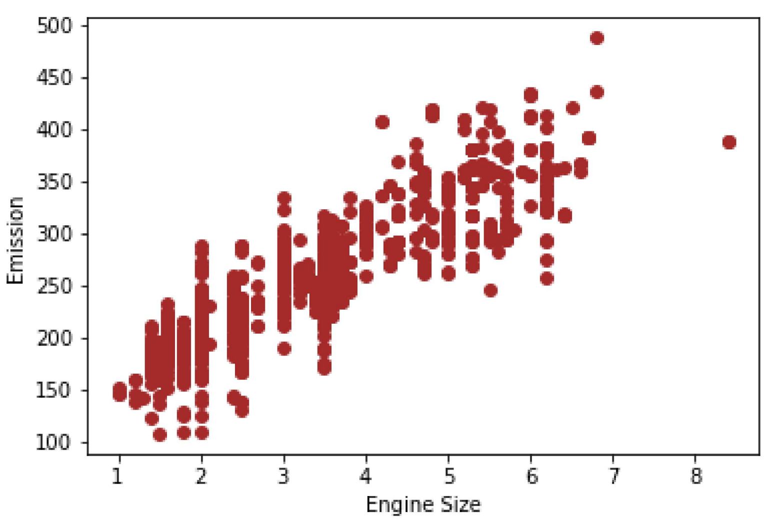
```
view = cdf[["CYLINDERS", "ENGINESIZE", "FUELCONSUMPTION_COMB", "CO2EMISSIONS"]]
view.hist()
plt.show()
```



Now, lets plot each of these features vs the Emission, to see how linear is their relation:

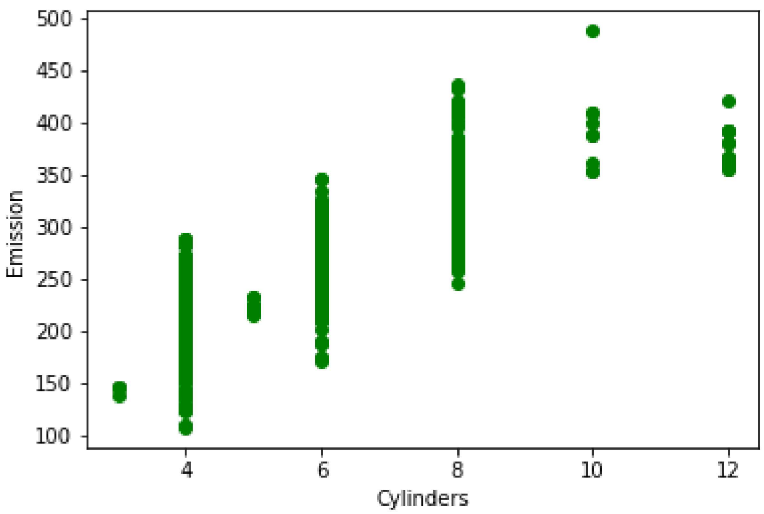
In [7]:

```
#ENGINESIZE vs EMISSION

plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS, color="brown")
plt.xlabel("Engine Size")
plt.ylabel("Emission")
plt.show()
```



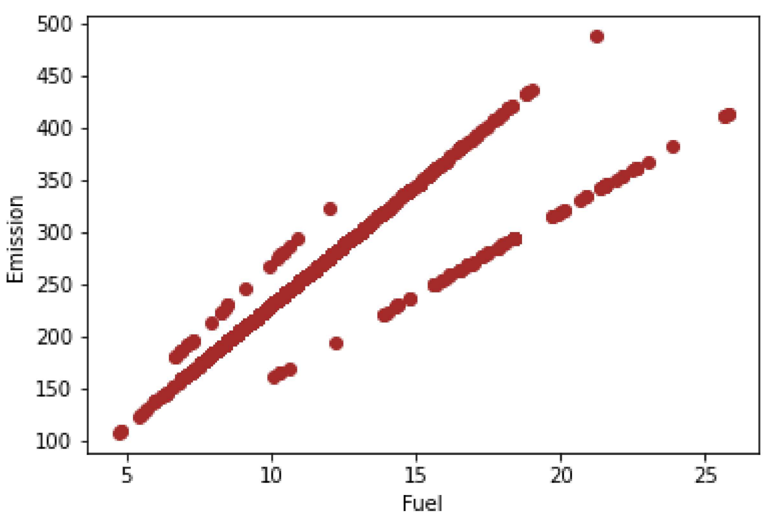
In [8]:

```
#CYLINDERS against CO2EMISSIONS

plt.scatter(cdf.CYLINDERS, cdf.CO2EMISSIONS, color="green")
plt.xlabel("Cylinders")
plt.ylabel("Emission")
plt.show()
```



In [9]:

```
#FuelCONSUMPTION_COMB against CO2EMISSIONS

plt.scatter(cdf.FUELCONSUMPTION_COMB, cdf.CO2EMISSIONS, color="brown")
plt.xlabel("Fuel")
plt.ylabel("Emission")
plt.show()
```



Now, lets plot each of these features vs the Emission, to see how linear is their relation:

Creating train and test dataset

In [10]:

```
msk = np.random.rand(len(cdf)) < 0.8
train = cdf[msk]
test = cdf[~msk]
train.head()
```

Out[10]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232

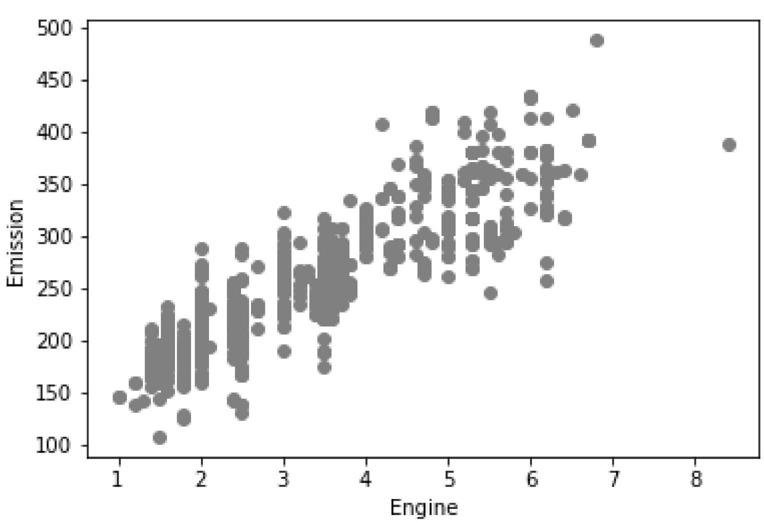
Simple Regression Model

Linear Regression fits a linear model with coefficients $\theta = (\theta_1, ..., \theta_n)$ to minimize the 'residual sum of squares' between the independent x in the dataset, and the dependent y by the linear approximation.

Train data distribution

In [11]:

```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color="grey")
plt.xlabel("Engine")
plt.ylabel("Emission")
plt.show()
```



Modeling

Using sklearn package to model data

In [12]:

```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()

x = train[["ENGINESIZE"]]
y = train[["CO2EMISSIONS"]]

lm.fit(x,y)
```

Out[12]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [13]:

```
regr_x = lm.coef_
regr_x
```

Out[13]:

```
array([[39.03755305]])
```

In [14]:

```
regr_y = lm.intercept_
regr_y
```

Out[14]:

```
array([126.00464656])
```

In [ ]:

```

```

In [15]:

```
Yhat=38.92382721 + 125.41496732*x
Yhat
```

Out[15]:

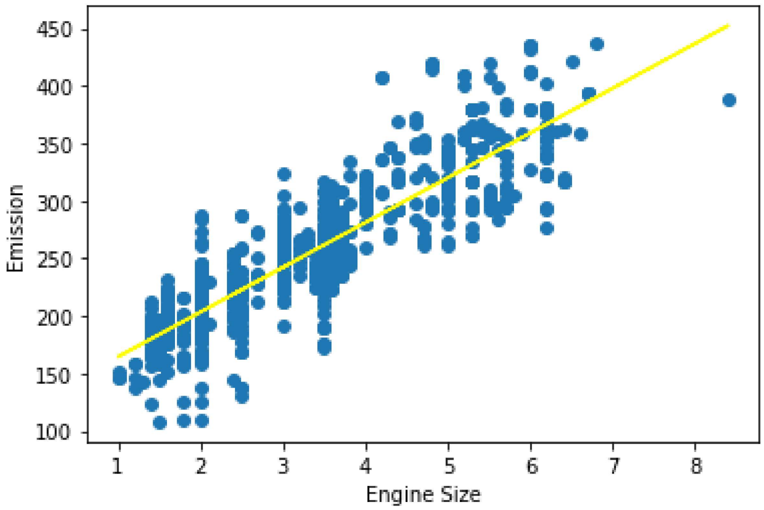
	ENGINESIZE
0	289.753762
1	339.919749
4	477.876213
5	477.876213
6	477.876213
...	...
1060	415.168729
1062	415.168729
1063	440.251723
1064	415.168729
1066	440.251723

839 rows × 1 columns

We can now plot the fit line over the data:

In [51]:

```
#to get the line, use this formulat
y_prediction=lm.predict(x)

plt.scatter(x, y)
plt.plot(x, y_prediction, color="yellow")
plt.xlabel("Engine Size")
plt.ylabel("Emission")
plt.show()
```



In [ ]:

```
#Now we have a linear regression graph that shows that engine size positively effects CO2
 emissions:
#The bigger the engine size, the more CO2 Emission
```

Evaluation

we compare the actual values and predicted values to calculate the accuracy of a regression model. Evaluation metrics provide a key role in the development of a model, as it provides insight to areas that require improvement.

There are different model evaluation metrics, lets use MSE here to calculate the accuracy of our model based on the test set:

- Mean absolute error: It is the mean of the absolute value of the errors. This is the easiest of the metrics to understand since it’s just average error.
- Mean Squared Error (MSE): Mean Squared Error (MSE) is the mean of the squared error. It’s more popular than Mean absolute error because the focus is geared more towards large errors. This is due to the squared term exponentially increasing larger errors in comparison to smaller ones.
- Root Mean Squared Error (RMSE): This is the square root of the Mean Square Error.
- R-squared is not error, but is a popular metric for accuracy of your model. It represents how close the data are to the fitted regression line. The higher the R-squared, the better the model fits your data. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse).

In [17]:

```
lm.fit(x, y)
# Find the R^2
print('The R-square is: ', lm.score(x, y))
```

```
The R-square is:  0.7693586192547599
```