In [ ]:

```
# import lib
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
import statsmodels.api as sm
import seaborn as sns


from pandas import DataFrame
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn import metrics

#FarhanRamdhani
#H071171527
#IlmuKomputer-UH
%matplotlib inline
```

In [2]:

```
# Read data from csv
data = pd.read_csv('co2.csv')

# preview co2 head
data.head()
```

Out[2]:

	MODEL_YEAR	MAKE	MODEL	VEHICLE_CLASS	ENGINE_SIZE	CYLINDERS	TRANSMI
0	2000	ACURA	1.6EL	COMPACT	1.6	4	
1	2000	ACURA	1.6EL	COMPACT	1.6	4	
2	2000	ACURA	3.2TL	MID-SIZE	3.2	6	
3	2000	ACURA	3.5RL	MID-SIZE	3.5	6	
4	2000	ACURA	INTEGRA	SUBCOMPACT	1.8	4	

In [3]:

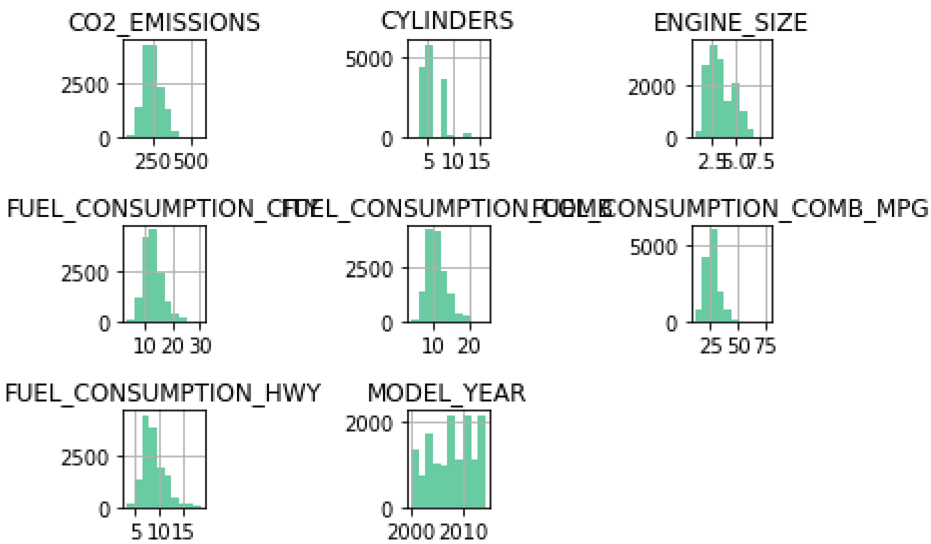
```
# summarize
data.describe()
#df.describe(include = 'all')
```

Out[3]:

	MODEL_YEAR	ENGINE_SIZE	CYLINDERS	FUEL_CONSUMPTION_CITY	FUEL_CONSU
count	14343.000000	14343.000000	14343.000000	14343.000000	
mean	2007.623022	3.474308	5.989054	12.914279	
std	4.140223	1.310875	1.776277	3.549669	
min	2000.000000	0.800000	2.000000	3.500000	
25%	2004.000000	2.400000	4.000000	10.600000	
50%	2008.000000	3.400000	6.000000	12.400000	
75%	2011.000000	4.300000	8.000000	14.900000	
max	2014.000000	8.400000	16.000000	30.600000	

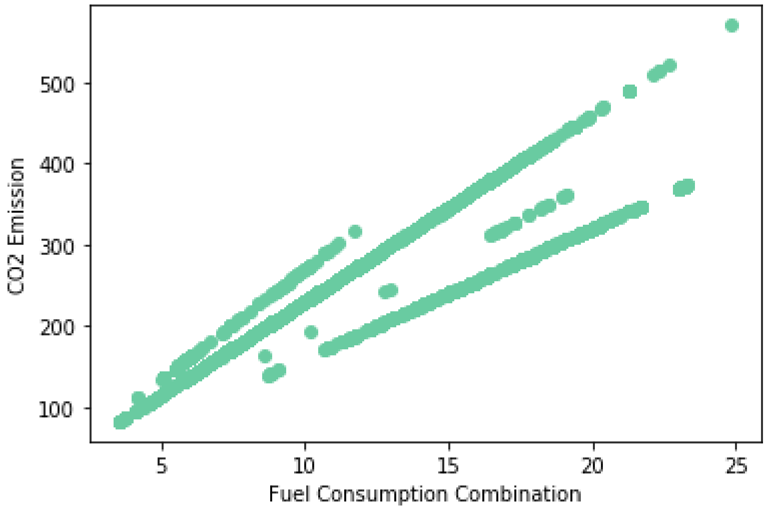
In [4]:

```
data.hist(color = '#69cba1')
plt.tight_layout()
plt.show()
```



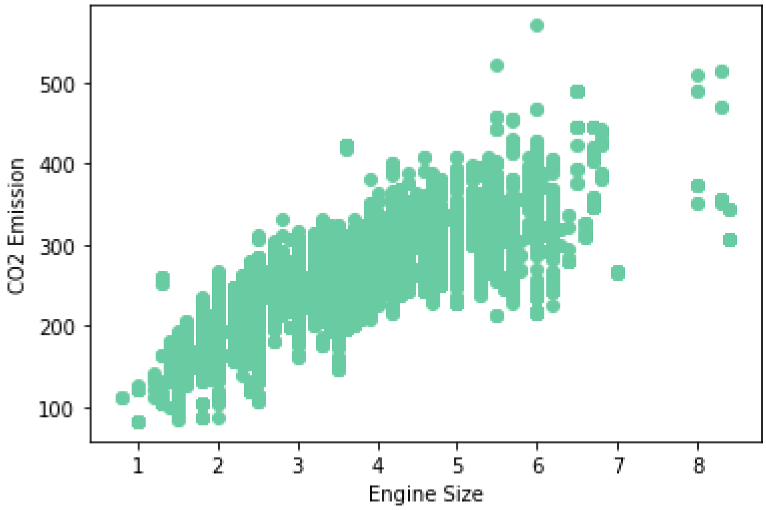
In [5]:

```
plt.scatter(data.FUEL_CONSUMPTION_COMB, data.CO2_EMISSIONS,  color='#69cba1')
plt.xlabel("Fuel Consumption Combination")
plt.ylabel("CO2 Emission")
plt.show()
```



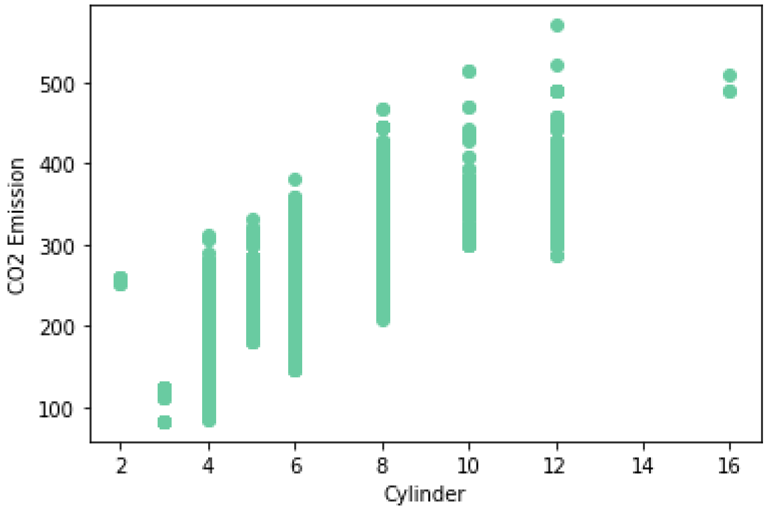
In [6]:

```
plt.scatter(data.ENGINE_SIZE, data.CO2_EMISSIONS,  color='#69cba1')
plt.xlabel("Engine Size")
plt.ylabel("CO2 Emission")
plt.show()
```



In [7]:

```
plt.scatter(data.CYLINDERS, data.CO2_EMISSIONS,  color='#69cba1')
plt.xlabel("Cylinder")
plt.ylabel("CO2 Emission")
plt.show()
```



In [8]:

```
#make dataframe with multiple regg attributes
dataToReg = data[['ENGINE_SIZE','CYLINDERS','FUEL_CONSUMPTION_COMB','CO2_EMISSIONS']]
dataToReg.head()
```

Out[8]:

	ENGINE_SIZE	CYLINDERS	FUEL_CONSUMPTION_COMB	CO2_EMISSIONS
0	1.6	4	8.1	186
1	1.6	4	7.6	175
2	3.2	6	10.0	230
3	3.5	6	11.5	264
4	1.8	4	8.6	198

In [9]:

```
# split data 70% train, 30% test
train, test = train_test_split(dataToReg, test_size=0.3)
print("before split: ",dataToReg.shape,", training data size: ",train.shape, "[70%], te
sting data size: ",test.shape,"[30%]")
```

```
before split:  (14343, 4) , training data size:  (10040, 4) [70%], testing
data size:  (4303, 4) [30%]
```

In [10]:

```
#training x and y from training data
xTraining = train[['ENGINE_SIZE','CYLINDERS', 'FUEL_CONSUMPTION_COMB']]
yTraining = train['CO2_EMISSIONS']

#testing x and y from testing data
xTesting = test[['ENGINE_SIZE','CYLINDERS', 'FUEL_CONSUMPTION_COMB']]
yTesting = test['CO2_EMISSIONS']
```

In [11]:

```
#multiple regression model
multireg = linear_model.LinearRegression()
multireg.fit(xTraining, yTraining) #fit(enginesize, cylinders, fuelconsumption)

print(multireg) #mutlireg model
print('Intercept: \n', multireg.intercept_) #multireg intercept
print('Coefficients: \n', multireg.coef_) #multireg coefficient
print('Variance score: {}'.format(multireg.score(xTesting, yTesting)))  #multireg varia
nce (1 means perfect)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
Intercept: 
 43.1020146866945
Coefficients: 
 [ 5.18800175  5.16507155 14.14927589]
Variance score: 0.8551895387297931
```

In [12]:

```
#create y from x testing with multireg prediction
yPredicted = pd.DataFrame(multireg.predict(xTesting), columns=['CO2_EMISSIONS'])
yPredicted.head()
```

Out[12]:

	CO2_EMISSIONS
0	281.945128
1	249.401794
2	235.016664
3	252.373122
4	278.503125

In [13]:

```
yTesting = pd.DataFrame(yTesting)
yTesting.head()
```

Out[13]:

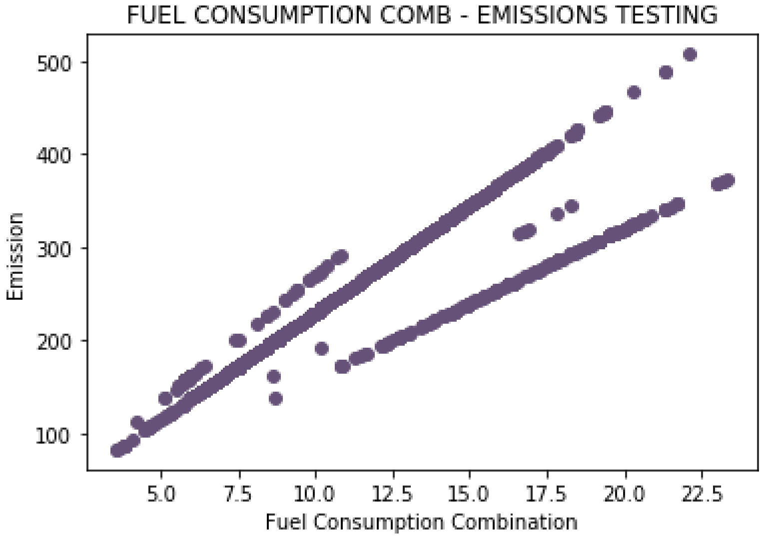
	CO2_EMISSIONS
7101	219
6693	262
2589	235
3882	264
10775	269

In [14]:

```
#plot color
cTrain = '#665178'
cTest = '#A9CDC3'
```

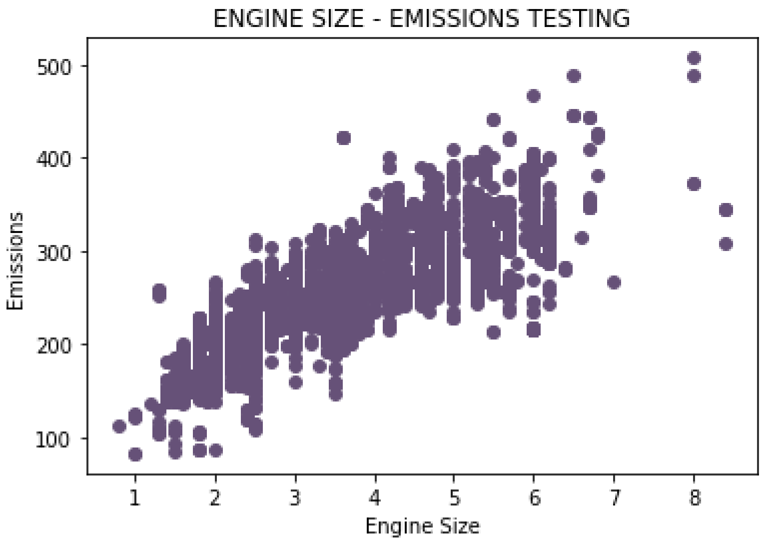
In [15]:

```
plt.scatter(xTesting.FUEL_CONSUMPTION_COMB, yTesting.CO2_EMISSIONS,  color=cTrain)
plt.title("FUEL CONSUMPTION COMB - EMISSIONS TESTING")
plt.xlabel("Fuel Consumption Combination")
plt.ylabel("Emission")
plt.show()
```



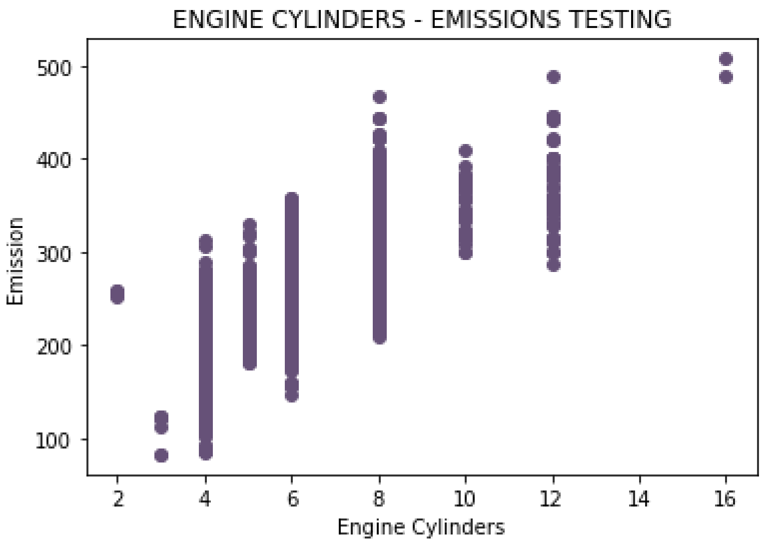
In [16]:

```
plt.scatter(xTesting.ENGINE_SIZE, yTesting.CO2_EMISSIONS,  color=cTrain)
plt.title("ENGINE SIZE - EMISSIONS TESTING")
plt.xlabel("Engine Size")
plt.ylabel("Emissions")
plt.show()
```



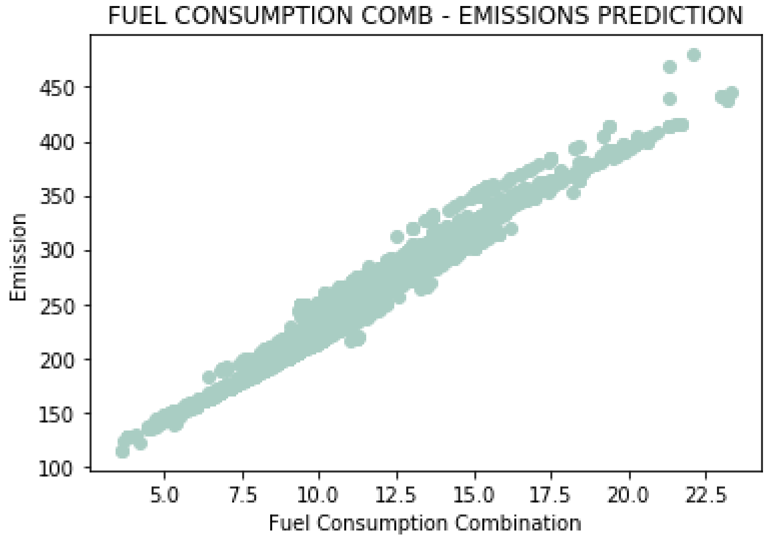
In [17]:

```
plt.scatter(xTesting.CYLINDERS, yTesting.CO2_EMISSIONS,  color=cTrain)
plt.title("ENGINE CYLINDERS - EMISSIONS TESTING")
plt.xlabel("Engine Cylinders")
plt.ylabel("Emission")
plt.show()
```



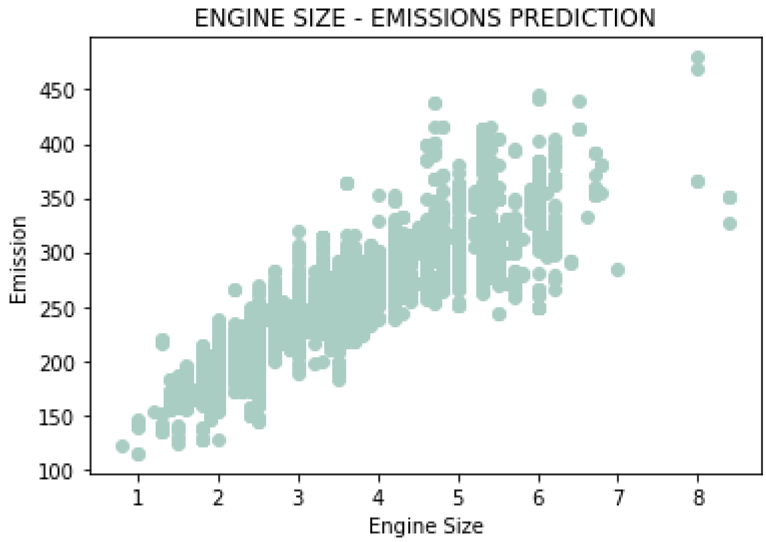
In [18]:

```
plt.scatter(xTesting.FUEL_CONSUMPTION_COMB, yPredicted.CO2_EMISSIONS,  color=cTest)
plt.title("FUEL CONSUMPTION COMB - EMISSIONS PREDICTION")
plt.xlabel("Fuel Consumption Combination")
plt.ylabel("Emission")
plt.show()
```



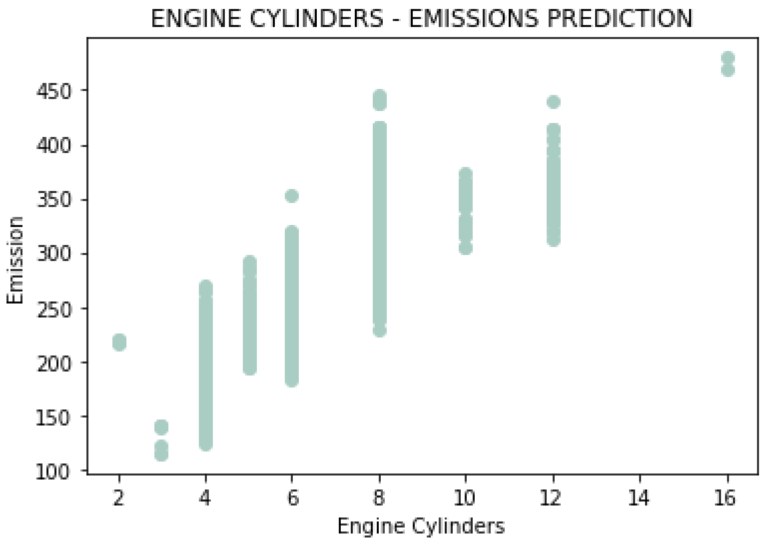
In [19]:

```
plt.scatter(xTesting.ENGINE_SIZE, yPredicted.CO2_EMISSIONS,  color=cTest)
plt.title("ENGINE SIZE - EMISSIONS PREDICTION")
plt.xlabel("Engine Size")
plt.ylabel("Emission")
plt.show()
```



In [20]:

```
plt.scatter(xTesting.CYLINDERS, yPredicted.CO2_EMISSIONS,  color=cTest)
plt.title("ENGINE CYLINDERS - EMISSIONS PREDICTION")
plt.xlabel("Engine Cylinders")
plt.ylabel("Emission")
plt.show()
```



In [21]:

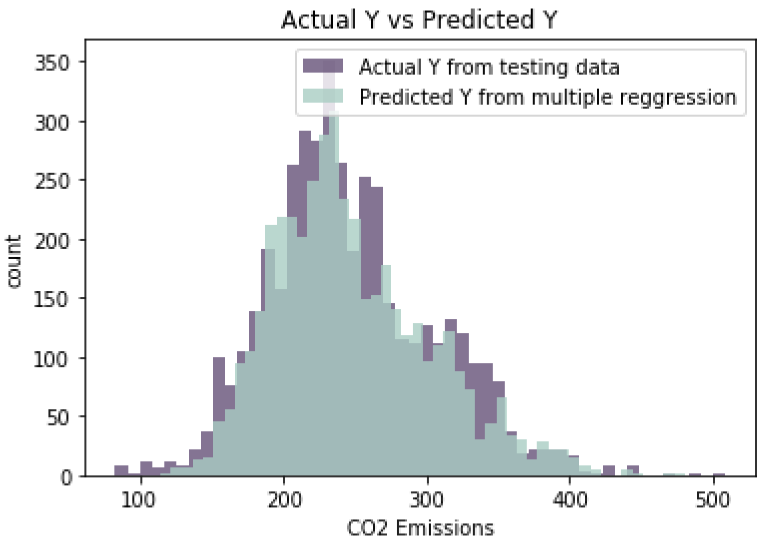
```
comparableData = pd.DataFrame({'Actual_CO2EMISSIONS': yTesting.values.flatten(), 'Predi
cted_CO2EMISSIONS': yPredicted.values.flatten()})
comparableData.describe()
```

Out[21]:

	Actual_CO2EMISSIONS	Predicted_CO2EMISSIONS
count	4303.000000	4303.000000
mean	247.457123	248.069516
std	58.072899	54.803818
min	83.000000	114.722624
25%	209.000000	207.801771
50%	238.000000	238.223846
75%	283.000000	281.639054
max	508.000000	479.946171

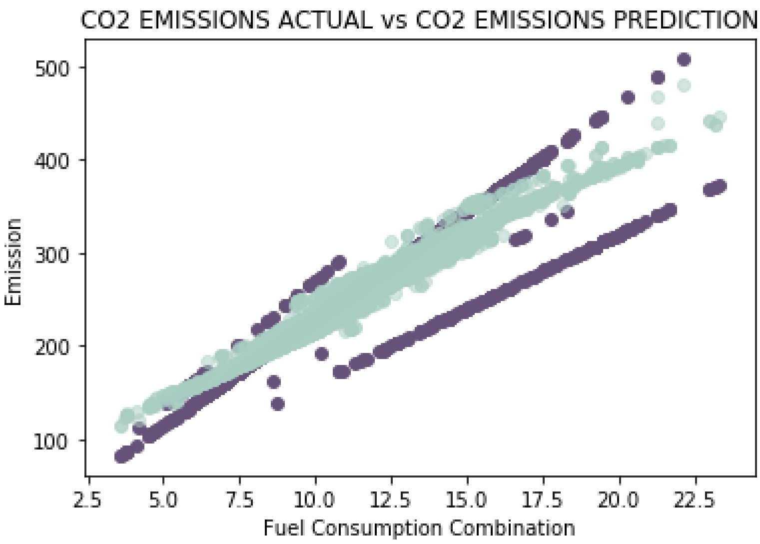
In [22]:

```
plt.hist(comparableData.Actual_CO2EMISSIONS, bins = 50, alpha=0.8, label='Actual Y from
testing data', color = cTrain)
plt.hist(comparableData.Predicted_CO2EMISSIONS, bins = 50, alpha=0.8, label='Predicted
 Y from multiple reggression', color = cTest)
plt.legend(loc='upper right')
plt.title("Actual Y vs Predicted Y")
plt.xlabel("CO2 Emissions")
plt.ylabel("count")
plt.show()
```



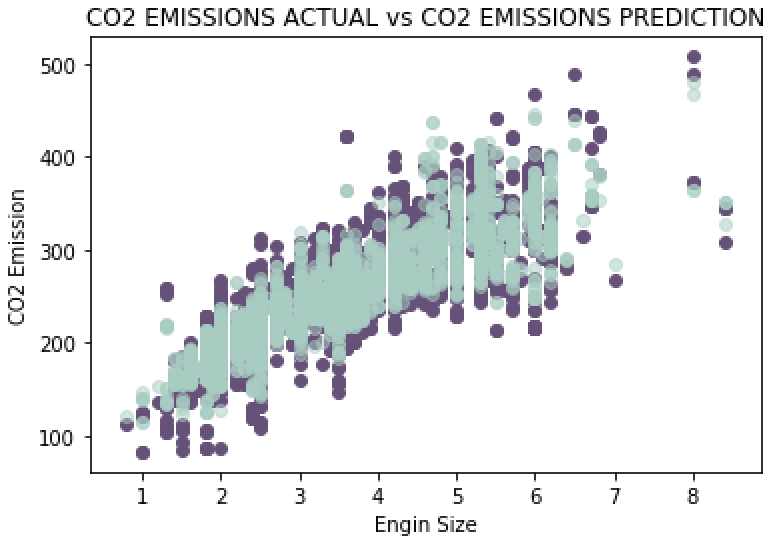
In [23]:

```
plt.scatter(xTesting.FUEL_CONSUMPTION_COMB, comparableData.Actual_CO2EMISSIONS, color=c
Train)
plt.scatter(xTesting.FUEL_CONSUMPTION_COMB, comparableData.Predicted_CO2EMISSIONS, colo
r=cTest, alpha = 0.5)
plt.title("CO2 EMISSIONS ACTUAL vs CO2 EMISSIONS PREDICTION")
plt.xlabel("Fuel Consumption Combination")
plt.ylabel("Emission")
plt.show()
```



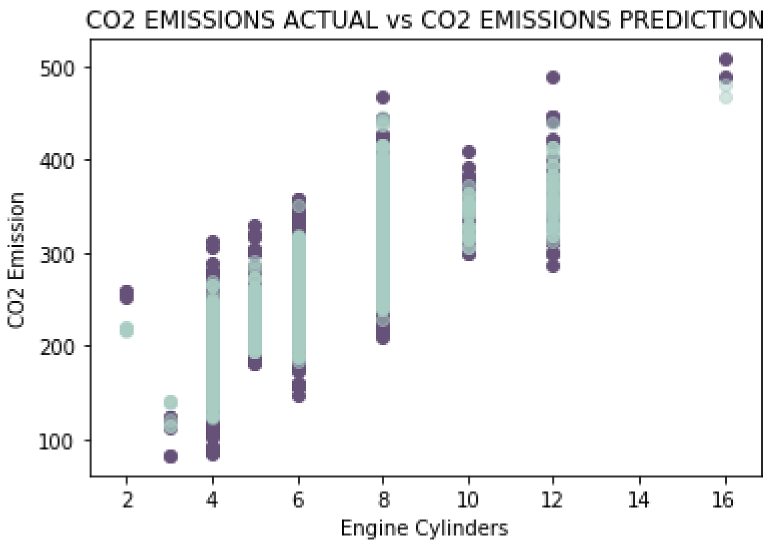
In [24]:

```
plt.scatter(xTesting.ENGINE_SIZE, comparableData.Actual_CO2EMISSIONS, color=cTrain)
plt.scatter(xTesting.ENGINE_SIZE, comparableData.Predicted_CO2EMISSIONS, color=cTest, a
lpha = 0.5)
plt.title("CO2 EMISSIONS ACTUAL vs CO2 EMISSIONS PREDICTION")
plt.xlabel("Engin Size")
plt.ylabel("CO2 Emission")
plt.show()
```



In [25]:

```
plt.scatter(xTesting.CYLINDERS, comparableData.Actual_CO2EMISSIONS, color=cTrain)
plt.scatter(xTesting.CYLINDERS, comparableData.Predicted_CO2EMISSIONS, color=cTest, alp
ha = 0.5)
plt.title("CO2 EMISSIONS ACTUAL vs CO2 EMISSIONS PREDICTION")
plt.xlabel("Engine Cylinders")
plt.ylabel("CO2 Emission")
plt.show()
```

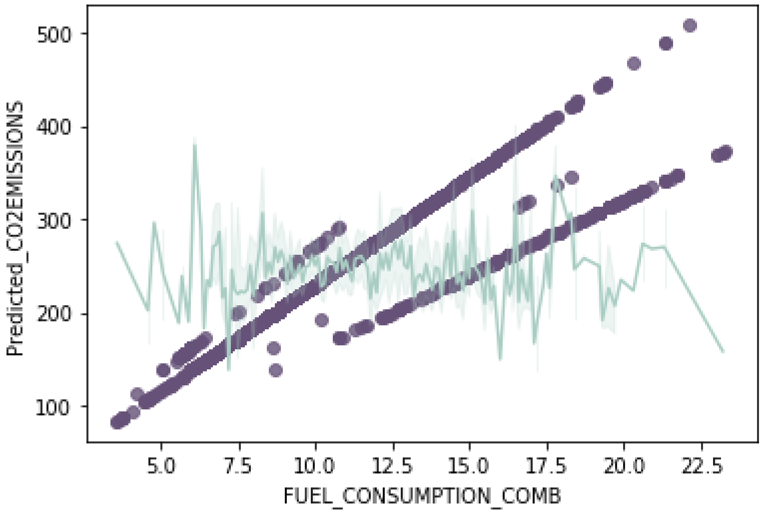


In [26]:

```
sns.regplot(xTesting.FUEL_CONSUMPTION_COMB, comparableData.Actual_CO2EMISSIONS, fit_reg
=False, color = cTrain)
sns.lineplot(xTesting.FUEL_CONSUMPTION_COMB, comparableData.Predicted_CO2EMISSIONS, col
or = cTest)
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ba17c06908>
```

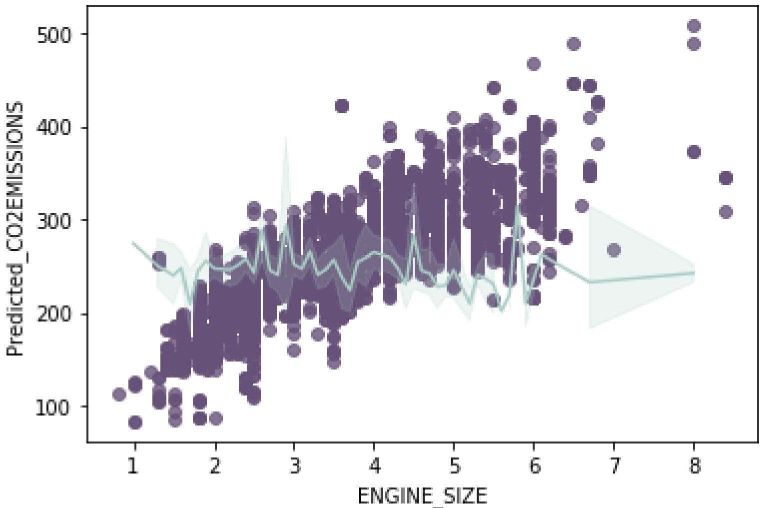


In [27]:

```
sns.regplot(xTesting.ENGINE_SIZE, comparableData.Actual_CO2EMISSIONS, fit_reg=False, co
lor = cTrain)
sns.lineplot(xTesting.ENGINE_SIZE, comparableData.Predicted_CO2EMISSIONS, color = cTest
)
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ba17c08408>
```

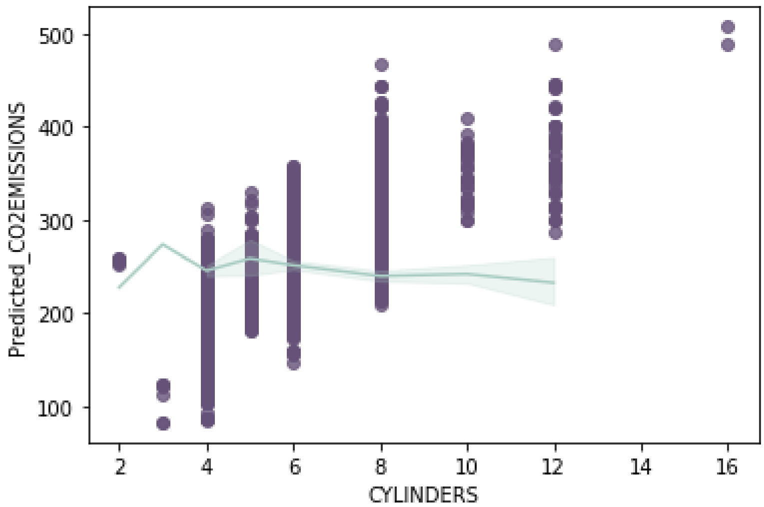


In [28]:

```
sns.regplot(xTesting.CYLINDERS, comparableData.Actual_CO2EMISSIONS, fit_reg=False, colo
r = cTrain)
sns.lineplot(xTesting.CYLINDERS, comparableData.Predicted_CO2EMISSIONS, color = cTest)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ba17b6dd48>
```



In [29]:

```
print('Mean Absolute Error:', metrics.mean_absolute_error(comparableData.Actual_CO2EMIS
SIONS, comparableData.Predicted_CO2EMISSIONS))
print('Mean Squared Error:', metrics.mean_squared_error(comparableData.Actual_CO2EMISSI
ONS, comparableData.Predicted_CO2EMISSIONS))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(comparableData.Act
ual_CO2EMISSIONS, comparableData.Predicted_CO2EMISSIONS)))
```

```
Mean Absolute Error: 14.033897683979582
Mean Squared Error: 488.25422641697577
Root Mean Squared Error: 22.096475429737108
```