In [10]:
```
import pandas as pd

df=pd.read_csv('dataset/fuel_consumption_dataset.csv')

# df.info()
# df.head()
# df.tail()
df.describe()
```

Out[10]:

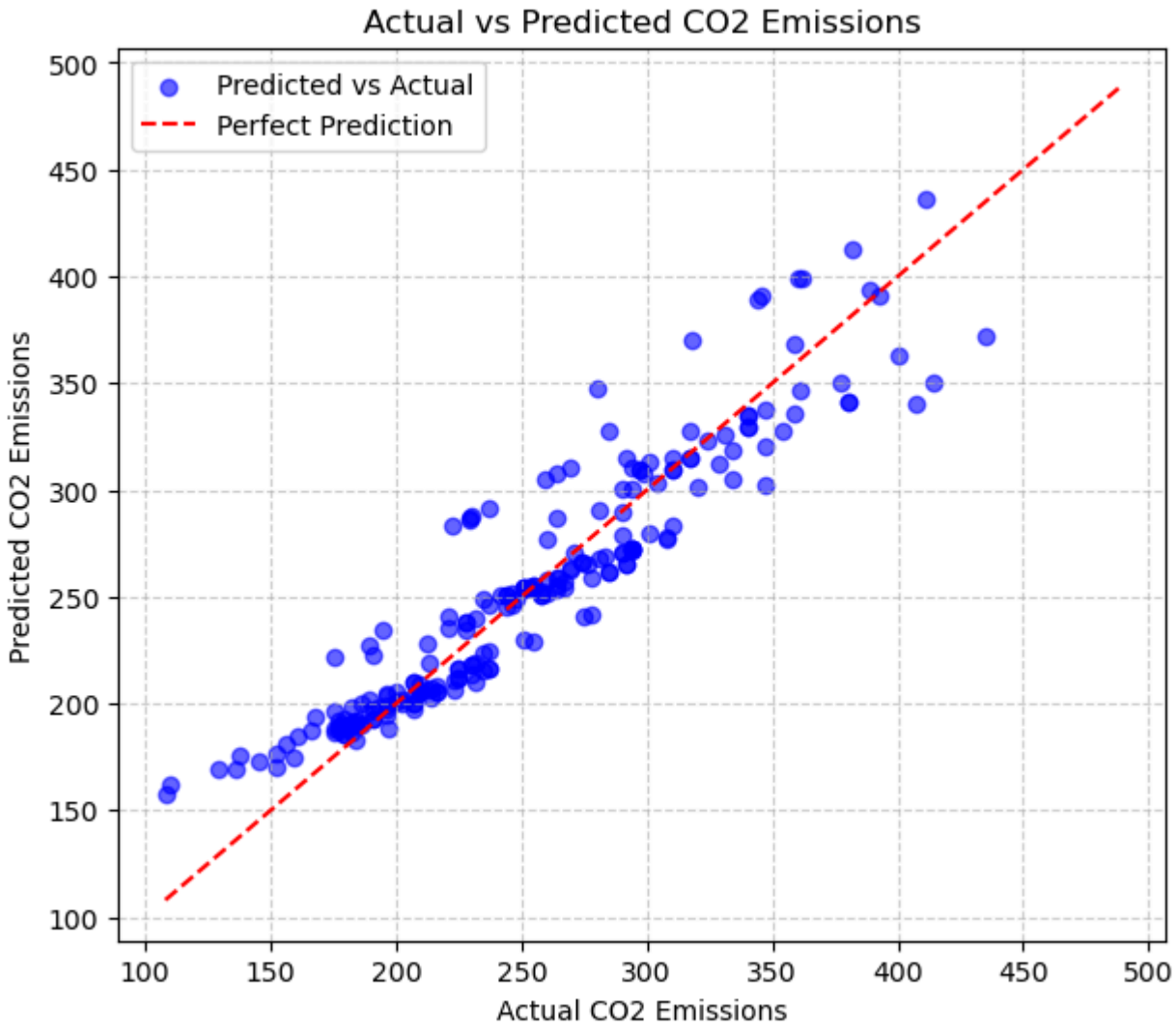
	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTIC
count	1067.0	1067.000000	1067.000000	1067.000000	1067.000000	1067.000000	
mean	2014.0	3.346298	5.794752	13.296532	9.474602	11.580881	
std	0.0	1.415895	1.797447	4.101253	2.794510	3.485595	
min	2014.0	1.000000	3.000000	4.600000	4.900000	4.700000	
25%	2014.0	2.000000	4.000000	10.250000	7.500000	9.000000	
50%	2014.0	3.400000	6.000000	12.600000	8.800000	10.900000	
75%	2014.0	4.300000	8.000000	15.550000	10.850000	13.350000	
max	2014.0	8.400000	12.000000	30.200000	20.500000	25.800000	

In [8]:

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Select features (X) and target (y)
X = df[['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_COMB']]
y = df['CO2EMISSIONS']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Metrics
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
# rmse = np.sqrt(mse)

print(" Model Performance Metrics")
print("R² Score:", r2)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
# print("Root Mean Squared Error (RMSE):", rmse)


# Plot Actual vs Predicted
plt.figure(figsize=(7, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='blue', label="Predicted vs Actual")
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--', label="Perfect Prediction")
plt.xlabel("Actual CO2 Emissions")
plt.ylabel("Predicted CO2 Emissions")
plt.title("Actual vs Predicted CO2 Emissions")
plt.legend()
```

```
plt.grid(True, linestyle="--", alpha=0.6)
plt.show()
```

```
 Model Performance Metrics
R² Score (Accuracy): 0.8759705206914069
Mean Absolute Error (MAE): 16.7215939835165
Mean Squared Error (MSE): 512.8551370148303
```

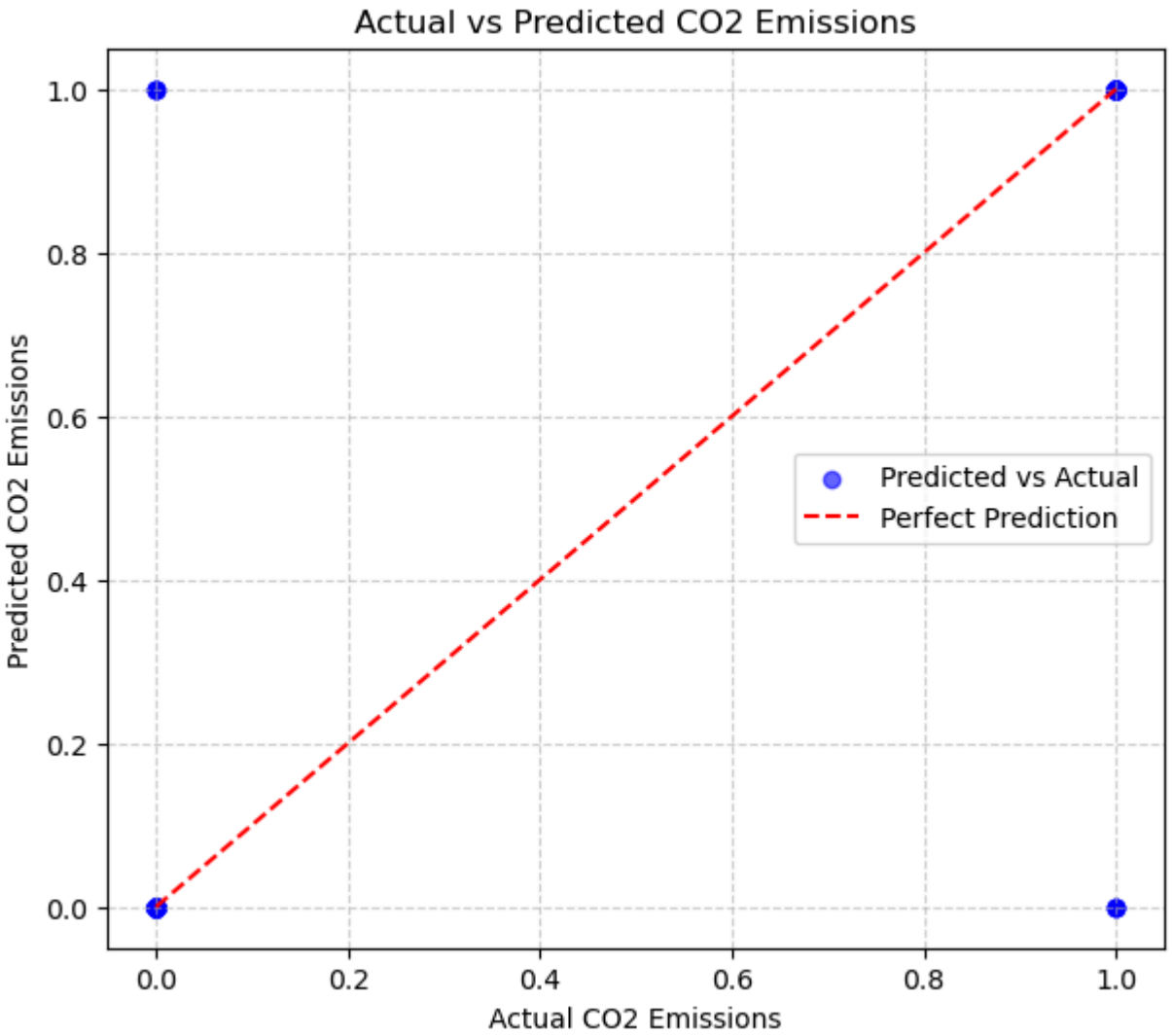


In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

# Load dataset
file_path = "dataset/fuel_consumption_dataset.csv"
df = pd.read_csv(file_path)

# Create binary target: High (1) if CO2 > mean, else Low (0)
threshold = df['CO2EMISSIONS'].mean()
df['HighEmission'] = (df['CO2EMISSIONS'] > threshold).astype(int)

# Features and target
X = df[['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_COMB']]
y = df['HighEmission']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Logistic Regression model
model = LogisticRegression(max_iter=2000)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))

# Detailed classification report
print("\n Classification Report:")
print(classification_report(y_test, y_pred))


# Plot Actual vs Predicted
plt.figure(figsize=(7, 6))
```

```
plt.scatter(y_test, y_pred, alpha=0.6, color='blue', label="Predicted vs Actual")
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--', label="Perfect Prediction")
plt.xlabel("Actual CO2 Emissions")
plt.ylabel("Predicted CO2 Emissions")
plt.title("Actual vs Predicted CO2 Emissions")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.6)
plt.show()
```

```
Accuracy: 0.9532710280373832
Precision: 0.9473684210526315
Recall: 0.9473684210526315
F1 Score: 0.9473684210526315

 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       119
           1       0.95      0.95      0.95        95

    accuracy                           0.95       214
   macro avg       0.95      0.95      0.95       214
weighted avg       0.95      0.95      0.95       214
```


Actual vs Predicted CO2 Emissions
Predicted CO2 Emissions
Actual CO2 Emissions
0.0
0.2
0.4
0.6
0.8
1.0
Predicted vs Actual
Perfect Prediction

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1067 entries, 0 to 1066
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   MODELYEAR                 1067 non-null   int64  
 1   MAKE                      1067 non-null   object 
 2   MODEL                     1067 non-null   object 
 3   VEHICLECLASS              1067 non-null   object 
 4   ENGINESIZE                1067 non-null   float64
 5   CYLINDERS                 1067 non-null   int64  
 6   TRANSMISSION              1067 non-null   object 
 7   FUELTYPE                  1067 non-null   object 
 8   FUELCONSUMPTION_CITY      1067 non-null   float64
 9   FUELCONSUMPTION_HWY       1067 non-null   float64
 10  FUELCONSUMPTION_COMB      1067 non-null   float64
 11  FUELCONSUMPTION_COMB_MPG  1067 non-null   int64  
 12  CO2EMISSIONS              1067 non-null   int64  
dtypes: float64(4), int64(4), object(5)
memory usage: 108.5+ KB
```

1. Display the first 10 rows of the dataset
2. Show all unique vehicle classes
3. Find the average CO2 emissions grouped by fuel type
4. List all cars from the year 2018
5. Show top 5 makes with the highest average engine size
6. Count the number of cars per cylinder type
7. Filter cars where CO2 emissions are greater than 300
8. Find the vehicle with the maximum highway fuel consumption
9. Group by make and model, and find the average city fuel consumption
10. Find the correlation between engine size and CO2 emissions

In [18]:

```python
import pandas as pd

# Load the dataset
df=pd.read_csv('dataset/fuel_consumption_dataset.csv')


# 1. Display the first 10 rows of the dataset
query1 = df.head(10)

# 2. Show all unique vehicle classes
query2 = df['VEHICLECLASS'].unique()

# 3. Find the average CO2 emissions grouped by fuel type
query3 = df.groupby('FUELTYPE')['CO2EMISSIONS'].mean()

# 4. List all cars from the year 2018
query4 = df[df['MODELYEAR'] == 2018]

# 5. Show top 5 makes with the highest average engine size
query5 = df.groupby('MAKE')['ENGINESIZE'].mean().sort_values(ascending=False).head(5)

# 6. Count the number of cars per cylinder type
query6 = df['CYLINDERS'].value_counts()

# 7. Filter cars where CO2 emissions are greater than 300
query7 = df[df['CO2EMISSIONS'] > 300]

# 8. Find the vehicle with the maximum highway fuel consumption
query8 = df.loc[df['FUELCONSUMPTION_HWY'].idxmax()]

# 9. Group by make and model, and find the average city fuel consumption
query9 = df.groupby(['MAKE', 'MODEL'])['FUELCONSUMPTION_CITY'].mean()

# 10. Find the correlation between engine size and CO2 emissions
query10 = df[['ENGINESIZE', 'CO2EMISSIONS']].corr().iloc[0,1]

# Example to print one of the queries
```

```
print(query5)
```

```
MAKE
SRT             8.400000
ROLLS-ROYCE     6.657143
LAMBORGHINI     5.633333
BENTLEY         5.350000
ASTON MARTIN    5.214286
Name: ENGINESIZE, dtype: float64
```