Car Price Prediction

Problem Statement

A Chinese automobile company Geely Auto aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.

They have contracted an automobile consulting company to understand the factors on which the pricing of cars depends. Specifically, they want to understand the factors affecting the pricing of cars in the American market, since those may be very different from the Chinese market. The company wants to know:

```
- Which variables are significant in predicting the price of a car
- How well those variables describe the price of a car
```

Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the Americal market.

Business Goal

You are required to model the price of cars with the available independent variables. It will be used by the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels. Further, the model will be a good way for management to understand the pricing dynamics of a new market.

In [48]:
```
import warnings
warnings.filterwarnings('ignore')

#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 1: Reading and Understanding the Data

Let's start with the following steps:

1. Importing data using the pandas library
2. Understanding the structure of the data

In [49]:
```
cars = pd.read_csv('CarPrice_Assignment.csv')
cars.head()
```

Out[49]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd
3	4	2	audi 100 ls	gas	std	four	sedan	fwd
4	5	2	audi 100ls	gas	std	four	sedan	4wd

5 rows × 26 columns

In [50]:
```
cars.shape
```

Out[50]:
```
(205, 26)
```

In [51]:
```
cars.describe()
```

Out[51]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000

In [52]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

Step 2 : Data Cleaning and Preparation

In [53]:
```
#Splitting company name from CarName column
CompanyName = cars['CarName'].apply(lambda x : x.split(' ')[0])
cars.insert(3,"CompanyName",CompanyName)
cars.drop(['CarName'],axis=1,inplace=True)
cars.head()
```

Out[53]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewhe
0	1	3	alfa-romero	gas	std	two	convertible	rw
1	2	3	alfa-romero	gas	std	two	convertible	rw
2	3	1	alfa-romero	gas	std	two	hatchback	rw
3	4	2	audi	gas	std	four	sedan	fw
4	5	2	audi	gas	std	four	sedan	4w

5 rows × 26 columns

In [54]:
```
cars.CompanyName.unique()
```

Out[54]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'maxda', 'mazda', 'buick', 'mercury',
       'mitsubishi', 'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche',
       'porcshce', 'renault', 'saab', 'subaru', 'toyota', 'toyouta',
       'vokswagen', 'volkswagen', 'vw', 'volvo'], dtype=object)
```

Fixing invalid values

- There seems to be some spelling error in the CompanyName column.
  - `maxda = mazda`
  - `Nissan = nissan`
  - `porsche = porcshce`
  - `toyota = toyouta`
  - `vokswagen = volkswagen = vw`

In [55]:
```
cars.CompanyName = cars.CompanyName.str.lower()

def replace_name(a,b):
    cars.CompanyName.replace(a,b,inplace=True)

replace_name('maxda','mazda')
replace_name('porcshce','porsche')
replace_name('toyouta','toyota')
replace_name('vokswagen','volkswagen')
replace_name('vw','volkswagen')

cars.CompanyName.unique()
```

Out[55]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'nissan', 'peugeot', 'plymouth', 'porsche', 'renault', 'saab',
       'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

In [56]:
```
#Checking for duplicates
cars.loc[cars.duplicated()]
```

Out[56]:

car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel

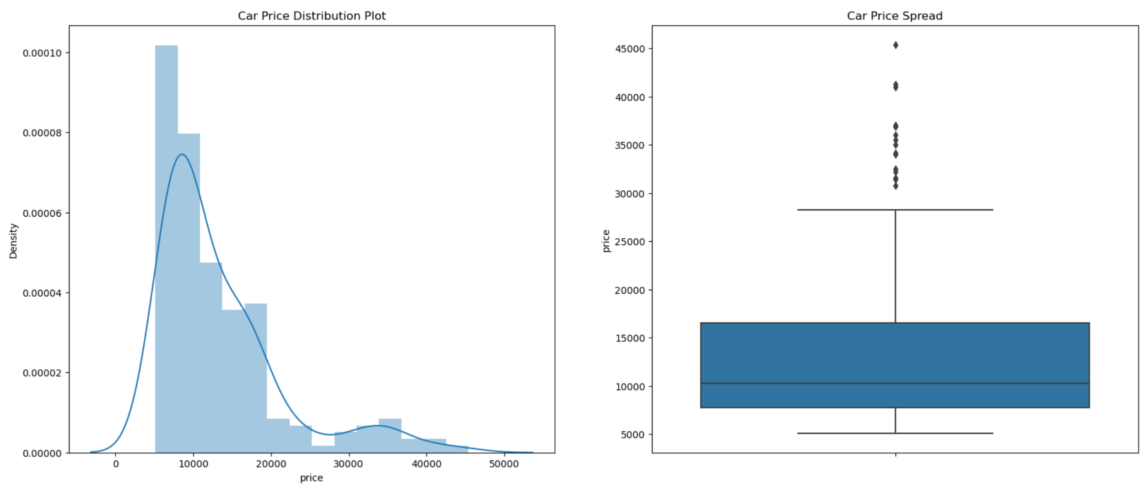
0 rows × 26 columns

In [57]:
```
cars.columns
```

Out[57]:
```
Index(['car_ID', 'symboling', 'CompanyName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbas
e',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'strok
e',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaymp
g',
       'price'],
      dtype='object')
```

Step 3: Visualizing the data

In [58]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Car Price Distribution Plot')
sns.distplot(cars.price)

plt.subplot(1,2,2)
plt.title('Car Price Spread')
sns.boxplot(y=cars.price)

plt.show()
```



In [59]:
```
print(cars.price.describe(percentiles = [0.25,0.50,0.75,0.85,0.90,1]))
```

```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
85%      18500.000000
90%      22563.000000
100%     45400.000000
max      45400.000000
Name: price, dtype: float64
```

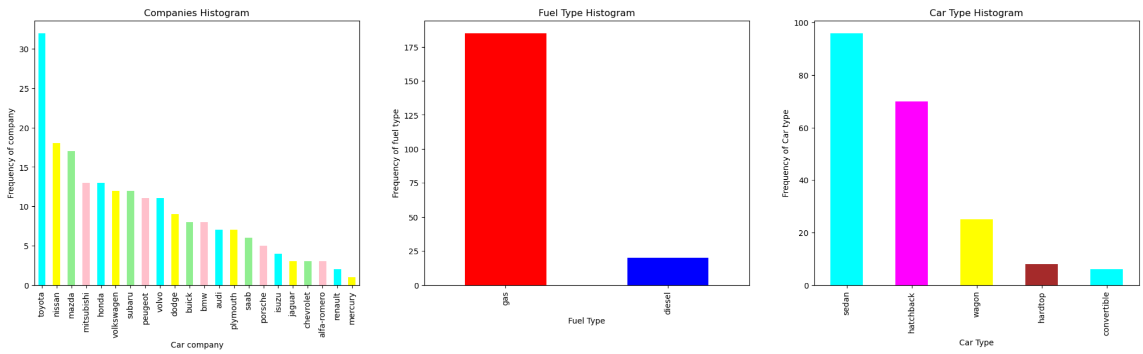
Inference :

1. The plot seemed to be right-skewed, meaning that the most prices in the dataset are low(Below 15,000).
2. There is a significant difference between the mean and the median of the price distribution.
3. The data points are far spread out from the mean, which indicates a high variance in the car prices.(85% of the prices are below 18,500, whereas the remaining 15% are between 18,500 and 45,400.)

Step 3.1 : Visualising Categorical Data

```
- CompanyName
- Symboling
- fueltype
- enginetype
- carbody
- doornumber
- enginelocation
- fuelsystem
- cylindernumber
- aspiration
- drivewheel
```

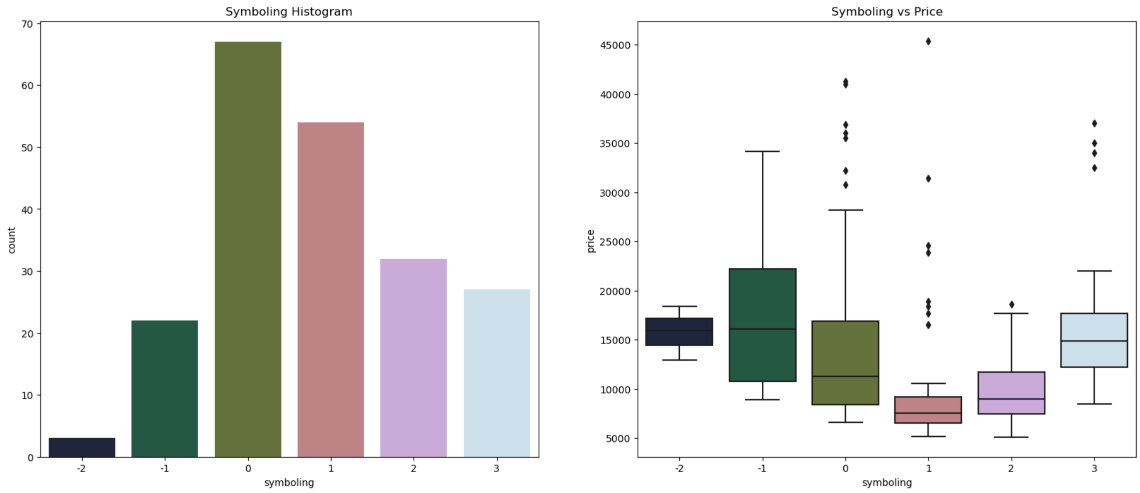
In [60]:
```
plt.figure(figsize=(25, 6))

company_colors = ['cyan', 'yellow', 'lightgreen', 'pink']
fueltype_colors = ['red', 'blue']
carbody_colors = ['cyan', 'magenta', 'yellow', 'brown']

plt.subplot(1, 3, 1)
plt1 = cars.CompanyName.value_counts().plot(kind='bar', color=company_color
plt.title('Companies Histogram')
plt1.set(xlabel='Car company', ylabel='Frequency of company')

plt.subplot(1, 3, 2)
plt2 = cars.fueltype.value_counts().plot(kind='bar', color=fueltype_colors)
plt.title('Fuel Type Histogram')
plt2.set(xlabel='Fuel Type', ylabel='Frequency of fuel type')

plt.subplot(1, 3, 3)
plt3 = cars.carbody.value_counts().plot(kind='bar', color=carbody_colors)
plt.title('Car Type Histogram')
plt3.set(xlabel='Car Type', ylabel='Frequency of Car type')

plt.show()
```



Inference :

1. `Toyota` seemed to be favored car company.
2. Number of `gas` fueled cars are more than `diesel` .
3. `sedan` is the top car type prefered.

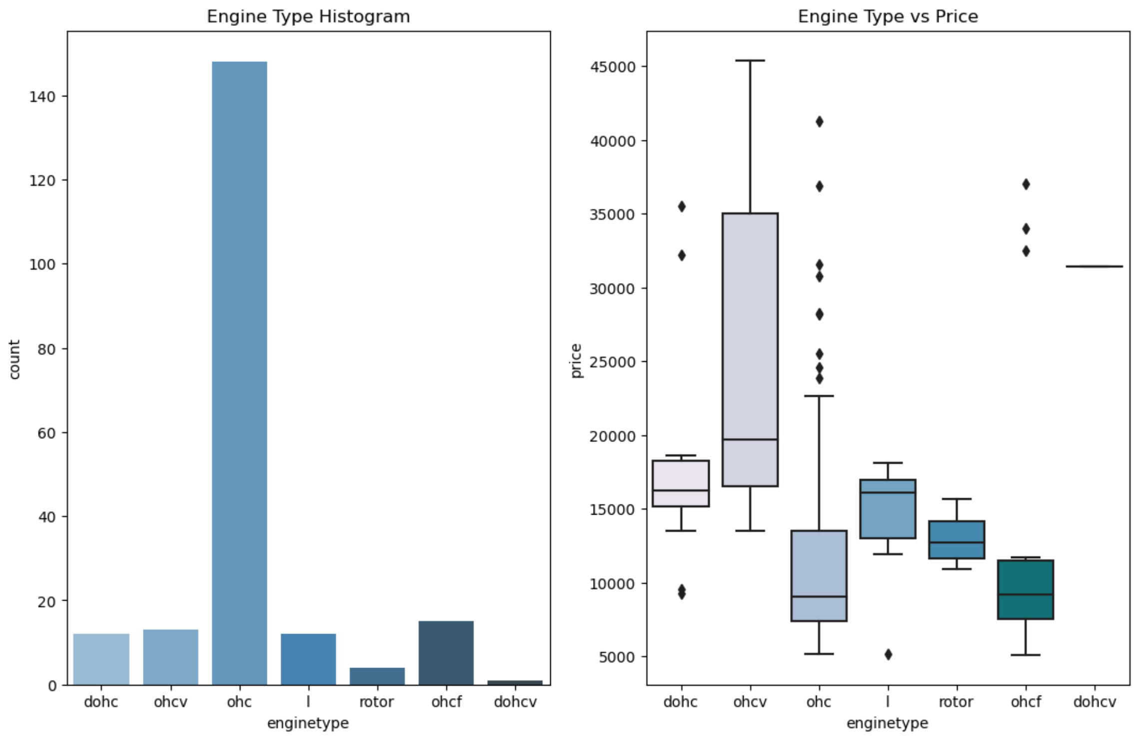
In [61]:
```
plt.figure(figsize=(20, 8))

plt.subplot(1, 2, 1)
plt.title('Symboling Histogram')
sns.countplot(x=cars['symboling'], palette="cubehelix")

plt.subplot(1, 2, 2)
plt.title('Symboling vs Price')
sns.boxplot(x=cars['symboling'], y=cars['price'], palette="cubehelix")

plt.show()
```



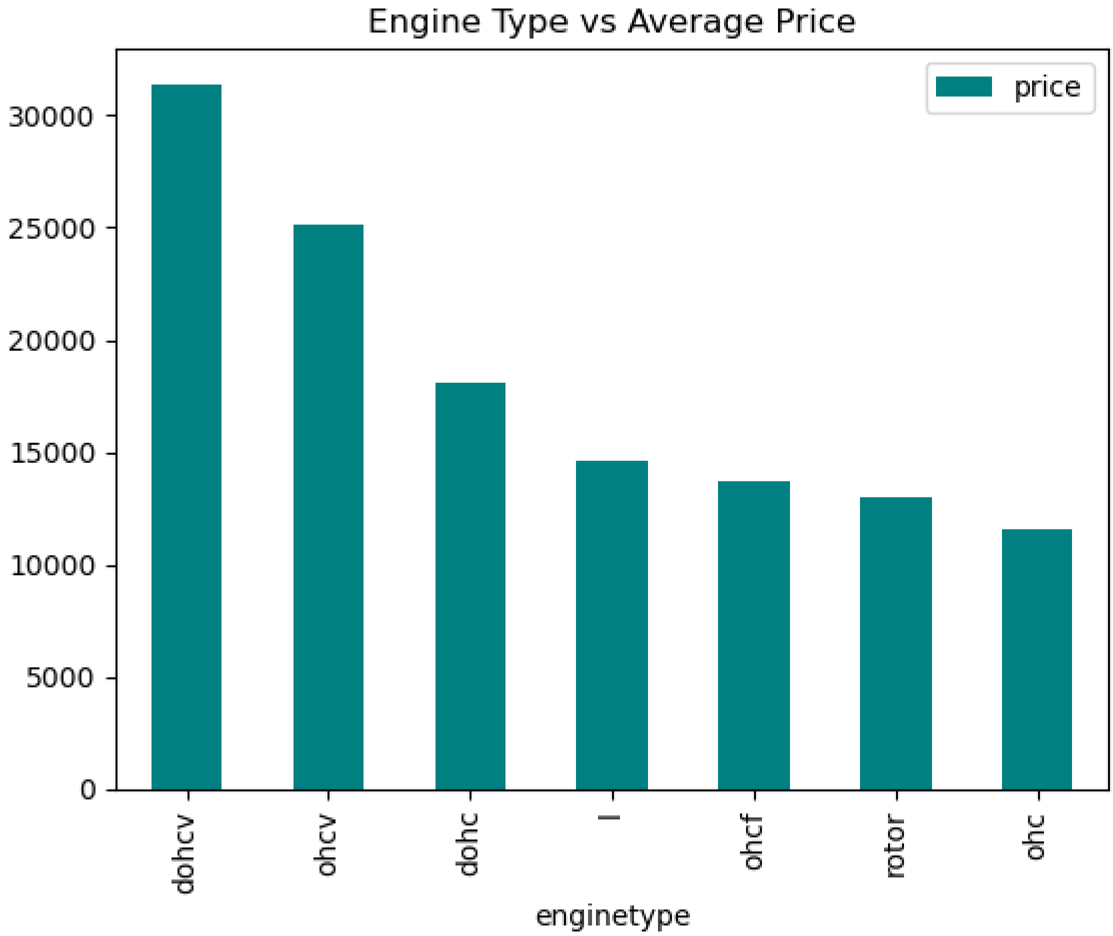
Inference :

1. It seems that the symboling with `0` and `1` values have high number of rows (i.e. They are most sold.)
2. The cars with `-1` symboling seems to be high priced (as it makes sense too, insurance risk rating -1 is quite good). But it seems that symboling with `3` value has the price range similar to `-2` value. There is a dip in price at symboling `1`.

In [62]:
```
plt.figure(figsize=(20, 8))

# First subplot: Count plot
plt.subplot(1, 3, 1)
plt.title('Engine Type Histogram')
sns.countplot(x=cars['enginetype'], palette="Blues_d")

# Second subplot: Boxplot
plt.subplot(1, 3, 2)
plt.title('Engine Type vs Price')
sns.boxplot(x=cars['enginetype'], y=cars['price'], palette="PuBuGn")

# Create a new figure for the third subplot
plt.figure(figsize=(8, 6))
df = pd.DataFrame(cars.groupby(['enginetype'])['price'].mean().sort_values(
df.plot.bar(color='teal')  # Adjust color as needed
plt.title('Engine Type vs Average Price')
plt.show()
```




```
<Figure size 800x600 with 0 Axes>
```

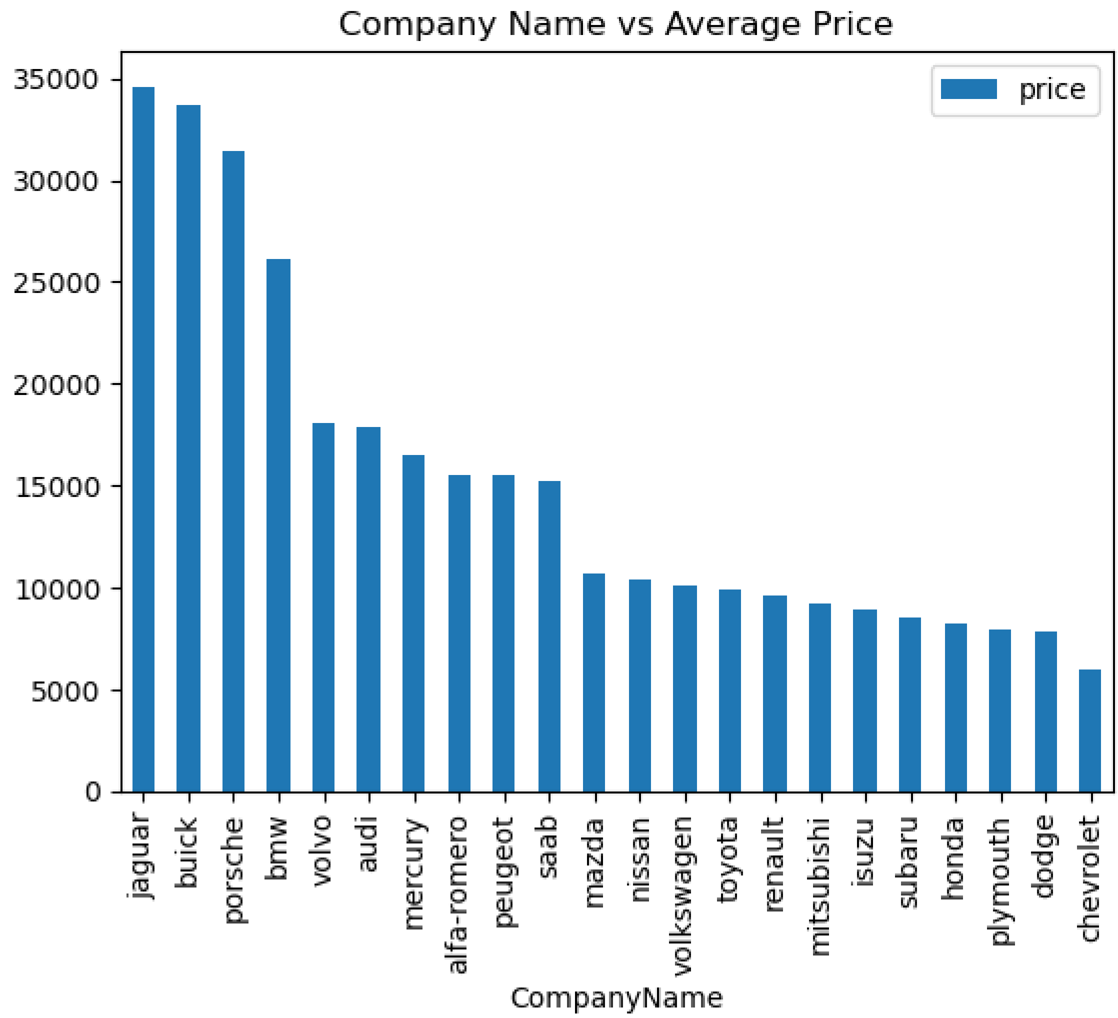


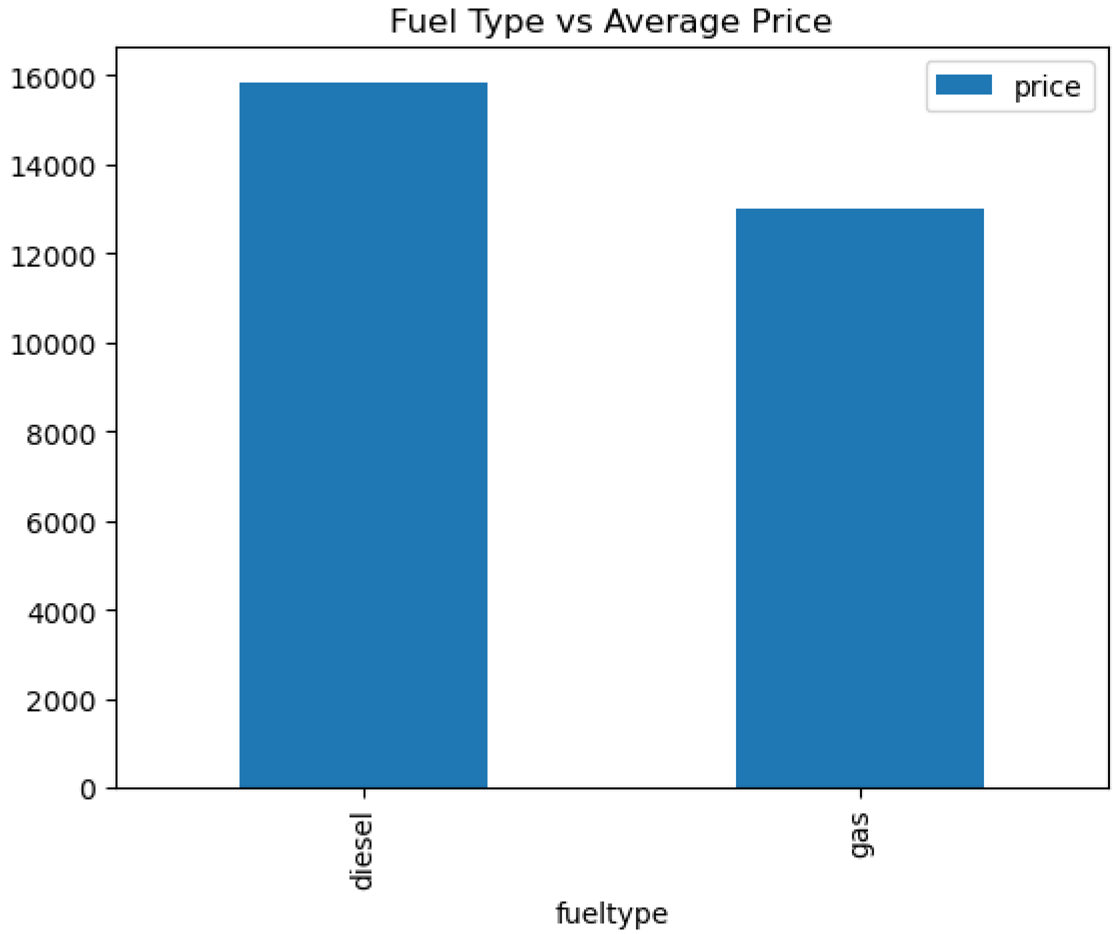
Inference :

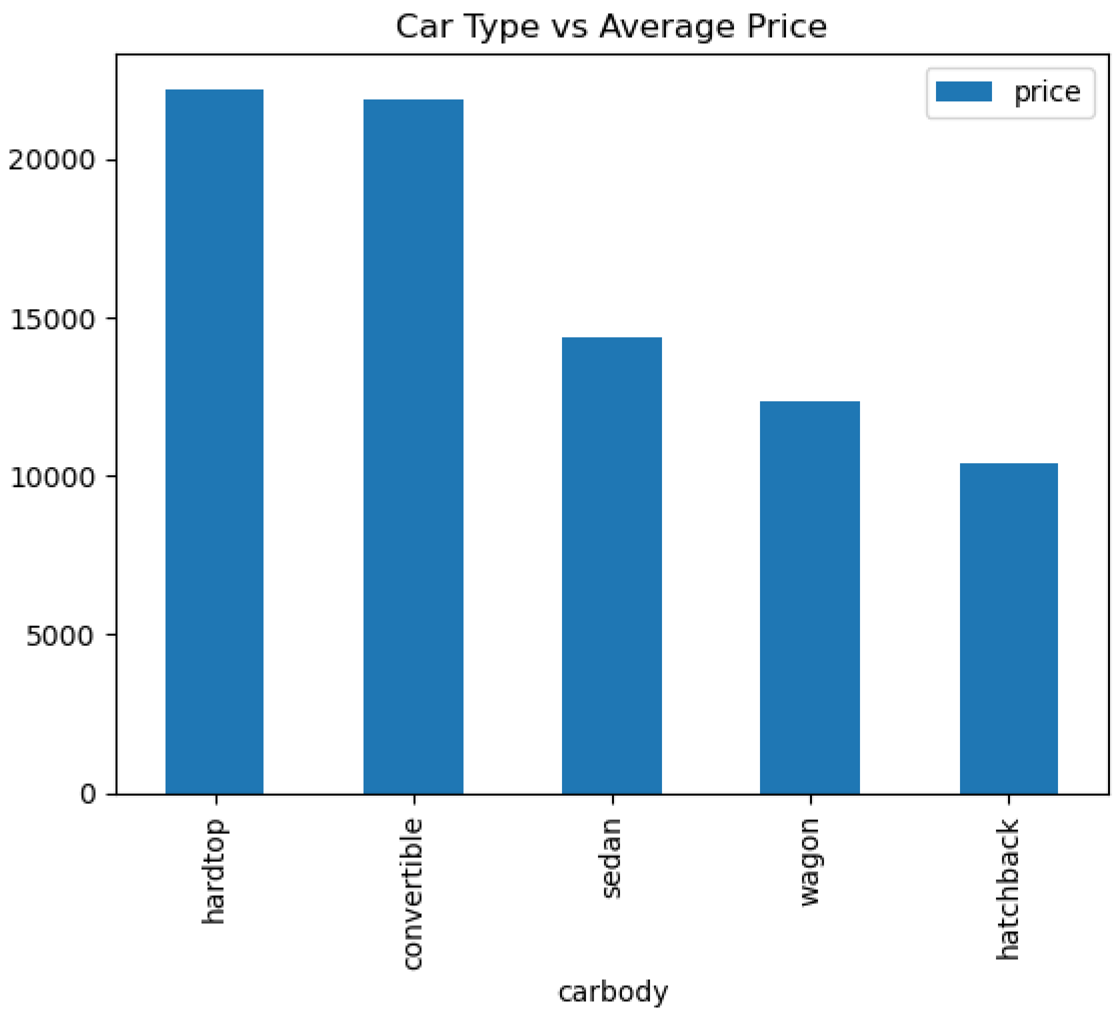
1. `ohc` Engine type seems to be most favored type.
2. `ohcv` has the highest price range (While `dohcv` has only one row), `ohc` and `ohcf` have the low price range.

In [63]:
```
plt.figure(figsize=(25, 6))

df = pd.DataFrame(cars.groupby(['CompanyName'])['price'].mean().sort_values
df.plot.bar()
plt.title('Company Name vs Average Price')
plt.show()

df = pd.DataFrame(cars.groupby(['fueltype'])['price'].mean().sort_values(as
df.plot.bar()
plt.title('Fuel Type vs Average Price')
plt.show()

df = pd.DataFrame(cars.groupby(['carbody'])['price'].mean().sort_values(asc
df.plot.bar()
plt.title('Car Type vs Average Price')
plt.show()
```

```
<Figure size 2500x600 with 0 Axes>
```




Fuel Type vs Average Price
price
16000
14000
12000
10000
8000
6000
4000
2000
0
diesel
gas
fueltype

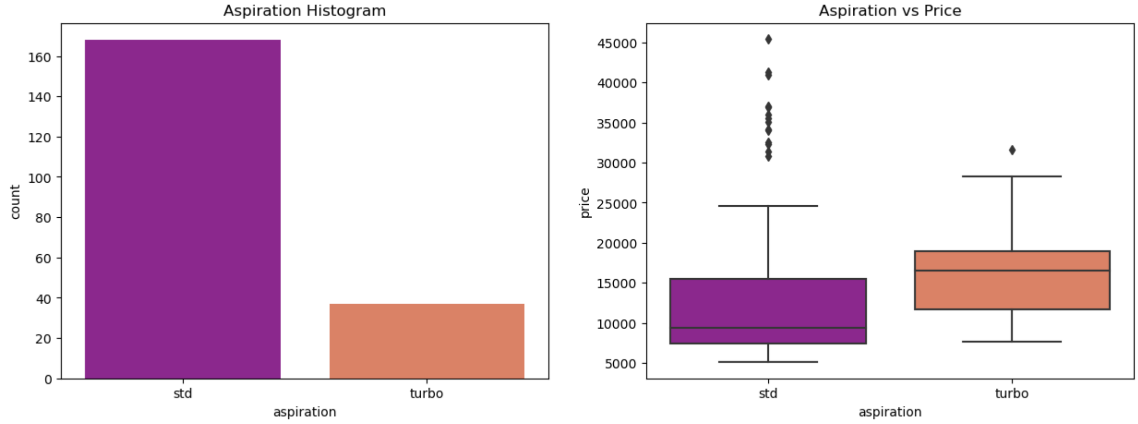

Car Type vs Average Price
price
20000
15000
10000
5000
0
hardtop
convertible
sedan
wagon
hatchback
carbody

Inference :

1. `Jaguar` and `Buick` seem to have highest average price.
2. `diesel` has higher average price than gas.
3. `hardtop` and `convertible` have higher average price.

In [74]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'cars' is your DataFrame

plt.figure(figsize=(15, 5))

# Create a new figure for the next set of subplots
plt.figure(figsize=(15, 5))

# Handle 'aspiration'
plt.subplot(1, 2, 1)
plt.title('Aspiration Histogram')
sns.countplot(x=cars['aspiration'], palette=("plasma"))

plt.subplot(1, 2, 2)
plt.title('Aspiration vs Price')

# Ensure 'price' and 'aspiration' contain valid data
if cars['price'].dtype == 'float64' and cars['aspiration'].dtype == 'object
    sns.boxplot(x=cars['aspiration'], y=cars['price'].dropna(), palette=("p
else:
    print("Check 'price' and 'aspiration' data types or handle missing/inco

plt.show()
```

```
<Figure size 1500x500 with 0 Axes>
```

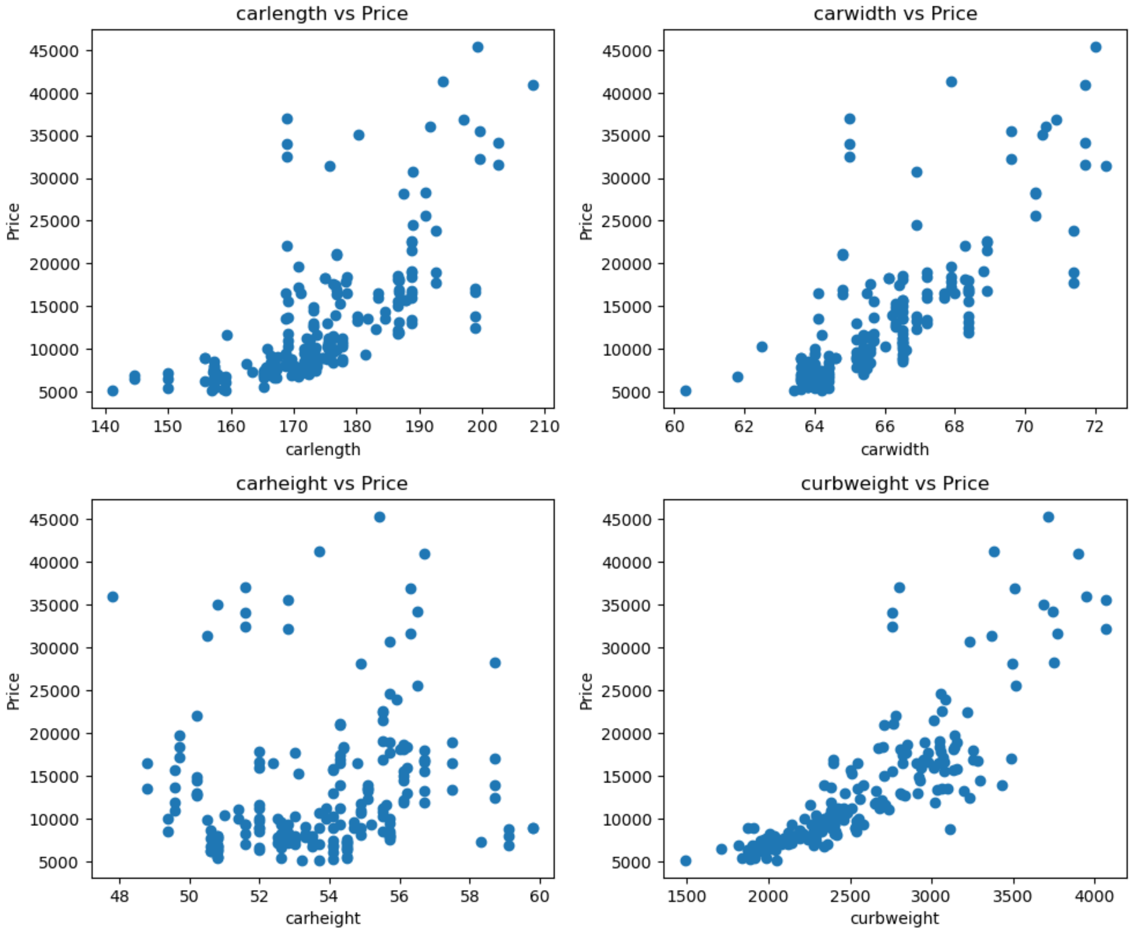


Inference :

1. It seems aspiration with `turbo` have higher price range than the `std` (though it has some high values outside the whiskers.)

Step 3.2 : Visualising numerical data

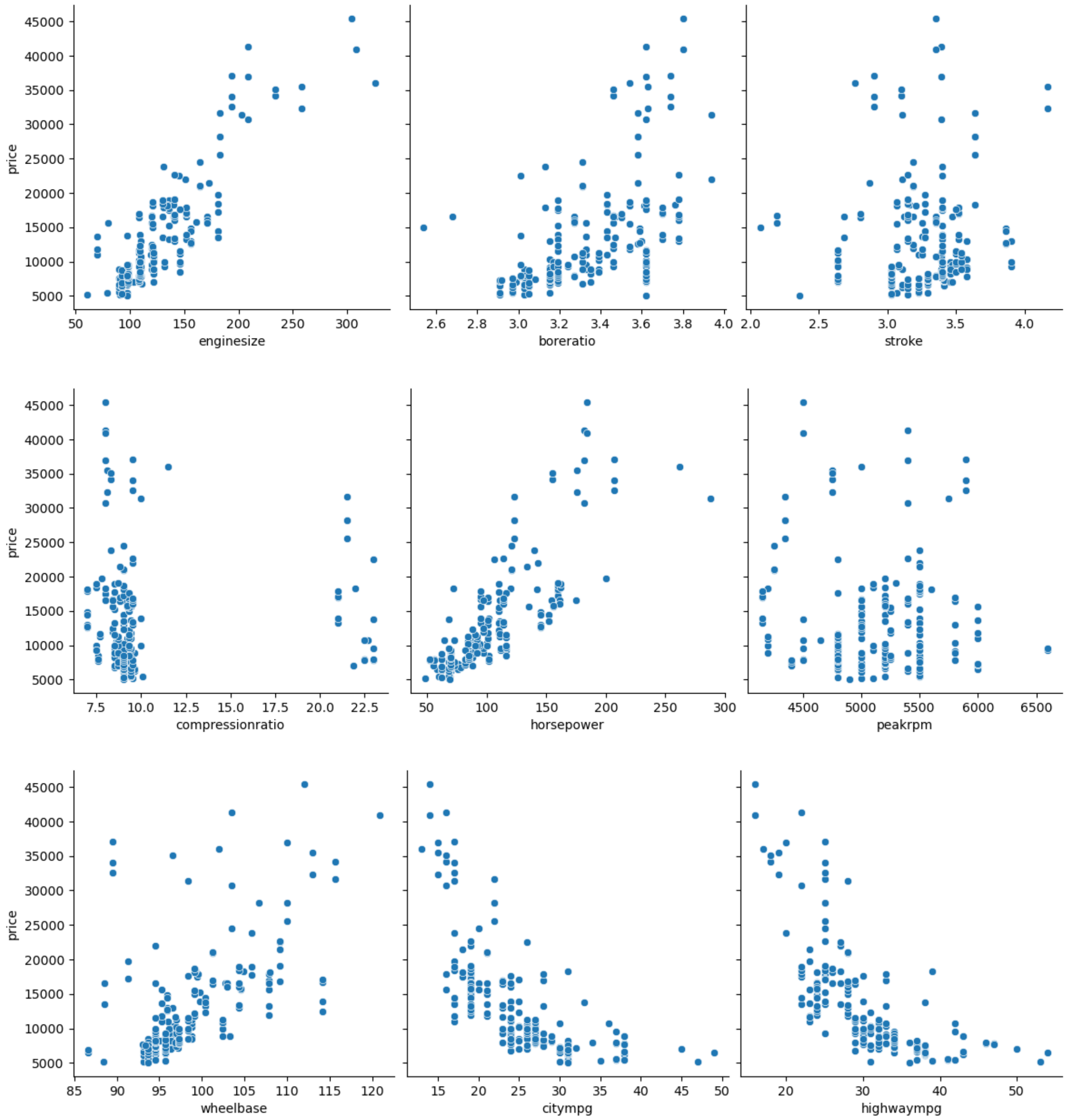
In [78]:
```python
def scatter(x,fig):
    plt.subplot(5,2,fig)
    plt.scatter(cars[x],cars['price'])
    plt.title(x+' vs Price')
    plt.ylabel('Price')
    plt.xlabel(x)

plt.figure(figsize=(10,20))

scatter('carlength', 1)
scatter('carwidth', 2)
scatter('carheight', 3)
scatter('curbweight', 4)

plt.tight_layout()
```



Inference :

1. `carwidth` , `carlength` and `curbweight` seems to have a poitive correlation with `price` .
2. `carheight` doesn't show any significant trend with price.

In [79]:
```
def pp(x,y,z):
    sns.pairplot(cars, x_vars=[x,y,z], y_vars='price',size=4, aspect=1, kin
    plt.show()

pp('enginesize', 'boreratio', 'stroke')
pp('compressionratio', 'horsepower', 'peakrpm')
pp('wheelbase', 'citympg', 'highwaympg')
```



Inference :

1. `enginesize` , `boreratio` , `horsepower` , `wheelbase` - seem to have a significant positive correlation with price.
2. `citympg` , `highwaympg` - seem to have a significant negative correlation with price.

In [80]:
```
np.corrcoef(cars['carlength'], cars['carwidth'])[0, 1]
```

Out[80]:
```
0.841118268481846
```

Step 4 : Deriving new features

In [81]:
```
#Fuel economy
cars['fueleconomy'] = (0.55 * cars['citympg']) + (0.45 * cars['highwaympg']
```

In [82]:
```
#Binning the Car Companies based on avg prices of each Company.
cars['price'] = cars['price'].astype('int')
temp = cars.copy()
table = temp.groupby(['CompanyName'])['price'].mean()
temp = temp.merge(table.reset_index(), how='left',on='CompanyName')
bins = [0,10000,20000,40000]
cars_bin=['Budget','Medium','Highend']
cars['carsrange'] = pd.cut(temp['price_y'],bins,right=False,labels=cars_bin
cars.head()
```

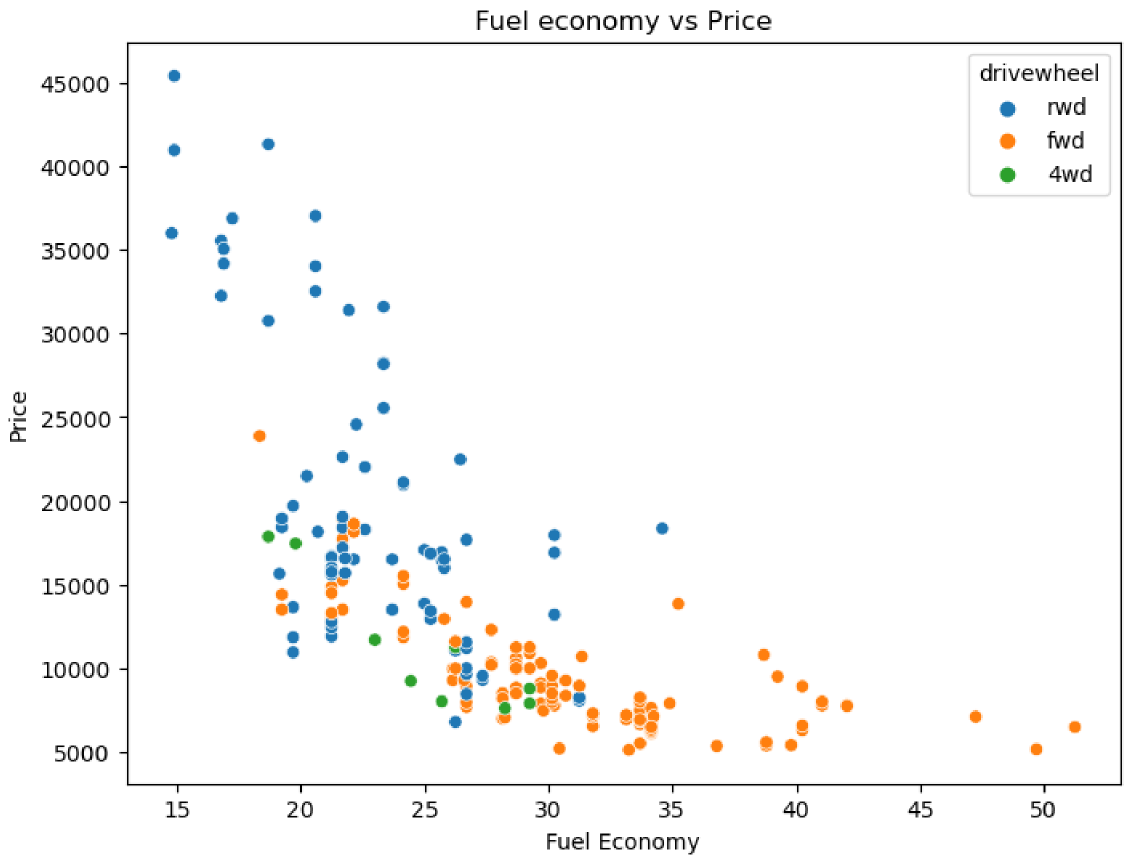
Out[82]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewhe
0	1	3	alfa-romero	gas	std	NaN	convertible	rw
1	2	3	alfa-romero	gas	std	NaN	convertible	rw
2	3	1	alfa-romero	gas	std	NaN	hatchback	rw
3	4	2	audi	gas	std	NaN	sedan	fw
4	5	2	audi	gas	std	NaN	sedan	4w

5 rows × 28 columns

Step 5 : Bivariate Analysis

In [83]:
```
plt.figure(figsize=(8,6))

plt.title('Fuel economy vs Price')
sns.scatterplot(x=cars['fueleconomy'],y=cars['price'],hue=cars['drivewheel'
plt.xlabel('Fuel Economy')
plt.ylabel('Price')

plt.show()
plt.tight_layout()
```



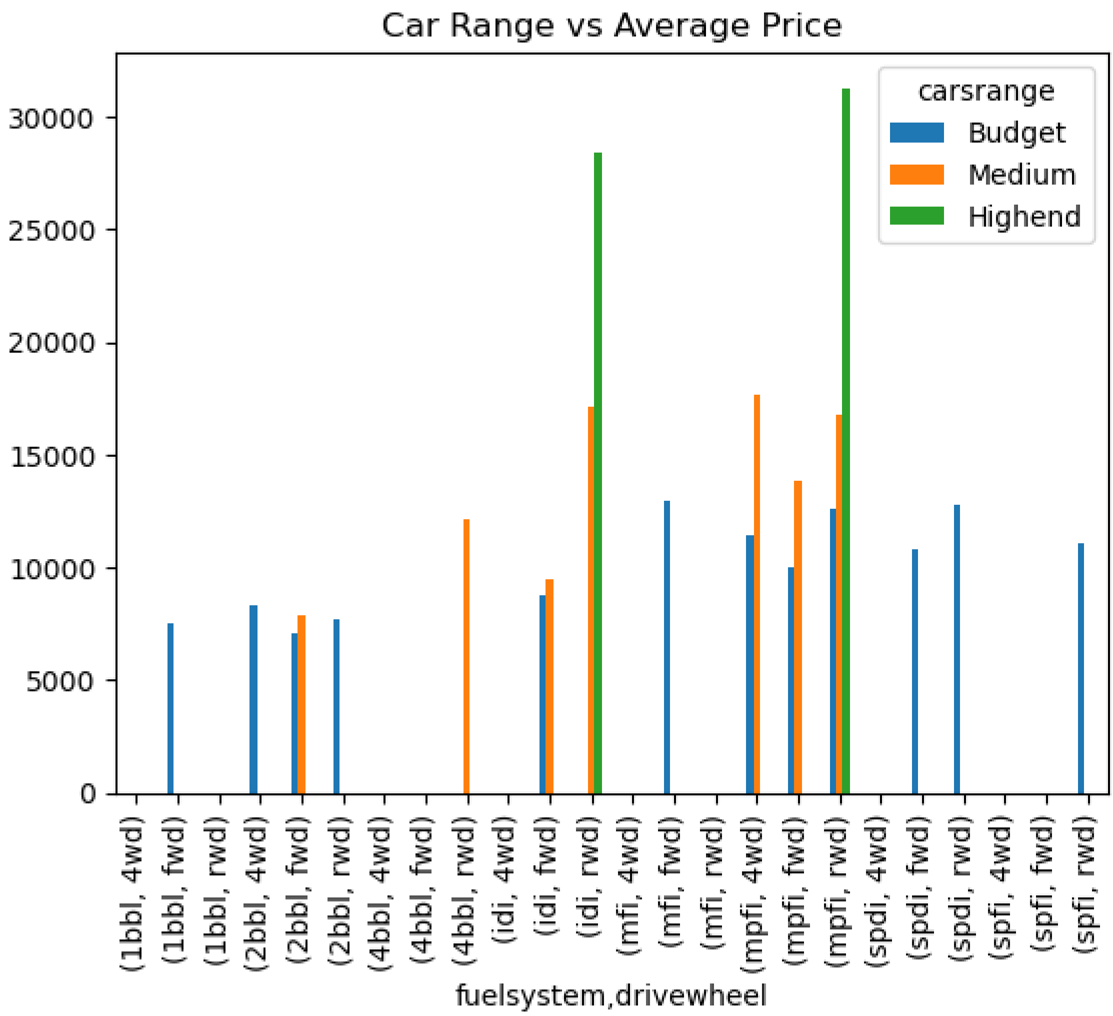

```
<Figure size 640x480 with 0 Axes>
```

Inference :

1. `fueleconomy` has an obvios `negative correlation` with price and is significant.

In [84]:
```
plt.figure(figsize=(25, 6))

df = pd.DataFrame(cars.groupby(['fuelsystem','drivewheel','carsrange'])['pr
df.plot.bar()
plt.title('Car Range vs Average Price')
plt.show()
```

```
<Figure size 2500x600 with 0 Axes>
```



Inference :

1. High ranged cars prefer `rwd` drivewheel with `idi` or `mpfi` fuelsystem.

List of significant variables after Visual analysis :

```
- Car Range
- Engine Type
- Fuel type
- Car Body
- Aspiration
- Cylinder Number
- Drivewheel
- Curbweight
- Car Length
- Car width
- Engine Size
- Boreratio
- Horse Power
- Wheel base
- Fuel Economy
```

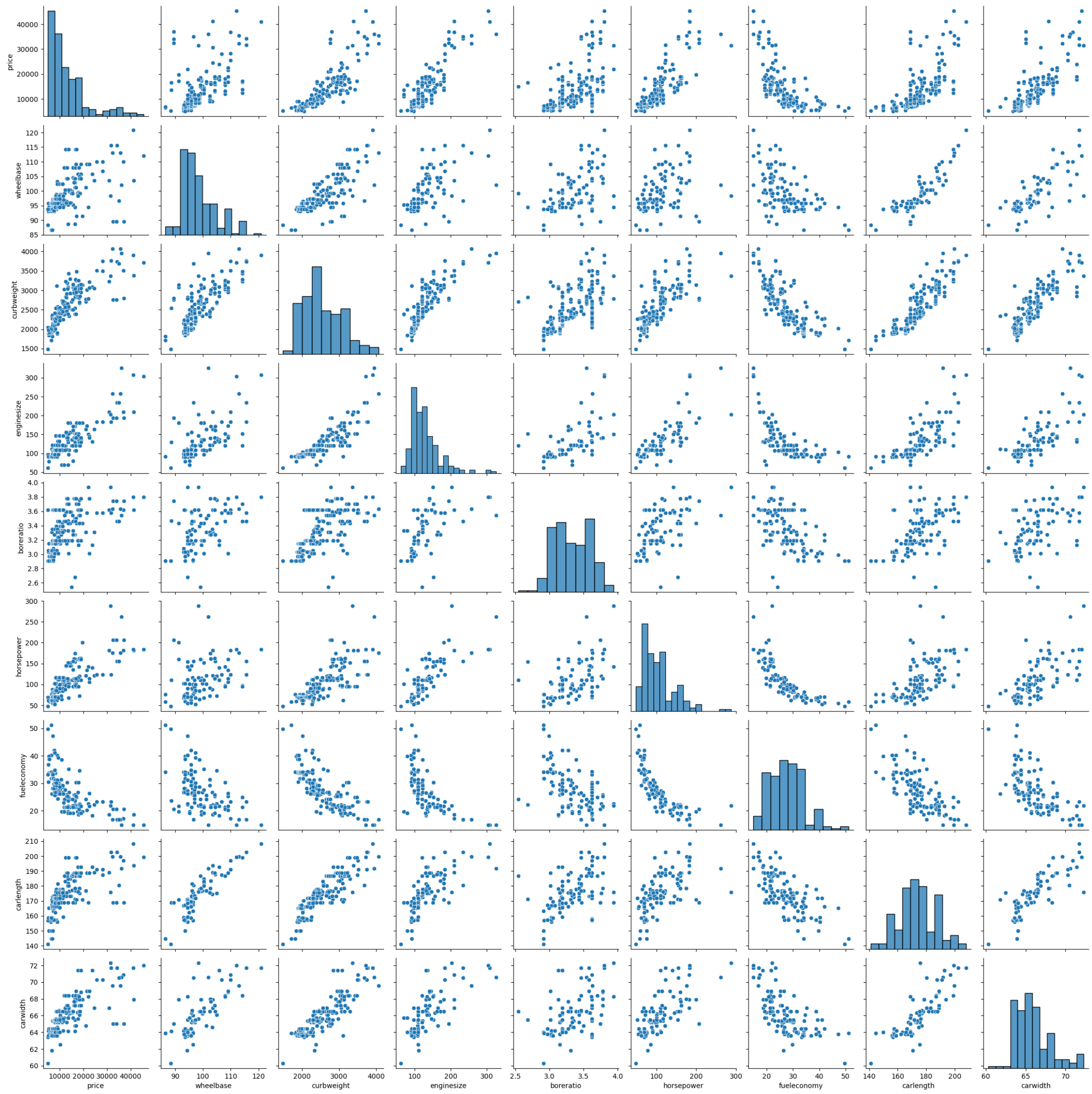
In [85]:

```
cars_lr = cars[['price', 'fueltype', 'aspiration','carbody', 'drivewheel','
                'curbweight', 'enginetype', 'cylindernumber', 'enginesize
                 'fueleconomy', 'carlength','carwidth', 'carsrange']]
cars_lr.head()
```

Out[85]:

	price	fueltype	aspiration	carbody	drivewheel	wheelbase	curbweight	enginetype	cy
0	13495	gas	std	convertible	rwd	88.6	2548	dohc	
1	16500	gas	std	convertible	rwd	88.6	2548	dohc	
2	16500	gas	std	hatchback	rwd	94.5	2823	ohcv	
3	13950	gas	std	sedan	fwd	99.8	2337	ohc	
4	17450	gas	std	sedan	4wd	99.4	2824	ohc	

In [86]:
```python
sns.pairplot(cars_lr)
plt.show()
```



Step 6 : Dummy Variables

In [87]:
```python
# Defining the map function
def dummies(x,df):
    temp = pd.get_dummies(df[x], drop_first = True)
    df = pd.concat([df, temp], axis = 1)
    df.drop([x], axis = 1, inplace = True)
    return df
# Applying the function to the cars_lr

cars_lr = dummies('fueltype',cars_lr)
cars_lr = dummies('aspiration',cars_lr)
cars_lr = dummies('carbody',cars_lr)
cars_lr = dummies('drivewheel',cars_lr)
cars_lr = dummies('enginetype',cars_lr)
cars_lr = dummies('cylindernumber',cars_lr)
cars_lr = dummies('carsrange',cars_lr)
```

In [88]:
```
cars_lr.head()
```

Out[88]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength
0	13495	88.6	2548	130	3.47	111	23.70	168.8
1	16500	88.6	2548	130	3.47	111	23.70	168.8
2	16500	94.5	2823	152	2.68	154	22.15	171.2
3	13950	99.8	2337	109	3.19	102	26.70	176.6
4	17450	99.4	2824	136	3.19	115	19.80	176.6

5 rows × 31 columns

In [89]:
```
cars_lr.shape
```

Out[89]: (205, 31)

Step 7 : Train-Test Split and feature scaling

In [90]:
```
from sklearn.model_selection import train_test_split

np.random.seed(0)
df_train, df_test = train_test_split(cars_lr, train_size = 0.7, test_size =
```

In [91]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepow
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

In [92]:
```
df_train.head()
```

Out[92]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carle
122	0.068818	0.244828	0.272692	0.139623	0.230159	0.083333	0.530864	0.42
125	0.466890	0.272414	0.500388	0.339623	1.000000	0.395833	0.213992	0.45
166	0.122110	0.272414	0.314973	0.139623	0.444444	0.266667	0.344307	0.44
1	0.314446	0.068966	0.411171	0.260377	0.626984	0.262500	0.244170	0.45
199	0.382131	0.610345	0.647401	0.260377	0.746032	0.475000	0.122085	0.77

5 rows × 31 columns

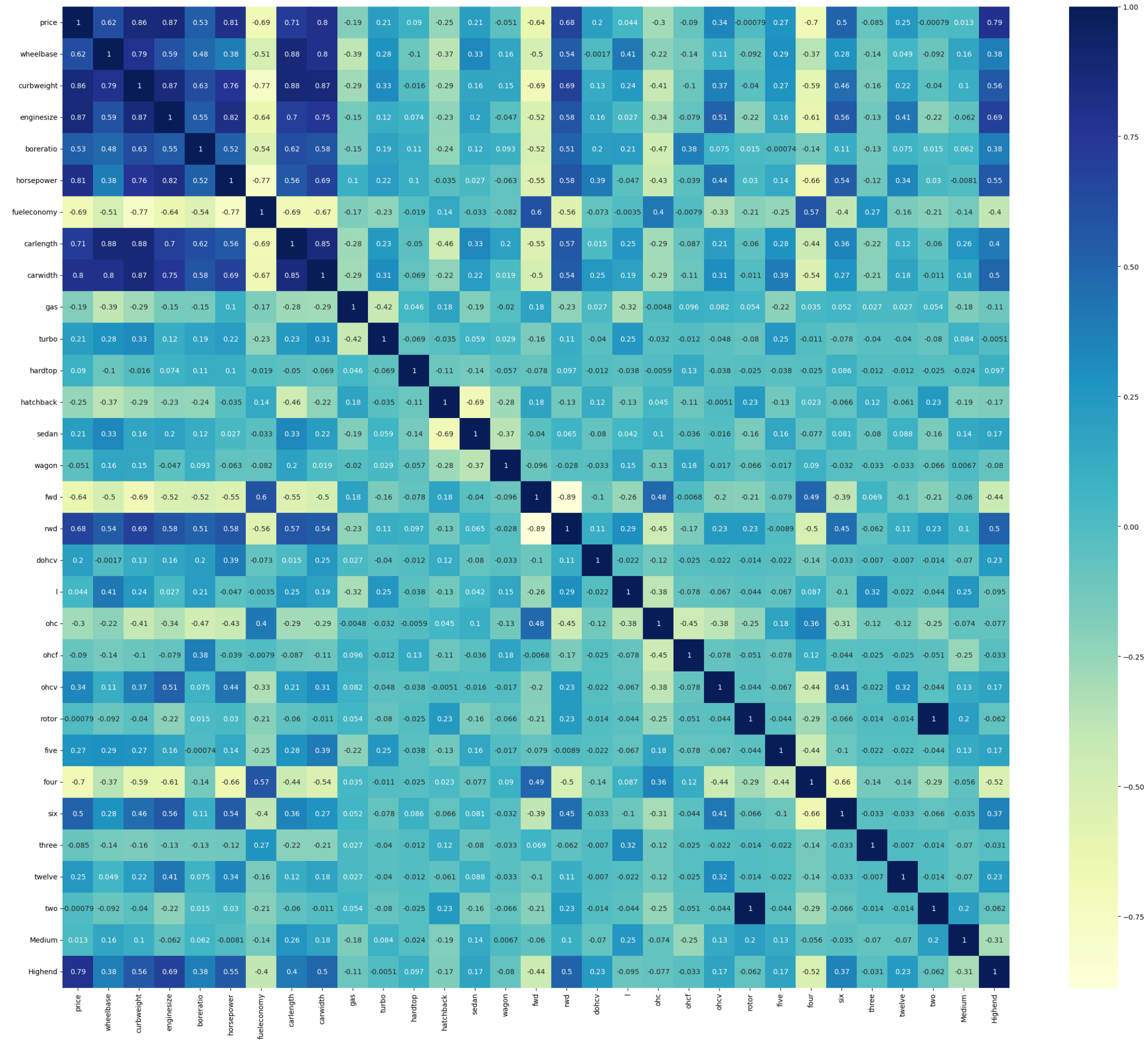
In [93]:
```
df_train.describe()
```

Out[93]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.00000
mean	0.219309	0.411141	0.407878	0.241351	0.497946	0.227302	0.35826
std	0.215682	0.205581	0.211269	0.154619	0.207140	0.165511	0.18598
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.067298	0.272414	0.245539	0.135849	0.305556	0.091667	0.19890
50%	0.140343	0.341379	0.355702	0.184906	0.500000	0.191667	0.34430
75%	0.313479	0.503448	0.559542	0.301887	0.682540	0.283333	0.51234
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

8 rows × 31 columns

In [94]:
```
#Correlation using heatmap
plt.figure(figsize = (30, 25))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



Highly correlated variables to price are - `curbweight` , `enginesize` , `horsepower` , `carwidth` and `highend` .

In [95]:
```
#Dividing data into X and y variables
y_train = df_train.pop('price')
X_train = df_train
```

Step 8 : Model Building

In [96]:
```
#RFE
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [98]:
```

# Create a linear regression model
lm = LinearRegression()

# Create RFE model and specify the number of features to select (e.g., 10)
rfe = RFE(lm, n_features_to_select=10)

# Fit the RFE model to your data
rfe.fit(X_train, y_train)
```

Out[98]:
```
RFE
RFE(estimator=LinearRegression(), n_features_to_select=10)
estimator: LinearRegression
LinearRegression()
LinearRegression
LinearRegression()
```

In [99]:
```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[99]:
```
[('wheelbase', False, 3),
 ('curbweight', True, 1),
 ('enginesize', False, 13),
 ('boreratio', False, 10),
 ('horsepower', True, 1),
 ('fueleconomy', True, 1),
 ('carlength', False, 11),
 ('carwidth', True, 1),
 ('gas', False, 17),
 ('turbo', False, 18),
 ('hardtop', False, 2),
 ('hatchback', True, 1),
 ('sedan', True, 1),
 ('wagon', True, 1),
 ('fwd', False, 16),
 ('rwd', False, 15),
 ('dohcv', True, 1),
 ('l', False, 19),
 ('ohc', False, 7),
 ('ohcf', False, 8),
 ('ohcv', False, 9),
 ('rotor', False, 20),
 ('five', False, 6),
 ('four', False, 4),
 ('six', False, 5),
 ('three', False, 14),
 ('twelve', True, 1),
 ('two', False, 21),
 ('Medium', False, 12),
 ('Highend', True, 1)]
```

In [100]:
```
X_train.columns[rfe.support_]
```

Out[100]:
```
Index(['curbweight', 'horsepower', 'fueleconomy', 'carwidth', 'hatchback',
       'sedan', 'wagon', 'dohcv', 'twelve', 'Highend'],
      dtype='object')
```

Building model using statsmodel, for the detailed statistics

In [101]:
```
X_train_rfe = X_train[X_train.columns[rfe.support_]]
X_train_rfe.head()
```

Out[101]:

	curbweight	horsepower	fueleconomy	carwidth	hatchback	sedan	wagon	dohcv	twe
122	0.272692	0.083333	0.530864	0.291667	0	1	0	0	
125	0.500388	0.395833	0.213992	0.666667	1	0	0	0	
166	0.314973	0.266667	0.344307	0.308333	1	0	0	0	
1	0.411171	0.262500	0.244170	0.316667	0	0	0	0	
199	0.647401	0.475000	0.122085	0.575000	0	0	1	0	

In [102]:
```
def build_model(X,y):
    X = sm.add_constant(X) #Adding the constant
    lm = sm.OLS(y,X).fit() # fitting the model
    print(lm.summary()) # model summary
    return X

def checkVIF(X):
    vif = pd.DataFrame()
    vif['Features'] = X.columns
    vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.s
    vif['VIF'] = round(vif['VIF'], 2)
    vif = vif.sort_values(by = "VIF", ascending = False)
    return(vif)
```

MODEL 1

In [103]:
```
X_train_new = build_model(X_train_rfe,y_train)
```

```
                            OLS Regression Results                            
==============================================================================
====
Dep. Variable:                  price   R-squared:                       
0.929
Model:                            OLS   Adj. R-squared:                  
0.923
Method:                 Least Squares   F-statistic:                     1
72.1
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           1.29
e-70
Time:                        22:45:30   Log-Likelihood:                 20
5.85
No. Observations:                 143   AIC:                            -3
89.7
Df Residuals:                     132   BIC:                            -3
57.1
Df Model:                          10                                         
Covariance Type:            nonrobust                                         
==============================================================================
=====
                  coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
-----
const          -0.0947      0.042     -2.243      0.027      -0.178      -
0.011
curbweight      0.2657      0.069      3.870      0.000       0.130       
0.402
horsepower      0.4499      0.074      6.099      0.000       0.304       
0.596
fueleconomy     0.0933      0.052      1.792      0.075      -0.010       
0.196
carwidth        0.2609      0.062      4.216      0.000       0.138       
0.383
hatchback      -0.0929      0.025     -3.707      0.000      -0.143      -
0.043
sedan          -0.0704      0.025     -2.833      0.005      -0.120      -
0.021
wagon          -0.0997      0.028     -3.565      0.001      -0.155      -
0.044
dohcv          -0.2676      0.079     -3.391      0.001      -0.424      -
0.112
twelve         -0.1192      0.067     -1.769      0.079      -0.253       
0.014
Highend         0.2586      0.020     12.929      0.000       0.219       
0.298
==============================================================================
====
Omnibus:                       43.093   Durbin-Watson:                   
1.867
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               13
0.648
Skew:                           1.128   Prob(JB):                     4.27
e-29
Kurtosis:                       7.103   Cond. No.                        
32.0
==============================================================================
====

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

p-vale of `twelve` seems to be higher than the significance value of 0.05, hence dropping it as it is insignificant in presence of other variables.

In [104]:
```
X_train_new = X_train_rfe.drop(["twelve"], axis = 1)
```

MODEL 2

In [105]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results                            
==============================================================================
====
Dep. Variable:                  price   R-squared:                       
0.927
Model:                            OLS   Adj. R-squared:                  
0.922
Method:                 Least Squares   F-statistic:                     1
87.9
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           4.25
e-71
Time:                        22:45:32   Log-Likelihood:                 20
4.17
No. Observations:                 143   AIC:                            -3
88.3
Df Residuals:                     133   BIC:                            -3
58.7
Df Model:                           9                                         
Covariance Type:            nonrobust                                         
==============================================================================
=====
                  coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
-----
const          -0.0764      0.041     -1.851      0.066      -0.158       
0.005
curbweight      0.2756      0.069      3.995      0.000       0.139       
0.412
horsepower      0.3997      0.069      5.824      0.000       0.264       
0.535
fueleconomy     0.0736      0.051      1.435      0.154      -0.028       
0.175
carwidth        0.2580      0.062      4.137      0.000       0.135       
0.381
hatchback      -0.0951      0.025     -3.766      0.000      -0.145      -
0.045
sedan          -0.0744      0.025     -2.983      0.003      -0.124      -
0.025
wagon          -0.1050      0.028     -3.744      0.000      -0.160      -
0.050
dohcv          -0.2319      0.077     -3.015      0.003      -0.384      -
0.080
Highend         0.2565      0.020     12.743      0.000       0.217       
0.296
==============================================================================
====
Omnibus:                       48.027   Durbin-Watson:                   
1.880
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               15
9.802
Skew:                           1.231   Prob(JB):                     1.99
e-35
Kurtosis:                       7.556   Cond. No.                         
29.6
==============================================================================
====

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [106]:
```
X_train_new = X_train_new.drop(["fueleconomy"], axis = 1)
```

MODEL 3

In [107]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                  price   R-squared:
0.926
Model:                            OLS   Adj. R-squared:
0.922
Method:                 Least Squares   F-statistic:                     2
09.5
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           7.85
e-72
Time:                        22:45:33   Log-Likelihood:                 20
3.07
No. Observations:                 143   AIC:                            -3
88.1
Df Residuals:                     134   BIC:                            -3
61.5
Df Model:                           8
Covariance Type:            nonrobust
==========================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
--------------------------------------------------------------------------
----
const         -0.0305      0.026     -1.165      0.246      -0.082
0.021
curbweight     0.2593      0.068      3.796      0.000       0.124
0.394
horsepower     0.3469      0.058      5.964      0.000       0.232
0.462
carwidth       0.2488      0.062      3.995      0.000       0.126
0.372
hatchback     -0.0922      0.025     -3.650      0.000      -0.142      -
0.042
sedan         -0.0711      0.025     -2.850      0.005      -0.120      -
0.022
wagon         -0.1047      0.028     -3.721      0.000      -0.160      -
0.049
dohcv         -0.1968      0.073     -2.689      0.008      -0.342      -
0.052
Highend        0.2610      0.020     13.083      0.000       0.222
0.301
==========================================================================
====
Omnibus:                       48.637   Durbin-Watson:
1.909
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               16
1.444
Skew:                           1.250   Prob(JB):                     8.77
e-36
Kurtosis:                       7.566   Cond. No.
27.2
==========================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [108]:
```
#Calculating the Variance Inflation Factor
checkVIF(X_train_new)
```

Out[108]:

	Features	VIF
0	const	26.90
1	curbweight	8.10
5	sedan	6.07
4	hatchback	5.63
3	carwidth	5.14
2	horsepower	3.61
6	wagon	3.58
8	Highend	1.63
7	dohcv	1.46

dropping `curbweight` because of high VIF value. (shows that curbweight has high multicollinearity.)

In [109]:
```
X_train_new = X_train_new.drop(["curbweight"], axis = 1)
```

MODEL 4

In [110]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                  price   R-squared:
0.918
Model:                            OLS   Adj. R-squared:
0.914
Method:                 Least Squares   F-statistic:                     2
15.9
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           4.70
e-70
Time:                        22:45:34   Log-Likelihood:                 19
5.77
No. Observations:                 143   AIC:                            -3
75.5
Df Residuals:                     135   BIC:                            -3
51.8
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const         -0.0319      0.027     -1.161      0.248      -0.086
0.022
horsepower     0.4690      0.051      9.228      0.000       0.368
0.569
carwidth       0.4269      0.043      9.944      0.000       0.342
0.512
hatchback     -0.1044      0.026     -3.976      0.000      -0.156      -
0.052
sedan         -0.0756      0.026     -2.896      0.004      -0.127      -
0.024
wagon         -0.0865      0.029     -2.974      0.003      -0.144      -
0.029
dohcv         -0.3106      0.070     -4.435      0.000      -0.449      -
0.172
Highend        0.2772      0.020     13.559      0.000       0.237
0.318
==============================================================================
====
Omnibus:                       43.937   Durbin-Watson:
2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               12
7.746
Skew:                           1.171   Prob(JB):                     1.82
e-28
Kurtosis:                       6.995   Cond. No.
18.0
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [111]:
```
checkVIF(X_train_new)
```

Out[111]:

	Features	VIF
0	const	26.89
4	sedan	6.06
3	hatchback	5.54
5	wagon	3.47
1	horsepower	2.50
2	carwidth	2.22
7	Highend	1.56
6	dohcv	1.21

dropping `sedan` because of high VIF value.

In [112]:
```
X_train_new = X_train_new.drop(["sedan"], axis = 1)
```

MODEL 5

In [113]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                  price   R-squared:
0.913
Model:                            OLS   Adj. R-squared:
0.909
Method:                 Least Squares   F-statistic:                     2
37.6
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           1.68
e-69
Time:                        22:45:36   Log-Likelihood:                 19
1.46
No. Observations:                 143   AIC:                            -3
68.9
Df Residuals:                     136   BIC:                            -3
48.2
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const         -0.0934      0.018     -5.219      0.000      -0.129      -
0.058
horsepower     0.5001      0.051      9.805      0.000       0.399
0.601
carwidth       0.3963      0.043      9.275      0.000       0.312
0.481
hatchback     -0.0373      0.013     -2.938      0.004      -0.062      -
0.012
wagon         -0.0170      0.017     -1.008      0.315      -0.050
0.016
dohcv         -0.3203      0.072     -4.460      0.000      -0.462      -
0.178
Highend        0.2808      0.021     13.402      0.000       0.239
0.322
==============================================================================
====
Omnibus:                       34.143   Durbin-Watson:
2.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                7
2.788
Skew:                           1.018   Prob(JB):                     1.56
e-16
Kurtosis:                       5.841   Cond. No.
16.4
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [114]:
```
checkVIF(X_train_new)
```

Out[114]:

	Features	VIF
0	const	10.82
1	horsepower	2.39
2	carwidth	2.09
6	Highend	1.55
3	hatchback	1.23
5	dohcv	1.21
4	wagon	1.11

dropping `wagon` because of high p-value.

In [115]:
```
X_train_new = X_train_new.drop(["wagon"], axis = 1)
```

MODEL 6

In [116]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                  price   R-squared:
0.912
Model:                            OLS   Adj. R-squared:
0.909
Method:                 Least Squares   F-statistic:                     2
84.8
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           1.57
e-70
Time:                        22:45:37   Log-Likelihood:                 19
0.93
No. Observations:                 143   AIC:                            -3
69.9
Df Residuals:                     137   BIC:                            -3
52.1
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const         -0.0970      0.018     -5.530      0.000      -0.132      -
0.062
horsepower     0.5013      0.051      9.832      0.000       0.401
0.602
carwidth       0.3952      0.043      9.252      0.000       0.311
0.480
hatchback     -0.0336      0.012     -2.764      0.006      -0.058      -
0.010
dohcv         -0.3231      0.072     -4.502      0.000      -0.465      -
0.181
Highend        0.2833      0.021     13.615      0.000       0.242
0.324
==============================================================================
====
Omnibus:                       36.097   Durbin-Watson:
2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                7
8.717
Skew:                           1.067   Prob(JB):                     8.07
e-18
Kurtosis:                       5.943   Cond. No.
16.3
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [117]:
```
checkVIF(X_train_new)
```

Out[117]:

	Features	VIF
0	const	10.39
1	horsepower	2.39
2	carwidth	2.08
5	Highend	1.53
4	dohcv	1.21
3	hatchback	1.13

MODEL 7

In [118]:
```
#Dropping dohcv to see the changes in model statistics
X_train_new = X_train_new.drop(["dohcv"], axis = 1)
X_train_new = build_model(X_train_new,y_train)
checkVIF(X_train_new)
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                  price   R-squared:
0.899
Model:                            OLS   Adj. R-squared:
0.896
Method:                 Least Squares   F-statistic:                     3
08.0
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           1.04
e-67
Time:                        22:45:38   Log-Likelihood:                 18
1.06
No. Observations:                 143   AIC:                            -3
52.1
Df Residuals:                     138   BIC:                            -3
37.3
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const         -0.0824      0.018     -4.480      0.000      -0.119      -
0.046
horsepower     0.4402      0.052      8.390      0.000       0.336
0.544
carwidth       0.3957      0.046      8.677      0.000       0.306
0.486
hatchback     -0.0414      0.013     -3.219      0.002      -0.067      -
0.016
Highend        0.2794      0.022     12.591      0.000       0.236
0.323
==============================================================================
====
Omnibus:                       29.385   Durbin-Watson:
1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               9
8.010
Skew:                           0.692   Prob(JB):                     5.22
e-22
Kurtosis:                       6.812   Cond. No.
12.9
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

Out[118]:

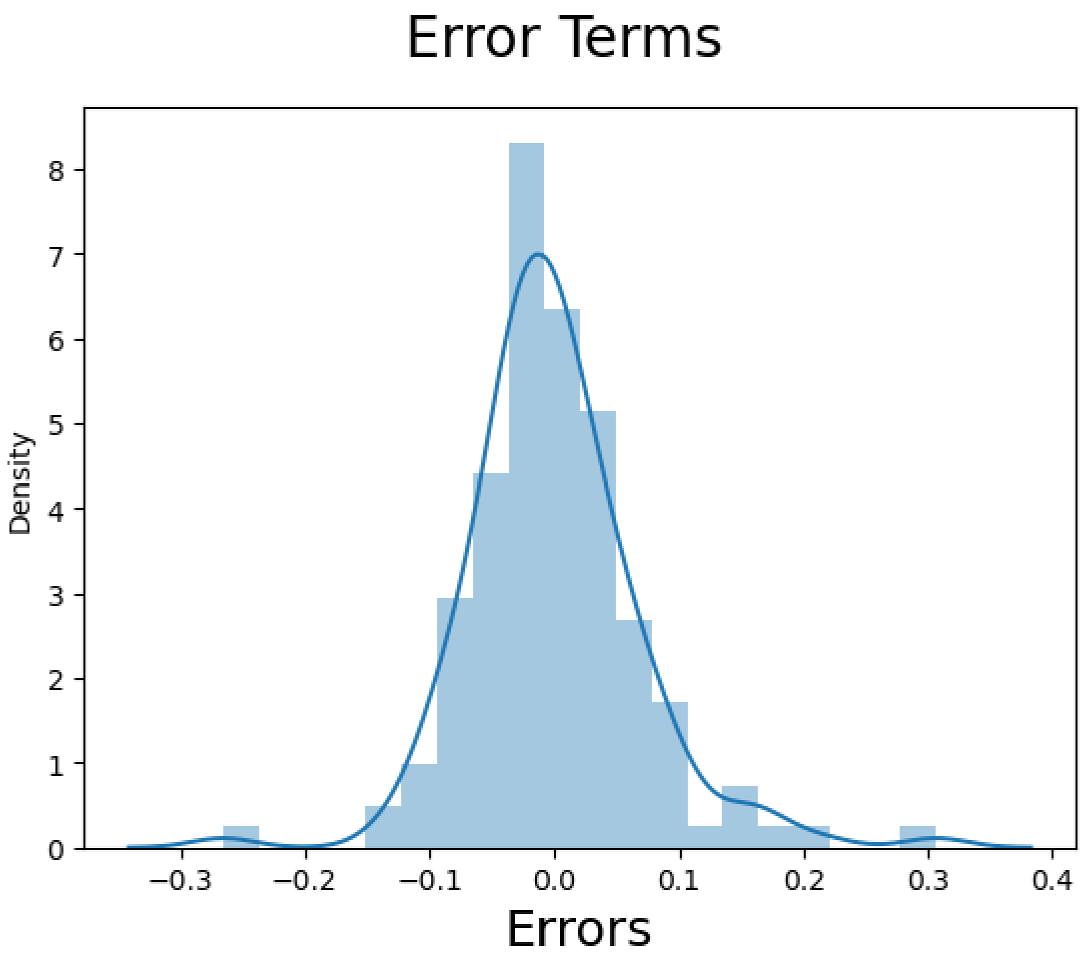
	Features	VIF
0	const	10.04
1	horsepower	2.22
2	carwidth	2.08
4	Highend	1.53
3	hatchback	1.10

Step 9 : Residual Analysis of Model

In [119]:
```
lm = sm.OLS(y_train,X_train_new).fit()
y_train_price = lm.predict(X_train_new)
```

In [120]:
```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
plt.xlabel('Errors', fontsize = 18)
```

Out[120]: Text(0.5, 0, 'Errors')



Error terms seem to be approximately normally distributed, so the assumption on the linear modeling seems to be fulfilled.

Step 10 : Prediction and Evaluation

In [121]:
```
#Scaling the test set
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepow
df_test[num_vars] = scaler.fit_transform(df_test[num_vars])
```

In [122]:
```
#Dividing into X and y
y_test = df_test.pop('price')
X_test = df_test
```

In [123]:
```
# Now let's use our model to make predictions.
X_train_new = X_train_new.drop('const',axis=1)
# Creating X_test_new dataframe by dropping variables from X_test
X_test_new = X_test[X_train_new.columns]

# Adding a constant variable
X_test_new = sm.add_constant(X_test_new)
```

In [124]:
```
# Making predictions
y_pred = lm.predict(X_test_new)
```

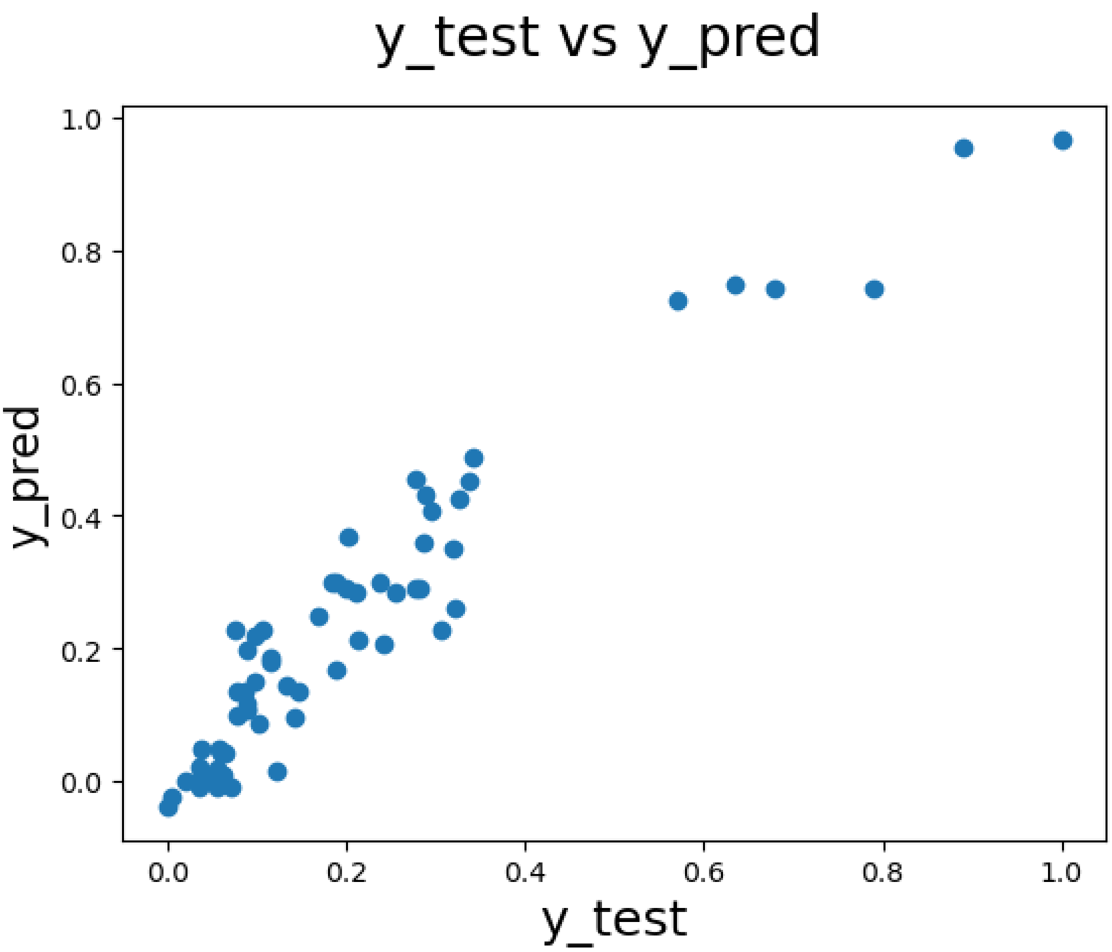
Evaluation of test via comparison of y_pred and y_test

In [125]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[125]: 0.8614595209022033

In [126]:
```
#EVALUATION OF THE MODEL
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)
```

Out[126]: Text(0, 0.5, 'y_pred')



Evaluation of the model using Statistics

In [127]:
```
print(lm.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.899
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     308.0
Date:                Thu, 15 Feb 2024   Prob (F-statistic):           1.04e-67
Time:                        22:45:44   Log-Likelihood:                 181.06
No. Observations:                 143   AIC:                            -352.1
Df Residuals:                     138   BIC:                            -337.3
Df Model:                           4                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0824      0.018     -4.480      0.000      -0.119      -0.046
horsepower     0.4402      0.052      8.390      0.000       0.336       0.544
carwidth       0.3957      0.046      8.677      0.000       0.306       0.486
hatchback     -0.0414      0.013     -3.219      0.002      -0.067      -0.016
Highend        0.2794      0.022     12.591      0.000       0.236       0.323
==============================================================================
Omnibus:                       29.385   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               98.010
Skew:                           0.692   Prob(JB):                     5.22e-22
Kurtosis:                       6.812   Cond. No.                         12.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Inference :

1. *R-sqaured and Adjusted R-squared (extent of fit)* - 0.899 and 0.896 - `90%` variance explained.

2. *F-stats and Prob(F-stats) (overall model fit)* - 308.0 and 1.04e-67(approx. 0.0) - Model fir is significant and explained `90%` variance is just not by chance.
3. *p-values* - p-values for all the coefficients seem to be less than the significance level of 0.05. - meaning that all the predictors are statistically significant.

In [ ]:

```

```

In [ ]:

```

```