Loading the data from cloud

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
import pandas as pd

df=pd.read_csv('/content/drive/MyDrive/CarPrice_Assignment.csv')
df.shape
```

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

In [ ]:
```
df.head()
```

Out[ ]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
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 [ ]:
```
df.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
```

In [ ]:

```
df.describe()
```

Out[ ]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	engi
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.9
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.6
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.0
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.0
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.0
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.0
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.0

In [ ]:

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

Out[ ]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
0	1	3	alfa-romero	gas	std	two	convertible	rwd
1	2	3	alfa-romero	gas	std	two	convertible	rwd
2	3	1	alfa-romero	gas	std	two	hatchback	rwd
3	4	2	audi	gas	std	four	sedan	fwd
4	5	2	audi	gas	std	four	sedan	4wd

5 rows × 26 columns

In [ ]:

```
df.CompanyName = df.CompanyName.str.lower()

def replace_name(a,b):
    df.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')

print(df.CompanyName.unique())
df.CompanyName.value_counts()
```

```
['alfa-romero' 'audi' 'bmw' 'chevrolet' 'dodge' 'honda' 'isuzu' 'jaguar'
 'mazda' 'buick' 'mercury' 'mitsubishi' 'nissan' 'peugeot' 'plymouth'
 'porsche' 'renault' 'saab' 'subaru' 'toyota' 'volkswagen' 'volvo']
```

Out[ ]:

```
toyota         32
nissan         18
mazda          17
mitsubishi     13
honda          13
volkswagen     12
subaru         12
peugeot        11
volvo          11
dodge           9
buick           8
bmw             8
audi            7
plymouth        7
saab            6
porsche         5
isuzu           4
jaguar          3
chevrolet       3
alfa-romero     3
renault         2
mercury         1
Name: CompanyName, dtype: int64
```

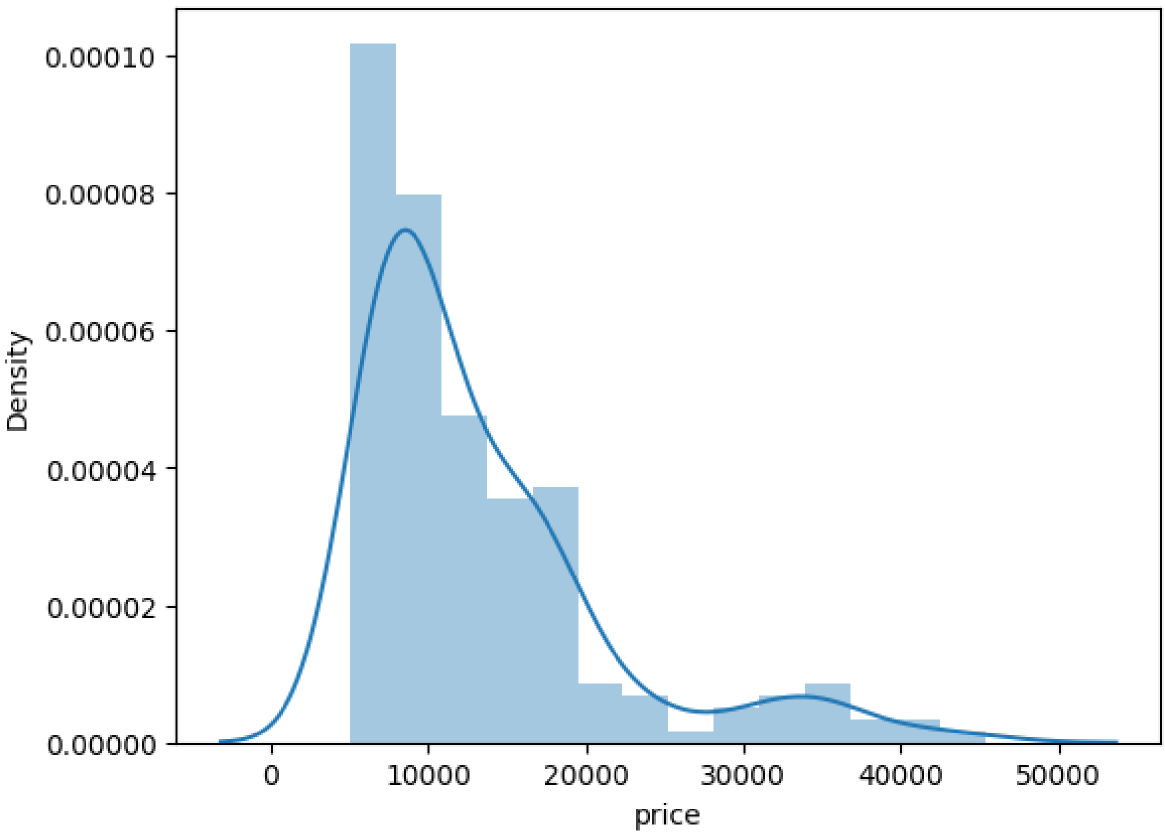
Exploratory data analysis

Histogram analysis for numerical variables

In [ ]:
```python
#Histogram for car price
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(df.price)
```

```
<ipython-input-8-3501d61547f0>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.price)
```

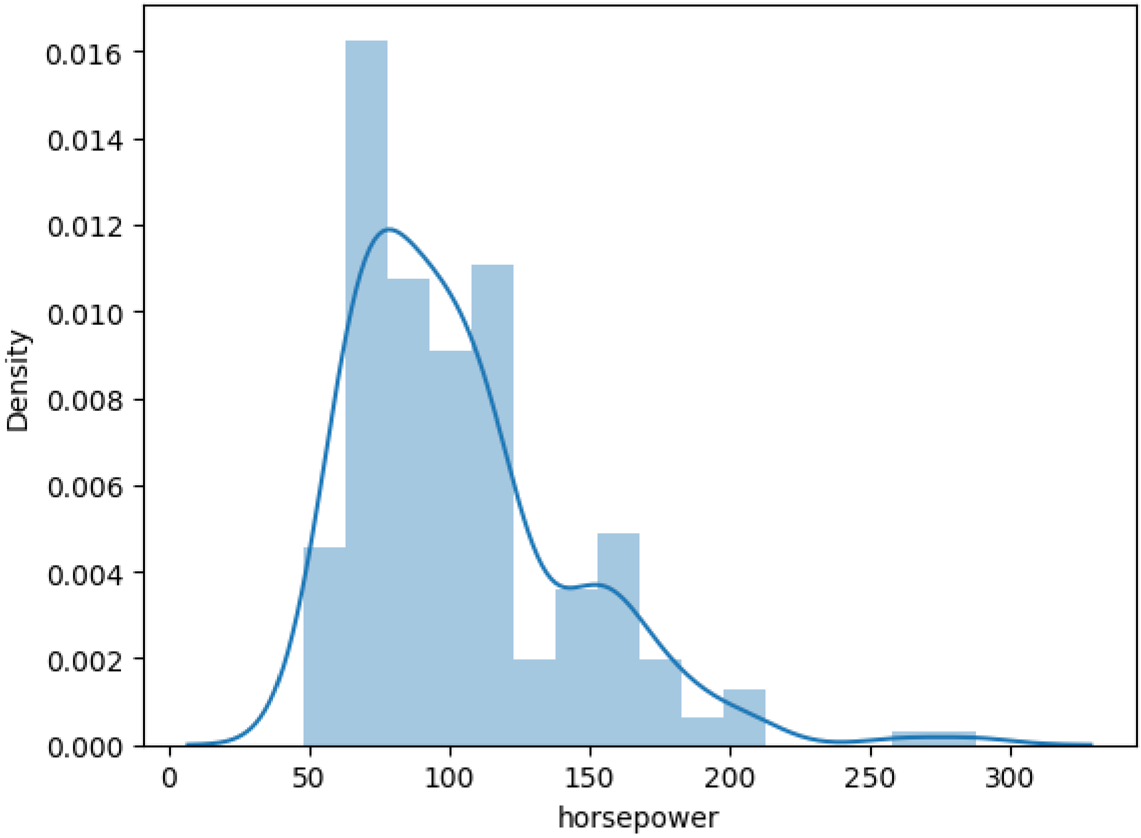
Out[ ]:
```
<Axes: xlabel='price', ylabel='Density'>
```



In [ ]:
```python
sns.distplot(df.horsepower)
```

```
<ipython-input-18-fe36dc2a0d7c>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.horsepower)
```

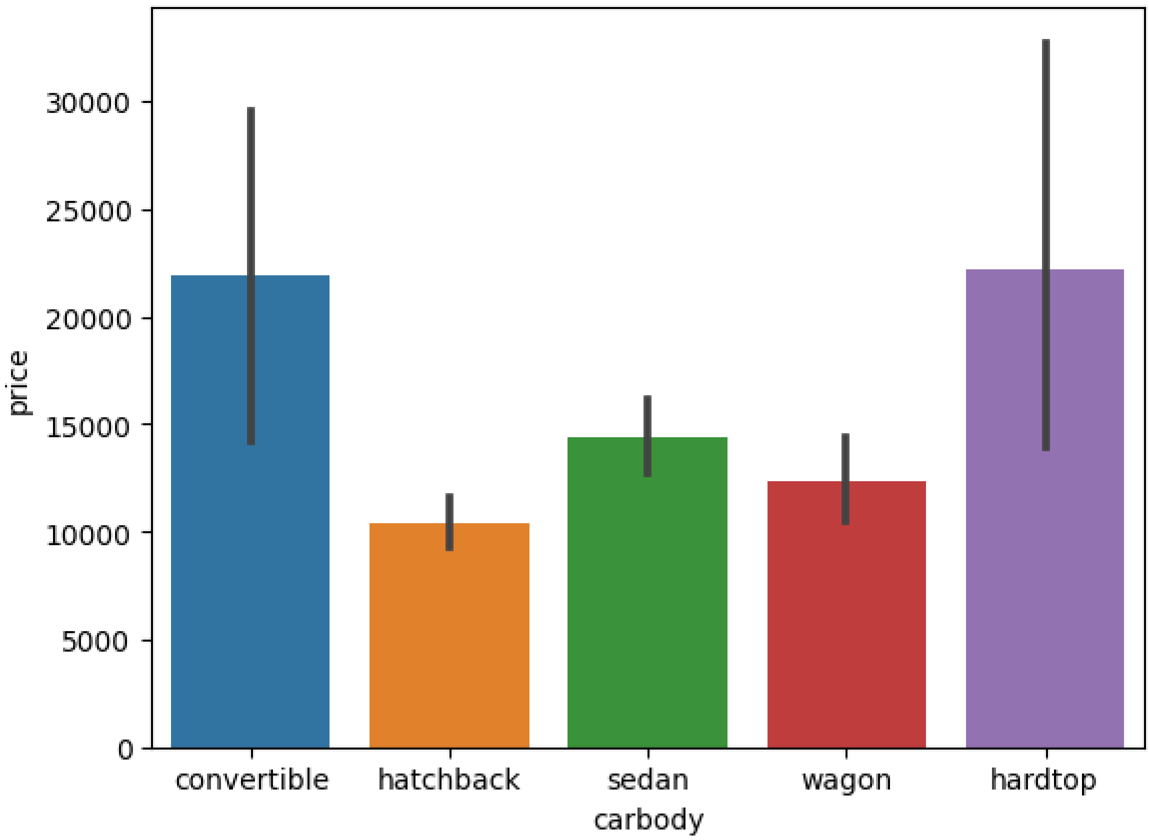
Out[ ]:
```
<Axes: xlabel='horsepower', ylabel='Density'>
```



Barplot analysis for categorical variables

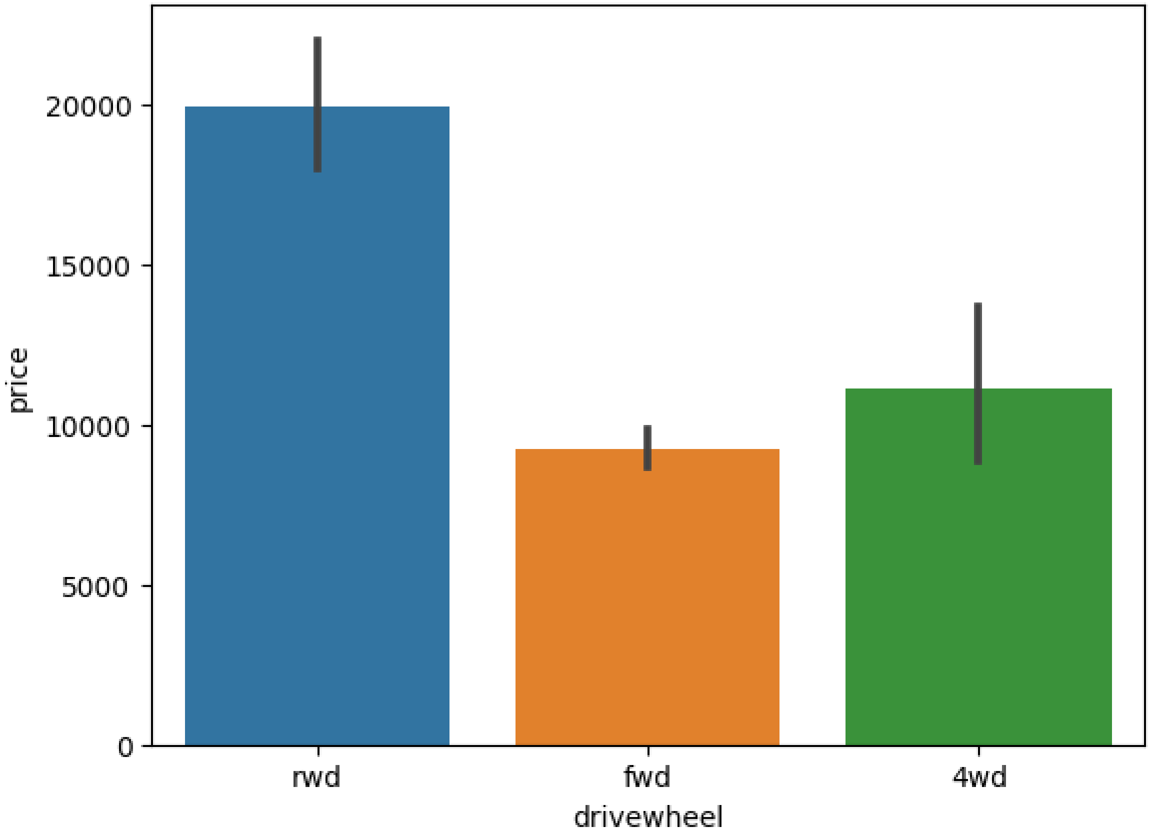
In [ ]:
```
sns.barplot(data=df,x='carbody',y='price')
```

Out[ ]:
```
<Axes: xlabel='carbody', ylabel='price'>
```



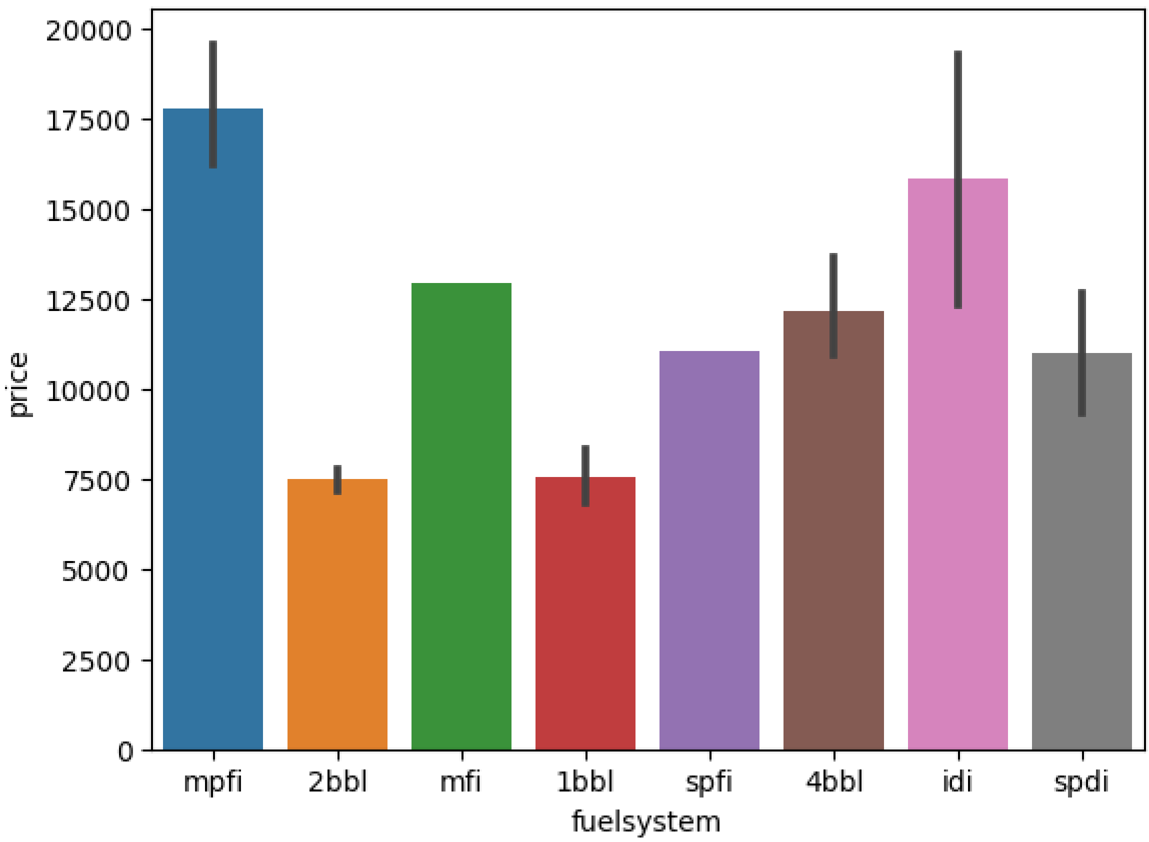
In [ ]:
```
sns.barplot(data=df,x='drivewheel',y='price')
```

Out[ ]: <Axes: xlabel='drivewheel', ylabel='price'>



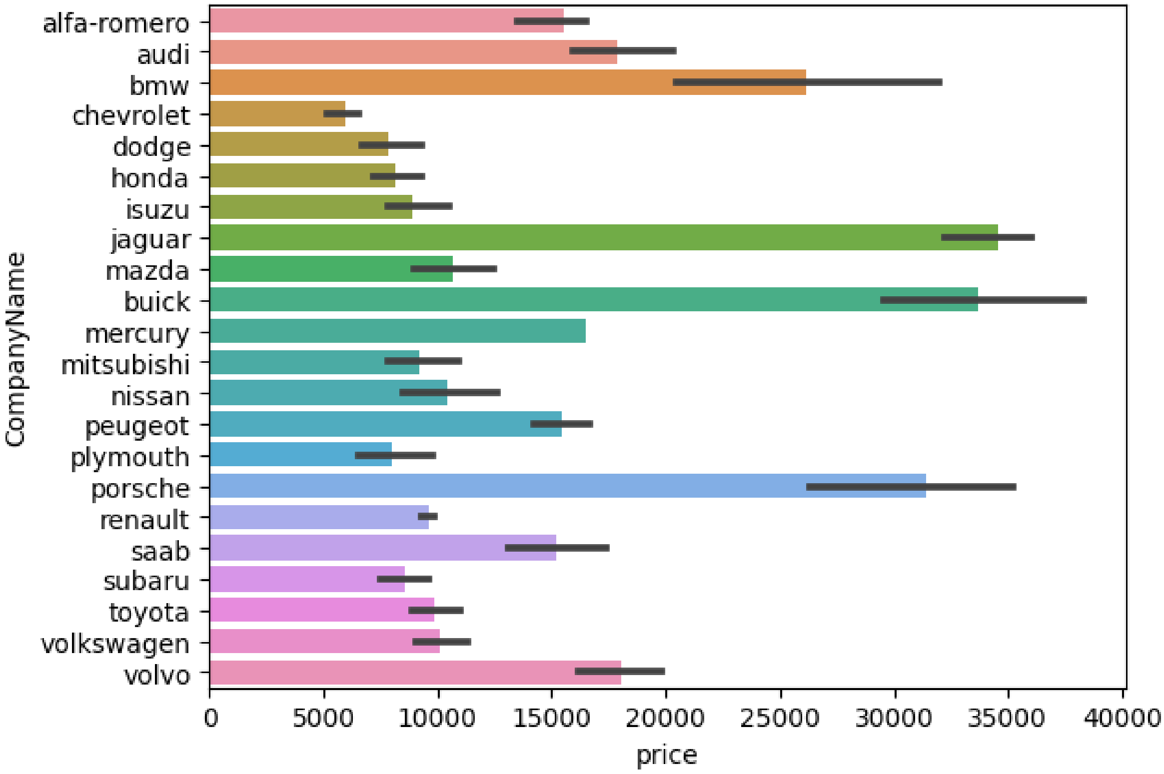
In [ ]:
```
sns.barplot(data=df,x='fuelsystem',y='price')
```

Out[ ]: <Axes: xlabel='fuelsystem', ylabel='price'>



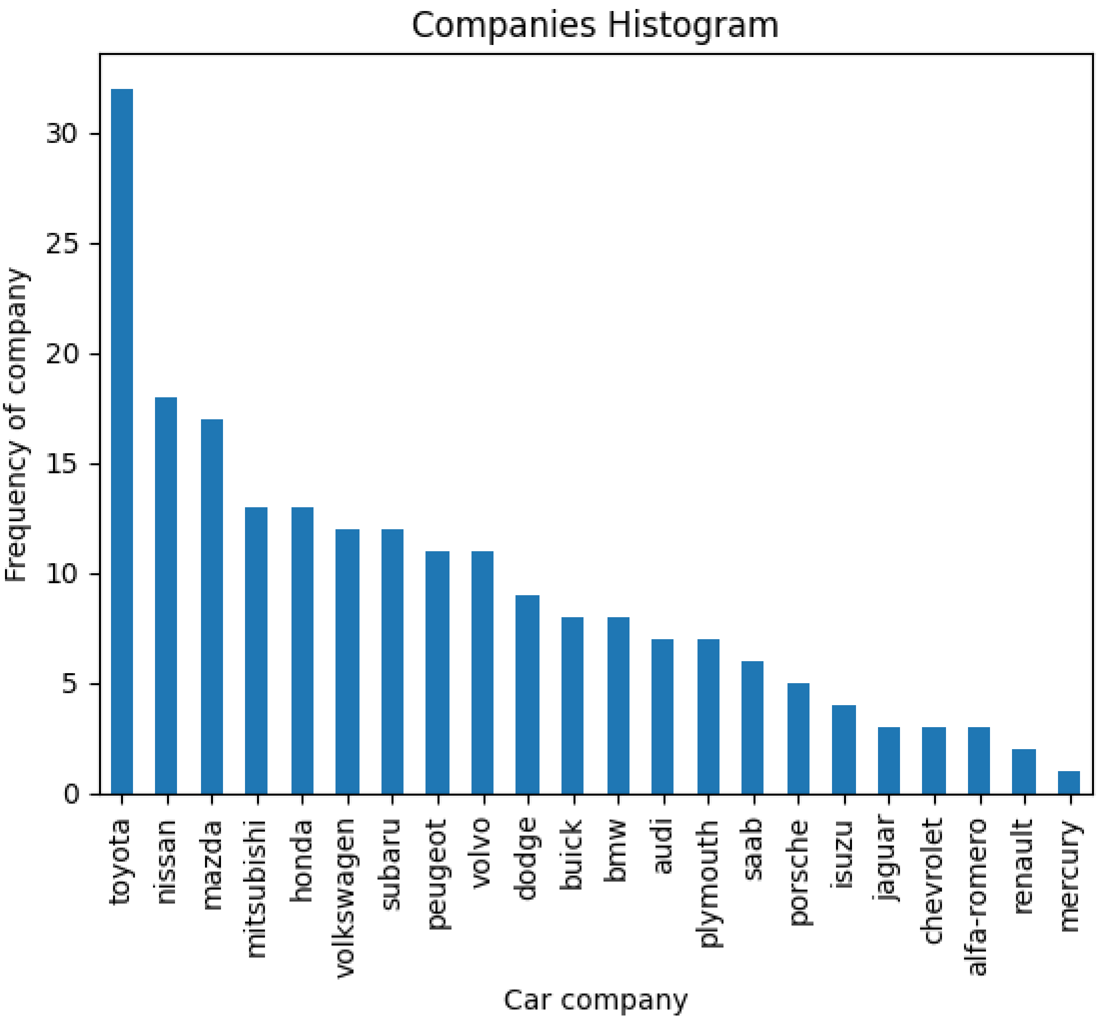
In [ ]:
```
sns.barplot(data=df,y='CompanyName',x='price',orient='h')
```

Out[ ]: <Axes: xlabel='price', ylabel='CompanyName'>



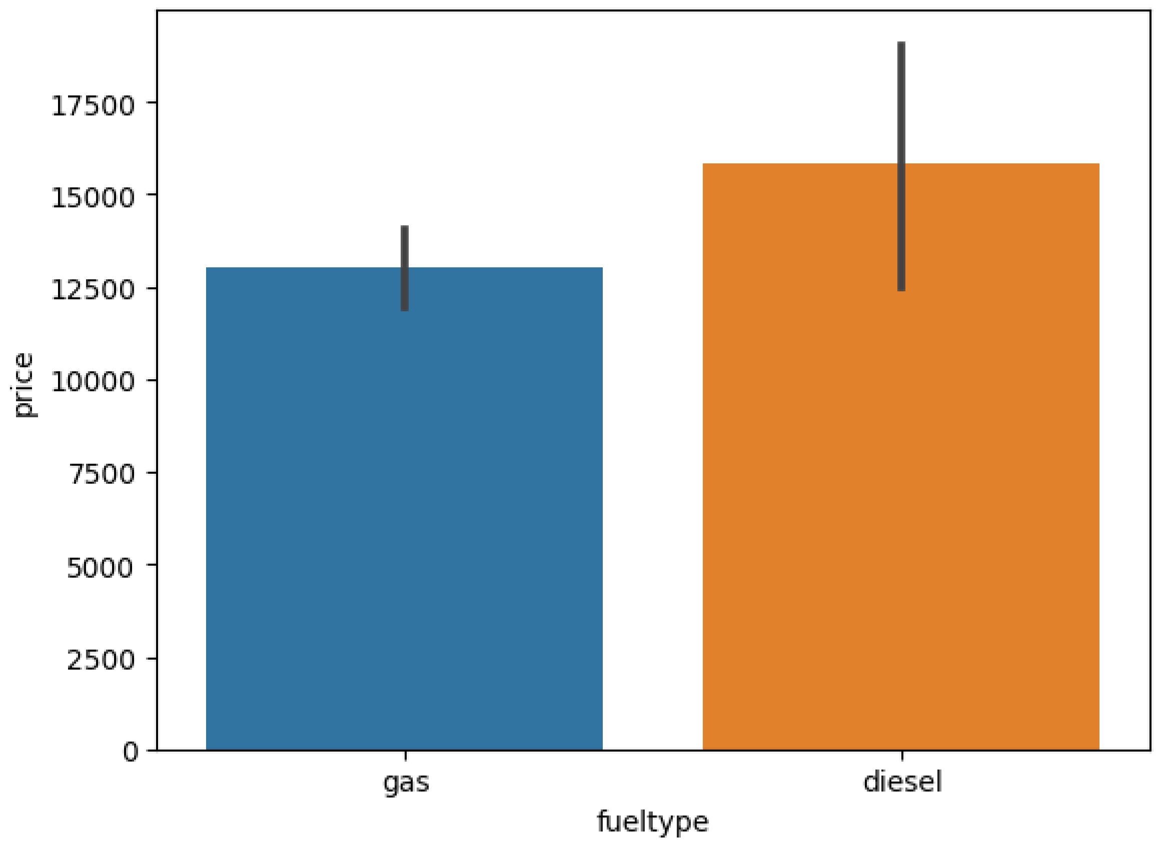
In [ ]:
```
plt01 = df.CompanyName.value_counts().plot(kind='bar')
plt.title('Companies Histogram')
plt01.set(xlabel = 'Car company', ylabel='Frequency of company')
```

Out[ ]:
```
[Text(0.5, 0, 'Car company'), Text(0, 0.5, 'Frequency of company')]
```



In [ ]:
```
sns.barplot(data=df,x='fueltype',y='price')
```

Out[ ]:
```
<Axes: xlabel='fueltype', ylabel='price'>
```

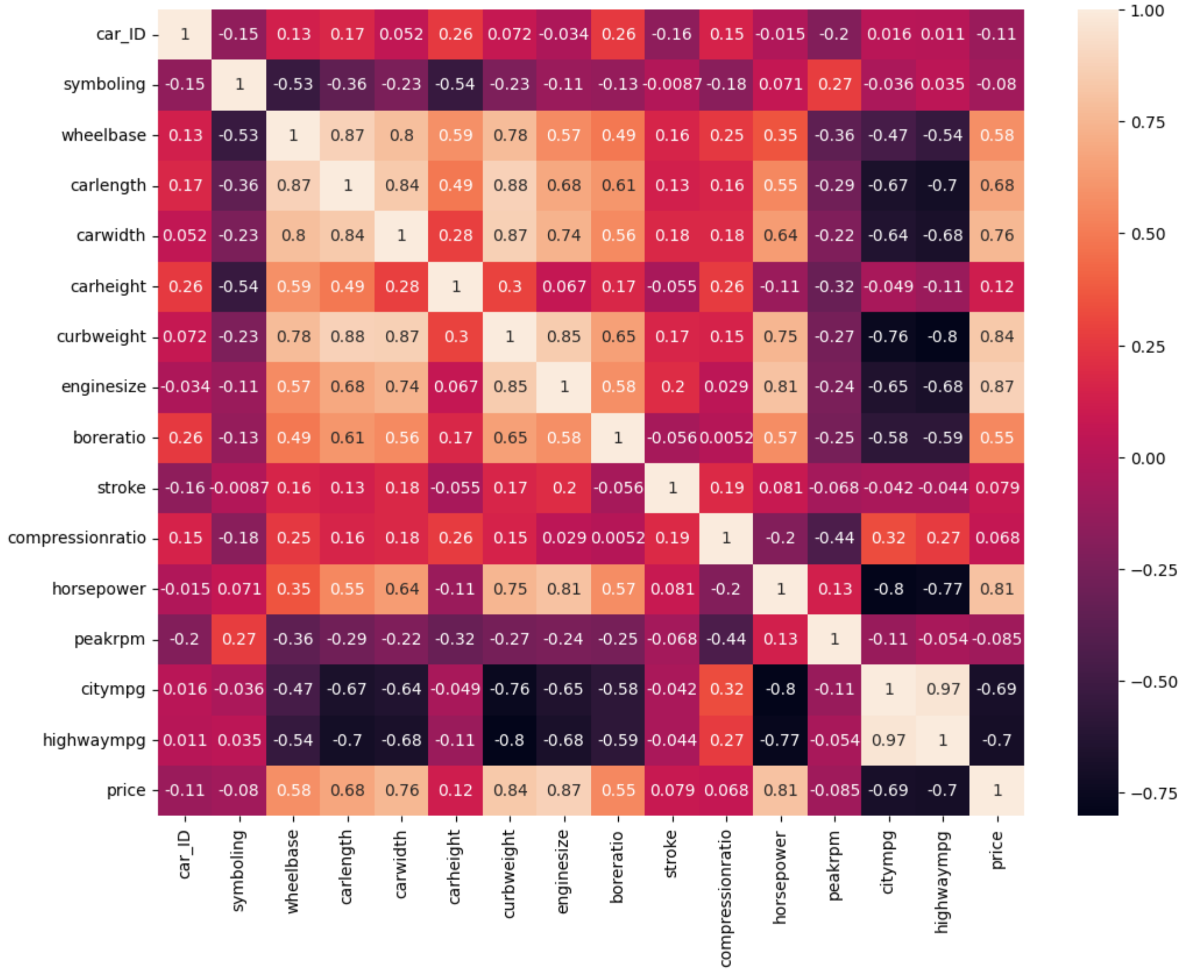


Correlation analysis using heatmap

In [ ]:
```
corr = df.corr()
plt.figure(figsize=(12, 9))
sns.heatmap(corr, annot = True)
```

```
<ipython-input-17-ca7bf234e79c>:1: FutureWarning: The default value of numeric_onl
y in DataFrame.corr is deprecated. In a future version, it will default to False.
Select only valid columns or specify the value of numeric_only to silence this war
ning.
  corr = df.corr()
```

Out[ ]:
```
<Axes: >
```



```
In [ ]: selected_cols=df[['wheelbase','carlength','carwidth','carheight','curbweight','engi

        #sns.pairplot(selected_cols)
```

Checking multicollinearity using VIF

```
In [ ]: #Defining model and checking multicollinearity of variables
        from statsmodels.stats.outliers_influence import variance_inflation_factor
        import statsmodels.api as sm
        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.shape[1])
            vif['VIF'] = round(vif['VIF'], 2)
            vif = vif.sort_values(by = "VIF", ascending = False)
            return(vif)

        checkVIF(selected_cols)
```

Out[ ]:

	Features	VIF
2	carwidth	2350.90
0	wheelbase	1903.66
1	carlength	1893.74
3	carheight	917.20
12	highwaympg	508.20
11	citympg	429.21
4	curbweight	403.02
6	boreratio	290.28
10	peakrpm	217.66
7	stroke	125.83
5	enginesize	68.77
9	horsepower	65.53
8	compressionratio	15.75

In [ ]:

```
#Dividing data into X and y variables
from sklearn.model_selection import train_test_split
y = selected_cols.pop('price')
X = selected_cols
X_train, X_test, y_train, y_test = train_test_split(X,y , random_state=104, train_s
```

In [ ]:

```
#Model1
md1=build_model(X_train,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.870
Model:                            OLS   Adj. R-squared:                  0.859
Method:                 Least Squares   F-statistic:                     77.19
Date:                Wed, 13 Sep 2023   Prob (F-statistic):           1.52e-59
Time:                        17:08:29   Log-Likelihood:                -1540.5
No. Observations:                 164   AIC:                             3109.
Df Residuals:                     150   BIC:                             3152.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
==
                       coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const            -3.894e+04    1.6e+04     -2.441      0.016   -7.05e+04   -7420.1
23
wheelbase          105.2326    105.964      0.993      0.322    -104.141     314.6
06
carlength          -64.6334     59.625     -1.084      0.280    -182.446      53.1
79
carwidth           412.0404    245.177      1.681      0.095     -72.406     896.4
86
carheight          214.9141    149.468      1.438      0.153     -80.421     510.2
49
curbweight           0.3725      1.820      0.205      0.838      -3.223       3.9
68
enginesize         152.3401     15.820      9.629      0.000     121.081     183.6
00
boreratio        -2627.9343   1382.323     -1.901      0.059   -5359.274     103.4
06
stroke           -4037.9994    904.330     -4.465      0.000   -5824.869   -2251.1
29
compressionratio   323.4195     87.106      3.713      0.000     151.305     495.5
34
horsepower          32.3381     16.266      1.988      0.049       0.197      64.4
79
peakrpm              2.3312      0.693      3.365      0.001       0.962       3.7
00
citympg           -330.7249    184.534     -1.792      0.075    -695.347      33.8
97
highwaympg         189.2060    161.815      1.169      0.244    -130.524     508.9
36
==============================================================================
Omnibus:                       17.839   Durbin-Watson:                   2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.158
Skew:                           0.068   Prob(JB):                     5.83e-16
Kurtosis:                       6.201   Cond. No.                     3.87e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
[2] The condition number is large, 3.87e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

```
checkVIF(md1)
```

Out[ ]:

	Features	VIF
0	const	4524.13
12	citympg	24.15
13	highwaympg	20.92
5	curbweight	15.42
2	carlength	9.44
10	horsepower	7.21
1	wheelbase	7.19
6	enginesize	6.71
3	carwidth	4.84
7	boreratio	2.47
4	carheight	2.29
9	compressionratio	2.24
11	peakrpm	2.04
8	stroke	1.43

Feature engineering

In [ ]:

```
#Dropping the columns with high multicollinearity
X_train_2 = X_train.drop(['citympg','highwaympg','curbweight'],axis=1)
```

In [ ]:

```
#Model2
md2=build_model(X_train_2,y_train)
checkVIF(md2)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.866
Model:                            OLS   Adj. R-squared:                  0.858
Method:                 Least Squares   F-statistic:                     99.15
Date:                Wed, 13 Sep 2023   Prob (F-statistic):           1.28e-61
Time:                        17:08:44   Log-Likelihood:                -1542.7
No. Observations:                 164   AIC:                             3107.
Df Residuals:                     153   BIC:                             3142.
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
==
                       coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
const            -4.827e+04   1.43e+04     -3.367      0.001   -7.66e+04   -1.99e+
04
wheelbase           75.7243    101.550      0.746      0.457    -124.896     276.3
45
carlength          -16.2208     53.224     -0.305      0.761    -121.370      88.9
28
carwidth           421.2682    243.569      1.730      0.086     -59.925     902.4
61
carheight          193.3858    149.489      1.294      0.198    -101.942     488.7
14
enginesize         149.8949     14.264     10.509      0.000     121.716     178.0
74
boreratio        -2226.0409   1371.615     -1.623      0.107   -4935.791     483.7
09
stroke           -3887.6179    900.697     -4.316      0.000   -5667.026   -2108.2
10
compressionratio   269.7576     71.262      3.785      0.000     128.973     410.5
42
horsepower          46.2314     14.465      3.196      0.002      17.654      74.8
09
peakrpm              2.4048      0.684      3.517      0.001       1.054       3.7
56
==============================================================================
Omnibus:                       16.333   Durbin-Watson:                   2.113
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               56.493
Skew:                           0.114   Prob(JB):                     5.40e-13
Kurtosis:                       5.866   Cond. No.                     3.10e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
[2] The condition number is large, 3.1e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Out[ ]:

	Features	VIF
0	const	3625.19
2	carlength	7.46
1	wheelbase	6.55
9	horsepower	5.66
5	enginesize	5.41
3	carwidth	4.74
6	boreratio	2.41
4	carheight	2.27
10	peakrpm	1.97
8	compressionratio	1.49
7	stroke	1.40

In [ ]:

```
X_train_3 =X_train_2.drop(['carlength'],axis=1)
```

In [ ]:

```
#Model3
md3=build_model(X_train_3,y_train)
checkVIF(md3)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.866
Model:                            OLS   Adj. R-squared:                  0.858
Method:                 Least Squares   F-statistic:                     110.8
Date:                Wed, 13 Sep 2023   Prob (F-statistic):           1.24e-62
Time:                        17:09:04   Log-Likelihood:                -1542.8
No. Observations:                 164   AIC:                             3106.
Df Residuals:                     154   BIC:                             3137.
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
==
                       coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const            -4.701e+04   1.37e+04     -3.433      0.001   -7.41e+04       -2e+
04
wheelbase           58.9255     85.037      0.693      0.489    -109.063     226.9
14
carwidth           404.4800    236.558      1.710      0.089     -62.838     871.7
98
carheight          179.2757    141.719      1.265      0.208    -100.689     459.2
40
enginesize         149.6247     14.194     10.541      0.000     121.584     177.6
65
boreratio        -2320.6541   1332.080     -1.742      0.083   -4952.162     310.8
54
stroke           -3923.4789    890.344     -4.407      0.000   -5682.343   -2164.6
15
compressionratio   271.3247     70.867      3.829      0.000     131.329     411.3
21
horsepower          45.5711     14.260      3.196      0.002      17.401      73.7
41
peakrpm              2.3988      0.681      3.520      0.001       1.053       3.7
45
==============================================================================
Omnibus:                       15.997   Durbin-Watson:                   2.107
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               54.874
Skew:                           0.092   Prob(JB):                     1.21e-12
Kurtosis:                       5.828   Cond. No.                     2.97e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
[2] The condition number is large, 2.97e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Out[ ]:

	Features	VIF
0	const	3327.46
8	horsepower	5.53
4	enginesize	5.39
1	wheelbase	4.62
2	carwidth	4.50
5	boreratio	2.29
3	carheight	2.05
9	peakrpm	1.97
7	compressionratio	1.48
6	stroke	1.38

Elimination based on multicollinearity,p-value and the correlation

In [ ]:
```
#Dropping the columns with high p-value
X_train_4 = X_train_3.drop(['wheelbase','carheight','boreratio'],axis=1)
```

In [ ]:
```
md4=build_model(X_train_4,y_train)
checkVIF(md4)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.860
Model:                            OLS   Adj. R-squared:                  0.855
Method:                 Least Squares   F-statistic:                     161.3
Date:                Wed, 13 Sep 2023   Prob (F-statistic):           1.75e-64
Time:                        17:09:18   Log-Likelihood:                -1546.3
No. Observations:                 164   AIC:                             3107.
Df Residuals:                     157   BIC:                             3128.
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
==
                       coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
const            -5.116e+04   1.13e+04     -4.527      0.000   -7.35e+04   -2.88e+
04
carwidth           583.7287    171.050      3.413      0.001     245.873     921.5
85
enginesize         149.4891     14.145     10.568      0.000     121.550     177.4
28
stroke           -3707.1353    826.909     -4.483      0.000   -5340.437   -2073.8
34
compressionratio   283.2975     71.006      3.990      0.000     143.046     423.5
49
horsepower          32.0878     13.130      2.444      0.016       6.154      58.0
21
peakrpm              2.5228      0.650      3.880      0.000       1.239       3.8
07
==============================================================================
Omnibus:                       12.372   Durbin-Watson:                   2.129
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               30.644
Skew:                           0.162   Prob(JB):                     2.22e-07
Kurtosis:                       5.093   Cond. No.                     2.42e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
[2] The condition number is large, 2.42e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Out[ ]:

	Features	VIF
0	const	2214.36
2	enginesize	5.23
5	horsepower	4.58
1	carwidth	2.30
6	peakrpm	1.75
4	compressionratio	1.45
3	stroke	1.16

In [ ]:
```
col=X_train_4.columns
X_test_1=X_test[['carwidth', 'enginesize', 'stroke', 'compressionratio', 'horsepowe
```

In [ ]:
```
# Predict using the trained model on the test set
# Making predictions
```

```python
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
fitted=lm.fit(X_train_4,y_train)
y_pred = lm.predict(X_test_1)


# You can also calculate metrics to evaluate the predictions, for example, Mean Squ
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

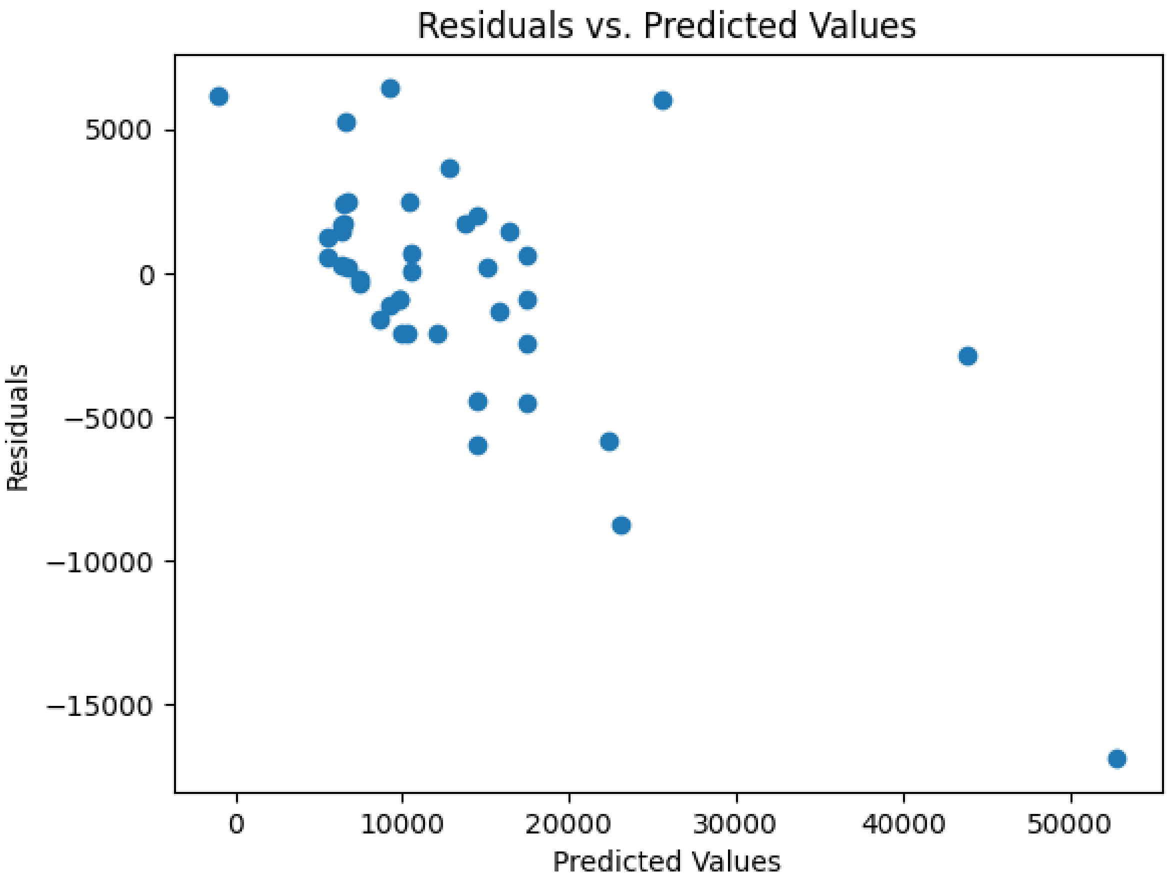
```
Mean Squared Error: 17045747.076587398
```

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

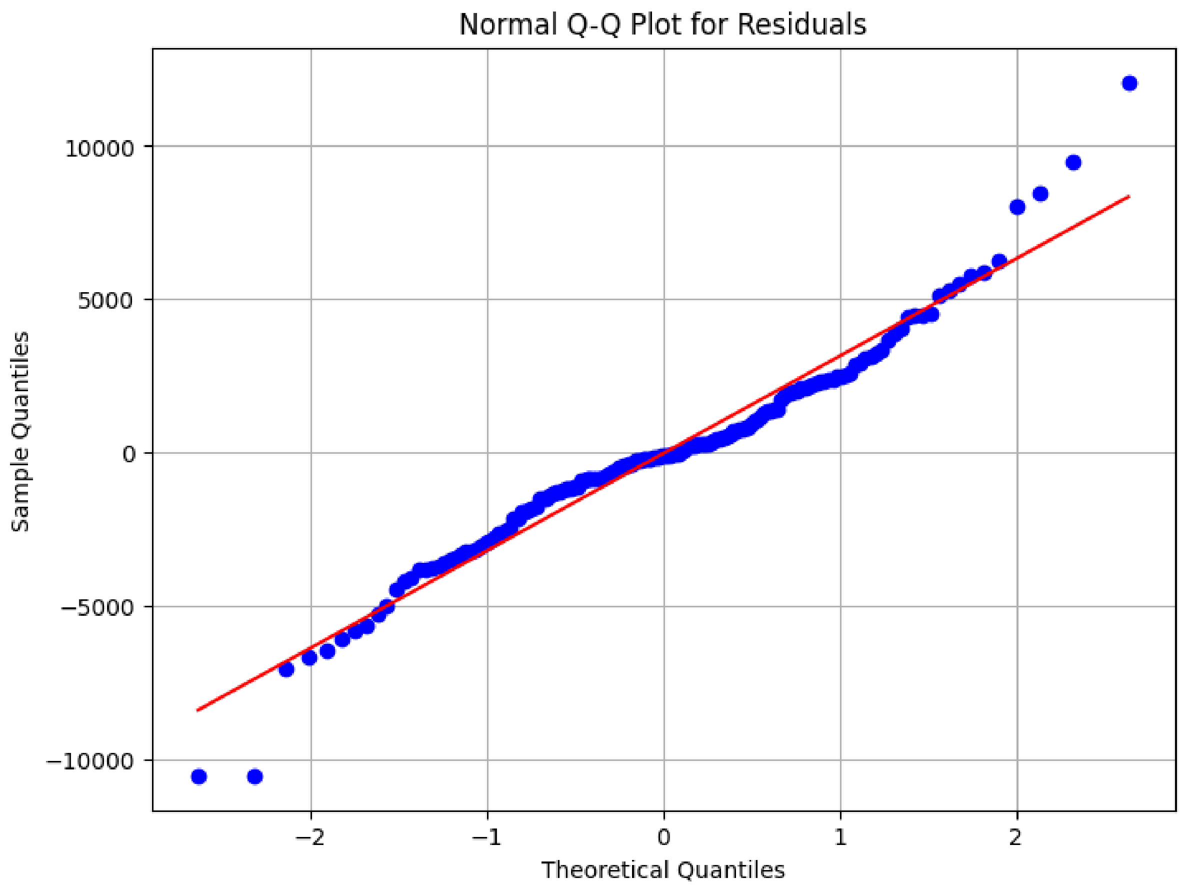
Out[ ]:
```
0.704915490316405
```

In [ ]:
```python
# Compute residuals
residuals = y_test - y_pred

# Compute Mean Squared Error (MSE) to assess model fit
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# Plot residuals vs. predicted values
plt.scatter(y_pred, residuals)
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residuals vs. Predicted Values")
plt.show()
```

```
Mean Squared Error: 17045747.076587398
```



In [ ]:
```python
import statsmodels.api as sm
import scipy.stats as stats

model = sm.OLS(y_train, X_train_4).fit()

# Get the residuals from the model
residuals = model.resid

# Create a Normal Q-Q plot for the residuals
plt.figure(figsize=(8, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("Normal Q-Q Plot for Residuals")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
plt.grid(True)
plt.show()
```



In [ ]:
```python
import numpy as np
user_input = []
for col in range(X_train_4.shape[1]):  # Iterate over the number of columns
    value = float(input(f"Enter value for {X_train_4.columns[col]}: "))
    user_input.append(value)

# Predict using the trained model
user_input_array = np.array(user_input).reshape(1, -1)  # Reshape for prediction
user_input_array_with_constant = sm.add_constant(user_input_array)  # Add constant
predicted_value = model.predict(user_input_array_with_constant)

print("Predicted car price:", predicted_value[0])
```

```
Enter value for carwidth: 64
Enter value for enginesize: 130
Enter value for stroke: 2.2
Enter value for compressionratio: 8
Enter value for horsepower: 100
Enter value for peakrpm: 4500
Predicted car price: 16873.61509462775
```

In [ ]: