CAR PRICE PREDICTION

- Numerous factors contribute to a car's price, encompassing brand reputation, car features, horsepower, mileage efficiency, and more.
- Car price prediction stands as a significant domain within machine learning research.
- If you seek to master the art of training a car price prediction model, this project presents a valuable learning opportunity.

In [60]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns
import matplotlib.pyplot as plt
```

In [61]:
```
# Load the dataset
data = pd.read_csv('CarPrice_Assignment.csv')
data
```

Out[61]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuel
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	
...	...	...	...	...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	

205 rows × 26 columns

In [62]:
```
# Display the first few rows of the dataset
data.head()
```

Out[62]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsy
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [63]:
```
# Display the last few rows of the dataset
data.tail()
```

Out[63]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsyst
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	n
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	n
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	n
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	n

5 rows × 26 columns

In [64]:
```
# Check for missing values
data.isnull().sum()
```

Out[64]:
```
car_ID            0
symboling         0
CarName           0
fueltype          0
aspiration        0
doornumber        0
carbody           0
drivewheel        0
enginelocation    0
wheelbase         0
carlength         0
carwidth          0
carheight         0
curbweight        0
enginetype        0
cylindernumber    0
enginesize        0
fuelsystem        0
boreratio         0
stroke            0
compressionratio  0
horsepower        0
peakrpm           0
citympg           0
highwaympg        0
price             0
dtype: int64
```

In [65]:
```
# Handle missing values.
data.dropna()
```

Out[65]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuel
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	
...	...	...	...	...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	

205 rows × 26 columns

In [66]:
```
# Check data types of the columns
print(data.dtypes)
```

```
car_ID              int64
symboling           int64
CarName            object
fueltype           object
aspiration         object
doornumber         object
carbody            object
drivewheel         object
enginelocation     object
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
enginetype         object
cylindernumber     object
enginesize          int64
fuelsystem         object
boreratio         float64
stroke            float64
compressionratio  float64
horsepower          int64
peakrpm             int64
citympg             int64
highwaympg          int64
price             float64
dtype: object
```

In [67]:
```
# Identify categorical features
categorical_features = [feature for feature in data.columns if data[feature].dtype=='O']
print(f'Categorical features: {categorical_features}')
```

```
Categorical features: ['CarName', 'fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'enginelocat
ion', 'enginetype', 'cylindernumber', 'fuelsystem']
```

In [68]:
```
# Identify numerical features
numerical_features = [feature for feature in data.columns if data[feature].dtype!='O']
print(f'Numerical features: {numerical_features}')
```

```
Numerical features: ['car_ID', 'symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight', 'e
nginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price
']
```

In [69]:
```
# Convert categorical variables to numerical values using one-hot encoding
data = pd.get_dummies(data, drop_first=True)
```

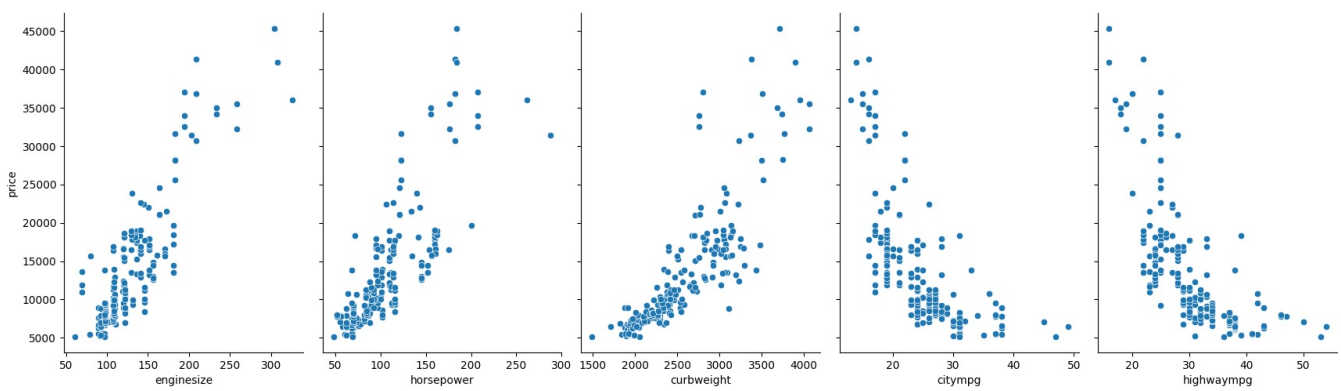
In [70]:
```
# Display the columns of the modified dataset
print(data.columns)
```

```
Index(['car_ID', 'symboling', 'wheelbase', 'carlength', 'carwidth',
       'carheight', 'curbweight', 'enginesize', 'boreratio', 'stroke',
       ...
       'cylindernumber_three', 'cylindernumber_twelve', 'cylindernumber_two',
       'fuelsystem_2bbl', 'fuelsystem_4bbl', 'fuelsystem_idi',
       'fuelsystem_mfi', 'fuelsystem_mpfi', 'fuelsystem_spdi',
       'fuelsystem_spfi'],
      dtype='object', length=191)
```

In [71]:
```
# Define features and target variable
X = data.drop(columns=['price'])
y = data['price']
```

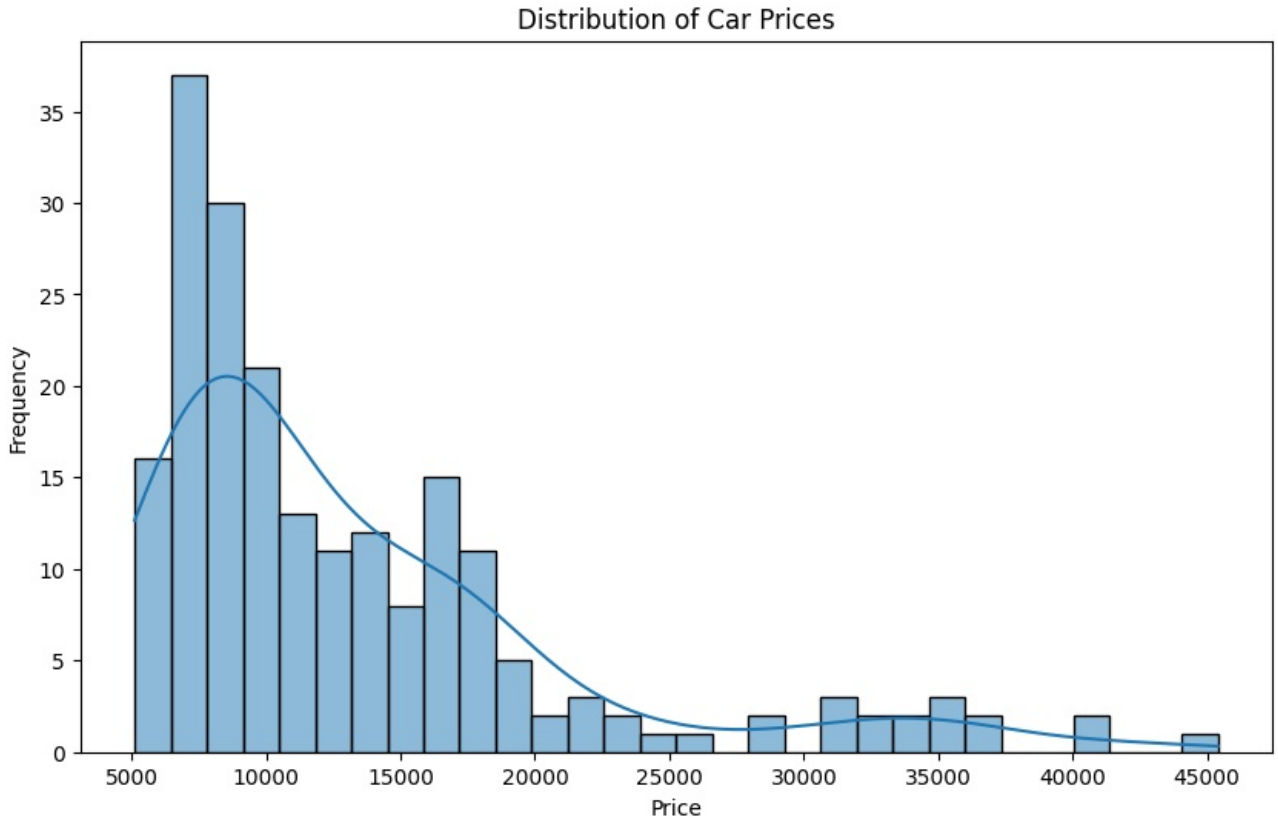
In [72]:
```
# Combine features and target variable for visualization purposes
data_combined = data.copy()
data_combined['price'] = y
```

In [73]:
```
# Pair Plot
sns.pairplot(data_combined, x_vars=['enginesize', 'horsepower', 'curbweight', 'citympg', 'highwaympg'], y_vars=
plt.show()
```

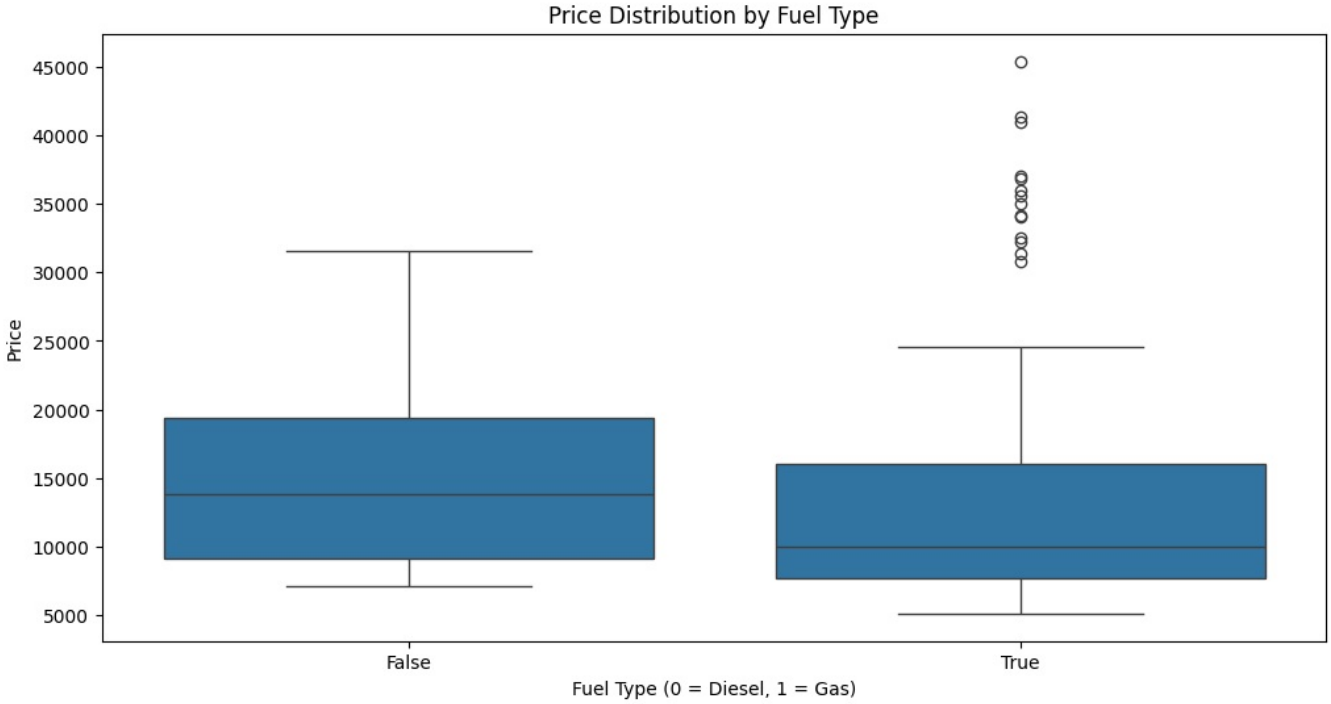


In [74]:
```
# Distribution Plot for Price
plt.figure(figsize=(10, 6))
```

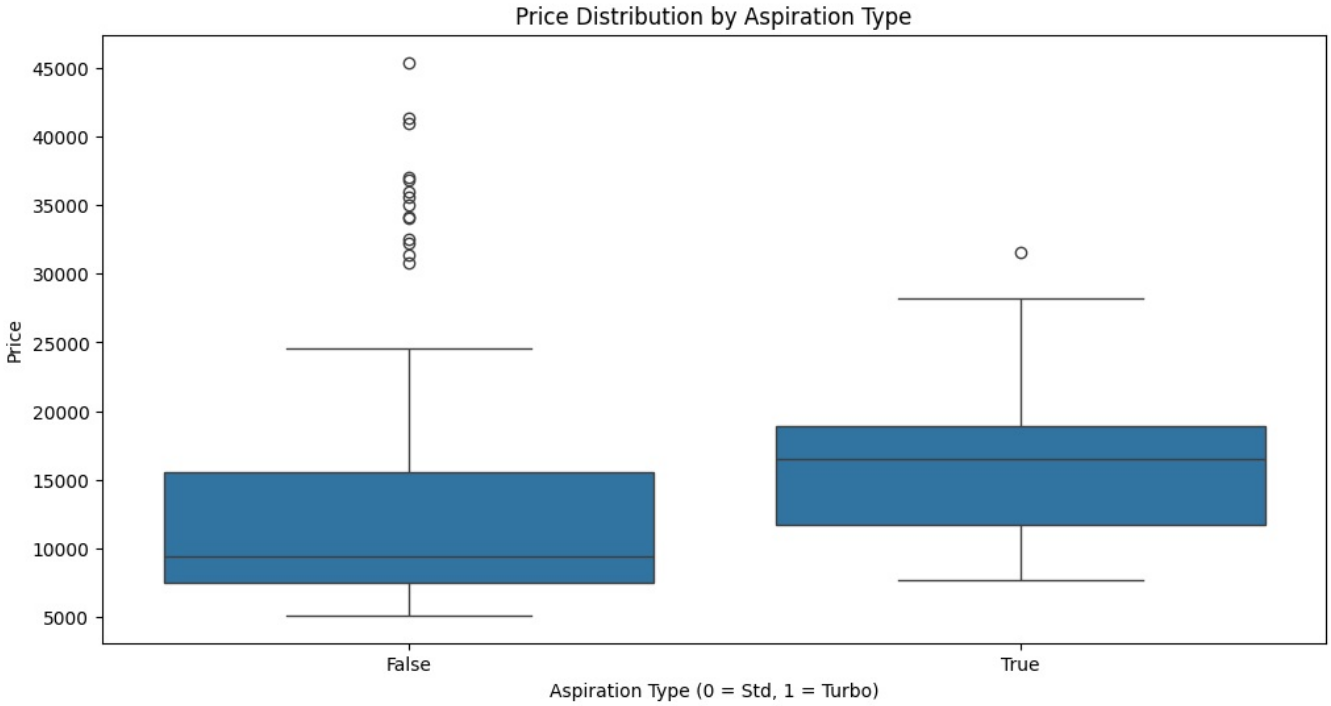
```
sns.histplot(y, bins=30, kde=True)
plt.title('Distribution of Car Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



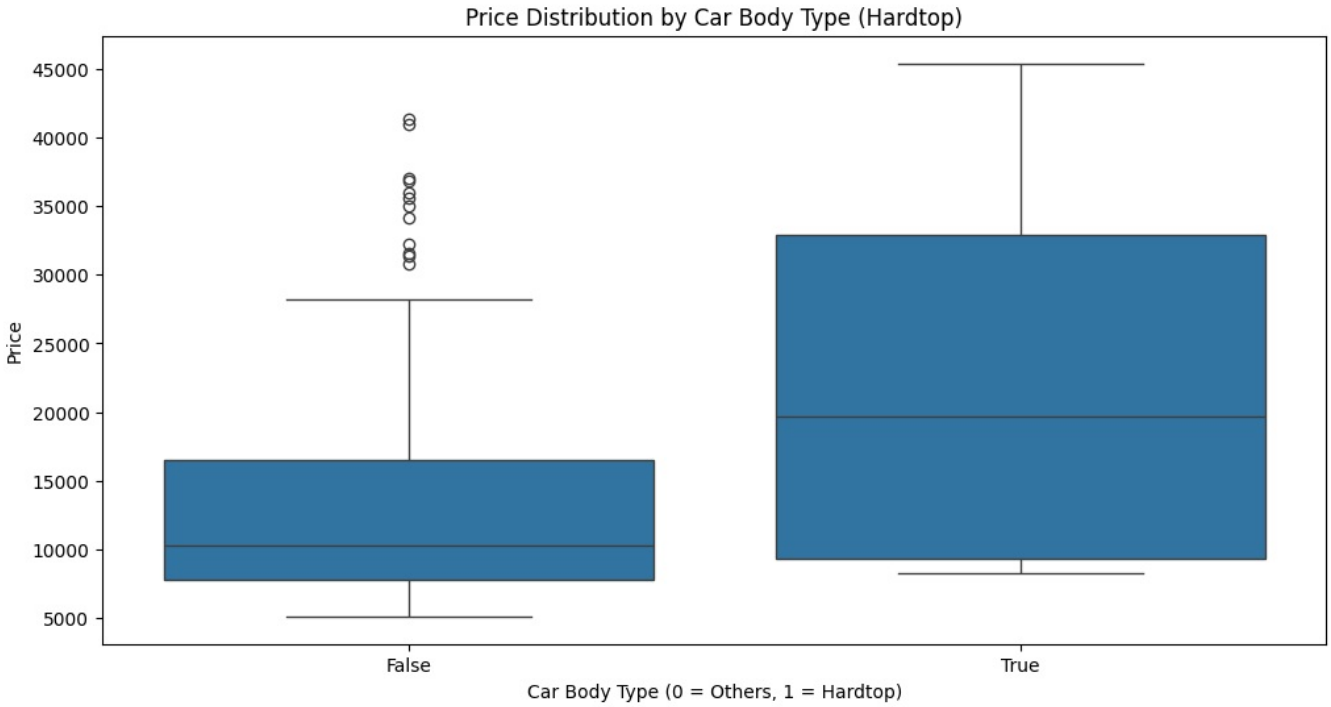
In [75]:
```
# Box Plot for categorical features
plt.figure(figsize=(12, 6))
sns.boxplot(x='fueltype_gas', y='price', data=data_combined)
plt.title('Price Distribution by Fuel Type')
plt.xlabel('Fuel Type (0 = Diesel, 1 = Gas)')
plt.ylabel('Price')
plt.show()
```



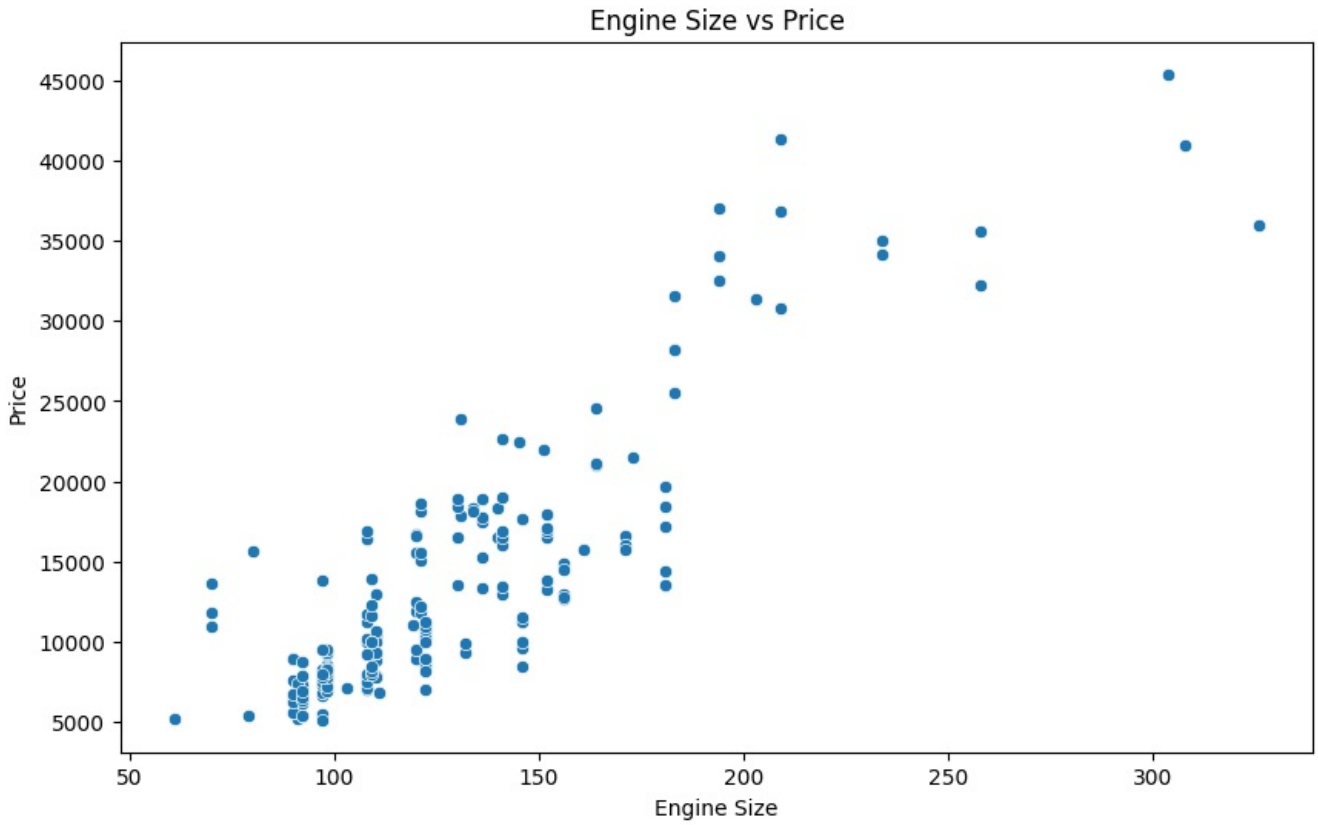
In [76]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='aspiration_turbo', y='price', data=data_combined)
plt.title('Price Distribution by Aspiration Type')
plt.xlabel('Aspiration Type (0 = Std, 1 = Turbo)')
plt.ylabel('Price')
plt.show()
```



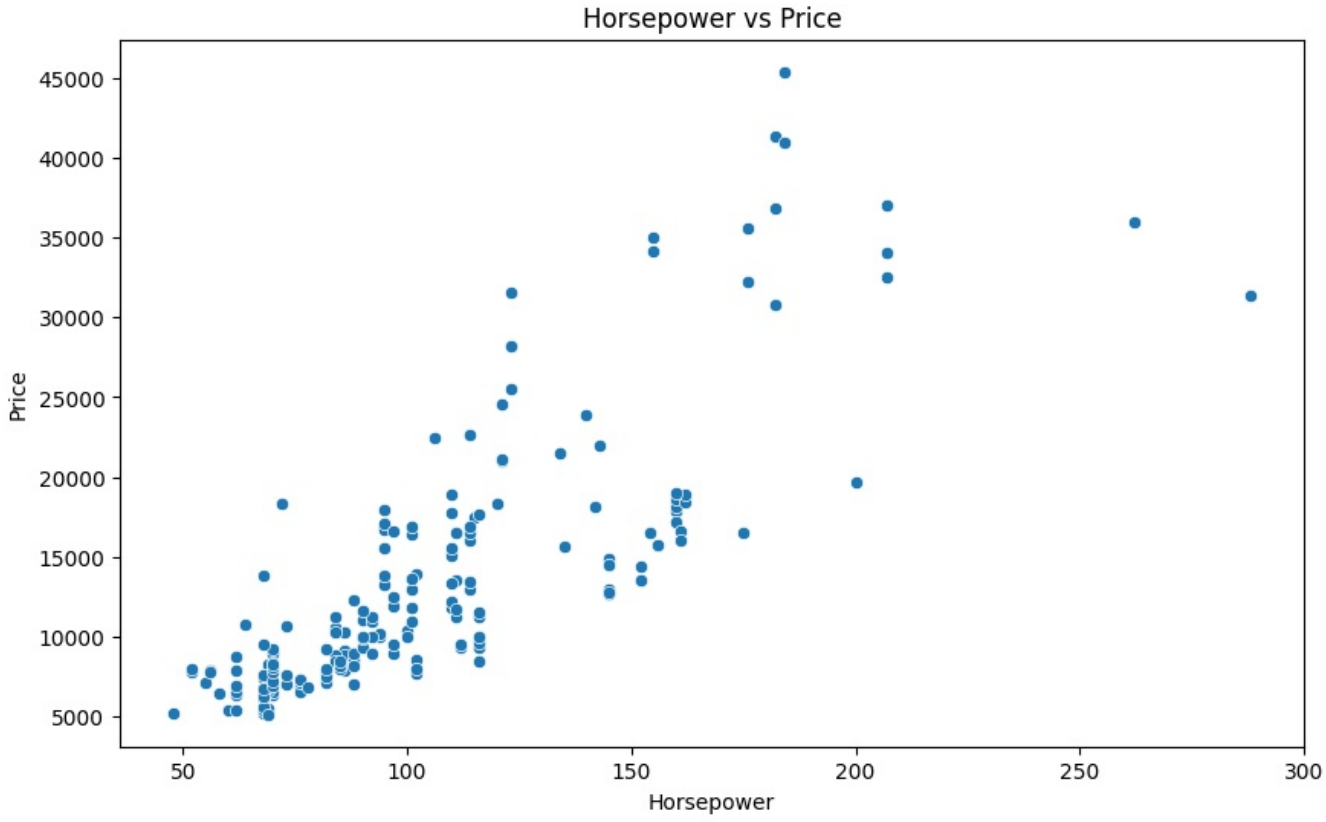
In [77]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='carbody_hardtop', y='price', data=data_combined)
plt.title('Price Distribution by Car Body Type (Hardtop)')
plt.xlabel('Car Body Type (0 = Others, 1 = Hardtop)')
plt.ylabel('Price')
plt.show()
```



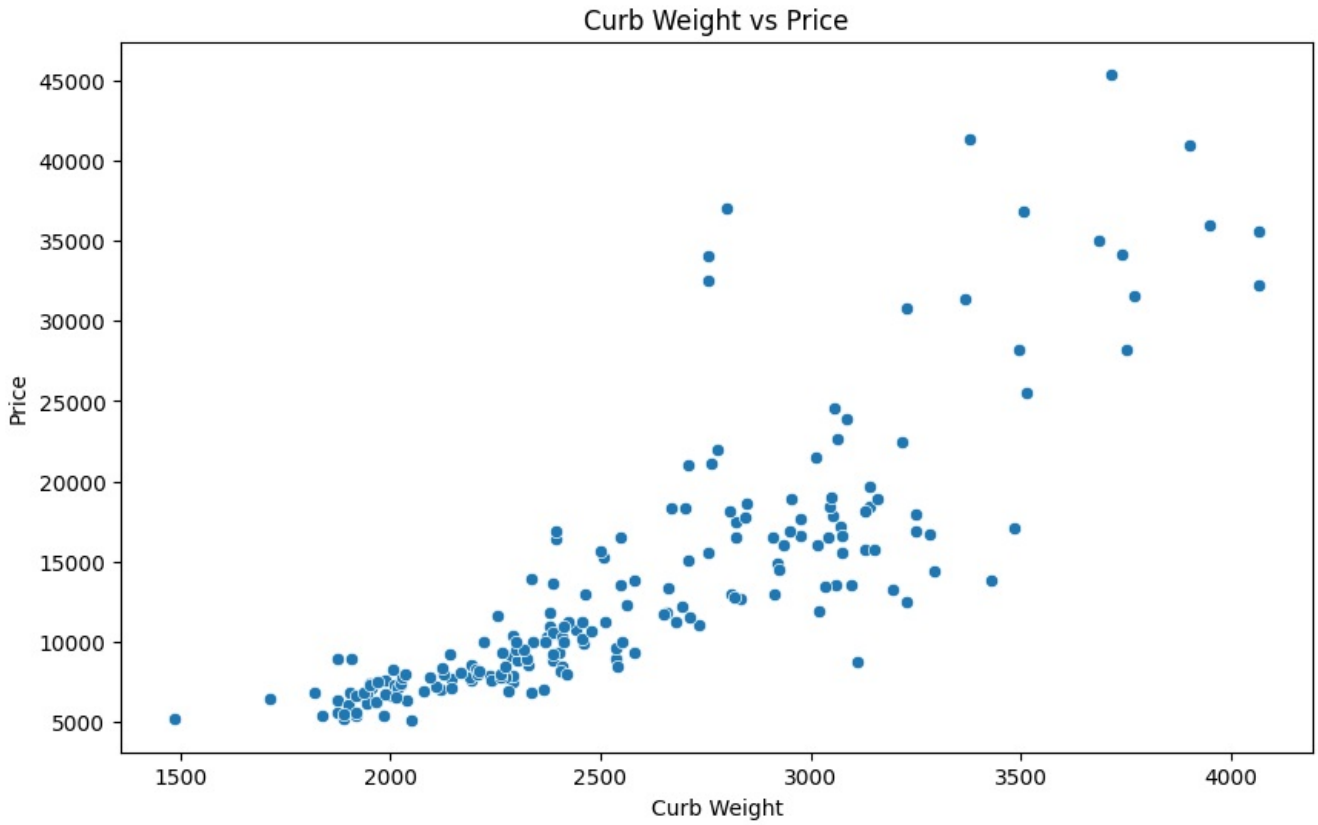
In [78]:
```
# Scatter Plot for key numerical features vs Price
plt.figure(figsize=(10, 6))
sns.scatterplot(x='enginesize', y='price', data=data_combined)
plt.title('Engine Size vs Price')
plt.xlabel('Engine Size')
plt.ylabel('Price')
plt.show()
```



In [79]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='horsepower', y='price', data=data_combined)
plt.title('Horsepower vs Price')
plt.xlabel('Horsepower')
plt.ylabel('Price')
plt.show()
```



In [80]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='curbweight', y='price', data=data_combined)
plt.title('Curb Weight vs Price')
plt.xlabel('Curb Weight')
plt.ylabel('Price')
plt.show()
```



In [81]:
```
X.head()
```

Out[81]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	...	cylindernumber_three	cylinder
0	1	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	...	False	
1	2	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	...	False	
2	3	1	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	...	False	
3	4	2	99.8	176.6	66.2	54.3	2337	109	3.19	3.40	...	False	
4	5	2	99.4	176.6	66.4	54.3	2824	136	3.19	3.40	...	False	

5 rows × 190 columns

In [82]:
```
y
```

Out[82]:
```
0      13495.0
1      16500.0
2      16500.0
3      13950.0
4      17450.0
        ...
200    16845.0
201    19045.0
202    21485.0
203    22470.0
204    22625.0
Name: price, Length: 205, dtype: float64
```

In [83]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [84]:
```
# Initialize the Linear Regression model
model = LinearRegression()
```

In [85]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[85]:
```
LinearRegression()
```

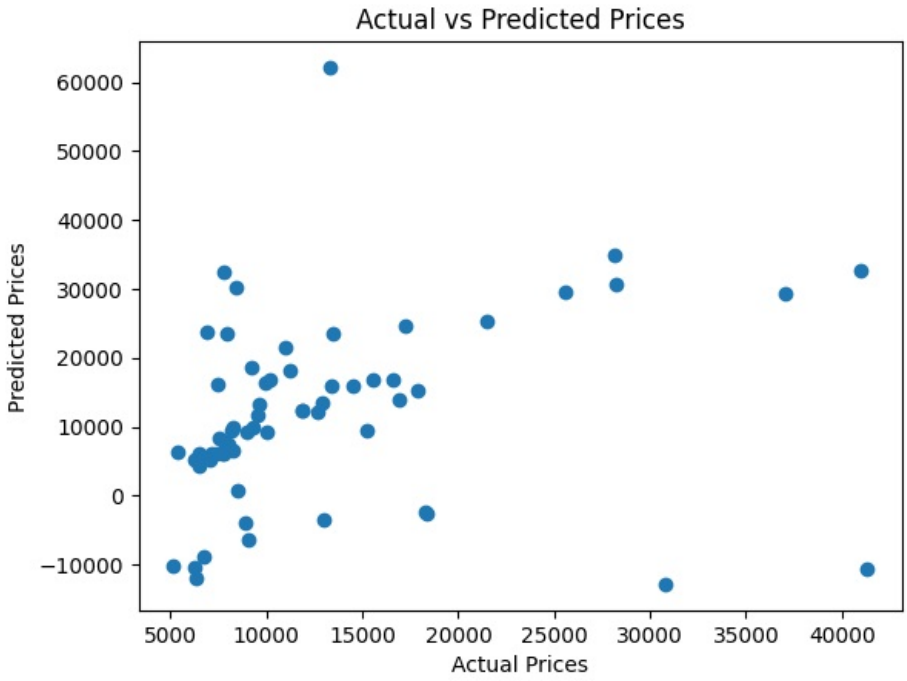
In [86]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
y_pred
```

Out[86]:
```
array([-12866.80133639,  15295.33767985,  11656.14223366,  12431.08997067,
        30684.16361391,   7866.2369073 ,   6136.1642045 ,   9840.6456589 ,
        16733.12471845,  32539.95440296,  62171.32473088,   6606.89890106,
        -2335.85827368,   9220.80861713,  32621.37231516,   6004.46838555,
       -10139.81311516,  12090.35434385,   9358.03337996,  16354.05570077,
          670.54637643,  23419.54489932,   9811.08872142,   4362.77798861,
        -8816.49636953, -10753.66003163,  13210.98896729,  15882.74865296,
         7391.6632841 ,  13532.42793038,  29537.50805037,   5162.97554505,
        23614.23764701,  25207.72242776,   5988.33054114,  34840.86882082,
        18114.64675501,  21618.89056114,  -3943.58076553,  15862.98807437,
        16071.46104951,  -2640.05783767,  16770.62107632,  23795.60474744,
         8358.90385882,  -6517.47558973, -10342.40840504,   5229.55435137,
        13956.83749147,  -3458.63351873, -12023.39307112,  16714.64954235,
         6029.25559781,   9227.68943633,   6118.5028523 ,   9481.92485867,
        30136.03116484,  12241.2569999 ,  29282.07859977,   6372.98002153,
        18516.24994161,  24570.48536687])
```

In [87]:
```
# Calculate the Mean Squared Error (MSE) and R^2 score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
print(f'R^2 Score: {r2}')
```

```
Mean Squared Error: 196438429.48
R^2 Score: -1.8352484848794606
```

In [88]:
```
# Plot the actual vs predicted prices
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted Prices')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: