Name: Amogh Javali

ID: NN/22/2355

Role: Data Analytics

In [4]:
```
!pip3 install pandas numpy
```

```
Requirement already satisfied: pandas in c:\users\amogh\appdata\local\programs\python\python312\lib\site-package
s (2.2.2)
Requirement already satisfied: numpy in c:\users\amogh\appdata\local\programs\python\python312\lib\site-packages
(1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\amogh\appdata\local\programs\python\python312\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\amogh\appdata\local\programs\python\python312\lib\site-p
ackages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\amogh\appdata\local\programs\python\python312\lib\site
-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\amogh\appdata\local\programs\python\python312\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
[notice] A new release of pip is available: 24.1.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
import pandas as pd
```

Data Reading

In [3]:
```
df = pd.read_csv("CarPrice_Assignment.csv")
```

Inspecting Data

In [4]:
```
df
```

Out[4]:

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

205 rows × 26 columns

In [5]:
```
df.head()
```

Out[5]:

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

5 rows × 26 columns

In [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke
count	205.000000	205.000000	205.000000	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	126.907317	3.329756	3.255415
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000

Data Cleaning

Checking for null values and Percentage

In [8]:
```
df.isnull()
```

Out[8]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesiz
0	False	False	False	False	False	False	False	False	False	False	...	Fals
1	False	False	False	False	False	False	False	False	False	False	...	Fals
2	False	False	False	False	False	False	False	False	False	False	...	Fals
3	False	False	False	False	False	False	False	False	False	False	...	Fals
4	False	False	False	False	False	False	False	False	False	False	...	Fals
...	...	...	...	...	...	...	...	...	...	...	...	
200	False	False	False	False	False	False	False	False	False	False	...	Fals
201	False	False	False	False	False	False	False	False	False	False	...	Fals
202	False	False	False	False	False	False	False	False	False	False	...	Fals
203	False	False	False	False	False	False	False	False	False	False	...	Fals
204	False	False	False	False	False	False	False	False	False	False	...	Fals

205 rows × 26 columns

In [9]:
```
null_counts = df.isnull().sum()
```

In [10]:
```
null_counts
```

Out[10]:
```
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 [11]:
```
null_percentage = (df.isnull().sum() / len(df)) * 100
```

In [12]:
```
null_percentage
```

Out[12]:
```
car_ID              0.0
symboling           0.0
CarName             0.0
fueltype            0.0
aspiration          0.0
doornumber          0.0
carbody             0.0
drivewheel          0.0
enginelocation      0.0
wheelbase           0.0
carlength           0.0
carwidth            0.0
carheight           0.0
curbweight          0.0
enginetype          0.0
cylindernumber      0.0
enginesize          0.0
fuelsystem          0.0
boreratio           0.0
stroke              0.0
compressionratio    0.0
horsepower          0.0
peakrpm             0.0
citympg             0.0
highwaympg          0.0
price               0.0
dtype: float64
```

In [13]:
```
null_df = pd.DataFrame({'Null Count': null_counts, 'Null Percentage': null_percentage})
```

In [14]:
```
null_df
```

Out[14]:

	Null Count	Null Percentage
car_ID	0	0.0
symboling	0	0.0
CarName	0	0.0
fueltype	0	0.0
aspiration	0	0.0
doornumber	0	0.0
carbody	0	0.0
drivewheel	0	0.0
enginelocation	0	0.0
wheelbase	0	0.0
carlength	0	0.0
carwidth	0	0.0
carheight	0	0.0
curbweight	0	0.0
enginetype	0	0.0
cylindernumber	0	0.0
enginesize	0	0.0
fuelsystem	0	0.0
boreratio	0	0.0
stroke	0	0.0
compressionratio	0	0.0
horsepower	0	0.0
peakrpm	0	0.0
citympg	0	0.0
highwaympg	0	0.0
price	0	0.0

In [15]:
```
null_df = null_df[null_df['Null Count'] > 0]
```

In [16]:
```
null_df
```

Out[16]:

	Null Count	Null Percentage

In [17]:
```
print(null_df)
```

```
Empty DataFrame
Columns: [Null Count, Null Percentage]
Index: []
```

In [18]:
```
threshold = 40
```

In [19]:
```
columns_to_drop = null_df[null_df['Null Percentage'] > threshold].index
```

In [20]:
```
df.drop(columns=columns_to_drop, axis=1, inplace=True)
```

In [21]:
```
print(f"Dropped columns: {columns_to_drop}")
```

```
Dropped columns: Index([], dtype='object')
```

In [22]:
```
columns_to_drop
```

Out[22]:
```
Index([], dtype='object')
```

In [23]:
```
df.dropna(thresh=df.shape[1] * 0.5, inplace=True)
```

In [24]:
```
print(df.dropna(thresh=df.shape[1] * 0.5, inplace=True))
```

```
None
```

Dropping 'enginelocation' Column as all cars engine location is Front only

In [25]:
```
df.drop(columns=['enginelocation'], axis=1, inplace=True)
```

In [26]:
```
df
```

Out[26]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	wheelbase	carlength	...	enginesize
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	88.6	168.8	...	130
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	88.6	168.8	...	130
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	94.5	171.2	...	152
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	99.8	176.6	...	109
4	5	2	audi 100ls	gas	std	four	sedan	4wd	99.4	176.6	...	136
...	...	...	...	...	...	...	...	...	...	...	...	..
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	109.1	188.8	...	141
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	109.1	188.8	...	141
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	109.1	188.8	...	173
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	109.1	188.8	...	145
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	109.1	188.8	...	141

205 rows × 25 columns

In [27]:
```
import seaborn as sns
import matplotlib.pyplot as plt

df['doornumber'] = df['doornumber'].map({'two': 2, 'four': 4})

plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[27], line 7
      4 df['doornumber'] = df['doornumber'].map({'two': 2, 'four': 4})
      6 plt.figure(figsize=(10,6))
----> 7 sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
      8 plt.title("Feature Correlation Heatmap")
      9 plt.show()

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.py:11049, in DataFrame.corr(self, method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.py:1993, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1991 if dtype is not None:
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1994 if result.dtype is not dtype:
   1995     result = np.asarray(result, dtype=dtype)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\internals\managers.py:1694, in BlockManager.as_array(self, dtype, copy, na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\internals\managers.py:1753, in BlockManager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

ValueError: could not convert string to float: 'alfa-romero giulia'
```

```
<Figure size 1000x600 with 0 Axes>
```

In [ ]:
```
print(df.dtypes)
```

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

df_numeric = df.select_dtypes(include=['number'])

plt.figure(figsize=(10,6))
sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```

Dropping Column 'doornumber' because we can see corelation between 'price' and 'doornumber' is '0.032' which is very low.

In [28]:
```
df.drop(columns=['doornumber'], axis=1, inplace=True)
```

In [29]:
```
df
```

Out[29]:

	car_ID	symboling	CarName	fueltype	aspiration	carbody	drivewheel	wheelbase	carlength	carwidth	...	enginesize	fu
0	1	3	alfa-romero giulia	gas	std	convertible	rwd	88.6	168.8	64.1	...	130	
1	2	3	alfa-romero stelvio	gas	std	convertible	rwd	88.6	168.8	64.1	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	hatchback	rwd	94.5	171.2	65.5	...	152	
3	4	2	audi 100 ls	gas	std	sedan	fwd	99.8	176.6	66.2	...	109	
4	5	2	audi 100ls	gas	std	sedan	4wd	99.4	176.6	66.4	...	136	
...	...	...	...	...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	sedan	rwd	109.1	188.8	68.9	...	141	
201	202	-1	volvo 144ea	gas	turbo	sedan	rwd	109.1	188.8	68.8	...	141	
202	203	-1	volvo 244dl	gas	std	sedan	rwd	109.1	188.8	68.9	...	173	
203	204	-1	volvo 246	diesel	turbo	sedan	rwd	109.1	188.8	68.9	...	145	
204	205	-1	volvo 264gl	gas	turbo	sedan	rwd	109.1	188.8	68.9	...	141	

205 rows × 24 columns

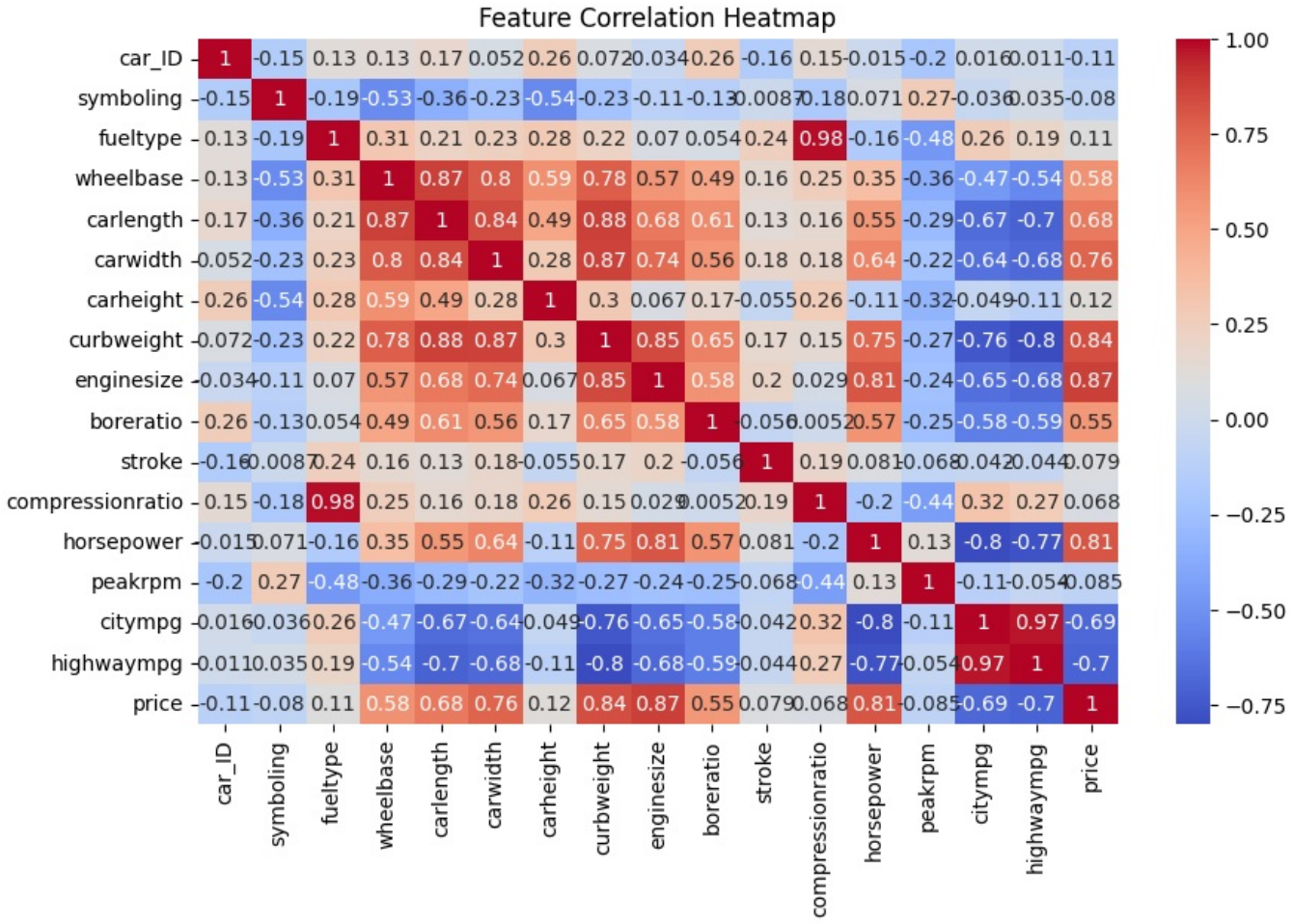
In [30]:
```
print(df['fueltype'].unique())
```

```
['gas' 'diesel']
```

converting char 'gas' and 'diesel' into numeric '0' and '1' to check correlation

In [31]:
```
df['fueltype'] = df['fueltype'].map({'gas': 0, 'diesel': 1})
```

In [32]:
```
import seaborn as sns
import matplotlib.pyplot as plt

df_numeric = df.select_dtypes(include=['number'])

plt.figure(figsize=(10,6))
sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```



Dropping 'fueltype' as it is not useful

In [33]:
```
df.drop(columns=['fueltype'], axis=1, inplace=True)
```

In [34]:
```
df
```

Out[34]:

	car_ID	symboling	CarName	aspiration	carbody	drivewheel	wheelbase	carlength	carwidth	carheight	...	enginesize
0	1	3	alfa-romero giulia	std	convertible	rwd	88.6	168.8	64.1	48.8	...	130
1	2	3	alfa-romero stelvio	std	convertible	rwd	88.6	168.8	64.1	48.8	...	130
2	3	1	alfa-romero Quadrifoglio	std	hatchback	rwd	94.5	171.2	65.5	52.4	...	152
3	4	2	audi 100 ls	std	sedan	fwd	99.8	176.6	66.2	54.3	...	109
4	5	2	audi 100ls	std	sedan	4wd	99.4	176.6	66.4	54.3	...	136
...	...	...	...	...	...	...	...	...	...	...	...	...
200	201	-1	volvo 145e (sw)	std	sedan	rwd	109.1	188.8	68.9	55.5	...	141
201	202	-1	volvo 144ea	turbo	sedan	rwd	109.1	188.8	68.8	55.5	...	141
202	203	-1	volvo 244dl	std	sedan	rwd	109.1	188.8	68.9	55.5	...	173
203	204	-1	volvo 246	turbo	sedan	rwd	109.1	188.8	68.9	55.5	...	145
204	205	-1	volvo 264gl	turbo	sedan	rwd	109.1	188.8	68.9	55.5	...	141

205 rows × 23 columns

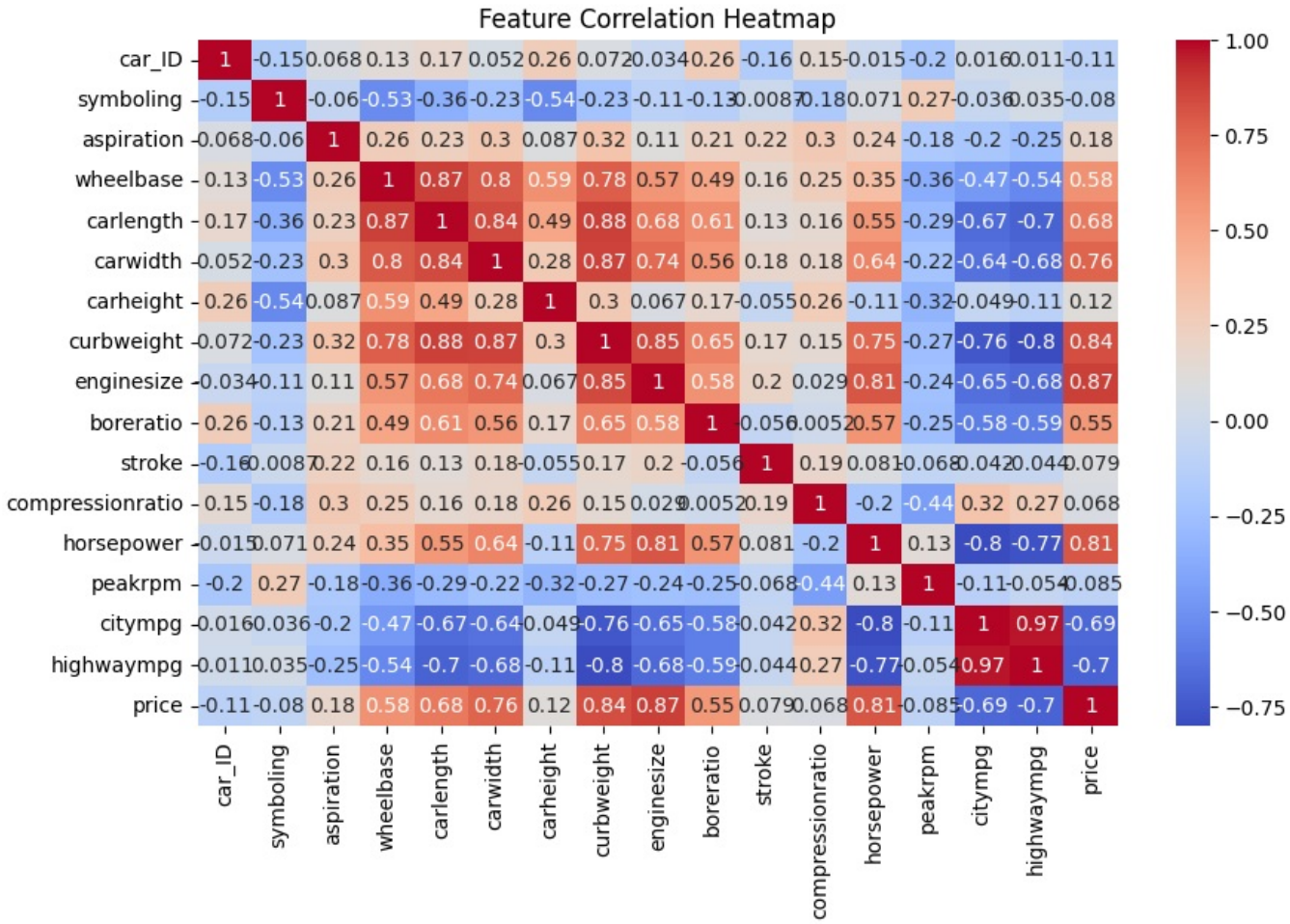
In [35]:
```
print(df['aspiration'].unique())
```

```
['std' 'turbo']
```

In [36]:
```
df['aspiration']=df['aspiration'].map({'std':0,'turbo':1})
```

In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt

df_numeric = df.select_dtypes(include=['number'])
```

```
plt.figure(figsize=(10,6))
sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```



Dropping 'aspiration' column

In [38]:
```
df.drop(columns=['aspiration'],axis=1,inplace=True)
```

Removing 'stroke' column

In [39]:
```
df.drop(columns=['stroke'],axis=1,inplace=True)
```

Sorting the table by 'price'(Ascending order)

In [40]:
```
df.sort_values(by='price', ascending=True, inplace=True)
```

In [41]:
```
df['price'].head()
```

Out[41]:
```
138    5118.0
18     5151.0
50     5195.0
150    5348.0
76     5389.0
Name: price, dtype: float64
```

In [42]:
```
df
```

Out[42]:

	car_ID	symboling	CarName	carbody	drivewheel	wheelbase	carlength	carwidth	carheight	curbweight	...	cylindernumb
138	139	2	subaru	hatchback	fwd	93.7	156.9	63.4	53.7	2050	...	fo
18	19	2	chevrolet impala	hatchback	fwd	88.4	141.1	60.3	53.2	1488	...	thr
50	51	1	maxda rx3	hatchback	fwd	93.1	159.1	64.2	54.1	1890	...	fo
150	151	1	toyota corona mark ii	hatchback	fwd	95.7	158.7	63.6	54.5	1985	...	fo
76	77	2	mitsubishi mirage	hatchback	fwd	93.7	157.3	64.4	50.8	1918	...	fo
...	...	...	...	...	...	...	...	...	...	...	...	
17	18	0	bmw x3	sedan	rwd	110.0	197.0	70.9	56.3	3505	...	
128	129	3	porsche boxter	convertible	rwd	89.5	168.9	65.0	51.6	2800	...	
73	74	0	buick century special	sedan	rwd	120.9	208.1	71.7	56.7	3900	...	eig
16	17	0	bmw x5	sedan	rwd	103.5	193.8	67.9	53.7	3380	...	
74	75	1	buick regal sport coupe (turbo)	hardtop	rwd	112.0	199.2	72.0	55.4	3715	...	eig

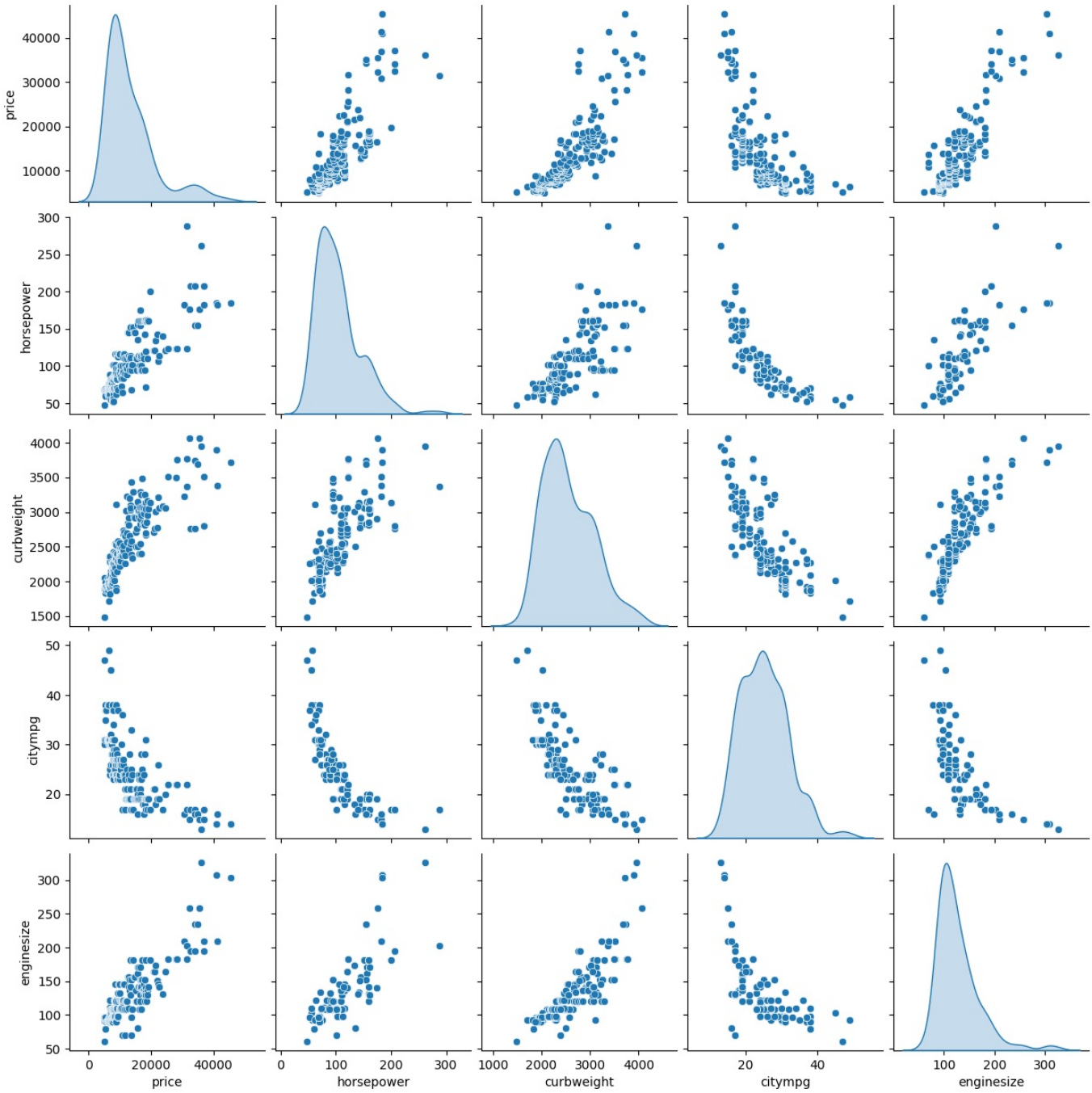
205 rows × 21 columns

Data Visualization

Pairplot

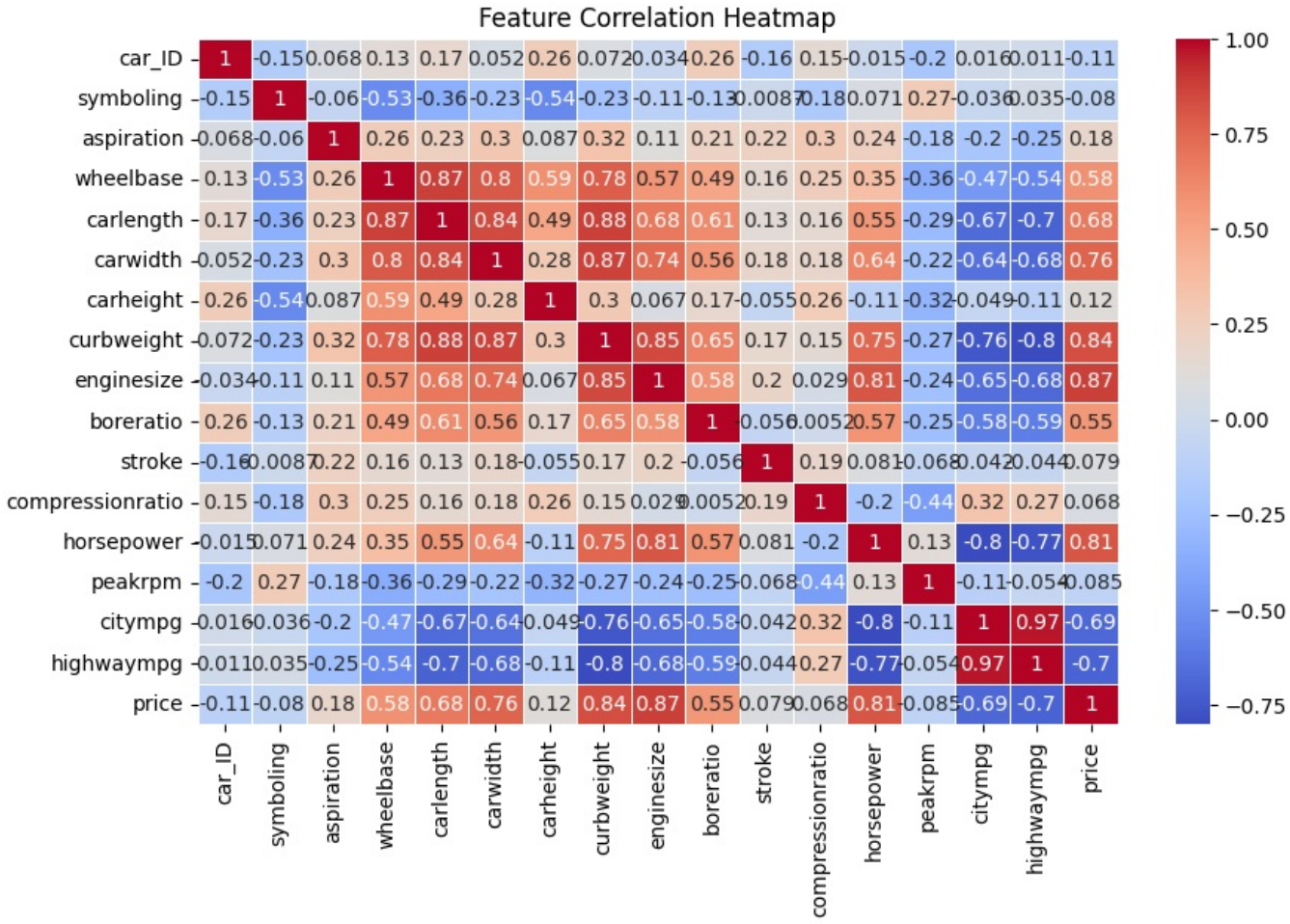
In [43]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [44]:
```
selected_features = ['price', 'horsepower', 'curbweight', 'citympg', 'enginesize']
sns.pairplot(df[selected_features], diag_kind='kde')
plt.show()
```



Heatmap

In [45]:
```
plt.figure(figsize=(10,6))
sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [46]:
```
df
```

Out[46]:

	car_ID	symboling	CarName	carbody	drivewheel	wheelbase	carlength	carwidth	carheight	curbweight	...	cylindernumb
138	139	2	subaru	hatchback	fwd	93.7	156.9	63.4	53.7	2050	...	fo
18	19	2	chevrolet impala	hatchback	fwd	88.4	141.1	60.3	53.2	1488	...	thr
50	51	1	maxda rx3	hatchback	fwd	93.1	159.1	64.2	54.1	1890	...	fo
150	151	1	toyota corona mark ii	hatchback	fwd	95.7	158.7	63.6	54.5	1985	...	fo
76	77	2	mitsubishi mirage	hatchback	fwd	93.7	157.3	64.4	50.8	1918	...	fo
...	...	...	...	...	...	...	...	...	...	...	...	
17	18	0	bmw x3	sedan	rwd	110.0	197.0	70.9	56.3	3505	...	s
128	129	3	porsche boxter	convertible	rwd	89.5	168.9	65.0	51.6	2800	...	s
73	74	0	buick century special	sedan	rwd	120.9	208.1	71.7	56.7	3900	...	eig
16	17	0	bmw x5	sedan	rwd	103.5	193.8	67.9	53.7	3380	...	s
74	75	1	buick regal sport coupe (turbo)	hardtop	rwd	112.0	199.2	72.0	55.4	3715	...	eig

205 rows × 21 columns

In [47]:
```
%store df
```

```
Stored 'df' (DataFrame)
C:\Users\amogh\AppData\Local\Programs\Python\Python312\Lib\site-packages\IPython\extensions\storemagic.py:229: UserWarning: This is now an optional IPython functionality, setting autorestore/df requires you to install the `pickleshare` library.
  db[ 'autorestore/' + arg ] = obj
```

In [49]:
```
!pip install pickleshare
```

```
Requirement already satisfied: pickleshare in c:\users\amogh\appdata\local\programs\python\python312\lib\site-pa
ckages (0.7.5)
[notice] A new release of pip is available: 24.1.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [50]:
```
%store df
```

```
Stored 'df' (DataFrame)
```

In [51]:
```
df.to_csv("updated_df.csv",index=False)
```

In [52]:
```
import pandas as pd
df = pd.read_csv("updated_df.csv")
```

In [53]:
```
df
```

Out[53]:

	car_ID	symboling	CarName	carbody	drivewheel	wheelbase	carlength	carwidth	carheight	curbweight	...	cylindernumb
0	139	2	subaru	hatchback	fwd	93.7	156.9	63.4	53.7	2050	...	fo
1	19	2	chevrolet impala	hatchback	fwd	88.4	141.1	60.3	53.2	1488	...	thr
2	51	1	maxda rx3	hatchback	fwd	93.1	159.1	64.2	54.1	1890	...	fo
3	151	1	toyota corona mark ii	hatchback	fwd	95.7	158.7	63.6	54.5	1985	...	fo
4	77	2	mitsubishi mirage	hatchback	fwd	93.7	157.3	64.4	50.8	1918	...	fo
...	...	...	...	...	...	...	...	...	...	...	...	
200	18	0	bmw x3	sedan	rwd	110.0	197.0	70.9	56.3	3505	...	s
201	129	3	porsche boxter	convertible	rwd	89.5	168.9	65.0	51.6	2800	...	s
202	74	0	buick century special	sedan	rwd	120.9	208.1	71.7	56.7	3900	...	eig
203	17	0	bmw x5	sedan	rwd	103.5	193.8	67.9	53.7	3380	...	s
204	75	1	buick regal sport coupe (turbo)	hardtop	rwd	112.0	199.2	72.0	55.4	3715	...	eig

205 rows × 21 columns

In [54]:
```
df.to_csv("updated_df.csv",index=False)
```

In [55]:
```
df
```

Out[55]:

	car_ID	symboling	CarName	carbody	drivewheel	wheelbase	carlength	carwidth	carheight	curbweight	...	cylindernumb
0	139	2	subaru	hatchback	fwd	93.7	156.9	63.4	53.7	2050	...	fo
1	19	2	chevrolet impala	hatchback	fwd	88.4	141.1	60.3	53.2	1488	...	thr
2	51	1	maxda rx3	hatchback	fwd	93.1	159.1	64.2	54.1	1890	...	fo
3	151	1	toyota corona mark ii	hatchback	fwd	95.7	158.7	63.6	54.5	1985	...	fo
4	77	2	mitsubishi mirage	hatchback	fwd	93.7	157.3	64.4	50.8	1918	...	fo
...	...	...	...	...	...	...	...	...	...	...	...	
200	18	0	bmw x3	sedan	rwd	110.0	197.0	70.9	56.3	3505	...	s
201	129	3	porsche boxter	convertible	rwd	89.5	168.9	65.0	51.6	2800	...	s
202	74	0	buick century special	sedan	rwd	120.9	208.1	71.7	56.7	3900	...	eig
203	17	0	bmw x5	sedan	rwd	103.5	193.8	67.9	53.7	3380	...	s
204	75	1	buick regal sport coupe (turbo)	hardtop	rwd	112.0	199.2	72.0	55.4	3715	...	eig

205 rows × 21 columns

Data Preperation

In [56]:
```
df['carCompany'] = df['CarName'].apply(lambda x: x.split(' ')[0].lower())
```

In [58]:
```
df['carCompany']
```

Out[58]:
```
0          subaru
1       chevrolet
2           maxda
3          toyota
4      mitsubishi
          ...
200           bmw
201       porsche
202         buick
203           bmw
204         buick
Name: carCompany, Length: 205, dtype: object
```

In [62]:
```
print(df['carCompany'].unique())
```

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

In [63]:
```
df['carCompany']=df['carCompany'].replace({
    'maxda': 'mazda',
    'vokswagen': 'volkswagen',
    'vw': 'volkswagen',
    'toyouta': 'toyota',
    'porcshce': 'porsche',
    'alfa-romero': 'alfa-romeo'
})
```

In [64]:
```
df['carCompany'].unique()
```

Out[64]:
```
array(['subaru', 'chevrolet', 'mazda', 'toyota', 'mitsubishi', 'honda',
       'nissan', 'plymouth', 'dodge', 'isuzu', 'volkswagen', 'renault',
       'saab', 'peugeot', 'volvo', 'alfa-romeo', 'audi', 'bmw', 'mercury',
       'porsche', 'buick', 'jaguar'], dtype=object)
```

Dropping 'car_ID' column

In [67]:
```
df.drop(columns=['car_ID'], axis=1, inplace=True)
```

In [68]:
```
df = pd.get_dummies(df, columns=['carCompany', 'enginetype', 'carbody'], drop_first=True)
```

In [69]:
```
df
```

Out[69]:

	symboling	CarName	drivewheel	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	...	eng
0	2	subaru	fwd	93.7	156.9	63.4	53.7	2050	four	97	...	
1	2	chevrolet impala	fwd	88.4	141.1	60.3	53.2	1488	three	61	...	
2	1	maxda rx3	fwd	93.1	159.1	64.2	54.1	1890	four	91	...	
3	1	toyota corona mark ii	fwd	95.7	158.7	63.6	54.5	1985	four	92	...	
4	2	mitsubishi mirage	fwd	93.7	157.3	64.4	50.8	1918	four	92	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
200	0	bmw x3	rwd	110.0	197.0	70.9	56.3	3505	six	209	...	
201	3	porsche boxter	rwd	89.5	168.9	65.0	51.6	2800	six	194	...	
202	0	buick century special	rwd	120.9	208.1	71.7	56.7	3900	eight	308	...	
203	0	bmw x5	rwd	103.5	193.8	67.9	53.7	3380	six	209	...	
204	1	buick regal sport coupe (turbo)	rwd	112.0	199.2	72.0	55.4	3715	eight	304	...	

205 rows × 49 columns

In [70]:
```
df.to_csv("updated_df.csv",index=False)
```

In [74]:
```
df.rename(columns={
    'carCompany_subaru': 'is_subaru',
    'carCompany_chevrolet': 'is_chevrolet',
    'carCompany_mazda': 'is_mazda',
    'carCompany_toyota': 'is_toyota',
    'carCompany_mitsubishi': 'is_mitsubishi',
    'carCompany_honda': 'is_honda',
    'carCompany_nissan': 'is_nissan',
    'carCompany_plymouth': 'is_plymouth',
    'carCompany_dodge': 'is_dodge',
    'carCompany_isuzu': 'is_isuzu',
    'carCompany_volkswagen': 'is_volkswagen',
    'carCompany_renault': 'is_renault',
    'carCompany_saab': 'is_saab',
    'carCompany_peugeot': 'is_peugeot',
    'carCompany_volvo': 'is_volvo',
    'carCompany_alfa-romeo': 'is_alfa_romeo',
    'carCompany_audi': 'is_audi',
    'carCompany_bmw': 'is_bmw',
    'carCompany_mercury': 'is_mercury',
    'carCompany_porsche': 'is_porsche',
    'carCompany_buick': 'is_buick',
    'carCompany_jaguar': 'is_jaguar'
}, inplace=True)
```

In [76]:
```
df.rename(columns={
    'enginetype_dohcv': 'is_dohcv',
    'enginetype_I': 'is_inline',
    'enginetype_ohc': 'is_ohc',
    'enginetype_ohcf': 'is_ohcf',
    'enginetype_ohcv': 'is_ohcv',
    'enginetype_rotor': 'is_rotor'
}, inplace=True)
```

In [77]:
```
df.rename(columns={
    'carbody_hardtop': 'is_hardtop',
    'carbody_sedan': 'is_sedan',
    'carbody_hatchback': 'is_hatchback',
    'carbody_wagon': 'is_wagon'
}, inplace=True)
```

In [78]:
```
df
```

Out[78]:

	symboling	CarName	drivewheel	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	...	is_c
0	2	subaru	fwd	93.7	156.9	63.4	53.7	2050	four	97	...	
1	2	chevrolet impala	fwd	88.4	141.1	60.3	53.2	1488	three	61	...	
2	1	maxda rx3	fwd	93.1	159.1	64.2	54.1	1890	four	91	...	
3	1	toyota corona mark ii	fwd	95.7	158.7	63.6	54.5	1985	four	92	...	
4	2	mitsubishi mirage	fwd	93.7	157.3	64.4	50.8	1918	four	92	...	
...	...	...	...	...	...	...	...	...	...	...	...	
200	0	bmw x3	rwd	110.0	197.0	70.9	56.3	3505	six	209	...	
201	3	porsche boxter	rwd	89.5	168.9	65.0	51.6	2800	six	194	...	
202	0	buick century special	rwd	120.9	208.1	71.7	56.7	3900	eight	308	...	
203	0	bmw x5	rwd	103.5	193.8	67.9	53.7	3380	six	209	...	
204	1	buick regal sport coupe (turbo)	rwd	112.0	199.2	72.0	55.4	3715	eight	304	...	

205 rows × 49 columns

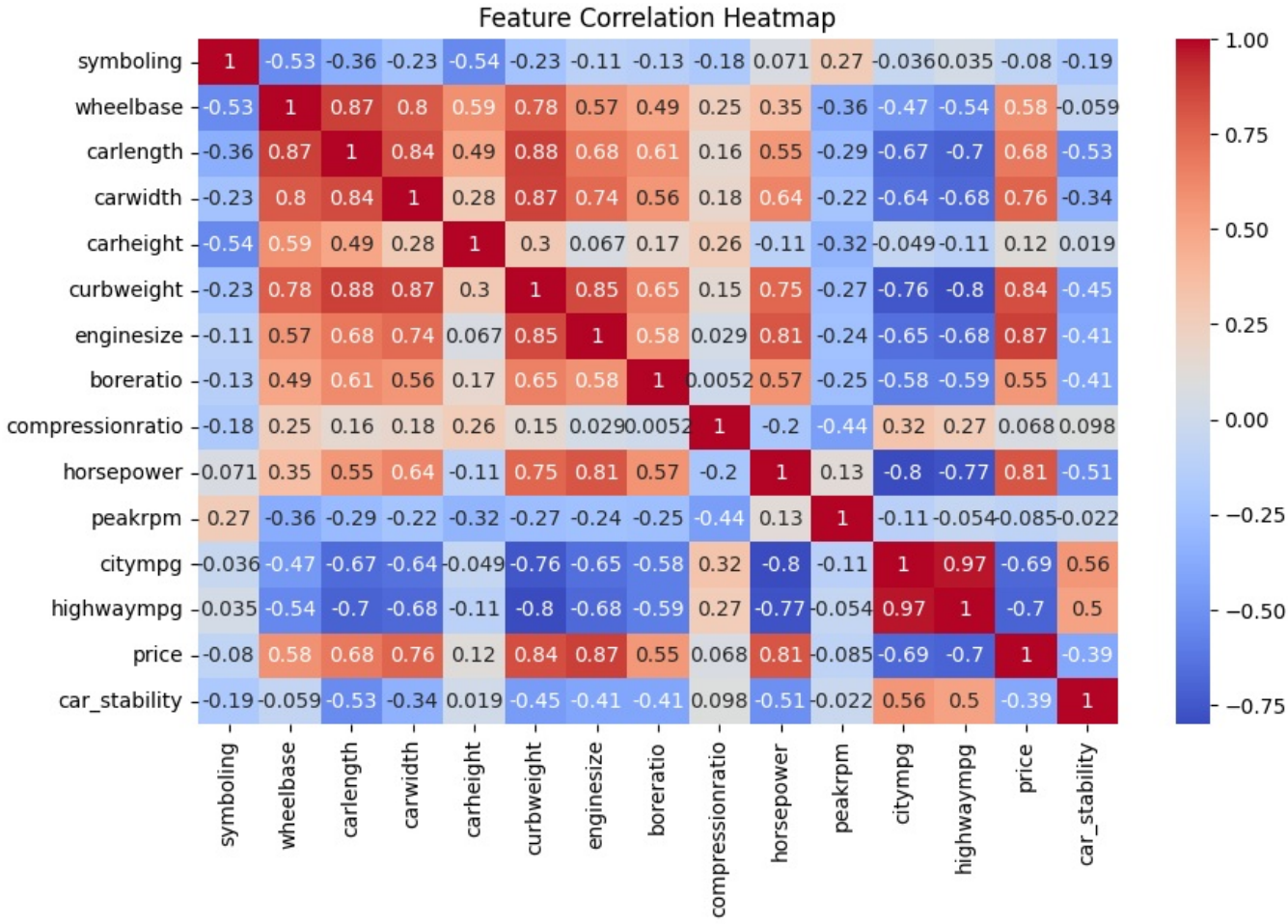
In [79]:
```
df.to_csv("updated_df.csv",index=False)
```

In [80]:
```
df['car_stability'] = df['wheelbase'] / df['carlength']
```

In [81]:
```
df['car_stability']
```

Out[81]:
```
0      0.597196
1      0.626506
2      0.585167
3      0.603025
4      0.595677
         ...
200    0.558376
201    0.529899
202    0.580971
203    0.534056
204    0.562249
Name: car_stability, Length: 205, dtype: float64
```

In [83]:
```
import seaborn as sns
import matplotlib.pyplot as plt

df_numeric = df.select_dtypes(include=['number'])

plt.figure(figsize=(10,6))
sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [84]:
```
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = df[['wheelbase', 'carlength', 'car_stability', 'price']].corr()

print(correlation_matrix)
```

```
              wheelbase  carlength  car_stability     price
wheelbase      1.000000   0.874587      -0.058602  0.577816
carlength      0.874587   1.000000      -0.533972  0.682920
car_stability -0.058602  -0.533972       1.000000 -0.389469
price          0.577816   0.682920      -0.389469  1.000000
```

Dropping 'wheelbase' because 'carlength' is more strongly correlated with 'price' than 'wheelbase'

In [85]:
```
df.drop(columns=['wheelbase'], inplace=True)
```

Dropping Highly Correlated Features to Avoid Redundancy

Dropping 'carwidth' and 'curbweight' as we already dropped 'wheelbase'

In [87]:
```
df.drop(columns=['carwidth', 'curbweight'], inplace=True)
```

Dropping 'highwaympg'

In [89]:
```
df.drop(columns=['highwaympg'], inplace=True)
```

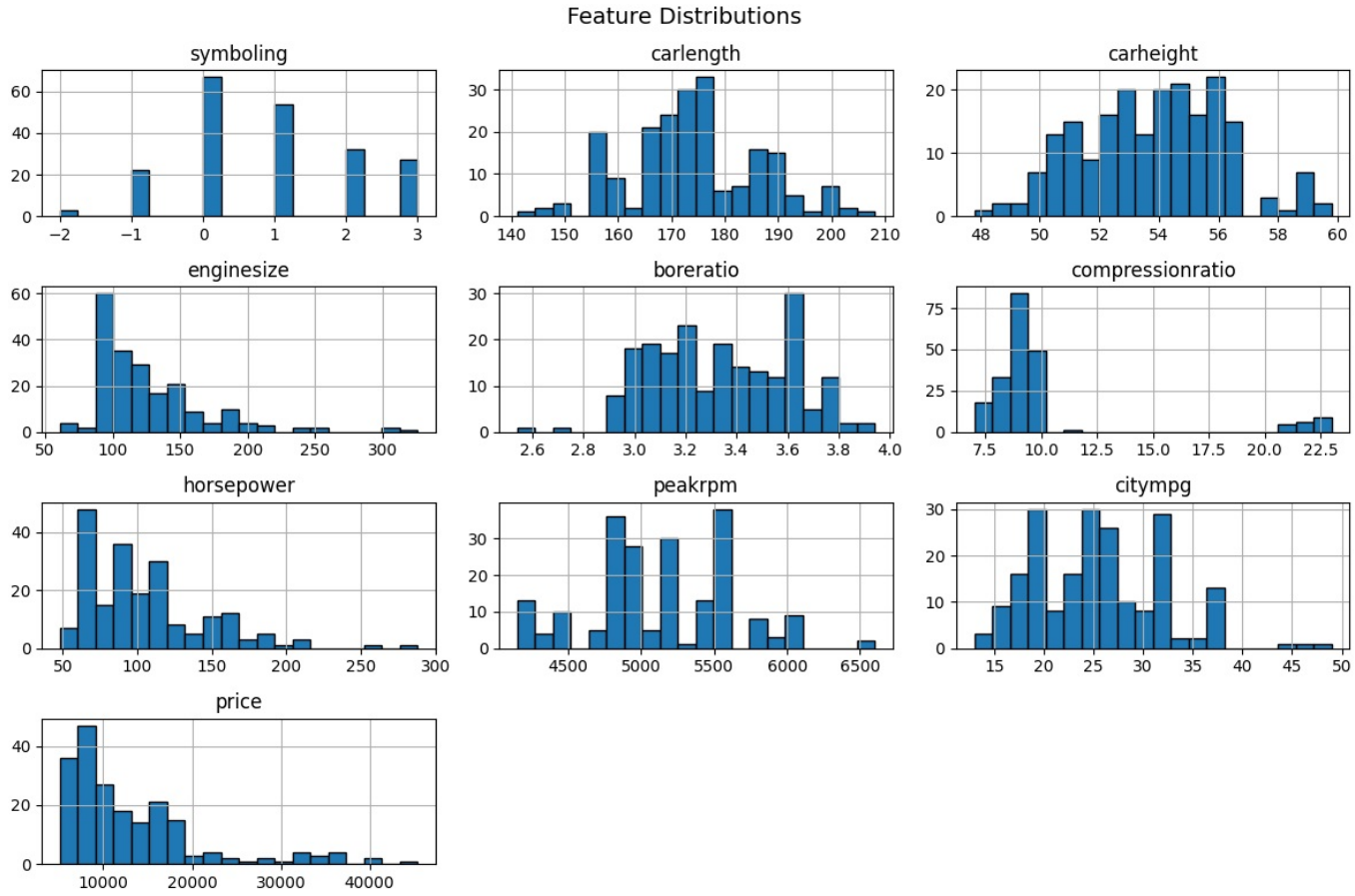
Dropping 'car_stability' to avoid redundancy

In [91]:
```
df.drop(columns=['car_stability'], inplace=True)
```

In [92]:
```
df.to_csv('updated_df.csv',index=False)
```

Data Analysis

In [97]:
```
import matplotlib.pyplot as plt

df.hist(figsize=(12, 8), bins=20, edgecolor='black')
plt.suptitle("Feature Distributions", fontsize=14)
plt.tight_layout()
plt.show()
```



In [99]:
```
brand_columns = [col for col in df.columns if col.startswith('is_')]
print(brand_columns)
```

```
['is_audi', 'is_bmw', 'is_buick', 'is_chevrolet', 'is_dodge', 'is_honda', 'is_isuzu', 'is_jaguar', 'is_mazda', '
is_mercury', 'is_mitsubishi', 'is_nissan', 'is_peugeot', 'is_plymouth', 'is_porsche', 'is_renault', 'is_saab', '
is_subaru', 'is_toyota', 'is_volkswagen', 'is_volvo', 'is_dohcv', 'is_ohc', 'is_ohcf', 'is_ohcv', 'is_rotor', 'i
s_hardtop', 'is_hatchback', 'is_sedan', 'is_wagon']
```

In [100…
```
df['carCompany'] = df[brand_columns].idxmax(axis=1).str.replace("is_", "")
```

In [101…
```
print(df[['carCompany']].head())
```

```
   carCompany
0      subaru
1   chevrolet
2       mazda
3      toyota
4  mitsubishi
```

In [103…
```
df
```

Out[103…

	symboling	CarName	drivewheel	carlength	carheight	cylindernumber	enginesize	fuelsystem	boreratio	compressionratio
0	2	subaru	fwd	156.9	53.7	four	97	2bbl	3.62	9.0
1	2	chevrolet impala	fwd	141.1	53.2	three	61	2bbl	2.91	9.5
2	1	maxda rx3	fwd	159.1	54.1	four	91	2bbl	3.03	9.0
3	1	toyota corona mark ii	fwd	158.7	54.5	four	92	2bbl	3.05	9.0
4	2	mitsubishi mirage	fwd	157.3	50.8	four	92	2bbl	2.97	9.4
...	...	...	...	...	...	...	...	...	...	...
200	0	bmw x3	rwd	197.0	56.3	six	209	mpfi	3.62	8.0
201	3	porsche boxter	rwd	168.9	51.6	six	194	mpfi	3.74	9.5
202	0	buick century special	rwd	208.1	56.7	eight	308	mpfi	3.80	8.0
203	0	bmw x5	rwd	193.8	53.7	six	209	mpfi	3.62	8.0
204	1	buick regal sport coupe (turbo)	rwd	199.2	55.4	eight	304	mpfi	3.80	8.0

205 rows × 46 columns

In [104…
```
df.to_csv("updated_df.csv",index=False)
```

In [112…
```
df = df.rename(columns={'enginetype_l': 'is_inline'})
```

In [114…
```
brand_columns = [col for col in df.columns if col.startswith('is_') and col not in ['is_dohcv', 'is_inline', 'is
print("Car Brand Columns:", brand_columns)
```

```
Car Brand Columns: ['is_audi', 'is_bmw', 'is_buick', 'is_chevrolet', 'is_dodge', 'is_honda', 'is_isuzu', 'is_jag
uar', 'is_mazda', 'is_mercury', 'is_mitsubishi', 'is_nissan', 'is_peugeot', 'is_plymouth', 'is_porsche', 'is_ren
ault', 'is_saab', 'is_subaru', 'is_toyota', 'is_volkswagen', 'is_volvo']
```

In [115…
```
df['carCompany'] = df[brand_columns].idxmax(axis=1).str.replace("is_", "")
```

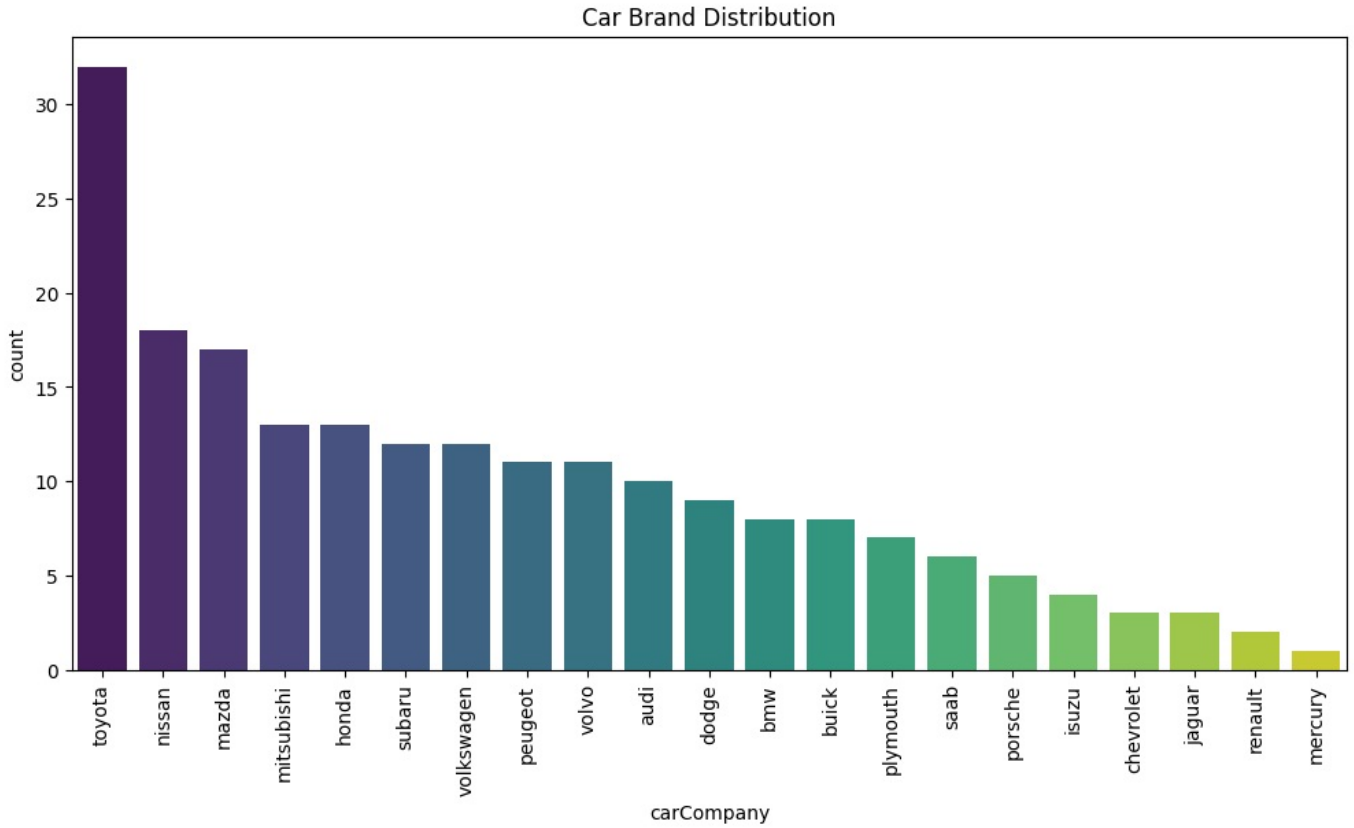
In [116…
```
print(df[['carCompany', 'is_dohcv', 'is_sedan']].head())
```

```
   carCompany  is_dohcv  is_sedan
0      subaru     False     False
1   chevrolet     False     False
2       mazda     False     False
3      toyota     False     False
4  mitsubishi     False     False
```

In [118…
```
df.to_csv("updated_df.csv",index=False)
```

In [119…
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))
sns.countplot(x=df['carCompany'], order=df['carCompany'].value_counts().index, palette="viridis")
plt.xticks(rotation=90)
plt.title("Car Brand Distribution")
plt.show()
```

```
C:\Users\amogh\AppData\Local\Temp\ipykernel_3952\2154864462.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df['carCompany'], order=df['carCompany'].value_counts().index, palette="viridis")
```



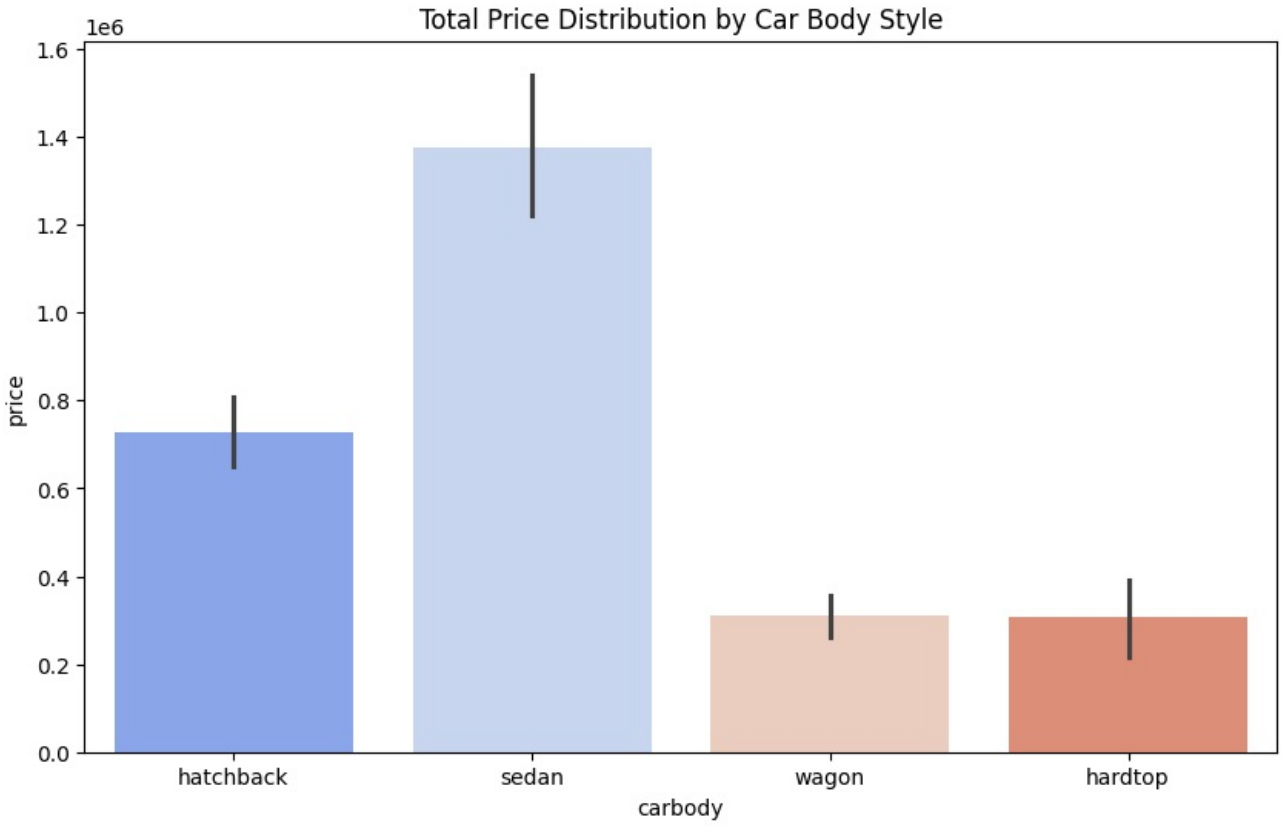
In [121…
```
carbody_columns = [col for col in df.columns if col.startswith('is_') and col in ['is_hardtop', 'is_sedan', 'is_
print("Car Body Columns:", carbody_columns)
```

```
Car Body Columns: ['is_hardtop', 'is_hatchback', 'is_sedan', 'is_wagon']
```

In [122…
```
df['carbody'] = df[carbody_columns].idxmax(axis=1).str.replace("is_", "")
```

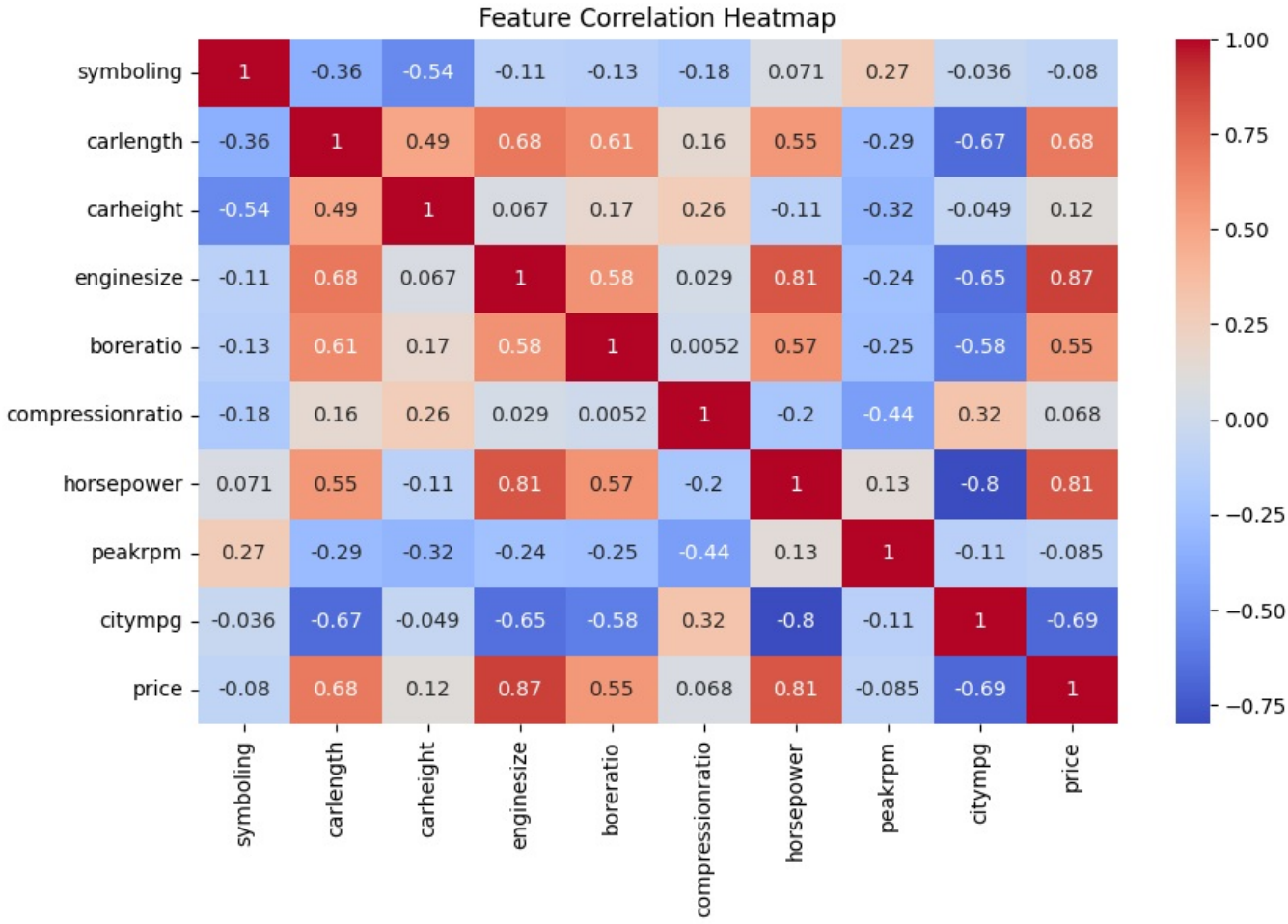
In [123…
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
sns.barplot(x=df['carbody'], y=df['price'], estimator=sum, palette="coolwarm")
plt.title("Total Price Distribution by Car Body Style")
plt.show()
```

```
C:\Users\amogh\AppData\Local\Temp\ipykernel_3952\227401964.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=df['carbody'], y=df['price'], estimator=sum, palette="coolwarm")
```



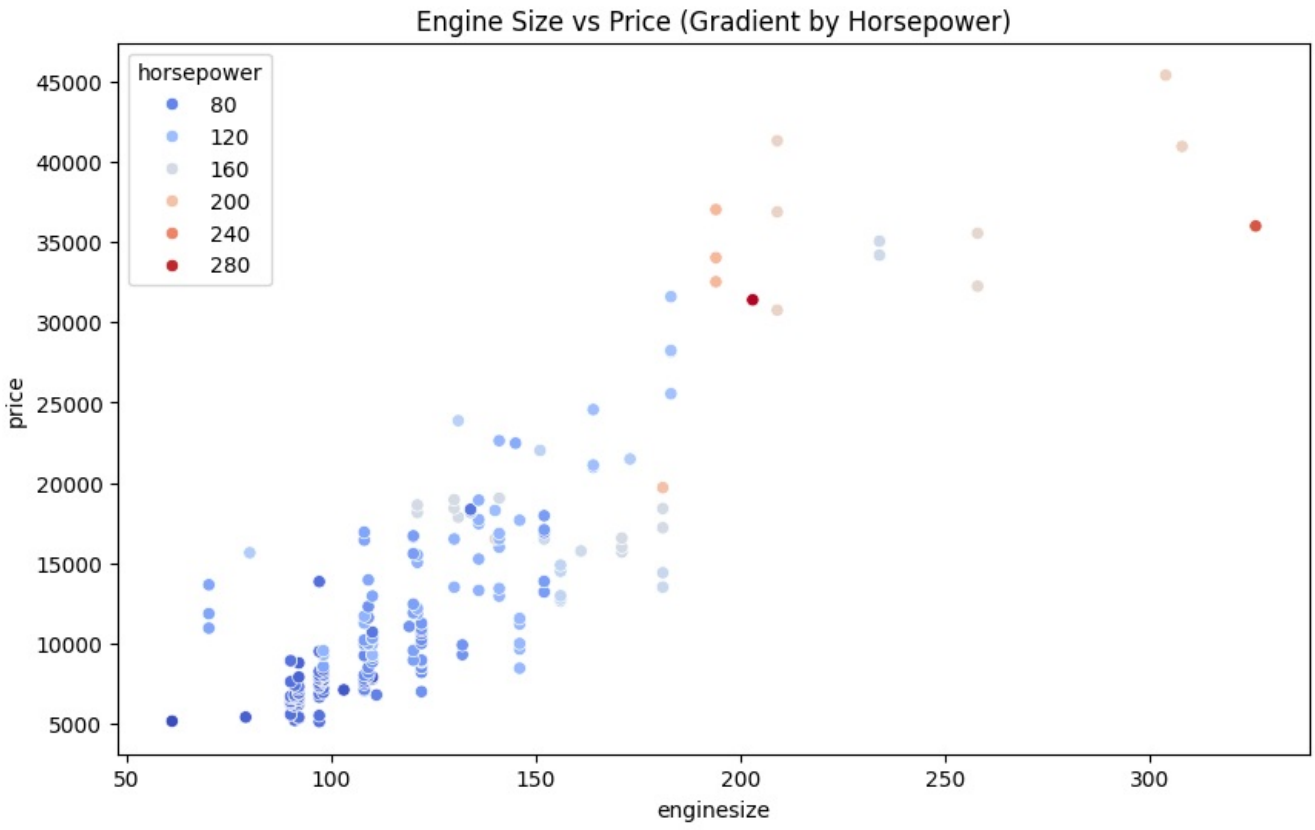
In [125…

```
import seaborn as sns
import matplotlib.pyplot as plt

df_numeric = df.select_dtypes(include=['number'])

plt.figure(figsize=(10,6))
sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [126…

```
plt.figure(figsize=(10,6))
sns.scatterplot(x=df['enginesize'], y=df['price'], hue=df['horsepower'], palette="coolwarm")
plt.title("Engine Size vs Price (Gradient by Horsepower)")
plt.show()
```



In [127…

```
df.to_csv("updated_df.csv",index=False)
```

Insight

From the above graphical representation, based on the Bar Chart, my insight is that 'Toyota' is the most sold car brand, while 'Mercury' is the least sold

Based on the 'carbody' vs. 'price' analysis, my insight is that sedan and hatchback cars are sold more frequently than wagons and hardtops

From the heatmap analysis, it's clear that engine size and horsepower are closely linked to higher car prices, meaning bigger and more powerful cars tend to cost more. On the other hand, fuel-efficient cars (higher MPG) are generally priced lower, showing a trade-off between performance and efficiency

From the scatter plot, we can see that cars with larger engine sizes generally have higher prices. Additionally, the gradient indicates that cars with higher horsepower also tend to be more expensive, suggesting that both engine size and power significantly impact a car's value

Key Insights

Most sold car brand: Toyota

Least sold car brand: Mercury

Most sold car body type: Sedan

Least sold car body type: Hardtop

Heatmap is very useful for identifying correlations

Cars with bigger engines and higher horsepower tend to have higher prices, and vice versa.

Thank You

Converting CSV to Excel

In [128…
```
!pip install openpyxl
```

```
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl.metadata (2.5 kB)
Collecting et-xmlfile (from openpyxl)
  Downloading et_xmlfile-2.0.0-py3-none-any.whl.metadata (2.7 kB)
Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
   ---------------------------------------- 0.0/250.9 kB ? eta -:--:--
   ---- ----------------------------------- 30.7/250.9 kB 1.4 MB/s eta 0:00:01
   --------- ------------------------------ 61.4/250.9 kB 825.8 kB/s eta 0:00:01
   -------------- ------------------------- 92.2/250.9 kB 871.5 kB/s eta 0:00:01
   ------------------------ --------------- 153.6/250.9 kB 1.0 MB/s eta 0:00:01
   ------------------------------- -------- 204.8/250.9 kB 1.0 MB/s eta 0:00:01
   ---------------------------------------- 250.9/250.9 kB 1.0 MB/s eta 0:00:00
Downloading et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-2.0.0 openpyxl-3.1.5
[notice] A new release of pip is available: 24.1.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [131…
```
import pandas as pd
df = pd.read_csv("updated_df.csv")
df.to_excel("updated_df.xlsx", index=False)
```

Thank you

In [ ]:
```
```