```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression


from google.colab import drive
drive.mount('/gdrive')

from google.colab import drive
drive.mount('/content/drive')
```

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

```
data = pd.read_csv("/content/drive/MyDrive/Trainity Assignments/Trainity Assignment - 7/Car_data.csv")
```

```
data.head()
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	N
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	

```
data.tail()
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Veh
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Mi
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Mi
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Mi
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Mi
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury	Mi

```
# function to get null value
def column_wise_null_percentage(df):
    output = round(df.isnull().sum()/len(df.index)*100,2)
    return output


# get missign values of all columns
NA_col = column_wise_null_percentage(data)
```

```
NA_col
```

```
Make                  0.00
Model                 0.00
Year                  0.00
Engine Fuel Type      0.03
Engine HP             0.58
Engine Cylinders      0.25
Transmission Type     0.00
Driven_Wheels         0.00
Number of Doors       0.05
Market Category      31.41
Vehicle Size          0.00
Vehicle Style         0.00
highway MPG           0.00
city mpg              0.00
Popularity            0.00
MSRP                  0.00
dtype: float64
```

```
data.index
```

```
RangeIndex(start=0, stop=11914, step=1)
```

```
data.dtypes
```

```
Make                  object
Model                 object
Year                   int64
Engine Fuel Type      object
Engine HP            float64
Engine Cylinders     float64
Transmission Type     object
Driven_Wheels         object
Number of Doors      float64
Market Category       object
Vehicle Size          object
Vehicle Style         object
highway MPG            int64
city mpg               int64
Popularity             int64
MSRP                   int64
dtype: object
```

```
#No of duplicate Rows
duplicate_rows_data = data[data.duplicated()]
print("number of duplicate rows: ", duplicate_rows_data.shape)
```

```
number of duplicate rows:  (715, 16)
```

```
# dropping duplicate rows
data = data.drop_duplicates()
```

```
#Calculating Missing values
print(data.isnull().sum())
```

```
Make                    0
Model                   0
Year                    0
Engine Fuel Type        3
Engine HP              69
Engine Cylinders       30
Transmission Type       0
Driven_Wheels           0
Number of Doors         6
Market Category      3376
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

```
# dropping the values
data = data.dropna()
data.count()
```

```
Make                 7735
Model                7735
Year                 7735
Engine Fuel Type     7735
Engine HP            7735
Engine Cylinders     7735
```

```
Transmission Type    7735
Driven_Wheels        7735
Number of Doors      7735
Market Category      7735
Vehicle Size         7735
Vehicle Style        7735
highway MPG          7735
city mpg             7735
Popularity           7735
MSRP                 7735
dtype: int64
```

```
#after dropping the values
print(data.isnull().sum())
```

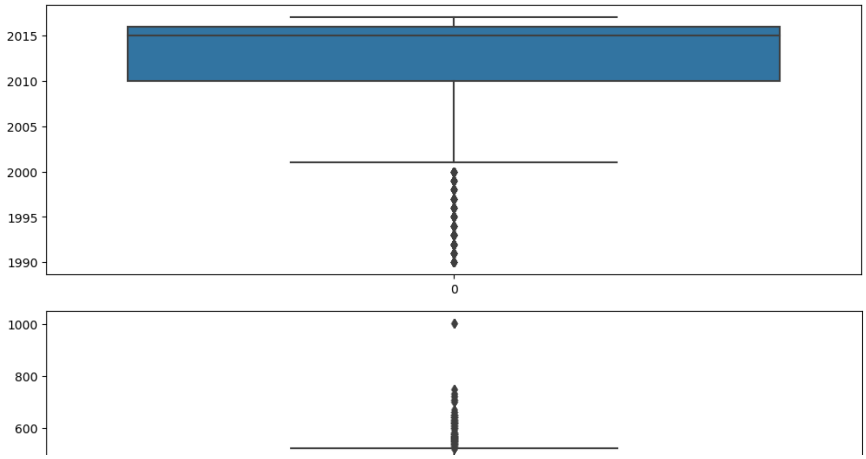
```
Make                 0
Model                0
Year                 0
Engine Fuel Type     0
Engine HP            0
Engine Cylinders     0
Transmission Type    0
Driven_Wheels        0
Number of Doors      0
Market Category      0
Vehicle Size         0
Vehicle Style        0
highway MPG          0
city mpg             0
Popularity           0
MSRP                 0
dtype: int64
```

```
for col in data.columns:
    print(col)
    print(data[col].unique()[:5])
    print(data[col].nunique())
    print('\n')
```

```
Engine Cylinders
[ 6.  4.  5.  8. 12.]
9


Transmission Type
['MANUAL' 'AUTOMATIC' 'AUTOMATED_MANUAL' 'DIRECT_DRIVE' 'UNKNOWN']
5


Driven_Wheels
['rear wheel drive' 'front wheel drive' 'all wheel drive'
 'four wheel drive']
4


Number of Doors
[2. 4. 3.]
3


Market Category
['Factory Tuner,Luxury,High-Performance' 'Luxury,Performance'
```

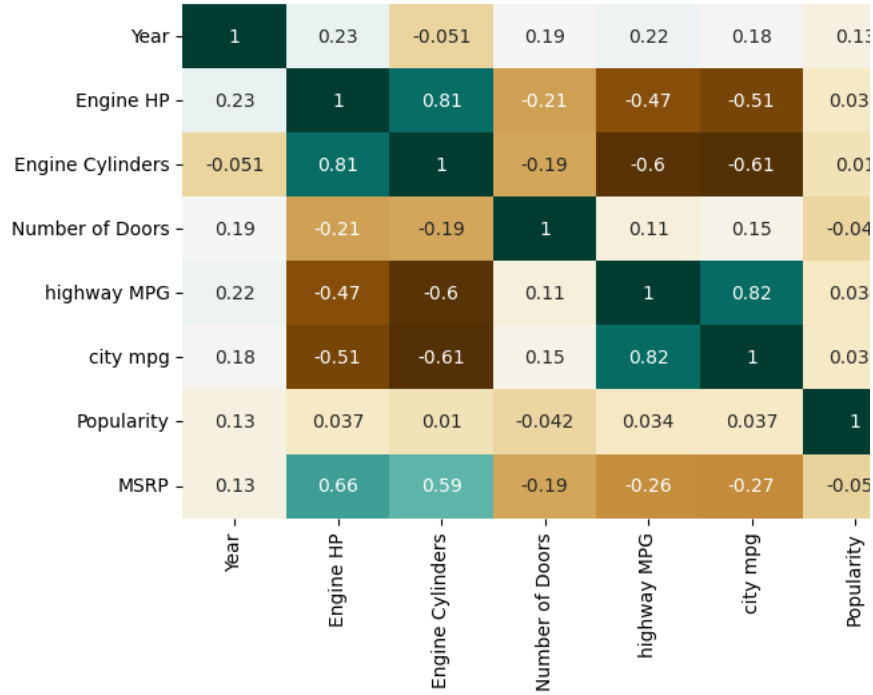
```
[3916 3105  819  617 1013]
47


MSRP
[46135 40650 36350 29450 34500]
4644
```

```
# Box plot for all variables for outlier detection
plt.figure(figsize=(12,4))
sns.boxplot(data['Year'])
plt.show()

plt.figure(figsize=(12,4))
sns.boxplot(data['Engine HP'])
plt.show()

plt.figure(figsize=(12,4))
sns.boxplot(data['Engine Cylinders'])
plt.show()

plt.figure(figsize=(12,4))
sns.boxplot(data['Popularity'])
plt.show()
```




```
plt.figure(figsize=(10,5))
c = data.corr()
sns.heatmap(c, cmap="BrBG", annot=True )
```

```
<ipython-input-18-ac065ee9baaa>:2: FutureWarning: The default value of numeric
  c = data.corr()
<Axes: >
```




```
data.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.ylabel('Number of cars')
plt.xlabel('Make');
```


```
grouped_df = data.groupby('Market Category')
```

```
car_model_counts = grouped_df['Model'].count()
popularity_scores = grouped_df['Popularity'].mean()
result_df = pd.DataFrame({'Car Model Count': car_model_counts, 'Popularity Score': popularity_scores})
result_df
```

Market Category	Car Model Count	Popularity Score
Crossover	1068	1539.475655
Crossover,Diesel	7	873.000000
Crossover,Exotic,Luxury,High-Performance	1	238.000000
Crossover,Exotic,Luxury,Performance	1	238.000000
Crossover,Factory Tuner,Luxury,High-Performance	26	1823.461538
...	...	...
Luxury,Hybrid	48	724.687500
Luxury,Performance	659	1293.062215
Luxury,Performance,Hybrid	11	2333.181818
Performance	503	1443.234592
Performance,Hybrid	1	155.000000

70 rows × 2 columns

To Determine : How does the popularity of a car model vary across different market categories?

Task 1.A: Create a pivot table that shows the number of car models in each market category and their corresponding popularity scores.

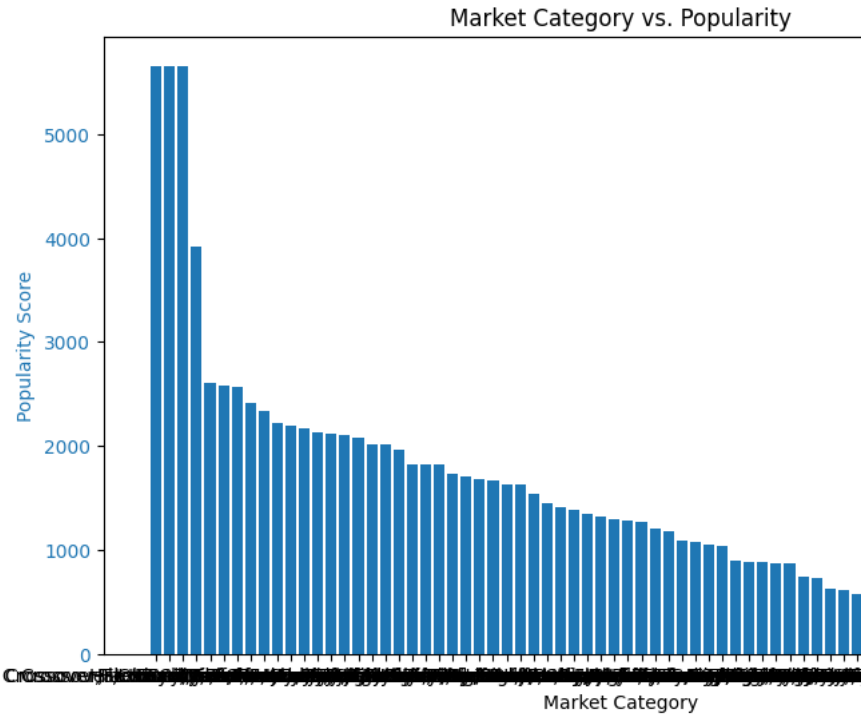
Task 1.B: Create a combo chart that visualizes the relationship between market category and popularity.

```
# Create the pivot table
pivot_table = data.pivot_table(index='Market Category', values='Model', aggfunc='count', margins=True, margins_name='Total')
pivot_table.columns = ['CarModelCount']

# Add popularity scores to the pivot table
popularity_scores = data.groupby('Market Category')['Popularity'].mean()
pivot_table['Popularity'] = popularity_scores

# Display the pivot table
print(pivot_table)
```

```
                                                CarModelCount   Popularity
Market Category
Crossover                                                1068  1539.475655
Crossover,Diesel                                            7   873.000000
Crossover,Exotic,Luxury,High-Performance                    1   238.000000
Crossover,Exotic,Luxury,Performance                         1   238.000000
Crossover,Factory Tuner,Luxury,High-Performance            26  1823.461538
...                                                       ...          ...
Luxury,Performance                                        659  1293.062215
Luxury,Performance,Hybrid                                  11  2333.181818
Performance                                               503  1443.234592
Performance,Hybrid                                          1   155.000000
Total                                                    7735          NaN

[71 rows x 2 columns]
```

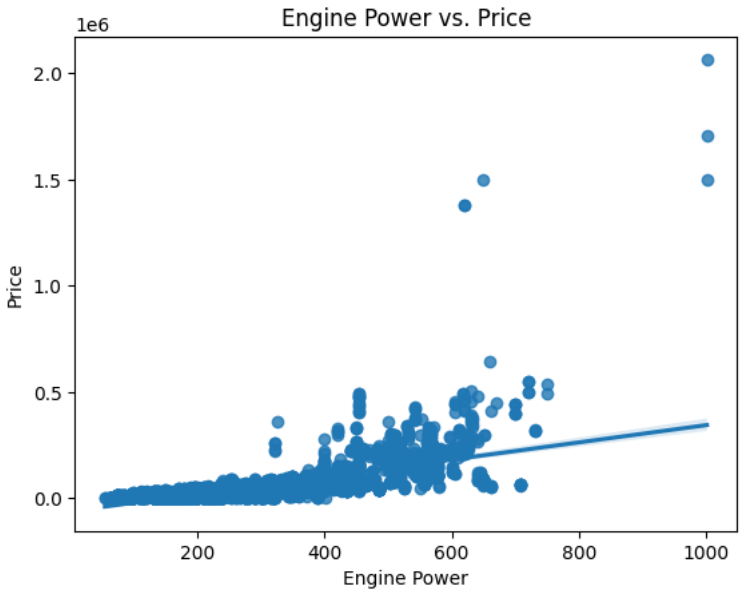
```
# Calculate the average popularity scores for each market category
popularity_scores = data.groupby('Market Category')['Popularity'].mean().reset_index()

# Sort the data by popularity score in descending order
sorted_df = popularity_scores.sort_values('Popularity', ascending=False)

# Create the combo chart
fig, ax1 = plt.subplots(figsize=(10, 6))

# Bar plot for popularity scores
```

```
ax1.bar(sorted_df['Market Category'], sorted_df['Popularity'], color='tab:blue')
ax1.set_xlabel('Market Category')
ax1.set_ylabel('Popularity Score', color='tab:blue')
ax1.tick_params(axis='y', labelcolor='tab:blue')

# Set title and display the chart
plt.title('Market Category vs. Popularity')
plt.show()
```



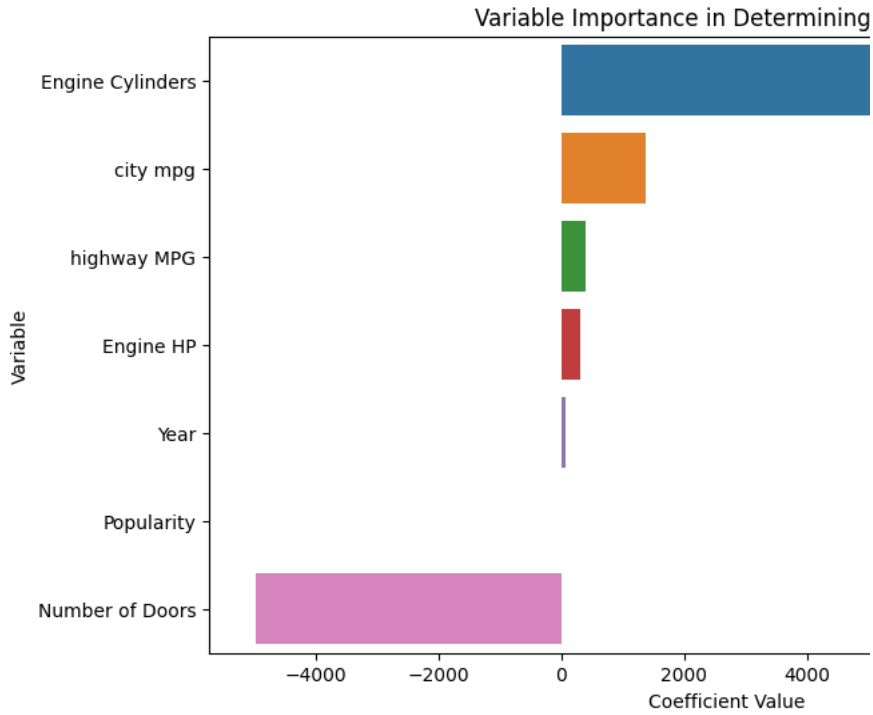
To Determine : What is the relationship between a car's engine power and its price?

```
sns.regplot(data=data, x='Engine HP', y='MSRP')

# Set labels and title
plt.xlabel('Engine Power')
plt.ylabel('Price')
plt.title('Engine Power vs. Price')

# Display the chart
plt.show()
```



To Determine : Which car features are most important in determining a car's price?

```
dataa=data.copy()
# Filter relevant columns with intergral values
relevant_columns = ['MSRP', 'Popularity', 'city mpg', 'highway MPG', 'Number of Doors', 'Engine Cylinders', 'Engine HP', 'Year
filtered_df = dataa[relevant_columns].dropna()

# Split the data into predictor variables (X) and the target variable (y)
X = filtered_df.drop('MSRP', axis=1)
y = filtered_df['MSRP']

# Perform linear regression
regression = LinearRegression()
regression.fit(X, y)

# Get the coefficient values for each variable
coefficient_values = pd.Series(regression.coef_, index=X.columns).sort_values(ascending=False)

# Create a bar chart to visualize the coefficient values
plt.figure(figsize=(10, 6))
sns.barplot(x=coefficient_values.values, y=coefficient_values.index, orient='h')
plt.xlabel('Coefficient Value')
plt.ylabel('Variable')
plt.title('Variable Importance in Determining Car Price')

# Display the chart
plt.show()
```



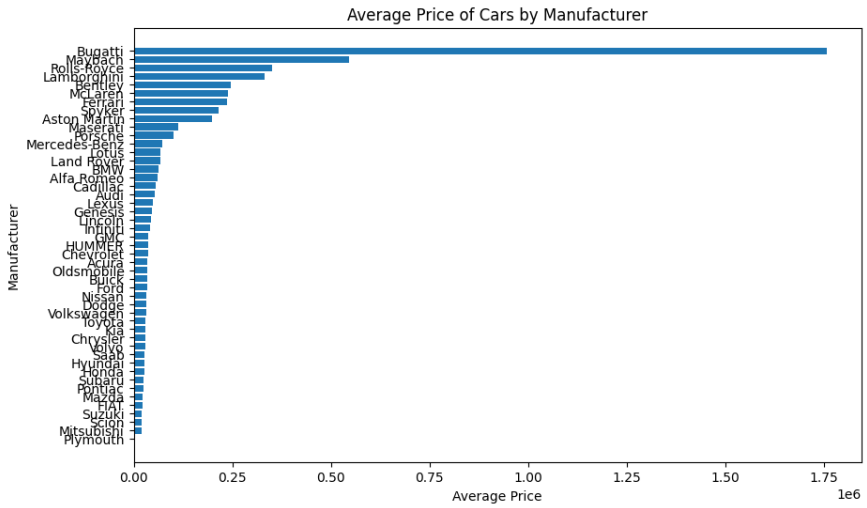
To Determine : How does the average price of a car vary across different manufacturers?

```
pivot_table = data.pivot_table(index='Make', values='MSRP', aggfunc='mean')

# Display the pivot table
print(pivot_table)
```

Make	MSRP
Acura	3.508749e+04
Alfa Romeo	6.160000e+04
Aston Martin	1.981235e+05
Audi	5.457412e+04
BMW	6.216256e+04
Bentley	2.471693e+05
Bugatti	1.757224e+06
Buick	3.377040e+04
Cadillac	5.636827e+04
Chevrolet	3.597082e+04
Chrysler	2.997887e+04
Dodge	3.118680e+04
FIAT	2.237066e+04
Ferrari	2.373838e+05
Ford	3.363895e+04
GMC	3.738575e+04
Genesis	4.661667e+04
HUMMER	3.646441e+04

```
Honda           2.697087e+04
Hyundai         2.727432e+04
Infiniti        4.264027e+04
Kia             3.014931e+04
Lamborghini     3.315673e+05
Land Rover      6.806709e+04
Lexus           4.754907e+04
Lincoln         4.356001e+04
Lotus           6.837714e+04
Maserati        1.136845e+05
Maybach         5.462219e+05
Mazda           2.324791e+04
McLaren         2.398050e+05
Mercedes-Benz   7.213503e+04
Mitsubishi      2.035282e+04
Nissan          3.290842e+04
Oldsmobile      3.486800e+04
Plymouth        4.189081e+03
Pontiac         2.472813e+04
Porsche         1.016224e+05
Rolls-Royce     3.511306e+05
Saab            2.787981e+04
Scion           2.039594e+04
Spyker          2.149900e+05
Subaru          2.583160e+04
Suzuki          2.120317e+04
Toyota          3.075312e+04
Volkswagen      3.089825e+04
Volvo           2.972468e+04
```

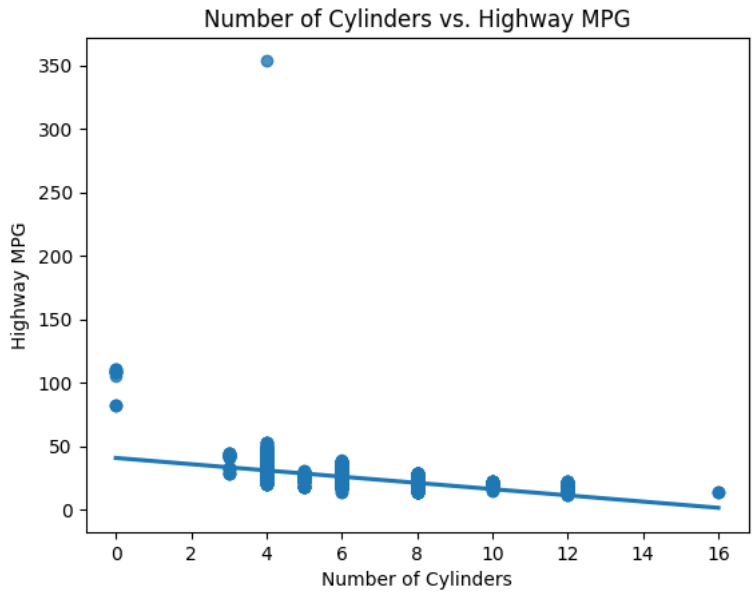
```
# Calculate the average price for each manufacturer
average_prices = data.groupby('Make')['MSRP'].mean().reset_index()

# Sort the data by average price in ascending order
sorted_df = average_prices.sort_values('MSRP', ascending=True)

# Create the bar chart
plt.figure(figsize=(10, 6))
plt.barh(sorted_df['Make'], sorted_df['MSRP'], color='tab:blue')
plt.xlabel('Average Price')
plt.ylabel('Manufacturer')
plt.title('Average Price of Cars by Manufacturer')

# Display the chart
plt.show()
```



To Determine : What is the relationship between fuel efficiency and the number of cylinders in a car's engine?

```
# Filter relevant columns
relevant_columns = ['Engine Cylinders', 'highway MPG']
filtered_df = data[relevant_columns]
```

```
# Create a scatter plot with a trendline
sns.regplot(data=filtered_df, x='Engine Cylinders', y='highway MPG')

# Set labels and title
plt.xlabel('Number of Cylinders')
plt.ylabel('Highway MPG')
plt.title('Number of Cylinders vs. Highway MPG')

# Display the chart
plt.show()

# Calculate the correlation coefficient
correlation_coefficient = filtered_df['Engine Cylinders'].corr(filtered_df['highway MPG'])

# Display the correlation coefficient
print("Correlation Coefficient:", correlation_coefficient)
```




```
Correlation Coefficient: -0.5978637854779184
```

To Determine : How does the distribution of car prices vary by brand and body style?

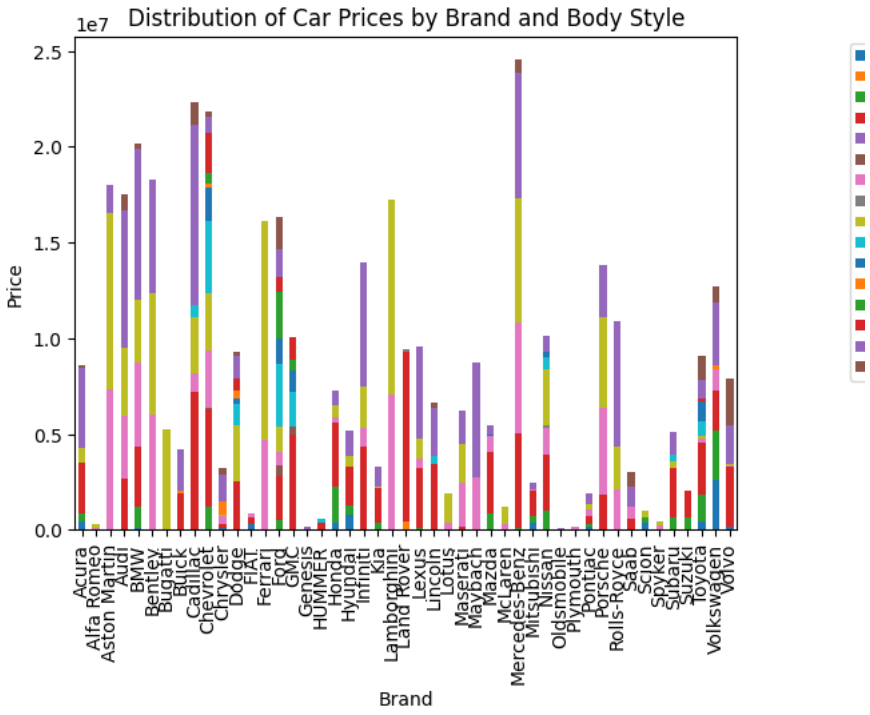
```
# Create pivot tables for total MSRP by brand and body style
pivot_brand1 = data.pivot_table(index='Make', columns='Vehicle Style', values='MSRP', aggfunc='sum', fill_value=0)

# Display the pivot tables
print("Total MSRP by Brand:")
print(pivot_brand1)
```

```
Total MSRP by Brand:
Vehicle Style  2dr Hatchback  2dr SUV  4dr Hatchback  4dr SUV  Cargo Minivan  \
Make
Acura                480917        0         357440  2663505              0
Alfa Romeo                0        0              0        0              0
Aston Martin              0        0              0        0              0
Audi                   4000        0              0  2674900              0
BMW                   80097        0        1103100  3160950              0
Bentley                   0        0              0        0              0
Bugatti                   0        0              0        0              0
Buick                     0        0              0  1944095              0
Cadillac                  0        0              0  7182555              0
Chevrolet              8000        0        1209735  5061440          45185
Chrysler              98805        0              0   181860              0
Dodge                 38000        0          16000  2462875          40685
FIAT                 325315        0              0   369305              0
Ferrari                   0        0              0        0              0
Ford                  24000        0         480155  2323070              0
GMC                       0    12785              0  4969304              0
Genesis                   0        0              0        0              0
HUMMER                    0        0              0   377490              0
Honda                413200        0        1846010  3356875              0
Hyundai              789650        0         528880  1994390              0
Infiniti                  0        0              0  4340200              0
Kia                       0        0         406960  1764140              0
Lamborghini               0        0              0        0              0
Land Rover                0   476394              0  8839200              0
Lexus                     0        0          94700  3152974              0
Lincoln                   0        0              0  3422570              0
Lotus                     0        0              0        0              0
Maserati                  0        0              0   155000              0
Maybach                   0        0              0        0              0
```

```
Mazda                 18000         0          853180  3175515              0
McLaren                   0         0               0        0              0
Mercedes-Benz             0         0          122800  4924810          28950
Mitsubishi           370169         0          334850  1352762              0
Nissan                14683         0         1023090  2890520              0
Oldsmobile                0         0               0        0              0
Plymouth              40000         0           14000        0              0
Pontiac              163505         0          162975   401550              0
Porsche               28827         0               0  1815200              0
Rolls-Royce               0         0               0        0              0
Saab                  12000         0           34586   541905              0
Scion                366325         0          282470        0              0
Spyker                    0         0               0        0              0
Subaru                12000         0          678060  2539900              0
Suzuki                44496         0          584387  1406621              0
Toyota               473750         0         1397750  2719805              0
Volkswagen          2606540         0         2566055  2084955              0
Volvo                157550         0               0  3131700              0

Vehicle Style  Cargo Van  Convertible  Convertible SUV     Coupe  \
Make
Acura                 0            0                0    793748
Alfa Romeo            0       129800                0    178200
Aston Martin          0      7321655                0   9258845
Audi                  0      3291405                0   3556290
BMW                   0      4403171                0   3304051
```

```
# Create the stacked column chart
plt.figure(figsize=(10, 6))
pivot_brand1.plot(kind='bar', stacked=True)

# Set labels and title
plt.xlabel('Brand')
plt.ylabel('Price')
plt.title('Distribution of Car Prices by Brand and Body Style')
plt.legend(loc='upper right',markerscale=0.5, fontsize=8,bbox_to_anchor=(1.5, 1))

# Display the chart
plt.show()
```

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



To Determine : Which car brands have the highest and lowest average MSRPs, and how does this vary by body style?

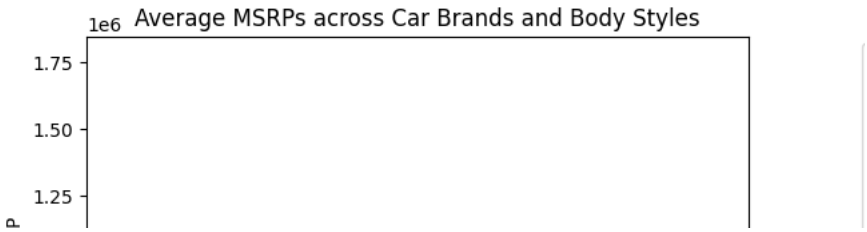
```
# Create pivot tables for average MSRP by brand and body style
pivot_brand2 = data.pivot_table(index='Make', columns='Vehicle Style', values='MSRP', aggfunc='mean',fill_value=0)
# Display the pivot tables
print("Average MSRP by Brand:")
print(pivot_brand2)
```

Vehicle Style	Cargo Minivan	Cargo Van	Convertible	Convertible SUV	\
Make					
Acura	0.0	0.000000	0.000000e+00	0.000000	
Alfa Romeo	0.0	0.000000	6.490000e+04	0.000000	
Aston Martin	0.0	0.000000	2.033793e+05	0.000000	
Audi	0.0	0.000000	7.002989e+04	0.000000	
BMW	0.0	0.000000	6.381407e+04	0.000000	
Bentley	0.0	0.000000	2.505362e+05	0.000000	
Bugatti	0.0	0.000000	0.000000e+00	0.000000	
Buick	0.0	0.000000	2.000000e+03	0.000000	
Cadillac	0.0	0.000000	7.040050e+04	0.000000	
Chevrolet	22592.5	30235.000000	6.283500e+04	0.000000	
Chrysler	0.0	0.000000	2.992036e+04	0.000000	
Dodge	20342.5	0.000000	2.000000e+03	0.000000	
FIAT	0.0	0.000000	2.591083e+04	0.000000	
Ferrari	0.0	0.000000	2.147187e+05	0.000000	
Ford	0.0	29893.611111	3.476224e+04	0.000000	
GMC	0.0	31863.214286	0.000000e+00	0.000000	
Genesis	0.0	0.000000	0.000000e+00	0.000000	
HUMMER	0.0	0.000000	0.000000e+00	0.000000	
Honda	0.0	0.000000	3.601929e+04	0.000000	
Hyundai	0.0	0.000000	0.000000e+00	0.000000	
Infiniti	0.0	0.000000	4.666905e+04	0.000000	
Kia	0.0	0.000000	0.000000e+00	0.000000	
Lamborghini	0.0	0.000000	3.364024e+05	0.000000	
Land Rover	0.0	0.000000	0.000000e+00	48577.000000	
Lexus	0.0	0.000000	5.245167e+04	0.000000	
Lincoln	0.0	0.000000	0.000000e+00	0.000000	
Lotus	0.0	0.000000	5.165750e+04	0.000000	
Maserati	0.0	0.000000	1.301646e+05	0.000000	
Maybach	0.0	0.000000	1.381375e+06	0.000000	
Mazda	0.0	0.000000	2.808081e+04	0.000000	
McLaren	0.0	0.000000	2.802250e+05	0.000000	
Mercedes-Benz	28950.0	0.000000	1.046175e+05	0.000000	
Mitsubishi	0.0	0.000000	3.259900e+04	0.000000	
Nissan	0.0	0.000000	3.907089e+04	43691.666667	
Oldsmobile	0.0	0.000000	0.000000e+00	0.000000	
Plymouth	0.0	0.000000	2.854367e+04	0.000000	
Pontiac	0.0	0.000000	2.472400e+04	0.000000	
Porsche	0.0	0.000000	1.155022e+05	0.000000	
Rolls-Royce	0.0	0.000000	4.282730e+05	0.000000	
Saab	0.0	0.000000	2.875582e+04	0.000000	
Scion	0.0	0.000000	0.000000e+00	0.000000	
Spyker	0.0	0.000000	2.199900e+05	0.000000	
Subaru	0.0	0.000000	0.000000e+00	0.000000	
Suzuki	0.0	0.000000	0.000000e+00	0.000000	
Toyota	0.0	0.000000	2.539500e+04	0.000000	
Volkswagen	0.0	0.000000	3.178614e+04	0.000000	
Volvo	0.0	0.000000	4.053333e+04	0.000000	

Vehicle Style	Coupe	Crew Cab Pickup	Extended Cab Pickup	\
Make				

```
#Create the clustered column chart
plt.figure(figsize=(10, 6))
pivot_brand2.plot(kind='bar')

# Set labels and title
plt.xlabel('Brand')
plt.ylabel('Average MSRP')
plt.title('Average MSRPs across Car Brands and Body Styles')
plt.legend(loc='upper right',markerscale=0.5, fontsize=8,bbox_to_anchor=(1.5, 1))

# Display the chart
plt.show()
```

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

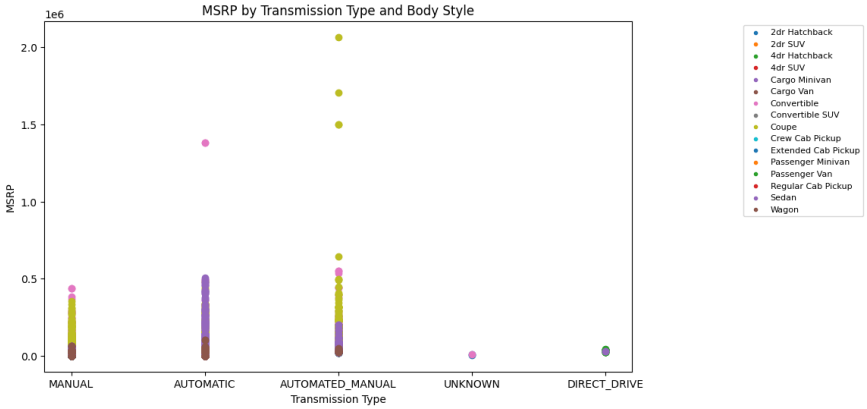


To Determine : How do the different feature such as transmission type affect the MSRP, and how does this vary by body style?

```
# Create a pivot table for the average MSRP by transmission type and body style
pivot_table1 = data.pivot_table(index='Transmission Type', columns='Vehicle Style', values='MSRP', aggfunc='mean')

# Display the pivot table
print("Average MSRP by Transmission Type and Body Style: \n")
print(pivot_table1)
```

```
Average MSRP by Transmission Type and Body Style:

Vehicle Style      2dr Hatchback  2dr SUV  4dr Hatchback       4dr SUV  \
Transmission Type
AUTOMATED_MANUAL    27470.416667      NaN   29347.045455  40451.153846
AUTOMATIC           20784.099010  35894.5   23888.735294  42971.875755
DIRECT_DRIVE                NaN      NaN   34511.923077           NaN
MANUAL              12840.655556  29222.5   17500.363636  23131.333333
UNKNOWN              7361.500000      NaN            NaN           NaN

Vehicle Style      Cargo Minivan     Cargo Van    Convertible  \
Transmission Type
AUTOMATED_MANUAL             NaN           NaN  134527.794872
AUTOMATIC                22964.0  30724.705882  112168.390438
DIRECT_DRIVE                 NaN           NaN            NaN
MANUAL                       NaN           NaN   69716.931159
UNKNOWN                      NaN           NaN    9567.000000

Vehicle Style      Convertible SUV          Coupe  Crew Cab Pickup  \
Transmission Type
AUTOMATED_MANUAL               NaN  245588.357143              NaN
AUTOMATIC             46134.333333   77070.494792     39606.576803
DIRECT_DRIVE                   NaN            NaN              NaN
MANUAL                         NaN   64786.502427     27361.000000
UNKNOWN                        NaN            NaN              NaN

Vehicle Style      Extended Cab Pickup  Passenger Minivan  Passenger Van  \
Transmission Type
AUTOMATED_MANUAL                   NaN                NaN            NaN
AUTOMATIC                33140.142012       26707.065574       35963.15
DIRECT_DRIVE                       NaN                NaN            NaN
MANUAL                   10650.555556                NaN            NaN
UNKNOWN                            NaN                NaN            NaN

Vehicle Style      Regular Cab Pickup         Sedan         Wagon
Transmission Type
AUTOMATED_MANUAL                  NaN  51531.081871  31985.277778
AUTOMATIC                29209.756944  56808.400000  33376.856115
DIRECT_DRIVE                      NaN  27822.500000           NaN
MANUAL                   18044.812500  22541.172414  23704.433962
UNKNOWN                           NaN           NaN           NaN
```

```
# Create a scatter plot chart
plt.figure(figsize=(10, 6))

# Iterate over each body style
for body_style, group in data.groupby('Vehicle Style'):
  plt.scatter(group['Transmission Type'], group['MSRP'], label=body_style)
# Set labels and title
plt.xlabel('Transmission Type')
plt.ylabel('MSRP')
plt.title('MSRP by Transmission Type and Body Style')
plt.legend(loc='upper right',markerscale=0.5, fontsize=8,bbox_to_anchor=(1.4, 1))

# Display the chart
plt.show()
```



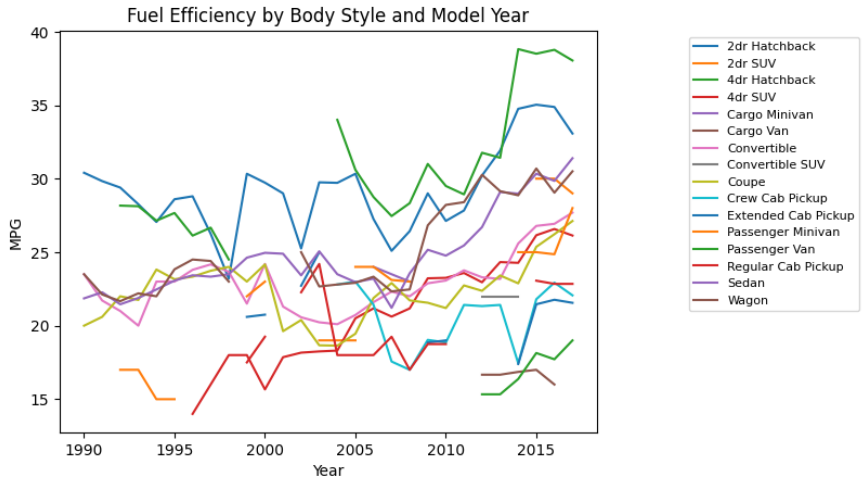
To Determine : How does the fuel efficiency of cars vary across different body styles and model years?

```
# Create a pivot table for the average MPG by body style and model year
pivot_table2 = data.pivot_table(index='Year', columns='Vehicle Style', values='highway MPG', aggfunc='mean')

# Display the pivot table
print("Average MPG by Body Style and Model Year:")
print(pivot_table2)
```

```
2015          30.337696  30.687500
2016          29.847500  29.046512
2017          31.390000  30.500000
```

```
# Create a line chart
plt.figure(figsize=(10, 6))

pivot_table2.plot()

# Set labels and title
plt.xlabel('Year')
plt.ylabel('MPG')
plt.title('Fuel Efficiency by Body Style and Model Year')
plt.legend(loc='upper right',markerscale=0.5, fontsize=8,bbox_to_anchor=(1.5, 1))

# Display the chart
plt.show()
```

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



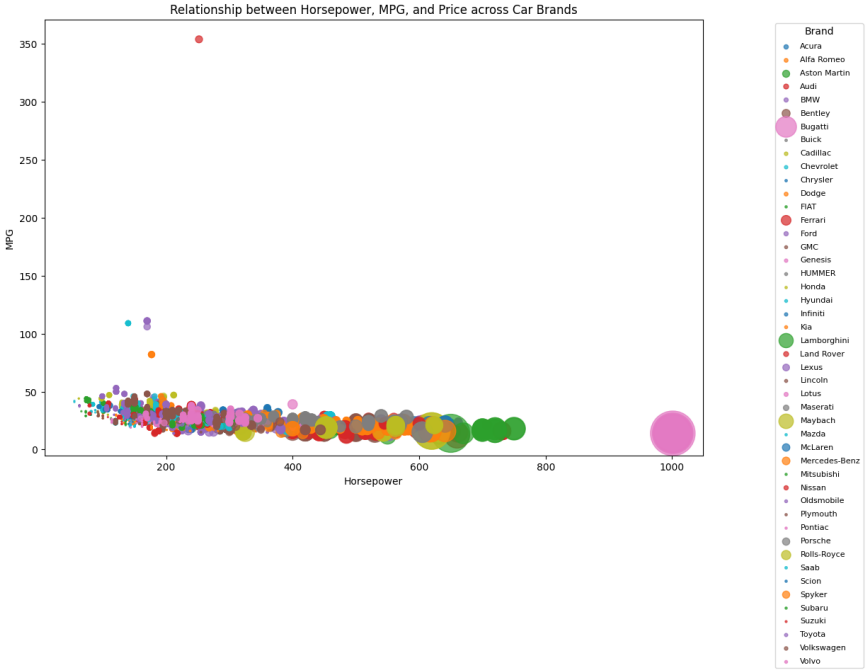
To Determine : How does the car's horsepower, MPG, and price vary across different Brands?

```
# Create a pivot table for the average horsepower, MPG, and MSRP by car brand
pivot_table3 = data.pivot_table(index='Make', values=['Engine HP', 'highway MPG', 'MSRP'], aggfunc='mean')

# Display the pivot table
print("Average Horsepower, MPG, and MSRP by Car Brand:")
print(pivot_table3)

scatter_df= pivot_table3.reset_index()
```

```
Average Horsepower, MPG, and MSRP by Car Brand:
                Engine HP          MSRP  highway MPG
Make
Acura          244.963415  3.508749e+04    28.219512
Alfa Romeo     237.000000  6.160000e+04    34.000000
Aston Martin   483.758242  1.981235e+05    18.934066
Audi           280.000000  5.457412e+04    28.928349
BMW            329.620370  6.216256e+04    29.126543
Bentley        533.851351  2.471693e+05    18.905405
Bugatti       1001.000000  1.757224e+06    14.000000
Buick          228.112000  3.377040e+04    27.512000
Cadillac       332.795455  5.636827e+04    25.244949
Chevrolet      284.761513  3.597082e+04    26.276316
Chrysler       241.388889  2.997887e+04    26.620370
Dodge          295.513423  3.118680e+04    22.996644
FIAT           146.894737  2.237066e+04    34.184211
Ferrari        509.911765  2.373838e+05    15.720588
Ford           276.144330  3.363895e+04    24.793814
GMC            279.985185  3.738575e+04    22.362963
Genesis        347.333333  4.661667e+04    25.333333
HUMMER         261.235294  3.646441e+04    17.294118
Honda          195.732342  2.697087e+04    32.234201
Hyundai        221.626316  2.727432e+04    28.510526
Infiniti       310.676829  4.264027e+04    24.795732
Kia            232.844037  3.014931e+04    29.036697
Lamborghini    614.076923  3.315673e+05    18.019231
Land Rover     322.517986  6.806709e+04    21.978417
Lexus          277.415842  4.754907e+04    25.876238
```

```
Lincoln         286.125000  4.356001e+04    24.144737
Lotus           271.535714  6.837714e+04    26.107143
Maserati        419.545455  1.136845e+05    20.163636
Maybach         590.500000  5.462219e+05    16.000000
Mazda           177.747863  2.324791e+04    29.311966
McLaren         610.400000  2.398050e+05    22.200000
Mercedes-Benz   353.500000  7.213503e+04    24.555882
Mitsubishi      173.441667  2.035282e+04    28.316667
Nissan          264.444805  3.290842e+04    26.970779
Oldsmobile      250.000000  3.486800e+04    23.666667
Plymouth        139.216216  4.189081e+03    26.297297
Pontiac         236.324675  2.472813e+04    25.194805
Porsche         392.794118  1.016224e+05    25.367647
Rolls-Royce     487.548387  3.511306e+05    19.129032
Saab            221.174312  2.787981e+04    26.376147
Scion           155.708333  2.039594e+04    32.812500
Spyker          400.000000  2.149900e+05    18.000000
Subaru          203.568528  2.583160e+04    28.304569
Suzuki          183.989583  2.120317e+04    25.864583
Toyota          222.949153  3.075312e+04    29.667797
Volkswagen      203.155718  3.089825e+04    32.352798
Volvo           234.560150  2.972468e+04    27.263158
```

```
# Create a bubble chart
plt.figure(figsize=(12, 8))

# Iterate over each brand
for brand, group in data.groupby('Make'):
    plt.scatter(group['Engine HP'], group['highway MPG'], s=group['MSRP'] / 1000, alpha=0.7, label=brand)

# Set labels and title
plt.xlabel('Horsepower')
plt.ylabel('MPG')
plt.title('Relationship between Horsepower, MPG, and Price across Car Brands')
plt.legend()

# Adjust the size of the bubbles in the legend
handles, labels = plt.gca().get_legend_handles_labels()
sizes = [40, 80, 120]  # Adjust the sizes as per your preference
plt.legend(handles, labels, scatterpoints=1, loc='upper right', markerscale=0.5, fontsize=8, title='Brand', labelspacing=0.8,

# Display the chart
plt.show()
```



✓ 0s completed at 02:29