```
import pandas as pd
cars_data = pd.read_csv('Cars_data.csv')
```

Data Understanding and Initial Exploration

```
cars_data.shape
```

```
(11914, 16)
```

```
# Display basic information about the dataset

cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Make               11914 non-null  object 
 1   Model              11914 non-null  object 
 2   Year               11914 non-null  int64  
 3   Engine Fuel Type   11911 non-null  object 
 4   Engine HP          11845 non-null  float64
 5   Engine Cylinders   11884 non-null  float64
 6   Transmission Type  11914 non-null  object 
 7   Driven_Wheels      11914 non-null  object 
 8   Number of Doors    11908 non-null  float64
 9   Market Category    8172 non-null   object 
 10  Vehicle Size       11914 non-null  object 
 11  Vehicle Style      11914 non-null  object 
 12  highway MPG        11914 non-null  int64  
 13  city mpg           11914 non-null  int64  
 14  Popularity         11914 non-null  int64  
 15  MSRP               11914 non-null  int64  
dtypes: float64(3), int64(5), object(8)
memory usage: 1.5+ MB
```

```
# Summary statistics of numerical columns

cars_data.describe()
```


```
{"summary":"{\n  \"name\": \"cars_data\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"Year\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3670.4255658356396,\n        \"min\": 7.579739887595646,\n
\"max\": 11914.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          2010.384337753903,\n          2015.0,\n
11914.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Engine HP\",\n      \"properties\": {\n        \"dtype\":
```

\"number\",\n   \"std\": 4091.8968322581513,\n   \"min\":
55.0,\n   \"max\": 11845.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   249.38607007176023,\n   227.0,\n
11845.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Engine Cylinders\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4199.640689133794,\n   \"min\":
0.0,\n   \"max\": 11884.0,\n   \"num_unique_values\": 7,\n
\"samples\": [\n   11884.0,\n   5.628828677213059,\n
6.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Number of Doors\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4209.087754200279,\n   \"min\":
0.8813153865835297,\n   \"max\": 11908.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
3.4360933825999327,\n   4.0,\n   0.8813153865835297\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"highway MPG\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
4189.644501132749,\n   \"min\": 8.863000766979342,\n
\"max\": 11914.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   26.637485311398354,\n   26.0,\n
11914.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"city mpg\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4200.901924587632,\n   \"min\":
7.0,\n   \"max\": 11914.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   19.73325499412456,\n   18.0,\n
11914.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Popularity\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 3956.693898024036,\n   \"min\":
2.0,\n   \"max\": 11914.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   1554.9111969111968,\n   1385.0,\n
11914.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"MSRP\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 720143.0484808605,\n   \"min\": 2000.0,\n
\"max\": 2065902.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   40594.737032063116,\n   29995.0,\n
11914.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
# Checking for missing values

cars_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      3742
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

```
# To display first few rows

cars_data.head(5)
```


```
{"summary":"{\n  \"name\": \"cars_data\",\n  \"rows\": 11914,\n
\"fields\": [\n    {\n      \"column\": \"Make\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 48,\n        \"samples\": [\n
\"Chevrolet\",\n          \"Land Rover\",\n          \"Bentley\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Model\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
915,\n        \"samples\": [\n          \"G35\",\n          \"Van\",\n
\"Flex\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7,\n        \"min\": 1990,\n        \"max\": 2017,\n
\"num_unique_values\": 28,\n        \"samples\": [\n          1990,\n
2000,\n          2016\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Engine Fuel Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"flex-fuel (premium unleaded
required/E85)\",\n          \"regular unleaded\",\n
\"electric\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Engine HP\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 109.19187025917257,\n        \"min\":
55.0,\n        \"max\": 1001.0,\n        \"num_unique_values\": 356,\n
\"samples\": [\n          145.0,\n          201.0,\n          219.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Engine Cylinders\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.7805593482463664,\n        \"min\": 0.0,\n        \"max\": 16.0,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          3.0,\n
```

```
4.0,\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Transmission Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"AUTOMATIC\",\n          \"UNKNOWN\",\n
\"AUTOMATED_MANUAL\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Driven_Wheels\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"front wheel drive\",\n          \"four wheel drive\",\n
\"rear wheel drive\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Number of Doors\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.8813153865835297,\n        \"min\":
2.0,\n        \"max\": 4.0,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          2.0,\n          4.0,\n          3.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Market Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 71,\n        \"samples\": [\n
\"Exotic,Luxury,Performance\",\n          \"Factory Tuner,Luxury,High-
Performance\",\n          \"Crossover,Flex Fuel,Luxury\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle Size\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Compact\",\n          \"Midsize\",\n          \"Large\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Vehicle Style\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 16,\n        \"samples\": [\n
\"Coupe\",\n          \"Convertible\",\n          \"2dr Hatchback\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"highway MPG\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8,\n        \"min\": 12,\n        \"max\": 354,\n
\"num_unique_values\": 59,\n        \"samples\": [\n          26,\n
20,\n          354\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"city mpg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 8,\n        \"min\": 7,\n
\"max\": 137,\n        \"num_unique_values\": 69,\n
\"samples\": [\n          9,\n          19,\n          41\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Popularity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1441,\n        \"min\": 2,\n        \"max\": 5657,\n
\"num_unique_values\": 48,\n        \"samples\": [\n          1385,\n
258,\n          520\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MSRP\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 60109,\n        \"min\": 2000,\n        \"max\": 2065902,\n
\"num_unique_values\": 6049,\n        \"samples\": [\n
42610,\n          32030,\n          89995\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"cars_data"}
```

Data Cleaning

```
# Replace null values in 'Market Category' with 'Not Specified'

cars_data['Market Category'].fillna('Not Specified', inplace = True)
```

```
<ipython-input-7-b9879e57606b>:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  cars_data['Market Category'].fillna('Not Specified', inplace = True)
```

```
# Handling missing values

cars_data.dropna(inplace = True)

# Checking for duplicates

cars_data.drop_duplicates(inplace = True)

# Convert 'Year' to a categorical variable if needed

cars_data['Year'] = cars_data['Year'].astype(str)

# Checking for missing values after cleaning

cars_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
```

```
# To drop Irrevalent column

cars_data.drop(columns = ['Engine Fuel Type'], inplace = True)

# To Verify changes

cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11097 entries, 0 to 11913
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Make               11097 non-null  object 
 1   Model              11097 non-null  object 
 2   Year               11097 non-null  object 
 3   Engine HP          11097 non-null  float64
 4   Engine Cylinders   11097 non-null  float64
 5   Transmission Type  11097 non-null  object 
 6   Driven_Wheels      11097 non-null  object 
 7   Number of Doors    11097 non-null  float64
 8   Market Category    11097 non-null  object 
 9   Vehicle Size       11097 non-null  object 
 10  Vehicle Style      11097 non-null  object 
 11  highway MPG        11097 non-null  int64  
 12  city mpg           11097 non-null  int64  
 13  Popularity         11097 non-null  int64  
 14  MSRP               11097 non-null  int64  
dtypes: float64(3), int64(4), object(8)
memory usage: 1.4+ MB
```

```
cars_data.shape
```

```
(11097, 15)
```

Data Transformation

```
# Creating a new feature 'Age' of the car

cars_data['Age'] = 2024 - cars_data['Year'].astype(int)

# Categorizing 'Engine HP' into bins

bins = [0,200,400,600]
labels = ['Low HP','Medium HP','High HP']
cars_data['HP Category'] = pd.cut(cars_data['Engine HP'], bins = bins,
labels = labels)

# verify Transformation
cars_data.head(5)
```

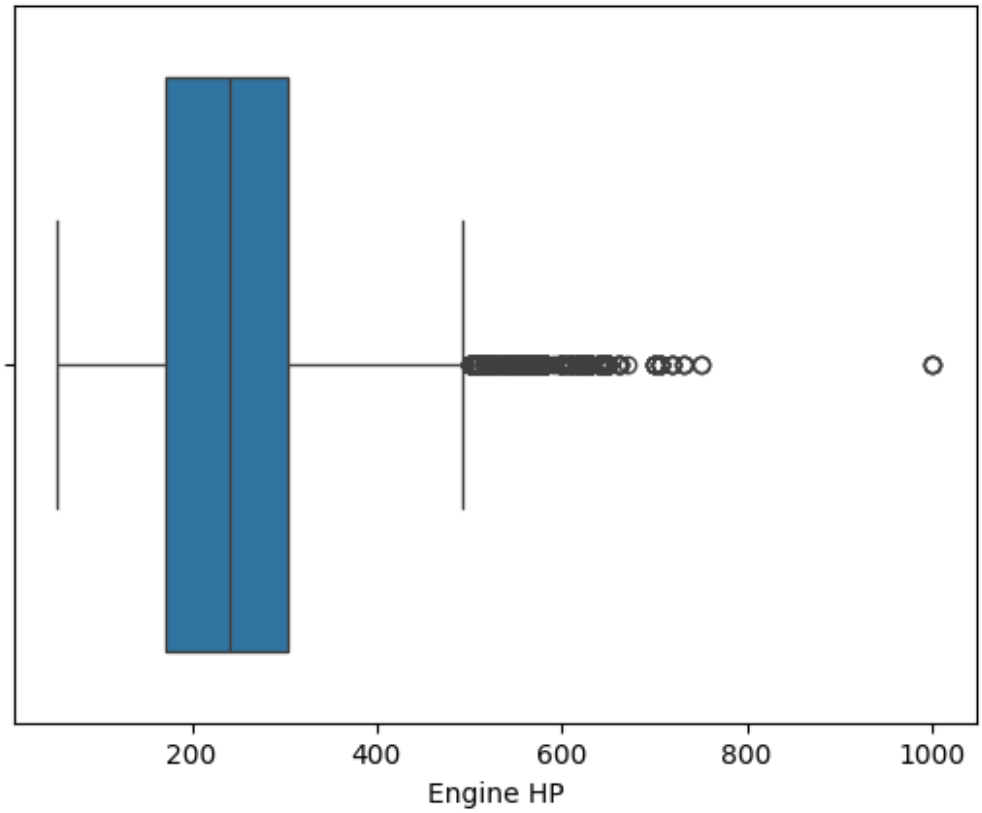

```
{"summary":"{\n  \"name\": \"cars_data\",\n  \"rows\": 11097,\n
\"fields\": [\n    {\n      \"column\": \"Make\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 47,\n        \"samples\": [\n
\"Chevrolet\",\n          \"Aston Martin\",\n          \"Bentley\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Model\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
904,\n        \"samples\": [\n          \"9000\",\n          \"IS
300\",\n          \"Continental Supersports\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Year\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"num_unique_values\": 28,\n
\"samples\": [\n          \"1990\",\n          \"2000\",\n
\"2016\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Engine HP\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 110.16487131183096,\n        \"min\":
55.0,\n        \"max\": 1001.0,\n        \"num_unique_values\": 355,\n
\"samples\": [\n          145.0,\n          201.0,\n          219.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Engine Cylinders\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.7661491513470076,\n        \"min\": 0.0,\n        \"max\": 16.0,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          3.0,\n
4.0,\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Transmission Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"AUTOMATIC\",\n          \"DIRECT_DRIVE\",\n
\"AUTOMATED_MANUAL\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Driven_Wheels\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"front wheel drive\",\n          \"four wheel drive\",\n
\"rear wheel drive\"\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Number of Doors\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.8746736913806462,\n        \"min\":
2.0,\n        \"max\": 4.0,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          2.0,\n          4.0,\n          3.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Market Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 71,\n        \"samples\": [\n
\"Exotic,Luxury,High-Performance\",\n          \"Factory
Tuner,Luxury,High-Performance\",\n          \"Crossover,Flex
Fuel,Luxury,Performance\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Vehicle Size\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Compact\",\n          \"Midsize\",\n
\"Large\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Vehicle Style\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 16,\n
\"samples\": [\n          \"Coupe\",\n          \"Convertible\",\n
\"2dr Hatchback\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"highway MPG\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7,\n        \"min\": 12,\n
\"max\": 354,\n        \"num_unique_values\": 44,\n
\"samples\": [\n          13,\n          15,\n          37\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"city mpg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 6,\n        \"min\": 7,\n
\"max\": 137,\n        \"num_unique_values\": 50,\n
\"samples\": [\n          28,\n          129,\n          40\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Popularity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1443,\n        \"min\": 2,\n
\"max\": 5657,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          1385,\n          259,\n          520\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MSRP\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 61730,\n
\"min\": 2000,\n        \"max\": 2065902,\n
\"num_unique_values\": 6013,\n        \"samples\": [\n
228625,\n          35080,\n          21590\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 7,\n        \"min\": 7,\n
\"max\": 34,\n        \"num_unique_values\": 28,\n        \"samples\":
```

```
[\n          34,\n          24,\n          8\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"HP Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"Medium
HP\",\n          \"Low HP\",\n          \"High HP\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"cars_data"}
```

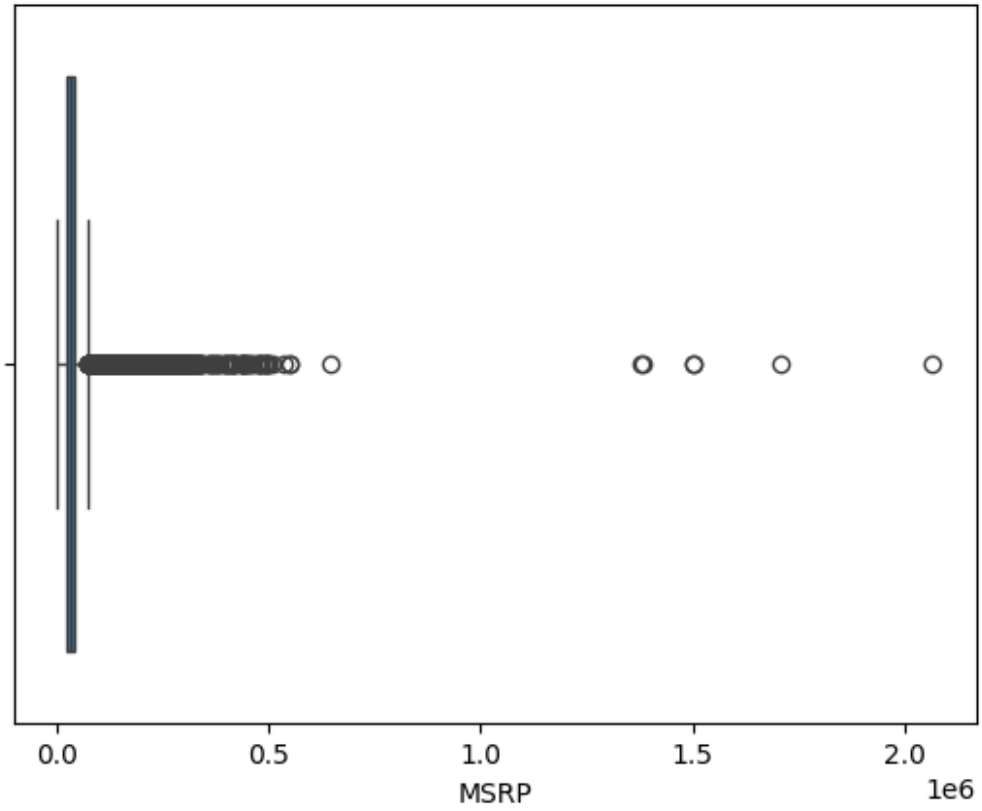

Exploratory Data Analysis (EDA)

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

# Checking Outliners

plt.figure()
sns.boxplot(x=cars_data['Engine HP'])
plt.show()
```




```
# Checking outliners

plt.figure()
```

```
sns.boxplot(x=cars_data['MSRP'])
plt.show()
```



Removing outliers

```
# Creating a new list for all Numberic columns

numberic_column = cars_data.select_dtypes(include =
['int64','float64']).columns.tolist()
numberic_column
```

```
['Engine HP',
 'Engine Cylinders',
 'Number of Doors',
 'highway MPG',
 'city mpg',
 'Popularity',
 'MSRP',
 'Age']
```

```
# Using IQR to find outliners

Q1 = cars_data[numberic_column].quantile(0.25)
Q3 = cars_data[numberic_column].quantile(0.75)
```

```
IQR = Q3 - Q1
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

IQR
```

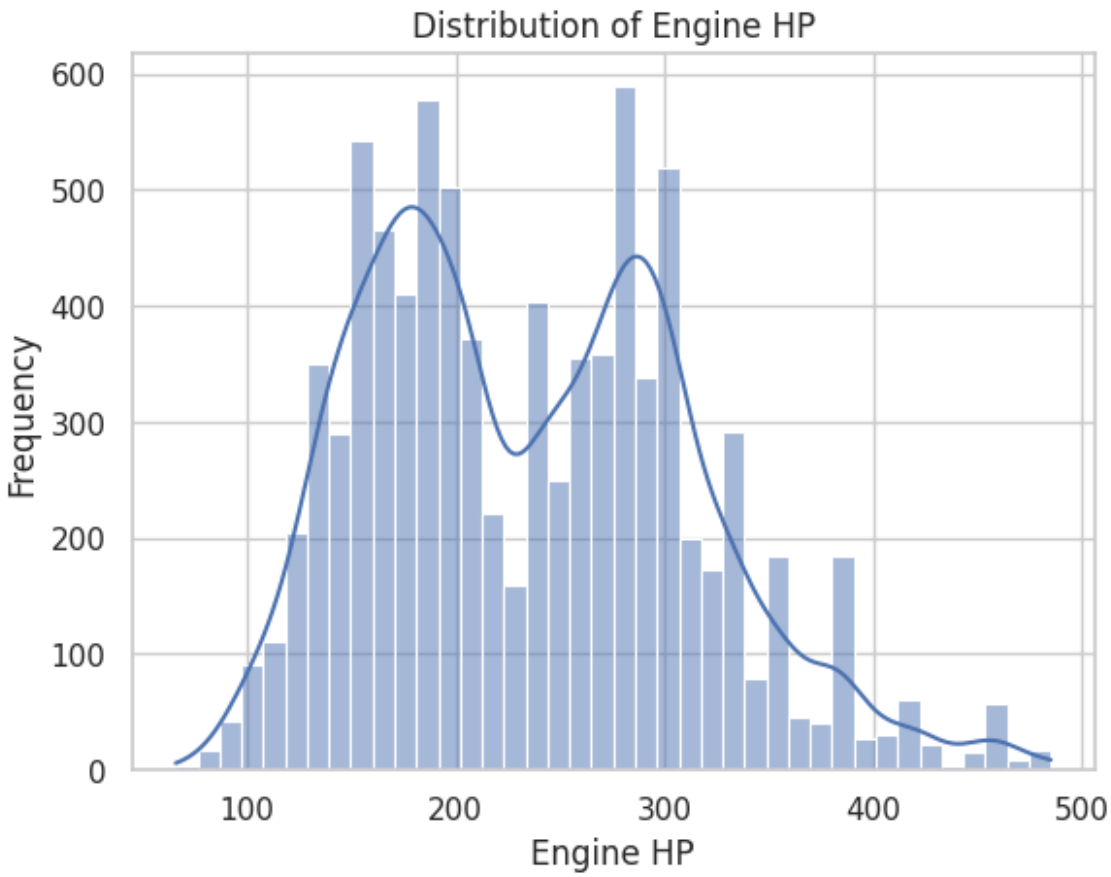
```
Engine HP             131.0
Engine Cylinders        2.0
Number of Doors         2.0
highway MPG             8.0
city mpg                6.0
Popularity           1460.0
MSRP                21480.0
Age                     9.0
dtype: float64
```

```
# giving new variable to data with no outliners

cars_data2 = cars_data[~((cars_data[numberic_column]< lower_limit) |
(cars_data[numberic_column]> upper_limit)).any(axis=1)]

cars_data2.shape,cars_data.shape
```

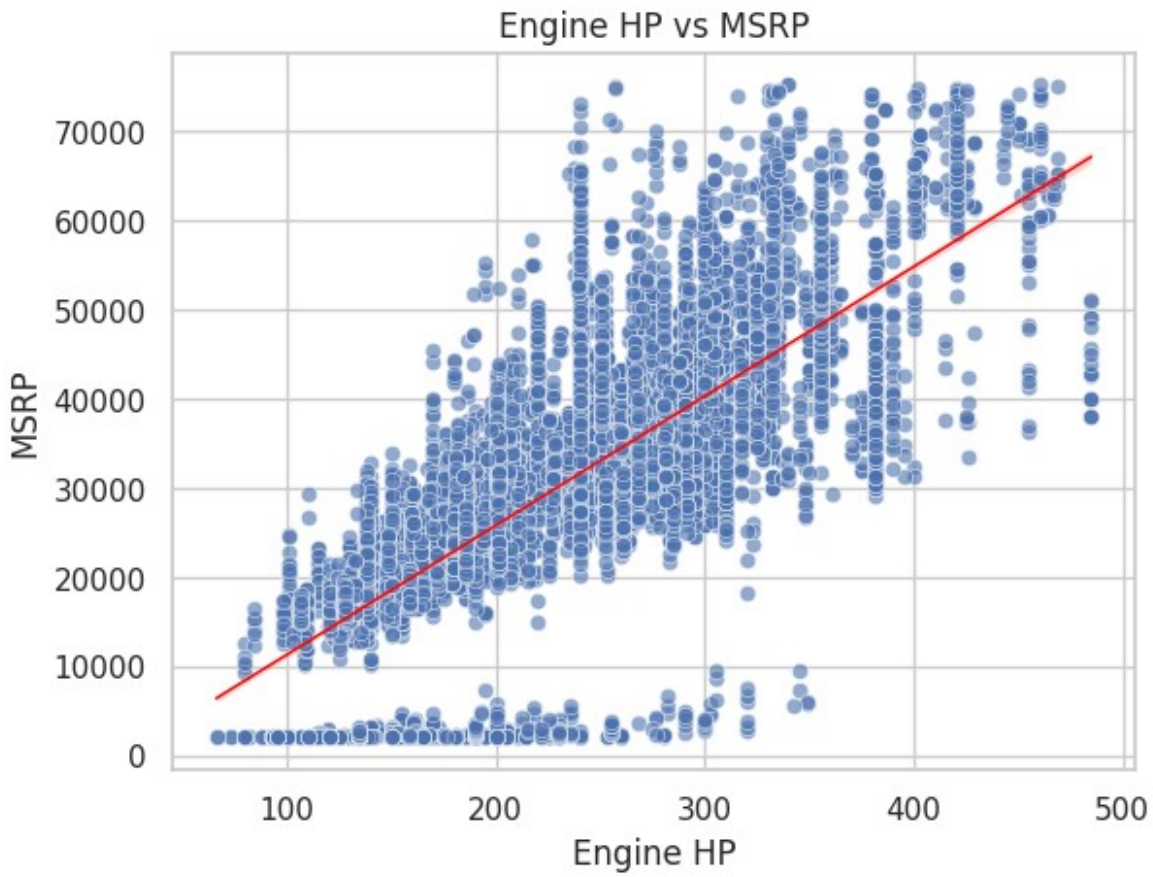
```
((8612, 17), (11097, 17))
```

```
# Frequency distribution of Engine HP
sns.set_style('whitegrid')
plt.figure()
sns.histplot(cars_data2['Engine HP'], kde = True, bins = 40)
plt.title('Distribution of Engine HP')
plt.xlabel('Engine HP')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```

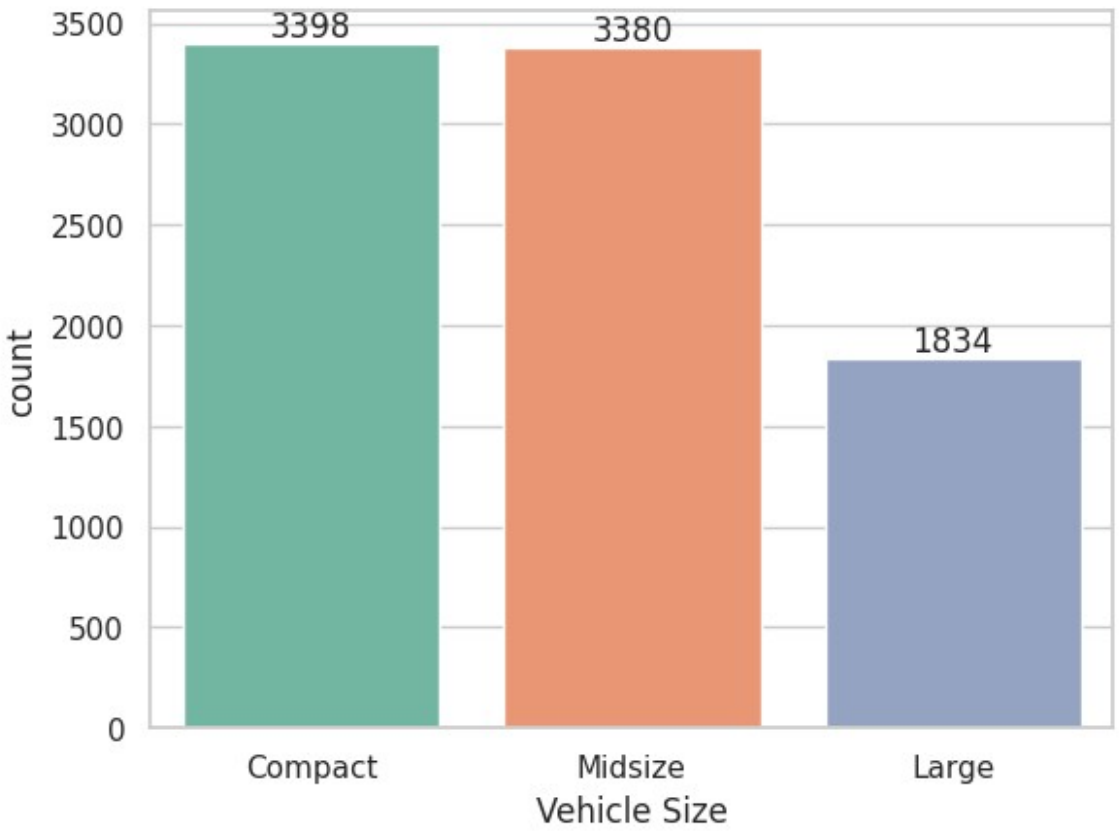



```
# Correlation between 'HP' and 'Price'

plt.figure()
sns.set(style="whitegrid")
sns.scatterplot(x = 'Engine HP', y = 'MSRP', data = cars_data2, alpha
= 0.6)
sns.regplot(x='Engine HP', y='MSRP', data=cars_data2, scatter=False,
color='red', line_kws={"lw":1})
plt.title('Engine HP vs MSRP')

Text(0.5, 1.0, 'Engine HP vs MSRP')
```



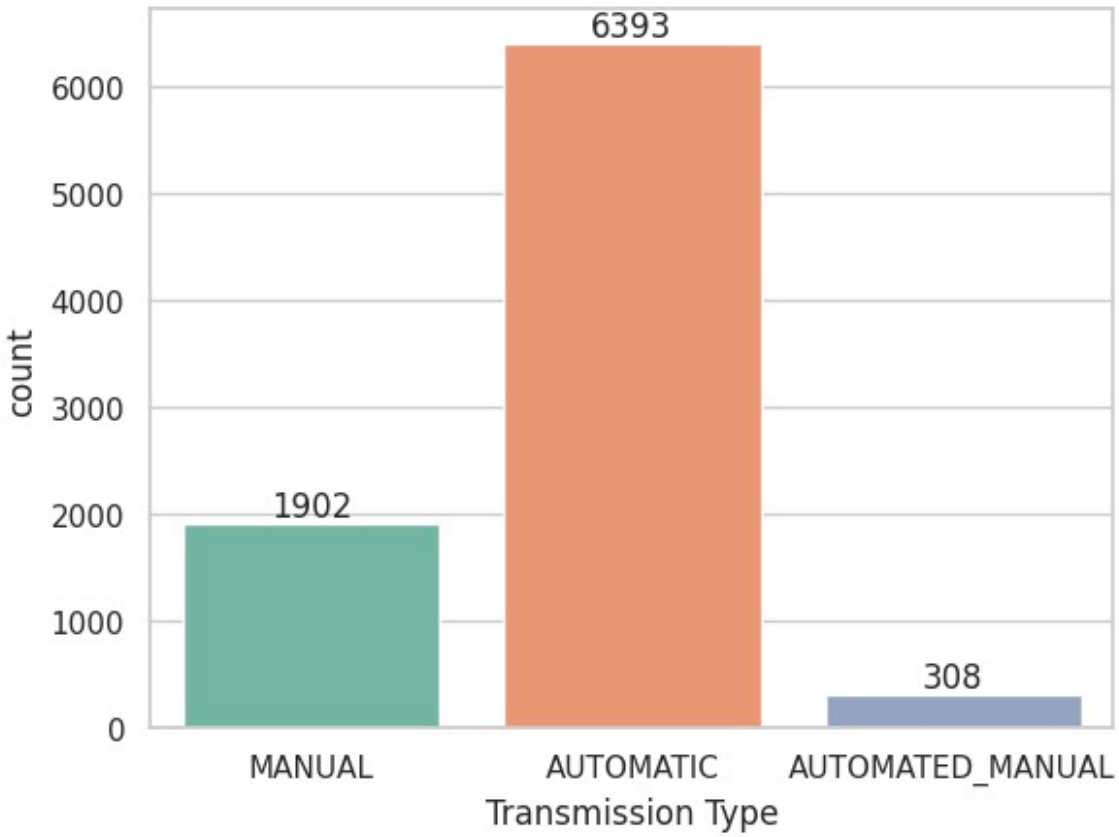

```
# Count plot of 'Vehicle Size'

plt.figure()
xa = sns.countplot(x = 'Vehicle Size', data = cars_data2,
palette='Set2' )

for bars in xa.containers:
    xa.bar_label(bars)
```

```
<ipython-input-43-b5bdef42dbdd>:4: 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.

  xa = sns.countplot(x = 'Vehicle Size', data = cars_data2,
palette='Set2' )
```

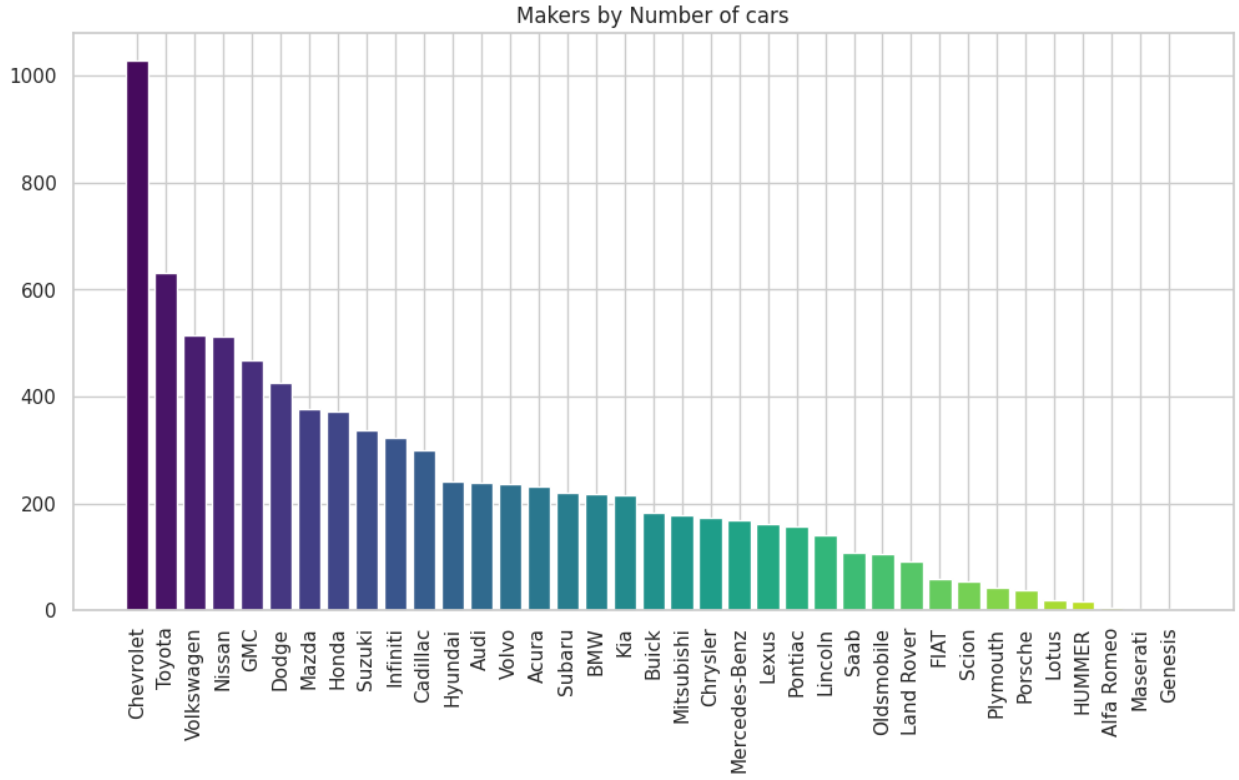



```
# countplot on transmission

fcd = cars_data2[cars_data2['Transmission Type'] != 'UNKNOWN']
plt.figure()
ax = sns.countplot(x = 'Transmission Type', data = fcd,
palette='Set2')

for bars in ax.containers:
    ax.bar_label(bars)
```

```
<ipython-input-44-924fba15ff20>: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.

  ax = sns.countplot(x = 'Transmission Type', data = fcd,
palette='Set2')
```

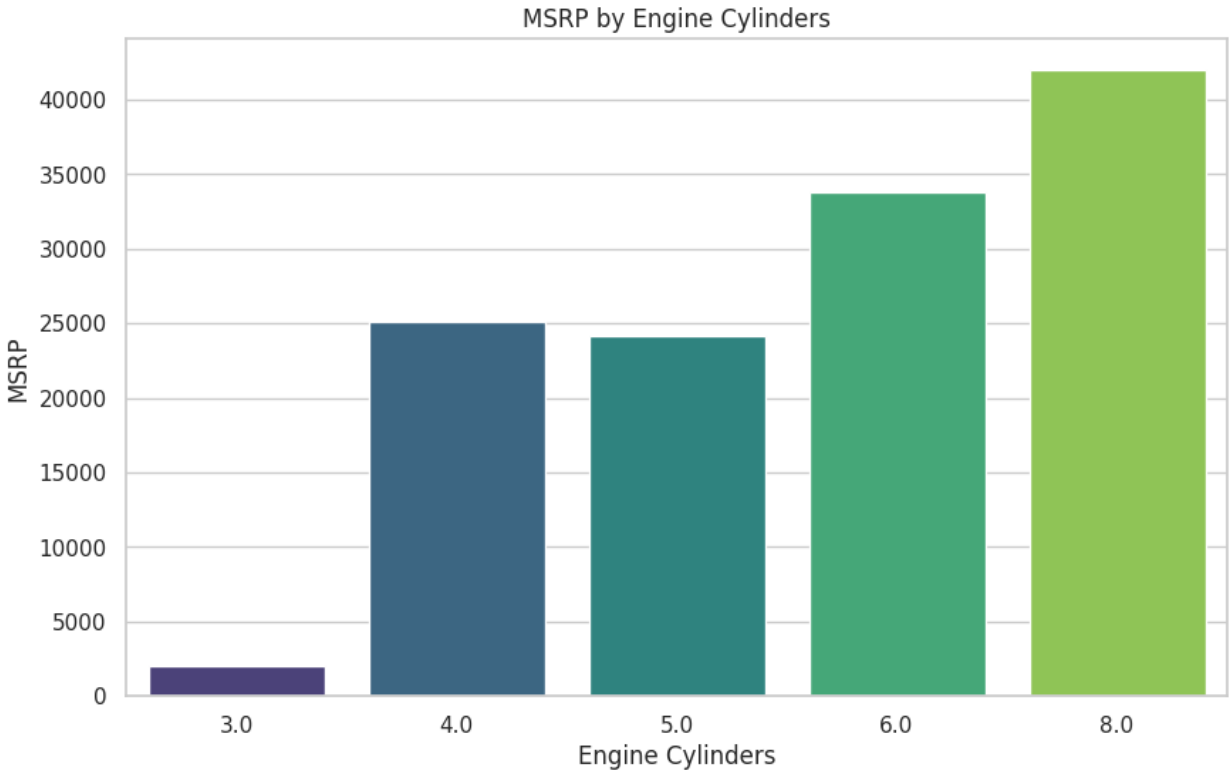



```
# Makers by number of cars
make_count = cars_data2['Make'].value_counts()
plt.figure(figsize=(12,6))
bars = plt.bar(make_count.index, make_count.values,
color=sns.color_palette('viridis', len(make_count)))
plt.xticks(rotation=90)
plt.title('Makers by Number of cars')
plt.show()
```



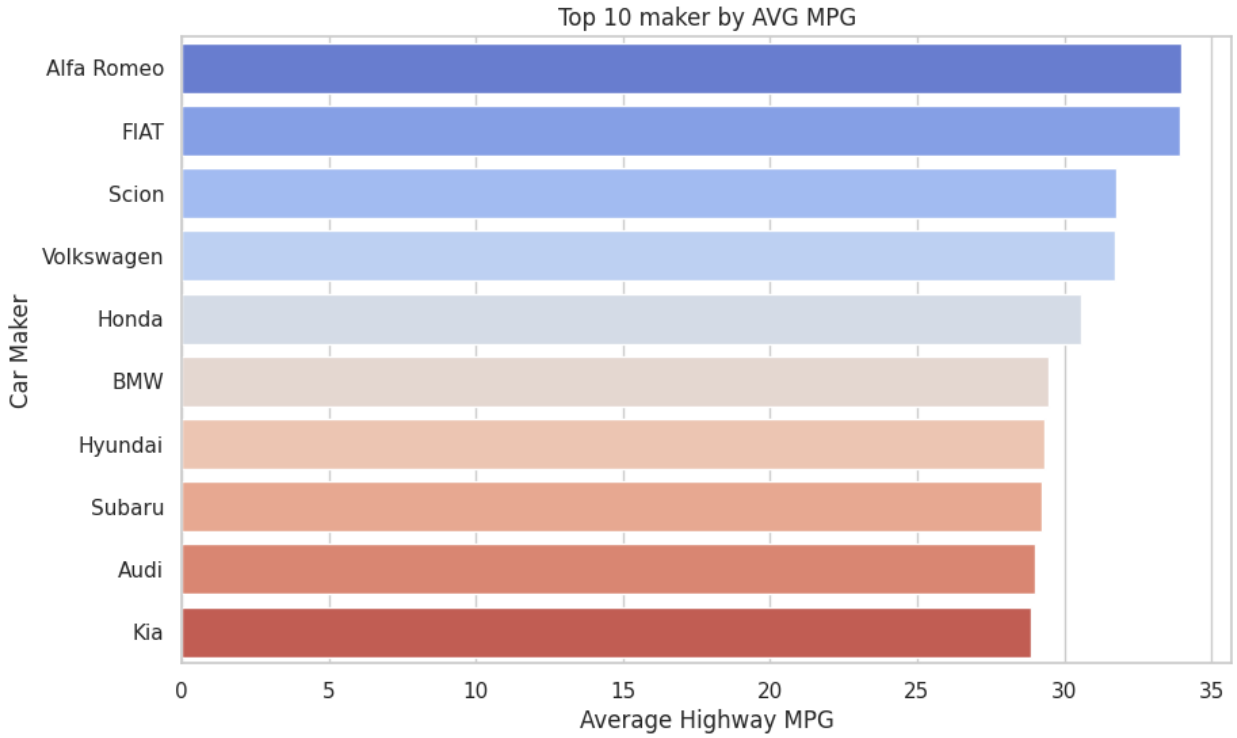

```
# Box plot of 'MSRP' by 'Vehicle Style'

plt.figure(figsize=(10,6))
mean_price = cars_data2.groupby(['Engine Cylinders'], as_index =
False)['MSRP'].mean().sort_values(by = 'MSRP', ascending = False)
sns.barplot(x = mean_price['Engine Cylinders'],y = mean_price['MSRP'],
palette='viridis')
plt.title('MSRP by Engine Cylinders')
plt.show()
```

```
<ipython-input-53-26902123e028>: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 = mean_price['Engine Cylinders'],y =
mean_price['MSRP'], palette='viridis')
```



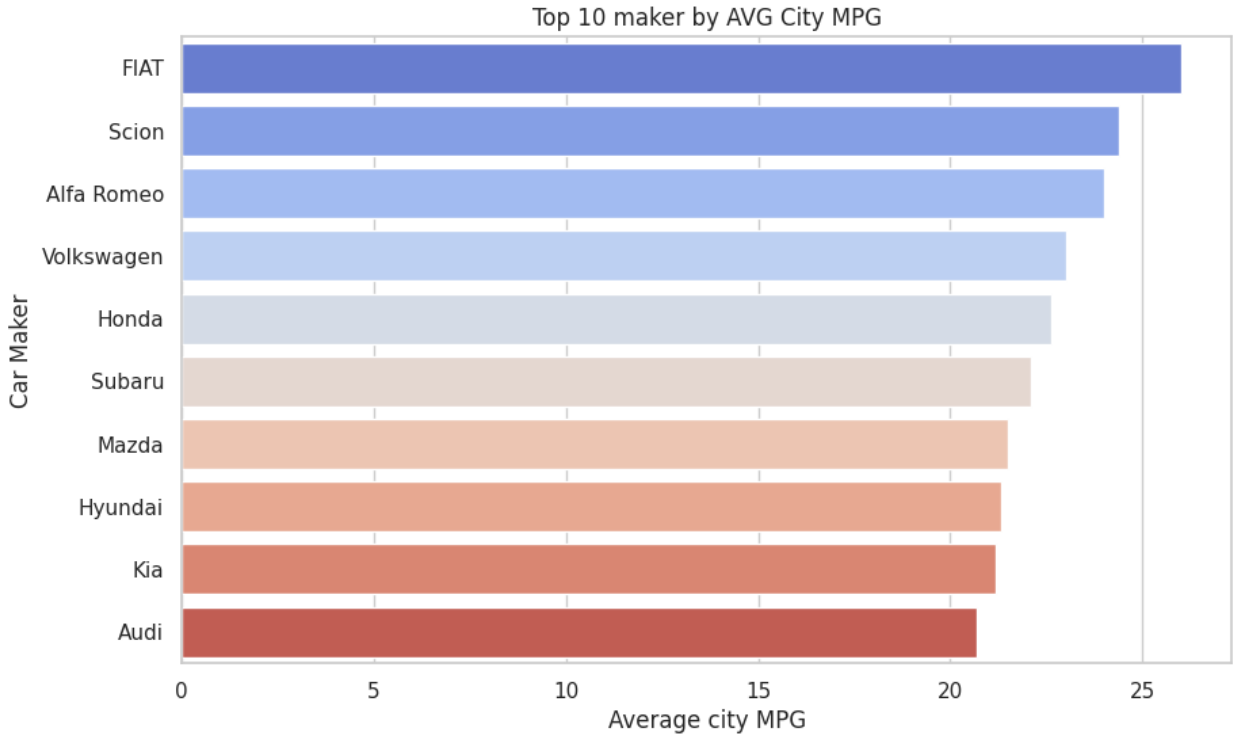

```
# Car maker by Average highway MPG

avg_mpg_by_make= cars_data2.groupby('Make')['highway
MPG'].mean().sort_values(ascending = False)
top_10_mpg_by_make = avg_mpg_by_make.head(10)
plt.figure(figsize = (10,6))
sns.barplot(x = top_10_mpg_by_make, y = top_10_mpg_by_make.index,
palette = 'coolwarm')
plt.title('Top 10 maker by AVG MPG')
plt.ylabel('Car Maker')
plt.xlabel('Average Highway MPG')
plt.show()
```

```
<ipython-input-37-f266ade09832>:6: FutureWarning:

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

  sns.barplot(x = top_10_mpg_by_make, y = top_10_mpg_by_make.index,
palette = 'coolwarm')
```

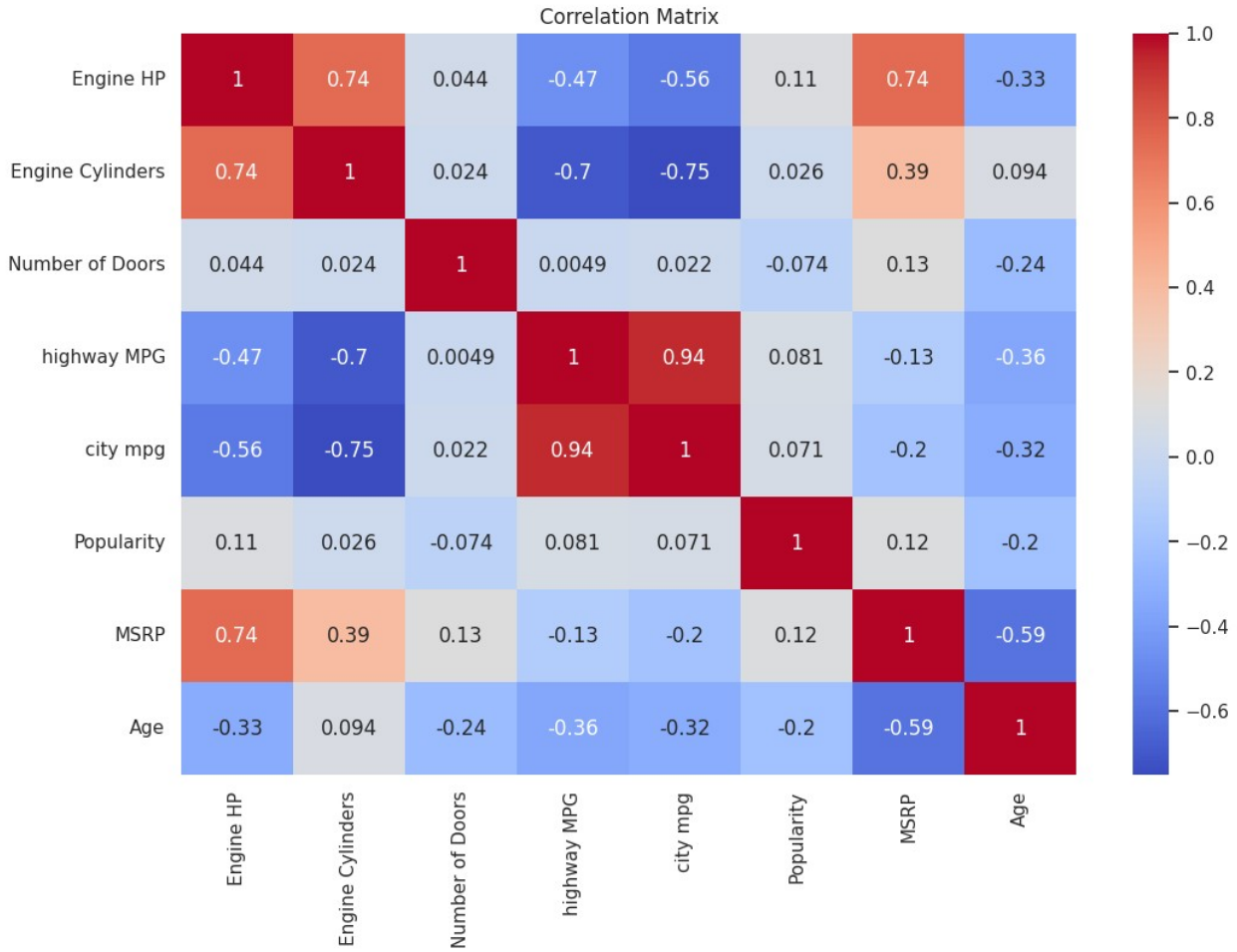



```
# Car Makers by average City MPG

avg_mpg_by_make= cars_data2.groupby('Make')['city
mpg'].mean().sort_values(ascending = False)
top_10_mpg_by_make = avg_mpg_by_make.head(10)
plt.figure(figsize = (10,6))
sns.barplot(x = top_10_mpg_by_make, y = top_10_mpg_by_make.index,
palette = 'coolwarm')
plt.title('Top 10 maker by AVG City MPG')
plt.ylabel('Car Maker')
plt.xlabel('Average city MPG')
plt.show()
```

```
<ipython-input-38-bf64df92101f>:6: FutureWarning:

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

  sns.barplot(x = top_10_mpg_by_make, y = top_10_mpg_by_make.index,
palette = 'coolwarm')
```




```
# Correlation Heatmap
plt.figure(figsize=(12, 8))
# Select only numeric features for correlation calculation
numeric_data = cars_data2.select_dtypes(include=['number'])
correlation_matrix = numeric_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
Engine HP
Engine Cylinders
Number of Doors
highway MPG
city mpg
Popularity
MSRP
Age
1 0.74 0.044 -0.47 -0.56 0.11 0.74 -0.33
0.74 1 0.024 -0.7 -0.75 0.026 0.39 0.094
0.044 0.024 1 0.0049 0.022 -0.074 0.13 -0.24
-0.47 -0.7 0.0049 1 0.94 0.081 -0.13 -0.36
-0.56 -0.75 0.022 0.94 1 0.071 -0.2 -0.32
0.11 0.026 -0.074 0.081 0.071 1 0.12 -0.2
0.74 0.39 0.13 -0.13 -0.2 0.12 1 -0.59
-0.33 0.094 -0.24 -0.36 -0.32 -0.2 -0.59 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6