Shaheed Zulfikar Ali Bhutto Institute of Science & Technology University

DEPARTMENT OF ROBOTICS & ARTIFICIAL INTELLIGENCE

Total Marks: 04

Obtained Marks: _________

Programming for Artificial Intelligence

Assignment # 04

Last date of Submission: 20 May 2024

Submitted To: Mr. Saad Irfan Khan

Student Name: Syed Muhammad Mustafa

Reg. Number: 22108130

DEPARTMENT OF ROBOTICS & ARTIFICIAL INTELLIGENCE

***Instructions**: Copied or shown assignments will be marked zero. Late submissions are not entertained in any case.*

CLO 4 – PLO B, D – C4

Scenario: Data Visualization with Matplotlib **(4 Marks)**

Apply Matplotlib to visualize a publicly available dataset related to AI or ML. You are required to do the following:

1. Choose a dataset of your choice.
2. Explore data, handle missing values, and preprocess.
3. Create a line plot, bar plot, scatter plot, histogram, and pie chart using Matplotlib.
4. Use advanced features like subplotting and customization.
5. Analyze and interpret each plot's insights and discuss Matplotlib's utility for AI projects.

Deliverables:

i. Jupyter/Colab Notebook with code and visualizations (softcopy on GCR).
ii. Dataset info, Code, annotated plots and a summary report (hardcopy submission) discussing insights and Matplotlib's effectiveness.

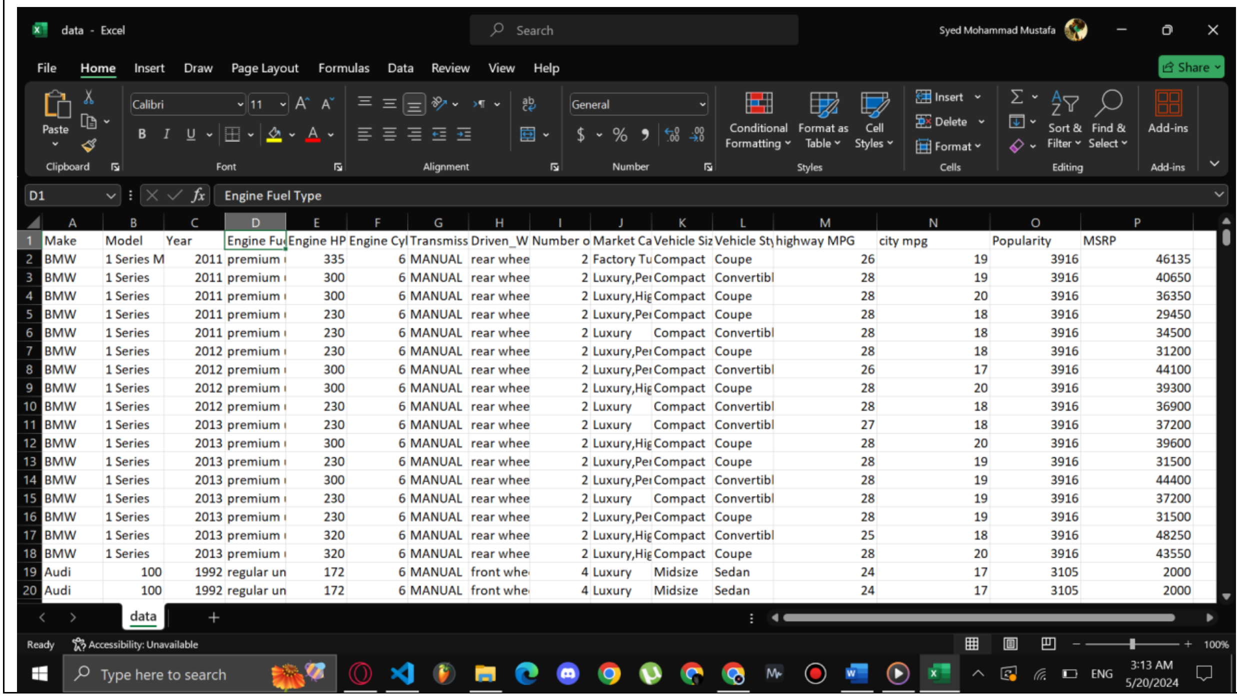
Dataset Information:

The dataset contains various features related to cars, such as Make, Model, Year, Engine Fuel Type, Engine HP, Engine Cylinders, Transmission Type, Driven Wheels, Number of Doors, Market Category, Vehicle Size, Vehicle Style, highway MPG, city mpg, Popularity, and MSRP. Here is a brief overview of the dataset:

Dataset:

Make: 48 unique values
Model: 723 unique values
Year: 28 unique values (int64)
Engine Fuel Type: 9 unique values
Engine HP: 330 unique values (float64)
Engine Cylinders: 10 unique values (float64)
Transmission Type: 5 unique values
Driven Wheels: 4 unique values
Number of Doors: 4 unique values (float64)
Market Category: 71 unique values
Vehicle Size: 3 unique values
Vehicle Style: 16 unique values
highway MPG: 58 unique values (int64)
city mpg: 68 unique values (int64)
Popularity: 48 unique values (int64)
MSRP: 4680 unique values (int64)

Dataset Screenshot:



Make	Model	Year	Engine Fu	Engine HP	Engine Cyl	Transmiss	Driven_W	Number o	Market Ca	Vehicle Siz	Vehicle Sty	highway MPG	city mpg	Popularity	MSRP
BMW	1 Series M	2011	premium	335	6	MANUAL	rear whee	2	Factory Tu	Compact	Coupe	26	19	3916	46135
BMW	1 Series	2011	premium	300	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Convertibl	28	19	3916	40650
BMW	1 Series	2011	premium	300	6	MANUAL	rear whee	2	Luxury,Hig	Compact	Coupe	28	20	3916	36350
BMW	1 Series	2011	premium	230	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Coupe	28	18	3916	29450
BMW	1 Series	2011	premium	230	6	MANUAL	rear whee	2	Luxury	Compact	Convertibl	28	18	3916	34500
BMW	1 Series	2012	premium	230	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Coupe	28	18	3916	31200
BMW	1 Series	2012	premium	300	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Convertibl	26	17	3916	44100
BMW	1 Series	2012	premium	300	6	MANUAL	rear whee	2	Luxury,Hig	Compact	Coupe	28	20	3916	39300
BMW	1 Series	2012	premium	230	6	MANUAL	rear whee	2	Luxury	Compact	Convertibl	28	18	3916	36900
BMW	1 Series	2013	premium	230	6	MANUAL	rear whee	2	Luxury	Compact	Convertibl	27	18	3916	37200
BMW	1 Series	2013	premium	300	6	MANUAL	rear whee	2	Luxury,Hig	Compact	Coupe	28	20	3916	39600
BMW	1 Series	2013	premium	230	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Coupe	28	19	3916	31500
BMW	1 Series	2013	premium	300	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Convertibl	28	19	3916	44400
BMW	1 Series	2013	premium	230	6	MANUAL	rear whee	2	Luxury	Compact	Convertibl	28	19	3916	37200
BMW	1 Series	2013	premium	230	6	MANUAL	rear whee	2	Luxury,Pe	Compact	Coupe	28	19	3916	31500
BMW	1 Series	2013	premium	320	6	MANUAL	rear whee	2	Luxury,Hig	Compact	Convertibl	25	18	3916	48250
BMW	1 Series	2013	premium	320	6	MANUAL	rear whee	2	Luxury,Hig	Compact	Coupe	28	20	3916	43550
Audi	100	1992	regular un	172	6	MANUAL	front whe	4	Luxury	Midsize	Sedan	24	17	3105	2000
Audi	100	1992	regular un	172	6	MANUAL	front whe	4	Luxury	Midsize	Sedan	24	17	3105	2000

Code:

```
import pandas as pd

df = pd.read_csv('Car Features and MSRP/data.csv')
df.head(5)
df.tail(5)
```

Output:

```
df.head(5)
[121] ✓ 0.0s  Open 'df' in Data Wrangler
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

```
df.tail(5)
[122] ✓ 0.0s  Open 'df' in Data Wrangler
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	46120
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	56670
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	50620
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	50920
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury	Midsize	Sedan	26	17	61	28995

```
df.shape
```

Code:

```
df.shape
df.columns
df.dtypes
```

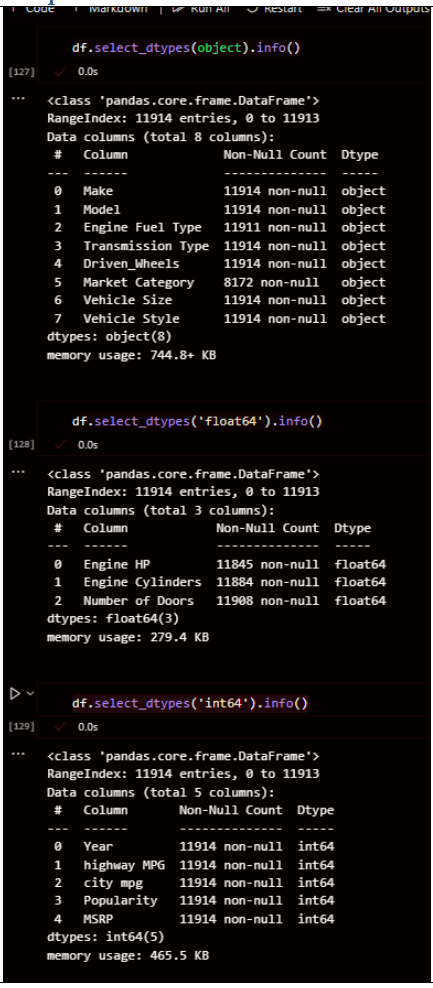
Output:

```
df.shape
[123] ✓ 0.0s
(11914, 16)

df.columns
[124] ✓ 0.0s
Index(['Make', 'Model', 'Year', 'Engine Fuel Type', 'Engine HP',
       'Engine Cylinders', 'Transmission Type', 'Driven_Wheels',
       'Number of Doors', 'Market Category', 'Vehicle Size', 'Vehicle Style',
       'highway MPG', 'city mpg', 'Popularity', 'MSRP'],
      dtype='object')

df.dtypes
[125] ✓ 0.0s
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
```

Code:

```
df.info()
df.select_dtypes(object).info()
df.select_dtypes('float64').info()
df.select_dtypes('int64').info()
```

Output:



```
df.select_dtypes(object).info()
[127]  0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Make               11914 non-null  object
 1   Model              11914 non-null  object
 2   Engine Fuel Type   11911 non-null  object
 3   Transmission Type  11914 non-null  object
 4   Driven_Wheels      11914 non-null  object
 5   Market Category    8172 non-null   object
 6   Vehicle Size       11914 non-null  object
 7   Vehicle Style      11914 non-null  object
dtypes: object(8)
memory usage: 744.8+ KB

df.select_dtypes('float64').info()
[128]  0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Engine HP         11845 non-null  float64
 1   Engine Cylinders  11884 non-null  float64
 2   Number of Doors   11908 non-null  float64
dtypes: float64(3)
memory usage: 279.4 KB

df.select_dtypes('int64').info()
[129]  0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Year         11914 non-null  int64
 1   highway MPG  11914 non-null  int64
 2   city mpg     11914 non-null  int64
 3   Popularity   11914 non-null  int64
 4   MSRP         11914 non-null  int64
dtypes: int64(5)
memory usage: 465.5 KB
```

```
0.0s
<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
```

Code:

```
_object_type_columns = df.select_dtypes(object).shape[1]
_float_type_columns = df.select_dtypes('float64').shape[1]
_int_type_columns = df.select_dtypes('int64').shape[1]

print(f'Columns with object data type  : {_object_type_columns}')
print(f'Columns with float64 data type : {_float_type_columns}')
print(f'Columns with int64 data type   : {_int_type_columns}')
print(f'Shape of the entire dataset    : {df.shape[1]} ')
df.describe()
```

Output:

```
_object_type_columns = df.select_dtypes(object).shape[1]
_float_type_columns = df.select_dtypes('float64').shape[1]
_int_type_columns = df.select_dtypes('int64').shape[1]

print(f'Columns with object data type  : {_object_type_columns}')
print(f'Columns with float64 data type : {_float_type_columns}')
print(f'Columns with int64 data type   : {_int_type_columns}')
print(f'Shape of the entire dataset    : {df.shape[1]} ')
[130] 0.0s

Columns with object data type  : 8
Columns with float64 data type : 3
Columns with int64 data type   : 5
Shape of the entire dataset    : 16

df.describe()
[131] 0.0s
```

	Year	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
count	11914.000000	11845.00000	11884.000000	11908.000000	11914.000000	11914.000000	11914.000000	1.191400e+04
mean	2010.384338	249.38607	5.628829	3.436093	26.637485	19.733255	1554.911197	4.059474e+04
std	7.579740	109.19187	1.780559	0.881315	8.863001	8.987798	1441.855347	6.010910e+04
min	1990.000000	55.00000	0.000000	2.000000	12.000000	7.000000	2.000000	2.000000e+03
25%	2007.000000	170.00000	4.000000	2.000000	22.000000	16.000000	549.000000	2.100000e+04
50%	2015.000000	227.00000	6.000000	4.000000	26.000000	18.000000	1385.000000	2.999500e+04
75%	2016.000000	300.00000	6.000000	4.000000	30.000000	22.000000	2009.000000	4.223125e+04
max	2017.000000	1001.00000	16.000000	4.000000	354.000000	137.000000	5657.000000	2.065902e+06

Code:

```
df.isnull().sum()
total_nullvalues = df.isnull().sum().sum()
print(f'Total Null Values in the dataset : {total_nullvalues}')
```

Output:

```
df.isnull().sum()
[134]  0.0s

...  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

total_nullvalues = df.isnull().sum().sum()
print(f'Total Null Values in the dataset : {total_nullvalues}')
[135]  0.0s

...  Total Null Values in the dataset : 3850
```

Code:

```
columns_with_nullvalues = df.columns[df.isnull().any()].tolist()
columns_with_nullvalues
print('Datatypes of columns with null values :')

for column in columns_with_nullvalues:
    print(f'{column}: {df[column].dtype}')
```

Output:

```
columns_with_nullvalues = df.columns[df.isnull().any()].tolist()
136]  0.0s

columns_with_nullvalues
137]  0.0s  Open 'columns_with_nullvalues' in Data Wrangler

...  ['Engine Fuel Type',
      'Engine HP',
      'Engine Cylinders',
      'Number of Doors',
      'Market Category']

print('Datatypes of columns with null values :')

for column in columns_with_nullvalues:
    print(f'{column}: {df[column].dtype}')
138]  0.0s

...  Datatypes of columns with null values :
     Engine Fuel Type: object
     Engine HP: float64
     Engine Cylinders: float64
     Number of Doors: float64
     Market Category: object
```

Code:

```python
print('Number of unique values in the features containing null values : ')
for column in columns_with_nullvalues:
    print(f'{column} : {df[column].nunique()}')
    print()
```

Output:

```
print('Number of unique values in the features containing null values : ')
for column in columns_with_nullvalues:
    print(f'{column} : {df[column].nunique()}')
    print()
0.0s

Number of unique values in the features containing null values :
Engine Fuel Type : 10

Engine HP : 356

Engine Cylinders : 9

Number of Doors : 3

Market Category : 71
```

Code:

```python
df_NullFreeMarketCategory = df.copy()
df_NullFreeMarketCategory = df_NullFreeMarketCategory.dropna(subset=['Market Category'])
```

```python
from sklearn.impute import SimpleImputer

impute_categorical = SimpleImputer(strategy='most_frequent')
df_NullFreeMarketCategory[['Engine Fuel Type']] = impute_categorical.fit_transform(df_NullFreeMarketCategory[['Engine Fuel Type']])

impute_numerical = SimpleImputer(strategy='mean')
df_NullFreeMarketCategory['Engine HP'] = impute_numerical.fit_transform(df_NullFreeMarketCategory[['Engine HP']])
df_NullFreeMarketCategory['Engine Cylinders'] = impute_numerical.fit_transform(df_NullFreeMarketCategory[['Engine Cylinders']])
df_NullFreeMarketCategory['Number of Doors'] = impute_numerical.fit_transform(df_NullFreeMarketCategory[['Number of Doors']])
```

```python
df.isnull().sum()
df_NullFreeMarketCategory.isnull().sum()
```

Output:

```
+ Code  + Markdown | ▷ Run All  ↺ Restart  ≡ Clear All Outputs |
df_NullFreeMarketCategory['Engine Cylinders'] = impute_
df_NullFreeMarketCategory['Number of Doors'] = impute_r

[141]  ✓ 0.0s

df.isnull().sum()
[142]  ✓ 0.0s

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

df_NullFreeMarketCategory.isnull().sum()
[143]  ✓ 0.0s

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
```

Code:

```
from tabulate import tabulate

table = []

for column  in df_NullFreeMarketCategory:
    table.append([column, df_NullFreeMarketCategory[column].nunique(), 
df_NullFreeMarketCategory[column].dtype])

def table_info():
    print(tabulate(table, headers = ['Feature Name', 'Unique Values', 'Data 
Type'], tablefmt = 'grid'))


table_info()
```

Output:

Feature Name	Unique Values	Data Type
Make	48	object
Model	723	object
Year	28	int64
Engine Fuel Type	9	object
Engine HP	330	float64
Engine Cylinders	10	float64
Transmission Type	5	object
Driven_Wheels	4	object
Number of Doors	4	float64
Market Category	71	object
Vehicle Size	3	object
Vehicle Style	16	object
highway MPG	58	int64
city mpg	68	int64

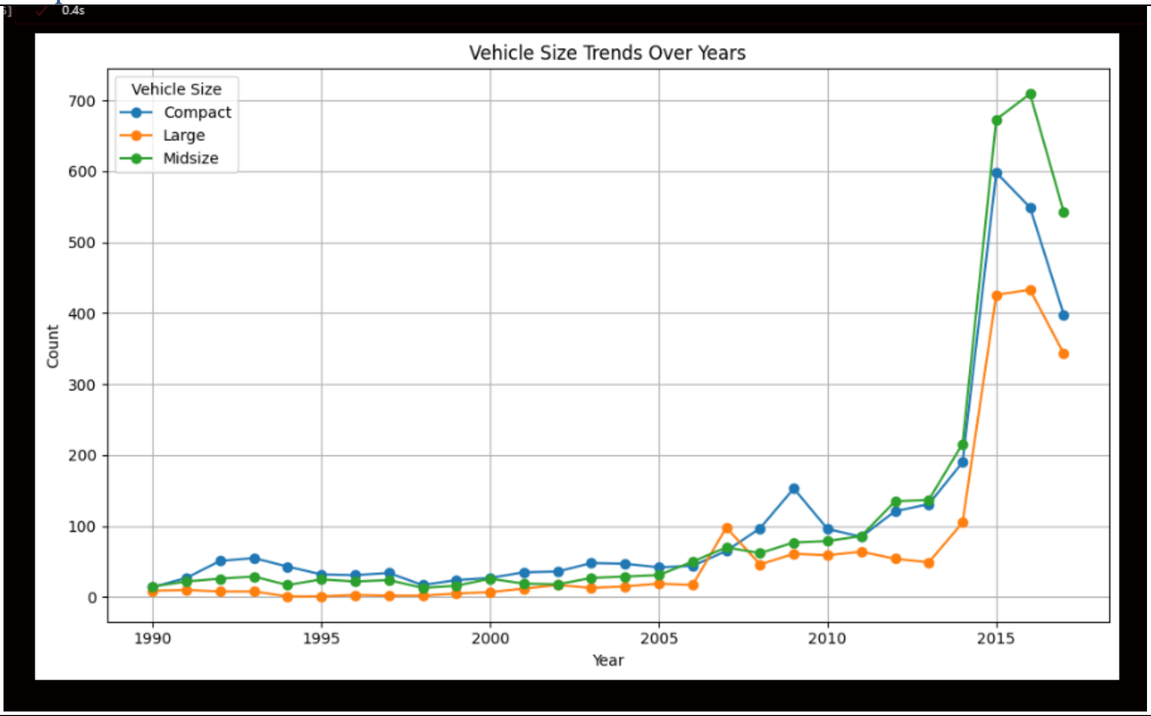
0.0s

Model	723	object
Year	28	int64
Engine Fuel Type	9	object
Engine HP	330	float64
Engine Cylinders	10	float64
Transmission Type	5	object
Driven_Wheels	4	object
Number of Doors	4	float64
Market Category	71	object
Vehicle Size	3	object
Vehicle Style	16	object
highway MPG	58	int64
city mpg	68	int64
Popularity	48	int64
MSRP	4680	int64

Code:

```
import matplotlib.pyplot as plt

data = df_NullFreeMarketCategory
counts = data.groupby(['Year', 'Vehicle Size']).size().unstack(fill_value=0)

counts.plot(kind='line', marker='o', figsize=(10, 6))

plt.title('Vehicle Size Trends Over Years')
plt.xlabel('Year')
plt.ylabel('Count')

plt.legend(title='Vehicle Size')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:



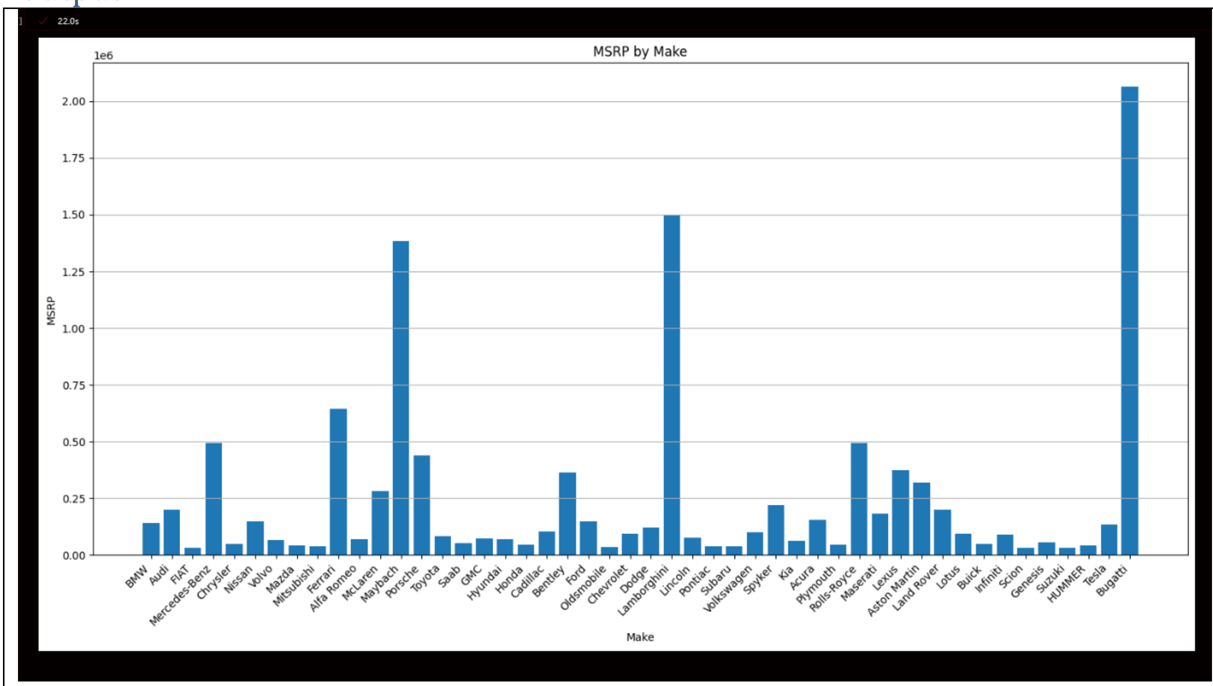
Insights:

1. We can see the highest vehicle size trends was mostly in the favour of 'Midsize'. The 'Large' did peak at mid 2005 - 2010, however it quickly nosedived
2. Dominance of Midsize Vehicles: As noted, the line plot indicates that the highest vehicle size trends were mostly in favor of 'Midsize' vehicles. This suggests that midsize vehicles have been consistently popular over the years, maintaining a significant portion of the market share.
3. Temporal Peaks and Declines: The observation regarding the 'Large' vehicle size peaking around mid-2005 to 2010, followed by a decline, indicates a temporal fluctuation in consumer preferences or market dynamics. This could be influenced by various factors such as economic conditions, fuel prices, consumer preferences, and automotive industry trends.

Code:

```
plt.figure(figsize=(15, 8))
plt.bar(data['Make'], data['MSRP'])
plt.title('MSRP by Make')
plt.xlabel('Make')
plt.ylabel('MSRP')
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.tight_layout()
plt.grid(axis='y')
plt.show()
```

Output:



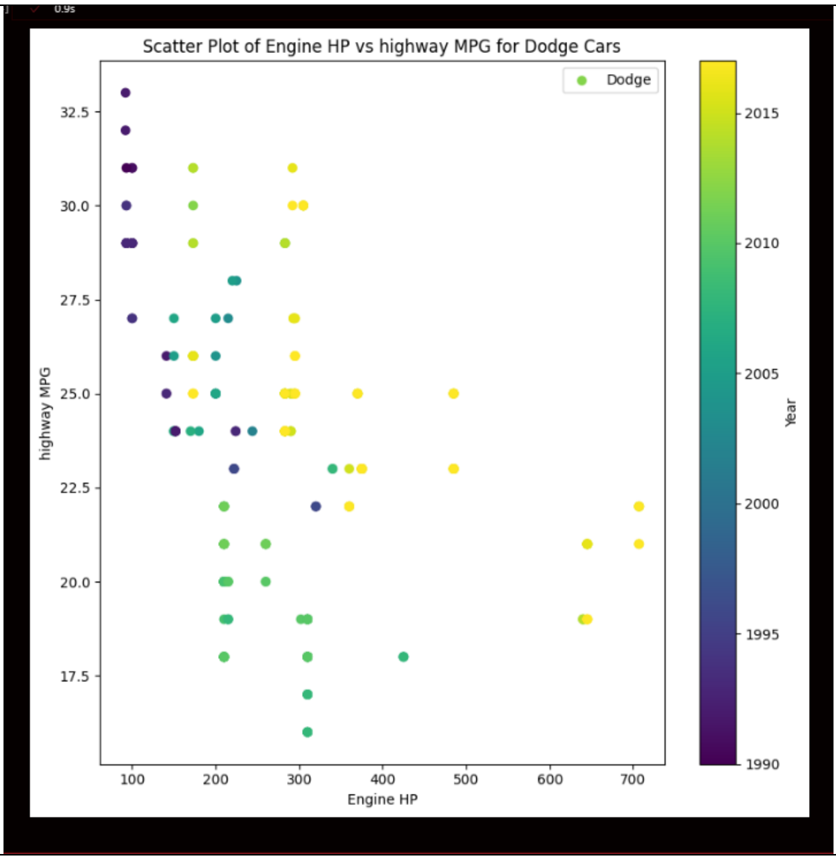
Insights:

```
1. We can cleary see the highest MSRP by make is of the 'Bugatti', folloewd by
'Lamborghini' and 'MayBach'
2. Top MSRP Manufacturers: As highlighted, the visualization clearly shows that
'Bugatti' has the highest average MSRP among all makes, followed by
'Lamborghini' and 'Maybach'. This suggests that vehicles from these
manufacturers are generally priced at a premium compared to others.
```

Code:

```
data_Dodge = data[data['Make']=='Dodge']

plt.figure(figsize=(8, 8))
plt.scatter(data_Dodge['Engine HP'], data_Dodge['highway MPG'],
c=data_Dodge['Year'], label="Dodge")
plt.xlabel("Engine HP")
plt.ylabel("highway MPG")
plt.legend()
plt.title("Scatter Plot of Engine HP vs highway MPG for Dodge Cars")
plt.colorbar(label='Year')


plt.tight_layout()
plt.show()
```

Output:



Insights:

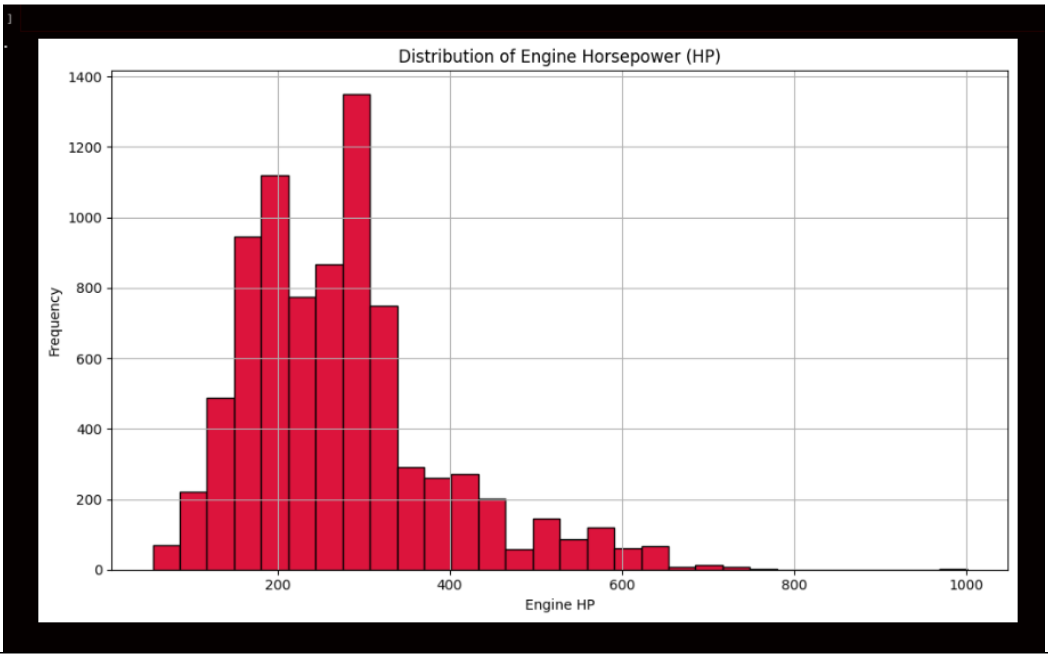
1. Trend Over Time: There seems to be a general trend of improvement in highway MPG over the years, particularly evident in the observations provided:
   - In 1990, Dodge cars typically achieved highway MPG between 22.5 and 32.5, with engine horsepower ranging from 100 to 200.
   - By 1995, there's a slight decrease in MPG, with a narrower range of around 22.5 to 25.0 MPG, but with a wider range of engine horsepower, typically between 200 and 400.
   - From 2000 to 2005, there's a wider spread of highway MPG, ranging from 17.5 to 27.5, and a wider range of engine horsepower from 150 to 400, suggesting some variability in Dodge's offerings during this period.
   - Between 2010 and 2015, there's a noticeable increase in both highway MPG and engine horsepower. MPG ranges from 20.0 to 30.0, with engine horsepower ranging from 180 to 700. This suggests a significant improvement in both fuel efficiency and engine performance over time.

2. Trade-off between HP and MPG: Generally, there seems to be a trade-off between engine horsepower and highway MPG, as evident from the scatter plot. Vehicles with higher horsepower tend to have lower MPG, and vice versa
3. Model Diversification: The spread of data points across different years suggests that Dodge has offered a diverse range of vehicle models with varying engine characteristics over time. This indicates that Dodge has adapted its product lineup to meet changing consumer preferences and regulatory requirements, resulting in a varied mix of vehicle options

Code:

```
data['Engine HP'].unique()

plt.figure(figsize=(10, 6))
plt.hist(data['Engine HP'], bins=30, color='crimson', edgecolor='black')
plt.title('Distribution of Engine Horsepower (HP)')
plt.xlabel('Engine HP')
plt.ylabel('Frequency')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:



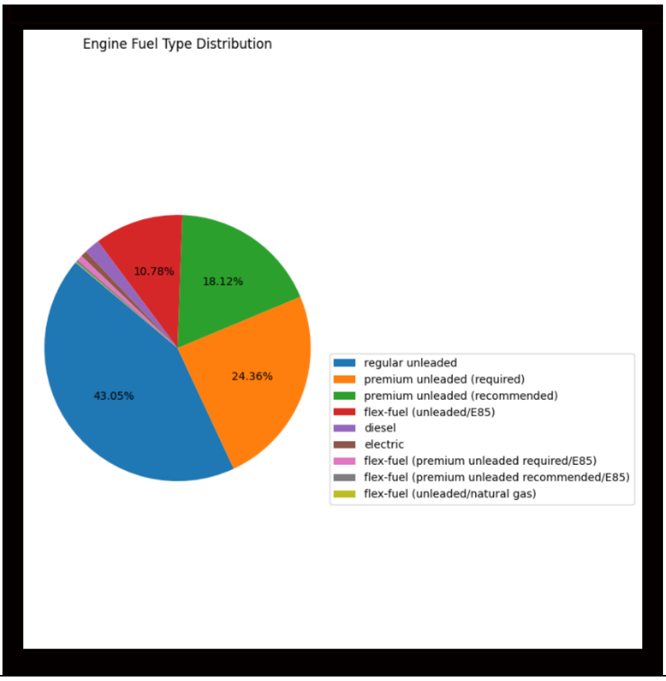
Insights:

```
1. Peak Frequency Range: The histogram peaks between 200 and 400 horsepower,
indicating that a significant number of vehicles in the dataset have engine
horsepower within this range. This suggests that engines with moderate to high
horsepower are prevalent among the vehicles in the dataset.
2. Distribution Shape: The distribution appears to be right-skewed, with a
gradual decrease in frequency as engine horsepower increases beyond the peak
range. This suggests that while there are many vehicles with moderate
horsepower, fewer vehicles have extremely high horsepower engines
```

Code:

```
df.columns
data['Engine Fuel Type'].unique()
plt.figure(figsize=(8, 8))
sizes = data['Engine Fuel Type'].value_counts().values
labels = data['Engine Fuel Type'].value_counts().index

total_count = sum(sizes)

threshold = 5

def autopct_fn(pct):
    return f'{pct:.2f}%' if pct > threshold else ''

plt.pie(sizes, labels=None, autopct=autopct_fn, startangle=140)
plt.title('Engine Fuel Type Distribution')
plt.axis('equal')

plt.legend(labels, loc="best", fontsize=10, bbox_to_anchor=(1, 0.5), labels=None)
plt.tight_layout()
plt.show()
```

Output:



Insights:

```
1.  Vehicles powered by regular unleaded fuel are widely manufactured and used
(Prevalence of Traditional Fuel Types)
2.  Premium unleaded fuels, both recommended and required, collectively account
for 42.48% of the distribution. This indicates a considerable market share for
vehicles that require or recommend higher-octane gasoline.
3.  Flex-fuel vehicles epresent 10.78% of the distribution.This suggests a
growing interest in vehicles that offer flexibility in fuel choice.
4.  Diesel, electric, and other flex-fuel variants represent relatively small
percentages of the distribution, each below 5%. This indicates that while there
is some adoption of alternative fuel technologies, they have not yet reached
widespread usage compared to traditional gasoline-powered vehicles
```

Summary Report:

Vehicle Size Trends: The dominance of 'Midsize' vehicles over the years indicates consumer preference for midsize cars, likely due to a balance of size, fuel efficiency, and cost. The peak and subsequent decline of 'Large' vehicles around 2005-2010 suggest changing market conditions or consumer preferences.

MSRP by Make: The high MSRP of brands like Bugatti, Lamborghini, and Maybach highlights their position as luxury and performance car manufacturers. This contrasts with more affordable brands, emphasizing market segmentation.

Market Categories: The concentration in top market categories suggests the presence of popular segments that car manufacturers target, reflecting trends and consumer demand in the automotive market.

Dodge Cars: The scatter plot of Dodge cars reveals the relationship between engine horsepower and fuel efficiency over time, showing technological advancements and shifts in consumer preferences.

Engine Horsepower Distribution: The histogram shows that most cars have engine horsepower between 100 and 400 HP, indicating a focus on moderate performance vehicles.

Fuel Type Distribution: The pie chart indicates a preference for regular unleaded fuel, followed by premium options. The distribution reflects consumer choices and market availability of different fuel types

Effectiveness of Matplotlib:

Visualization: Matplotlib excels at displaying a wide range of data types and relationships, simplifying the identification of trends, category comparisons, and understanding of distributions.

Customization: The library's extensive customization options for plots (such as colors, labels, and grid lines) improve both clarity and visual appeal.

Interpretation: Annotated plots offer a clear, visual summary of complex datasets, making it easier to interpret and communicate insights effectively.