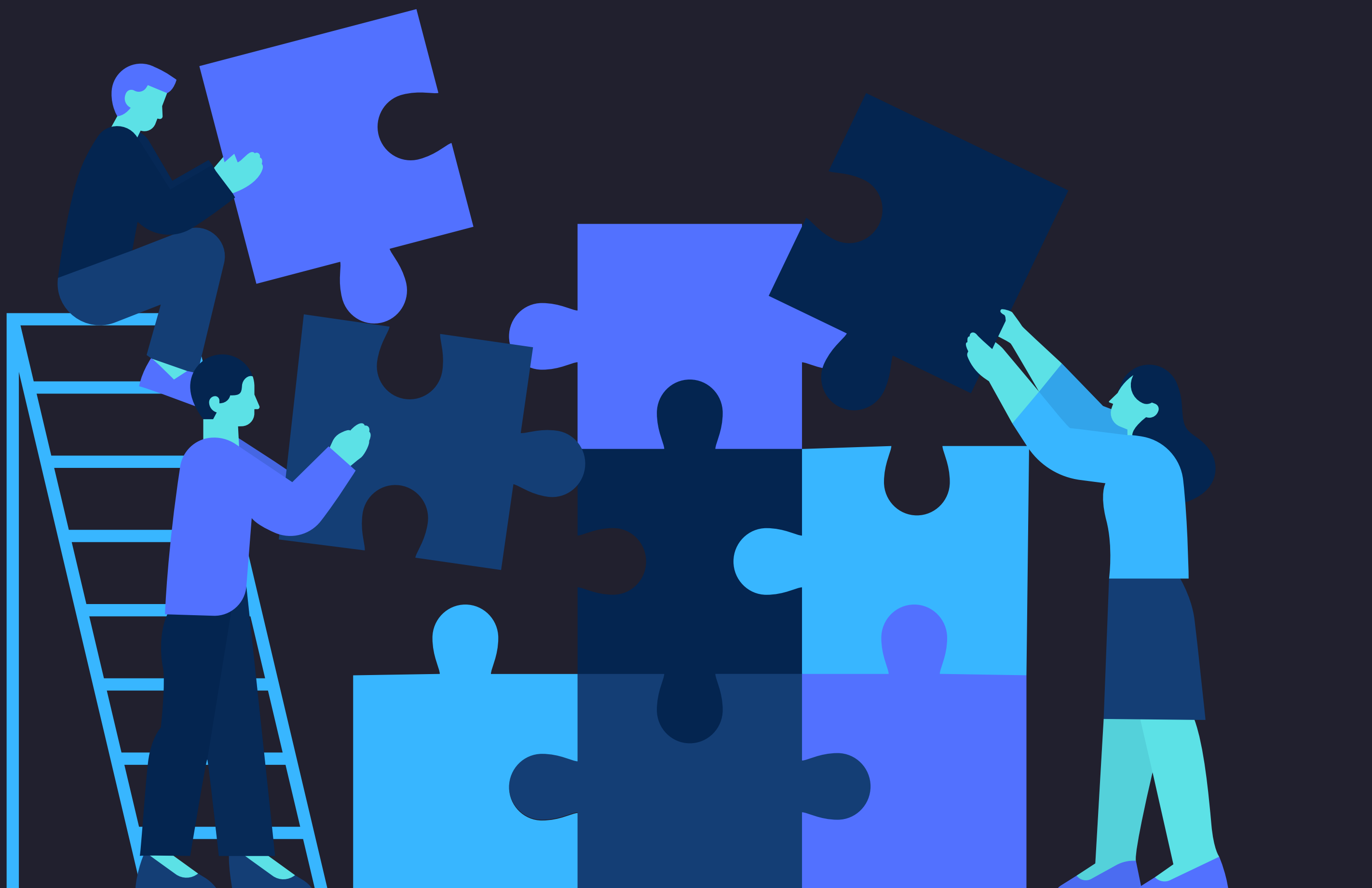
PYTHON-POWERED PROJECT FOR IN-DEPTH DATA ANALYSIS

EXPLORING INSIGHTS:

Data Cleaning
Data Analysis
Data Visualization

CARS DATASET



PYTHON-POWERED PROJECT FOR IN-DEPTH DATA ANALYSIS

- **Objective:** To analyze and visualize trends, patterns, and insights from a comprehensive cars dataset using Python and Pandas.
- **Data Handling:** Efficient preprocessing, cleaning, and manipulation of data for meaningful analysis.
- **Visualization:** Employing Matplotlib and Seaborn to create impactful graphs and charts that depict the data insights visually.
- **Key Insights:** Identifying relationships, trends, and actionable insights within the dataset (e.g., price distributions, engine performance).
- **Skills Demonstrated:** Showcasing proficiency in Python programming, data analysis techniques, and storytelling through data visualization.

Data Analysis & Visualization: Harnessing the Power of Python Libraries 📊

Leveraging robust Python libraries like Pandas for seamless data manipulation 🧮, Matplotlib for visually compelling graphs and plots 📉, and Seaborn for insightful statistical analysis 🔍, empowering precise and impactful data-driven decision-making.

Importing Libraries and Loading the Dataset 📂 📊

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('data.csv')
dfreal=pd.read_csv('data.csv')
```

[4] ✓ 0.1s Python

```
df.head()
```

[5] ✓ 0.0s Python

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0

Exploratory Data Analysis 📊

```python
df.describe()
```

[6] ✓ 0.0s Python

	Year	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	
count	11914.000000	11845.00000	11884.000000	11908.000000	11914.000000	11914.000000	119
mean	2010.384338	249.38607	5.628829	3.436093	26.637485	19.733255	15
std	7.579740	109.19187	1.780559	0.881315	8.863001	8.987798	14
min	1990.000000	55.00000	0.000000	2.000000	12.000000	7.000000	
25%	2007.000000	170.00000	4.000000	2.000000	22.000000	16.000000	5
50%	2015.000000	227.00000	6.000000	4.000000	26.000000	18.000000	13
75%	2016.000000	300.00000	6.000000	4.000000	30.000000	22.000000	20
max	2017.000000	1001.00000	16.000000	4.0000[illegible]	[illegible]54.000000	137.000000	56

Minimize

```python
df.info()
```

[7] ✓ 0.0s Python

```
<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
```

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

[8] ✓ 0.0s Python

```
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
```

Minimize

Data Cleaning

Handling Missing Data with Median Imputation 🛠 📊

```python
eng_cyl_median=df['Engine Cylinders'].median()
df['Engine Cylinders'].fillna(eng_cyl_median,inplace=True)
df['Engine HP'].fillna(df['Engine HP'].median(),inplace =True)
df['Number of Doors'].fillna(df['Number of Doors'].median(),inplace=True)
print(df.isnull().sum())
```

[9] ✓ 0.0s Python

```
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
```

```python
df[df["Model"]=="Verona"]
```

[10] ✓ 0.0s Python

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Numl Do
11321	Suzuki	Verona	2004	NaN	155.0	6.0	AUTOMATIC	front wheel drive	
11322	Suzuki	Verona	2004	NaN	155.0	6.0	AUTOMATIC	front wheel drive	
11323	Suzuki	Verona	2004	NaN	155.0	6.0	AUTOMATIC	front wheel drive	
11324	Suzuki	Verona	2005	regular unleaded	155.0	6.0	AUTOMATIC	front wheel drive	
11325	Suzuki	Verona	2005	regular unleaded	155.0	6.0	AUTOMATIC	front wheel drive	
11326	Suzuki	Verona	2005	regular unleaded	155.0	6.0	AUTOMATIC	front wheel drive	
11327	Suzuki	Verona	2005	regular unleaded	155.0	6.0	AUTOMATIC	front wheel drive	
11328	Suzuki	Verona	2006	regular unleaded	155.0	6.0	AUTOMATIC	front wheel drive	
11329	Suzuki	Verona	2006	regular unleaded	155.0	6.0	AUTOMATIC	front wheel drive	

Filtering Data and Handling Missing Values 🚗 🔧

```python
fuel_na=df[df["Engine Fuel Type"].isnull()]
df["Engine Fuel Type"].fillna("regular unleaded",inplace=True)
df.isnull().sum()
```

[14] ✓ 0.0s Python

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

Dropping Columns and Checking for Missing Values ✂️ 🔍

```python
df.drop(['Market Category'], axis=1, inplace=True)
df.isnull().sum()
```

[12] ✓ 0.0s Python

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

Minimize

```python
print("Rows before removing duplicates: ",dfreal.shape)
df[df.duplicated()]
df.drop_duplicates(inplace=True)
print("Rows after removing duplicates: ",df.shape)
```

[16] ✓ 0.0s Python

```
Rows before removing duplicates:  (11914, 16)
Rows after removing duplicates:  (11194, 15)
```

Top 10 Years by MSRP: Insights into Vehicle Pricing Trends 📅 💰

A comprehensive analysis of the years with the highest average Manufacturer's Suggested Retail Price (MSRP), showcasing key trends in pricing and significant market dynamics within the automotive industry.

```python
yearon_msrp=df.groupby('Year')['MSRP'].sum().sort_values(ascending=False).head(10)
print(yearon_msrp)
```

[17] ✓ 0.0s Python

```
Year
2016    99190613
2015    98069201
2017    68519509
2014    36299883
2012    21747656
2009    18316074
```

Top 10 Years by MSRP: Insights into Vehicle Pricing Trends 📅 💰

A comprehensive analysis of the years with the highest average Manufacturer's Suggested Retail Price (MSRP), showcasing key trends in pricing and significant market dynamics within the automotive industry.

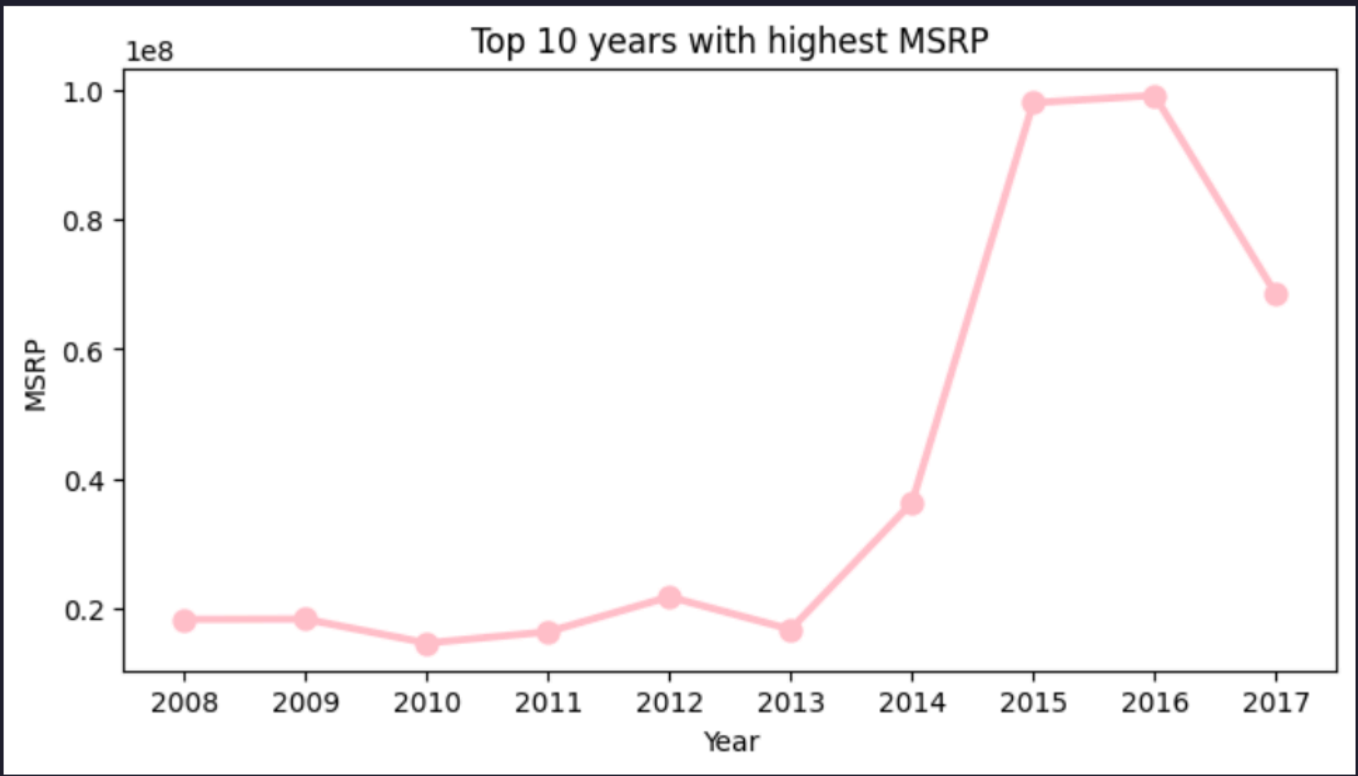
```python
yearon_msrp=df.groupby('Year')['MSRP'].sum().sort_values(ascending=False).head(10)
print(yearon_msrp)
```

[17] ✓ 0.0s Python

```
Year
2016    99190613
2015    98069201
2017    68519509
2014    36299883
2012    21747656
2009    18316074
2008    18274736
2013    16729622
2011    16349881
2010    14617934
Name: MSRP, dtype: int64
```

```python
yearon_msrp_df = yearon_msrp.reset_index()
plt.figure(figsize=(8, 4))
sns.pointplot(x='Year', y='MSRP', data=yearon_msrp_df, color='pink')
plt.title('Top 10 years with highest MSRP')
plt.show()
```

[18] ✓ 0.2s Python



Top 10 Most Expensive Cars by MSRP 🏎️ 💵

Unveiling the most luxurious and high-priced automobiles, ranked by their Manufacturer's Suggested Retail Price (MSRP).

```
Makeon_msrp=df.groupby('Make')['MSRP'].sum().sort_values(ascending=False).head(10)
print(Makeon_msrp)
```

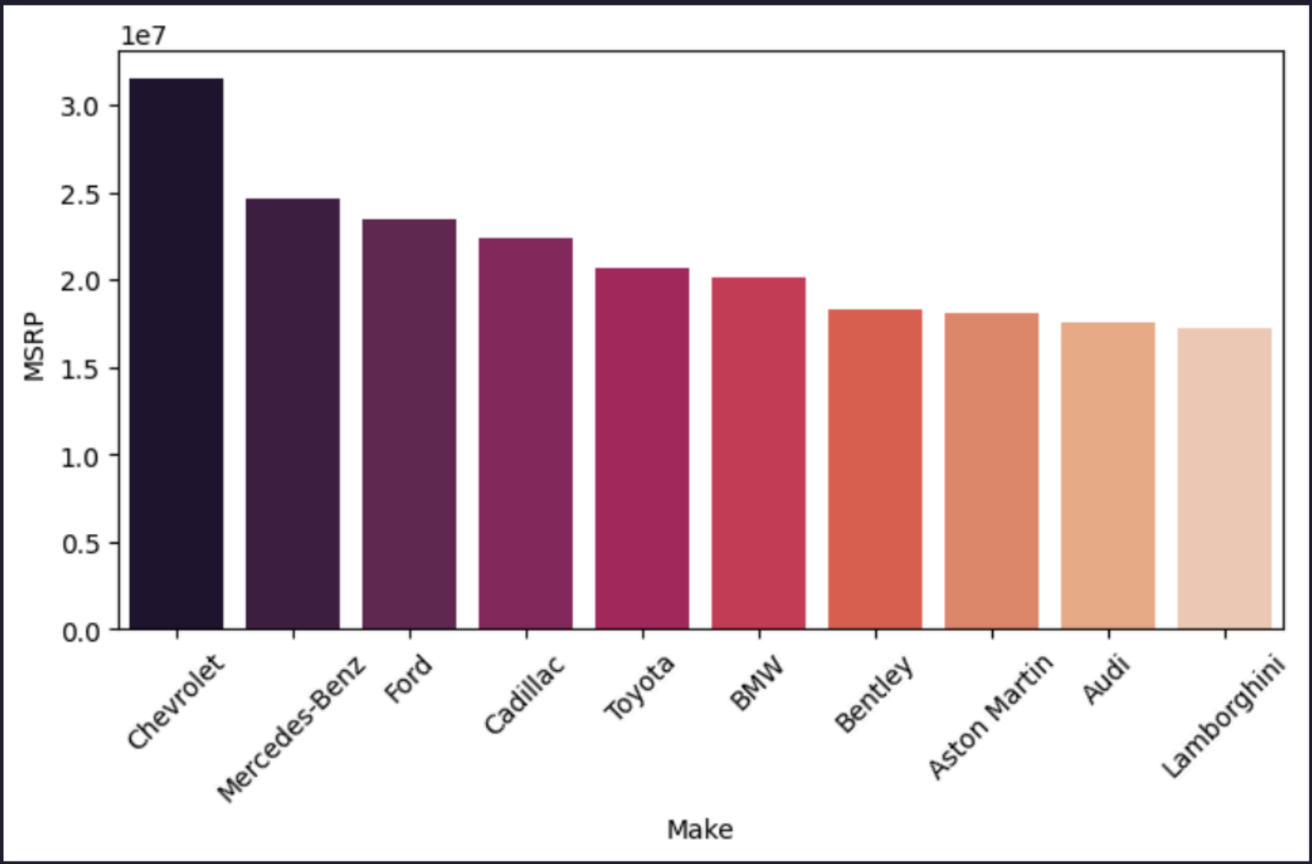
[19] ✓ 0.0s Python

Minimize

```
Make
Chevrolet        31487928
Mercedes-Benz    24575709
Ford             23490684
Cadillac         22321833
Toyota           20646567
BMW              20140669
Bentley          18290530
Aston Martin     18029235
Audi             17518293
Lamborghini      17241500
Name: MSRP, dtype: int64
```

```
Makeon_msrpdf=Makeon_msrp.reset_index()
plt.figure(figsize=(8, 4))
sns.barplot(x='Make', y='MSRP', data=Makeon_msrpdf,hue='Make', palette="rocket")
plt.xticks(rotation=45)
plt.show()
```

[28] ✓ 0.2s Python



Evolution: Most Popular Car Makers of the 20th & 21st Century 🚗

Exploring historical automotive trends to identify the most dominant and influential car manufacturers across two centuries.

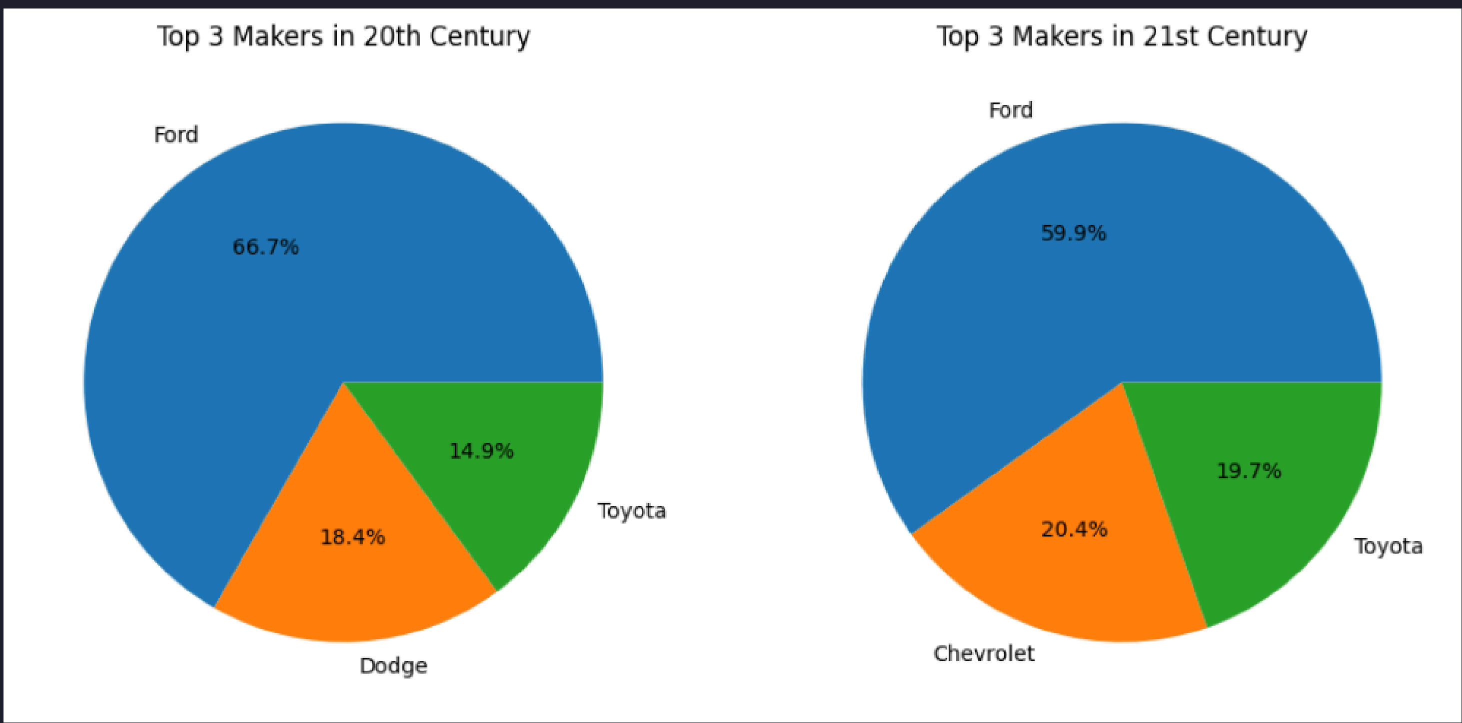
```
cen21df=df[df['Year']>=2000]
cen20df=df[df['Year']<2000]
popular_car_20=cen20df.groupby('Make')['Popularity'].sum().sort_values(ascending=False).head(3)
print("Top 3 Maker in 20th Century",popular_car_20)
popular_car_21=cen21df.groupby('Make')['Popularity'].sum().sort_values(ascending=False).head(3)
print("Top 3 Maker in 21th Century",popular_car_21)
```

[29] ✓ 0.0s Pyth

```
Top 3 Maker in 20th Century Make
Ford       718439
Dodge      198057
Toyota     160449
Name: Popularity, dtype: int64
Top 3 Maker in 21th Century Make
Ford          3942929
Chevrolet     1342065
Toyota        1293747
Name: Popularity, dtype: int64
```

```
popular_car_20df = popular_car_20.reset_index()
popular_car_21df = popular_car_21.reset_index()
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

axes[0].pie(popular_car_20df['Popularity'], labels=popular_car_20df['Make'], autopct='%1.1f%%')
axes[0].set_title('Top 3 Makers in 20th Century')

axes[1].pie(popular_car_21df['Popularity'], labels=popular_car_21df['Make'], autopct='%1.1f%%')
axes[1].set_title('Top 3 Makers in 21st Century')

plt.show()
```

[71] Pyth



The Power Behind the Wheels: Most Widely Used Engine Types ⚙️ 🔥

A deep dive into the most common and influential engine types that have powered vehicles across generations.

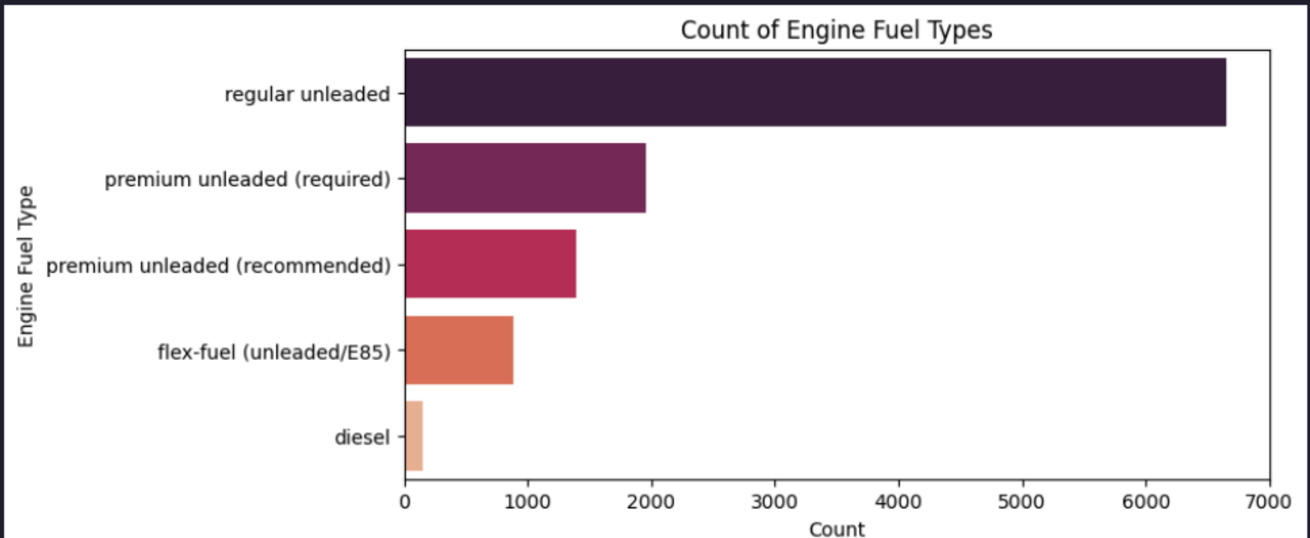
```
engine_type=df.groupby('Engine Fuel Type')['Engine Fuel Type'].count().sort_values(as
print(engine_type)
```

[30] ✓ 0.0s Python

```
Engine Fuel Type
regular unleaded                                  6656
premium unleaded (required)                       1956
premium unleaded (recommended)                    1392
flex-fuel (unleaded/E85)                           887
diesel                                             150
electric                                            66
flex-fuel (premium unleaded required/E85)           53
flex-fuel (premium unleaded recommended/E85)        26
flex-fuel (unleaded/natural gas)                     6
natural gas                                          2
Name: Engine Fuel Type, dtype: int64
```

```
t.figure(figsize=(8, 4))
gine_type5=engine_type.head()
s.barplot(y=engine_type5.index, x=engine_type5.values,hue=engine_type5.index, palette

t.xlabel("Count")
t.ylabel("Engine Fuel Type")
t.title("Count of Engine Fuel Types")
t.xticks(range(0, 8000, 1000))

Show the plot
t.show()
```

[35] ✓ 0.1s Python



Transmission Trends: Most Widely Used Automatic & Manual Systems 🔄 🚗

A comprehensive analysis of the most prevalent transmission systems, including automatic, manual, CVT, and dual-clutch, highlighting their impact on vehicle performance and driving dynamics.

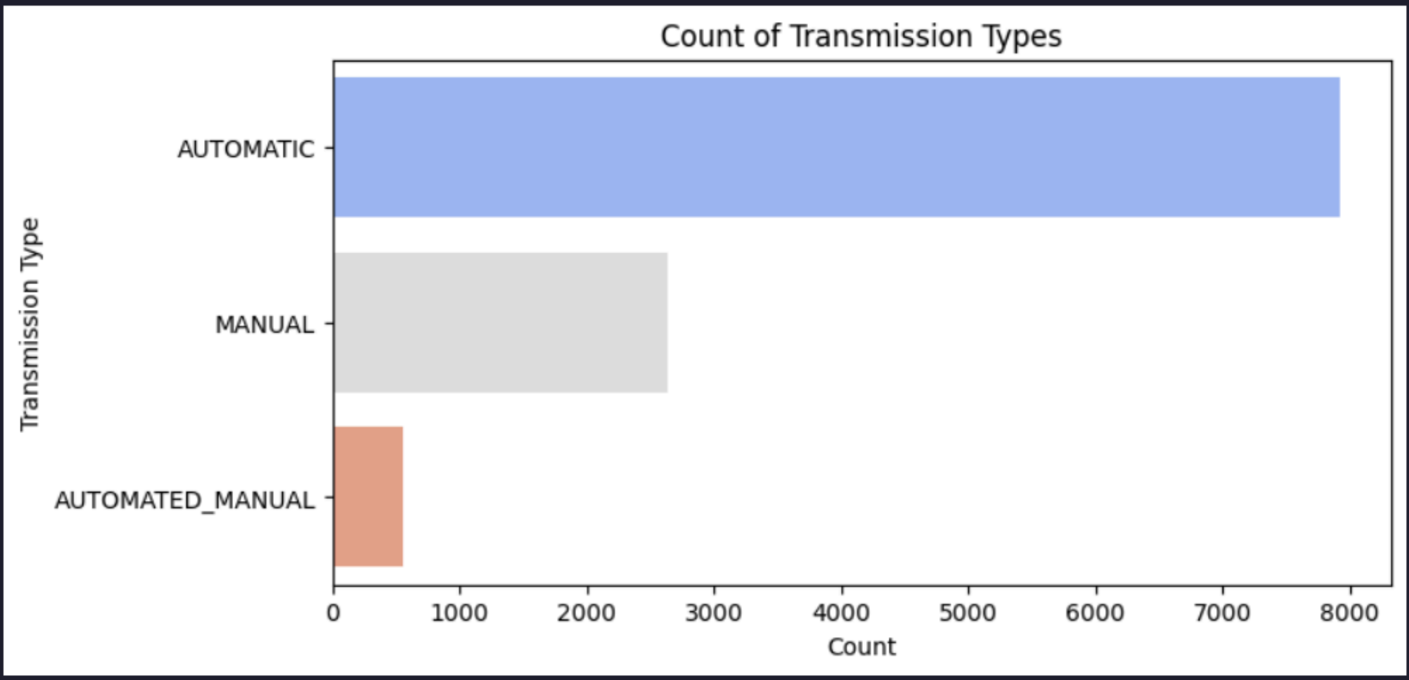
```python
transmission_type=df.groupby('Transmission Type')['Transmission Type'].count().sort_values(ascending=False
print(transmission_type)
```

[36] ✓ 0.0s Python

```
Transmission Type
AUTOMATIC           7928
MANUAL              2633
AUTOMATED_MANUAL     553
DIRECT_DRIVE          68
UNKNOWN               12
Name: Transmission Type, dtype: int64
```

```python
transmission_type3=transmission_type.head(3)
plt.figure(figsize=(8, 4))
sns.barplot(y=transmission_type3.index,x=transmission_type3.values,hue=transmission_type3.index, palette="
plt.xlabel("Count")
plt.ylabel("Transmission Type")
plt.title("Count of Transmission Types")
plt.show()
```

[40] ✓ 0.1s Python



Most Widely Used Drive Systems 🏁 🔧

An insight into the most popular drivetrain configurations that have shaped vehicle performance and handling over the years.

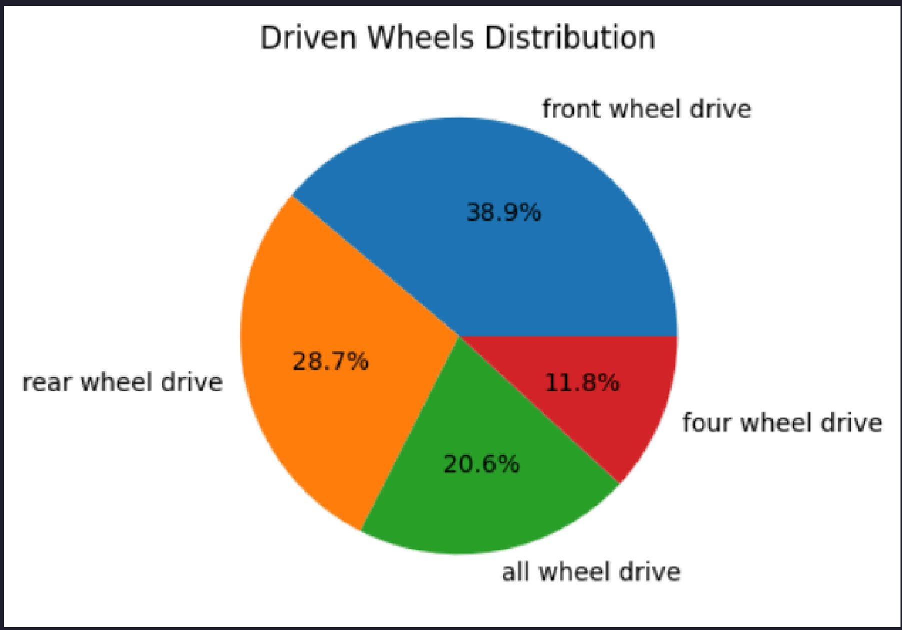
```python
driven_wheels=df.groupby('Driven_Wheels')['Driven_Wheels'].count().sort_values(ascend:
print(driven_wheels)
```

[41] ✓ 0.0s Python

```
Driven_Wheels
front wheel drive    4350
rear wheel drive     3211
all wheel drive      2308
four wheel drive     1325
Name: Driven_Wheels, dtype: int64
```

```python
plt.figure(figsize=(8, 4))
plt.pie(driven_wheels, labels=driven_wheels.index, autopct='%1.1f%%')
plt.title('Driven Wheels Distribution')
plt.show()
```

[42] ✓ 0.0s Python



Horsepower Spectrum: Top 5 Most Powerful & Least Powerful Car Models ⚙️ 🚀

An analytical comparison of the highest and lowest horsepower car models, highlighting the extremes of automotive performance and engineering.

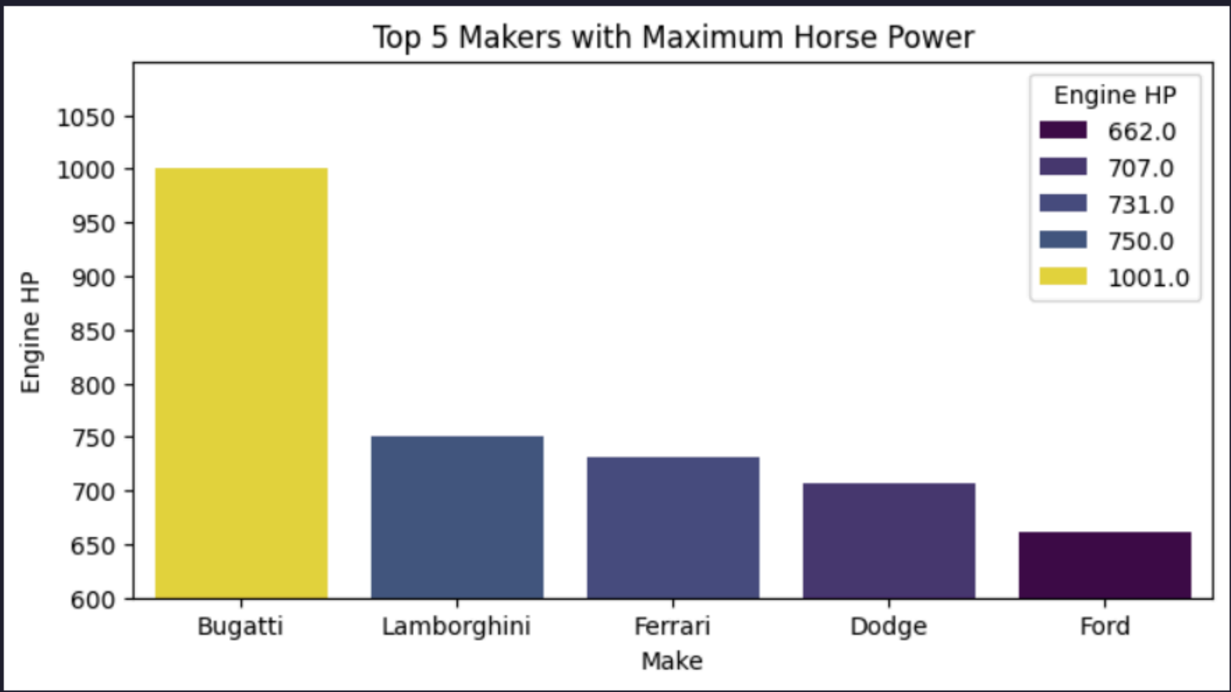
```python
maxhorse_power=df.groupby(['Make'])['Engine HP'].max().sort_values(ascending=False).head(5)
print(maxhorse_power)
minhorse_power=df.groupby('Make')['Engine HP'].min().sort_values(ascending=True).head(1)
print(minhorse_power)
```

[43] ✓ 0.0s Python

```
Make
Bugatti        1001.0
Lamborghini     750.0
Ferrari         731.0
Dodge           707.0
Ford            662.0
Name: Engine HP, dtype: float64
Make
Chevrolet    55.0
Name: Engine HP, dtype: float64
```

```python
maxhorse_powerdf = maxhorse_power.reset_index()
plt.figure(figsize=(8, 4))
sns.barplot(x='Make',y='Engine HP',data=maxhorse_powerdf,hue='Engine HP',palette="viridis")
plt.title('Top 5 Makers with Maximum Horse Power')
plt.yticks(range(600, 1100, 50))
plt.ylim(600, 1100)
plt.show()
```

[152] Python



Engine Efficiency: Average Cylinders by Vehicle Size 🔍 🔩

Examining the correlation between vehicle size and the average number of cylinders, providing insights into engine design and performance trends.

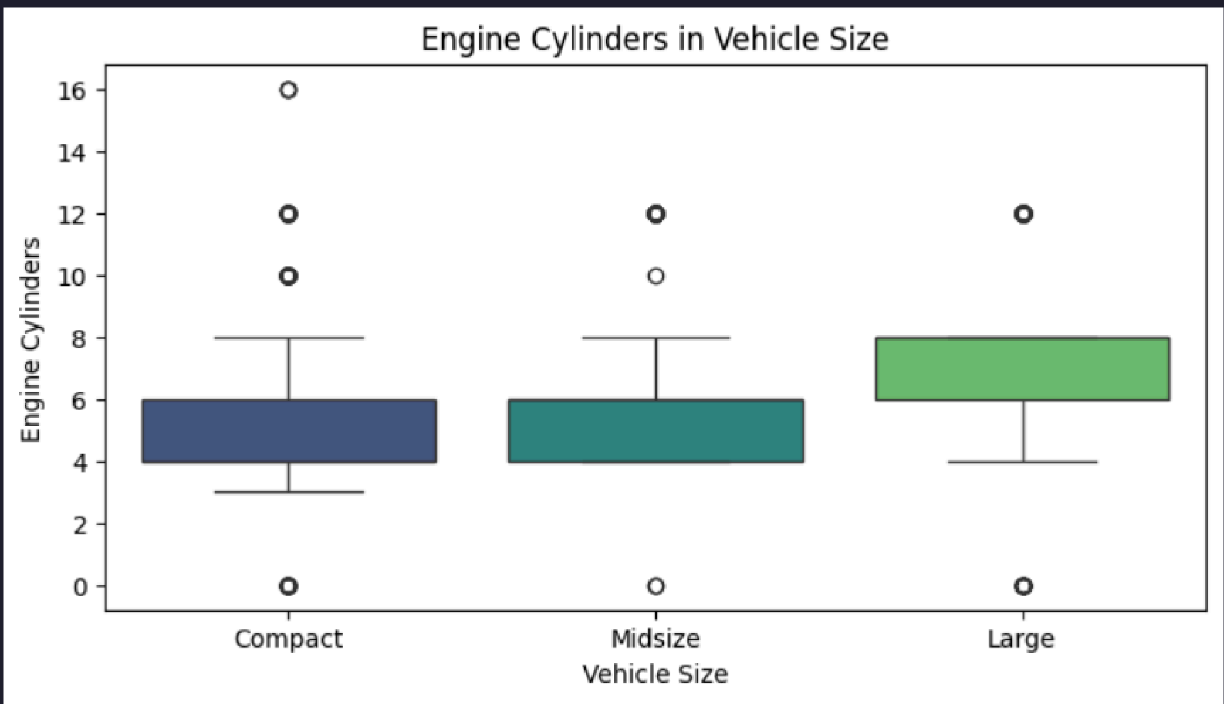
```python
avg_cylinders=df.groupby('Vehicle Size')['Engine Cylinders'].mean().sort_values(ascending=False)
print(avg_cylinders)
```

[44] ✓ 0.0s Python

```
Vehicle Size
Large      7.077152
Midsize    5.622541
Compact    4.838909
Name: Engine Cylinders, dtype: float64
```

```python
avg_cylindersdf = avg_cylinders.reset_index()
plt.figure(figsize=(8, 4))
sns.boxplot(x='Vehicle Size',y='Engine Cylinders',data=df,hue='Vehicle Size',palette="viridis")
plt.title('Engine Cylinders in Vehicle Size')
plt.show()
```

[155] Python



Comparative Analysis of Highway & City MPG Across Vehicle Styles ⛽ 📊

A comprehensive evaluation of fuel efficiency variations across different vehicle styles, highlighting performance differences in urban and highway driving conditions.

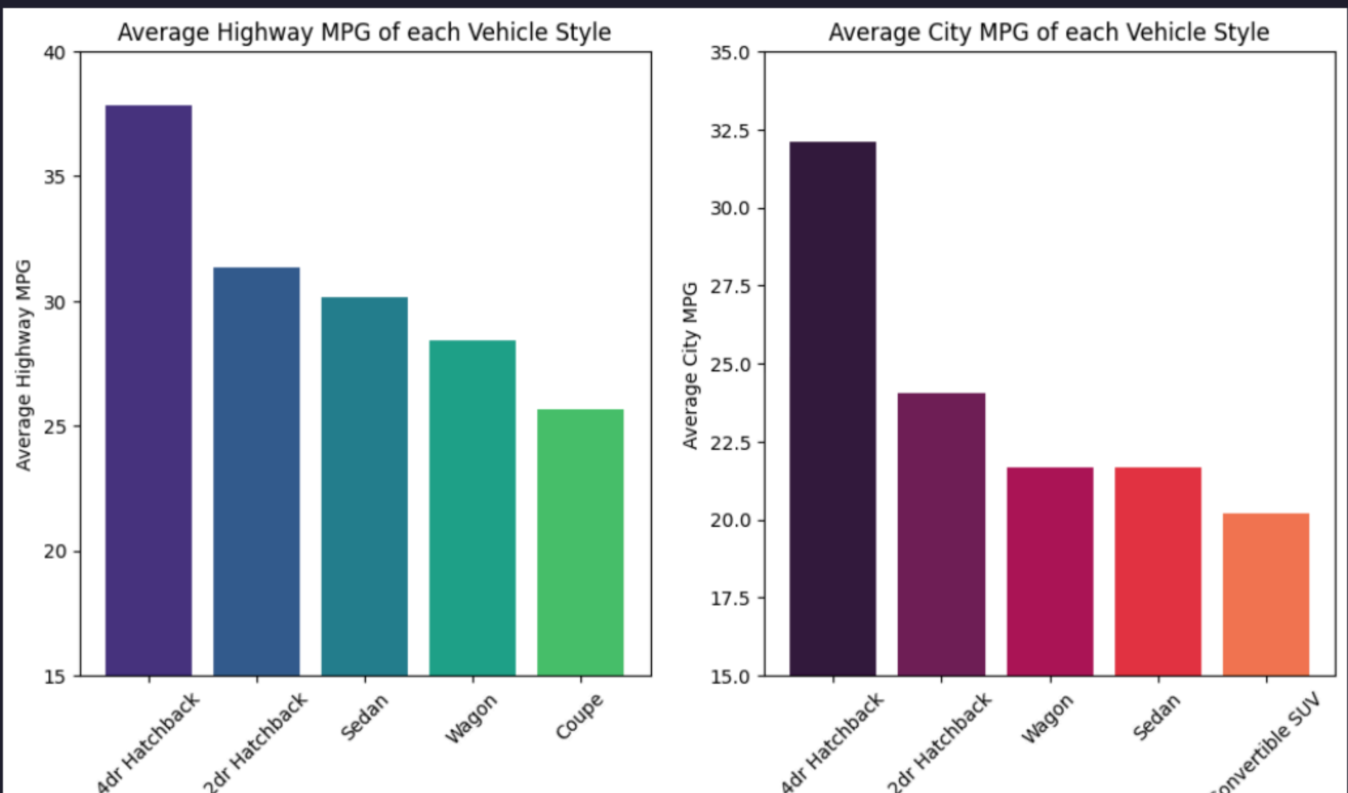
```python
print("Average Highway MPG of each Vehicle Style\n")
avg_hmpg=df.groupby('Vehicle Style')["highway MPG"].mean().sort_values(ascending=False).head(5)
print(avg_hmpg)
print("Average city MPG of each Vehicle Style\n")
avg_cmpg=df.groupby('Vehicle Style')["city mpg"].mean().sort_values(ascending=False).head(5)
print(avg_cmpg)
```

[45] ✓ 0.0s Python

```
Average Highway MPG of each Vehicle Style

Vehicle Style
4dr Hatchback    37.811463
2dr Hatchback    31.355231
Sedan            30.176761
Wagon            28.402135
Coupe            25.637447
Name: highway MPG, dtype: float64
Average city MPG of each Vehicle Style

Vehicle Style
4dr Hatchback      32.088989
2dr Hatchback      24.051095
Wagon              21.686833
Sedan              21.651056
Convertible SUV    20.178571
Name: city mpg, dtype: float64
```

```python
avg_hmpgdf = avg_hmpg.reset_index()
avg_cmpgdf = avg_cmpg.reset_index()
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
axes[0].bar(avg_hmpgdf['Vehicle Style'],avg_hmpgdf['highway MPG'],color=sns.color_palette('viridis'))
axes[0].set_title('Average Highway MPG of each Vehicle Style')
axes[0].set_xlabel('Vehicle Style')
axes[0].tick_params(axis='x', rotation=45)
axes[0].set_ylabel('Average Highway MPG')
axes[0].set_ylim(15, 40)

axes[1].bar(avg_cmpgdf['Vehicle Style'],avg_cmpgdf['city mpg'],color=sns.color_palette('rocket'))
axes[1].set_title('Average City MPG of each Vehicle Style')
axes[1].tick_params(axis='x', rotation=45)
axes[1].set_xlabel('Vehicle Style')
axes[1].set_ylabel('Average City MPG')
axes[1].set_ylim(15, 35)
plt.show()
```

[49] ✓ 0.2s Python



PYTHON-POWERED PROJECT FOR IN-DEPTH DATA ANALYSIS

THANK YOU

work.aroraparth@gmail.com

Portfolio

+91 8218135640