!pip install matplotlib

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

```
df = pd.read_csv("C:\Users\user\Downloads\Dataset_task1\data.csv")
```

df.shape

```
df.shape
```

```
(11914, 16)
```

```
df.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
```

```
df.describe()
```

	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	2.999000e+04
75%	2016.000000	300.00000	6.000000	4.000000	30.000000	22.000000	2009.000000	4.221725e+04
max	2017.000000	1001.00000	16.000000	4.000000	354.000000	137.000000	5657.000000	2.065902e+06

```
df.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
```

```
df.head()
```

	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.dropna(subset = ['Engine Fuel Type'], inplace = True)
```

```
df.dropna(subset = ['Engine HP'], inplace = True)
```

```
df.dropna(subset = ['Engine Cylinders'], inplace = True)
```

```
df.dropna(subset =['Number of Doors'], inplace = True)
```

df.isnull().sum()

```
df.shape
```

```
(11812, 16)
```

```
missing_values = df['Market Category'].isnull().sum() #finding missing values in market category column

total_rows = len(df) #calculating the total row using len()

missing_percentage = (missing_values / total_rows) * 100

print("Missing percentage in 'Market Category' column: {missing_percentage:.2f}%")
```

```
Missing percentage in 'Market Category' column: 31.59%
```

```
df['Market Category'].fillna('Unknown', inplace=True) #filling the missing values using unknown
```

df.isnull().sum()

```
df.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
```

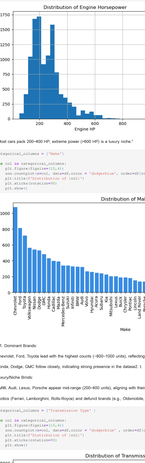
```
df.duplicated().sum() #checking duplicates
```

```
715
```

```
df.drop_duplicates(inplace = True) #removing duplicate values
```

```
# Plot for Engine Fuel Type
sns.countplot(x='Engine Fuel Type', data=df , color = 'darkgreen');
plt.xticks(rotation=90);
plt.title('Distribution of Engine Fuel Types')
plt.show()
```


```
# Plot for Engine HP
df['Engine HP'].hist(bins=50)
plt.title('Distribution of Engine Horsepower')
plt.xlabel('Engine HP')
plt.ylabel('Frequency')
plt.show()
```



"Most cars pack 200-400 HP; extreme power (>600 HP) is a luxury niche."

```
categorical_columns = ['Make']

for col in categorical_columns:
    plt.figure(figsize=(15,5))
    sns.countplot(data=df, x=col, color = 'darkgreen', order=df[col].value_counts().index)
    plt.title("Distribution of {col}")
    plt.xticks(rotation=90)
    plt.show()
```


1. Dominant Brands:

Chevrolet, Ford, Toyota lead with the highest counts (~800–1000 units), reflecting mass-market popularity.

Honda, Dodge, GMC follow closely, indicating strong presence in the dataset. 1.

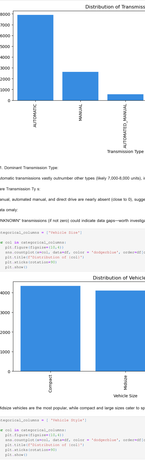
Luxury/Niche Brands:

BMW, Audi, Lexus, Porsche appear mid-range (200–400 units), aligning with their smaller marketshare.

Exotics (Ferrari, Lamborghini, Rolls-Royce) and defunct brands (e.g., Oldsmobile, Plymouth) are rare (<200 units), would be removed.

```
categorical_columns = ['Transmission Type']

for col in categorical_columns:
    plt.figure(figsize=(15,5))
    sns.countplot(data=df, x=col, color = 'darkgreen' , order=df[col].value_counts().index)
    plt.title("Distribution of {col}")
    plt.xticks(rotation=90)
    plt.show()
```



1. Dominant Transmission Type:

Automatic transmissions vastly outnumber other types (likely 7,000-8,000 units), indicating strong consumer preference for convenience.2.

Rare Transmission Ty e

Manual, automated manual, and direct drive are nearly absent (close to 0), suggesting niche markets (e.g., performance cars or E3. Vs).

Data oerialy:

"UNKNOWN" transmissions (if not zero) could indicate data gaps—worth investigating for accuracy.

```
categorical_columns = ['Vehicle Size']

for col in categorical_columns:
    plt.figure(figsize=(15,5))
    sns.countplot(data=df, x=col, color = 'darkgreen', order=df[col].value_counts().index)
    plt.title("Distribution of {col}")
    plt.xticks(rotation=90)
    plt.show()
```


"Midsize vehicles are the most popular, while compact and large sizes cater to specific needs like economy or cargo space."

```
categorical_columns = ['Vehicle Style']

for col in categorical_columns:
    plt.figure(figsize=(15,5))
    sns.countplot(data=df, x=col, color = 'darkgreen', order=df[col].value_counts().index)
    plt.title("Distribution of {col}")
    plt.xticks(rotation=90)
    plt.show()
```


"Sedans and SUVs lead in popularity, while niche styles (convertibles, vans) cater to specific use cases."

```
top_makes = df['Make'].value_counts().nlargest(10).index

sns.countplot(x='Make', data=df[df['Make'].isin(top_makes)].sort_values(by=top_makes , ), color = 'darkgreen')
plt.title('Top 10 Vehicle Brands')
plt.xticks(rotation=90)
plt.show()
```

-> The list shows duplicates (e.g., Toyota, Ford repeated) and typos (e.g., "Volkswagen" for Volkswagen, "MakesMakes") . -> Top brands like Toyota, Ford, and Chevrolet appear consistently, suggesting they dominate the datase

1.

```
plt.figure(figsize=(8,5))
sns.scatterplot(x='Engine HP', y='MSRP', hue='Vehicle Size', data=df)
plt.title('Engine Horsepower vs Car Price')
plt.xlabel('Engine HP')
plt.ylabel('Price ($)')
plt.show()
```


-> Positive Trend: Higher engine horsepower (HP) correlates with higher car prices. -> Size Matters: Large vehicles dominate the high-HP, high-price range; compacts are cheaper with lower HP.->

Outliers: Some high-HP cars are priced mid-range—potential performance bargains or data qui

"Longer, more powerful cars cost more, but outliers may reveal hidden gems."ns.