Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a method of analyzing datasets to understand their main characteristics. It involves summarizing data features, detecting patterns, and uncovering relationships through visual and statistical techniques.

IMPORT LABRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#to ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

READING DATASET

```
df = pd.read_csv("used_cars_data.csv")
```

ANALYZING THE DATA

```
df.shape                          # display the number of
observations(rows) and features(columns) in the dataset

(7253, 14)

df.head()                             # top 5 rows

   S.No.                                  Name    Location  Year  \
0      0            Maruti Wagon R LXI CNG      Mumbai  2010
1      1  Hyundai Creta 1.6 CRDi SX Option        Pune  2015
2      2                      Honda Jazz V     Chennai  2011
3      3                 Maruti Ertiga VDI     Chennai  2012
4      4   Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013

   Kilometers_Driven Fuel_Type Transmission Owner_Type     Mileage
Engine  \
0              72000       CNG       Manual      First  26.6 km/kg
998 CC
1              41000    Diesel       Manual      First  19.67 kmpl
1582 CC
2              46000    Petrol       Manual      First   18.2 kmpl
1199 CC
3              87000    Diesel       Manual      First  20.77 kmpl
1248 CC
4              40670    Diesel    Automatic     Second   15.2 kmpl
1968 CC
```

```
       Power  Seats  New_Price  Price
0  58.16 bhp    5.0        NaN   1.75
1  126.2 bhp    5.0        NaN  12.50
2   88.7 bhp    5.0  8.61 Lakh   4.50
3  88.76 bhp    7.0        NaN   6.00
4  140.8 bhp    5.0        NaN  17.74
```

```
df.tail()                        # last 5 rows
```

```
      S.No.                                              Name
Location  \
7248   7248                   Volkswagen Vento Diesel Trendline
Hyderabad
7249   7249                              Volkswagen Polo GT TSI
Mumbai
7250   7250                              Nissan Micra Diesel XV
Kolkata
7251   7251                              Volkswagen Polo GT TSI
Pune
7252   7252  Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...
Kochi

      Year  Kilometers_Driven Fuel_Type Transmission Owner_Type
Mileage  \
7248  2011              89411    Diesel       Manual      First  20.54
kmpl
7249  2015              59000    Petrol    Automatic      First  17.21
kmpl
7250  2012              28000    Diesel       Manual      First  23.08
kmpl
7251  2013              52262    Petrol    Automatic      Third   17.2
kmpl
7252  2014              72443    Diesel    Automatic      First   10.0
kmpl

       Engine      Power  Seats New_Price  Price
7248  1598 CC  103.6 bhp    5.0       NaN    NaN
7249  1197 CC  103.6 bhp    5.0       NaN    NaN
7250  1461 CC   63.1 bhp    5.0       NaN    NaN
7251  1197 CC  103.6 bhp    5.0       NaN    NaN
7252  2148 CC    170 bhp    5.0       NaN    NaN
```

```
df.info()                                  #information of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   S.No.              7253 non-null   int64
```

```
 1   Name               7253 non-null   object
 2   Location           7253 non-null   object
 3   Year               7253 non-null   int64
 4   Kilometers_Driven  7253 non-null   int64
 5   Fuel_Type          7253 non-null   object
 6   Transmission       7253 non-null   object
 7   Owner_Type         7253 non-null   object
 8   Mileage            7251 non-null   object
 9   Engine             7207 non-null   object
 10  Power              7207 non-null   object
 11  Seats              7200 non-null   float64
 12  New_Price          1006 non-null   object
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

Check for Duplication

```
df.nunique()
```

```
S.No.                7253
Name                 2041
Location               11
Year                   23
Kilometers_Driven    3660
Fuel_Type               5
Transmission            2
Owner_Type              4
Mileage               450
Engine                150
Power                 386
Seats                   9
New_Price             625
Price                1373
dtype: int64
```

Missing Values Calculation

```
df.isnull()                    # identify null values in the data
```

```
      S.No.   Name  Location   Year  Kilometers_Driven  Fuel_Type  \
0     False  False     False  False              False      False
1     False  False     False  False              False      False
2     False  False     False  False              False      False
3     False  False     False  False              False      False
4     False  False     False  False              False      False
...     ...    ...       ...    ...                ...        ...
7248  False  False     False  False              False      False
7249  False  False     False  False              False      False
7250  False  False     False  False              False      False
```

```
7251  False  False     False  False              False      False
7252  False  False     False  False              False      False

      Transmission  Owner_Type  Mileage  Engine  Power  Seats
New_Price  \
0            False       False    False   False  False  False
True
1            False       False    False   False  False  False
True
2            False       False    False   False  False  False
False
3            False       False    False   False  False  False
True
4            False       False    False   False  False  False
True
...            ...         ...      ...     ...    ...    ...         .
..
7248         False       False    False   False  False  False
True
7249         False       False    False   False  False  False
True
7250         False       False    False   False  False  False
True
7251         False       False    False   False  False  False
True
7252         False       False    False   False  False  False
True

      Price
0     False
1     False
2     False
3     False
4     False
...     ...
7248   True
7249   True
7250   True
7251   True
7252   True

[7253 rows x 14 columns]
```

```
df.isnull().sum()                #get the number of missing records in
each column
```

```
S.No.                    0
Name                     0
Location                 0
Year                     0
```

```
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 2
Engine                 46
Power                  46
Seats                  53
New_Price            6247
Price                1234
dtype: int64
```

```
(df.isnull().sum()/(len(df)))*100                    #calculate the
percentage of missing values
```

```
S.No.                0.000000
Name                 0.000000
Location             0.000000
Year                 0.000000
Kilometers_Driven    0.000000
Fuel_Type            0.000000
Transmission         0.000000
Owner_Type           0.000000
Mileage              0.027575
Engine               0.634220
Power                0.634220
Seats                0.730732
New_Price           86.129877
Price               17.013650
dtype: float64
```

Data Reduction

```
# Remove S.No. column from data
df = df.drop(['S.No.'], axis = 1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               7253 non-null   object
 1   Location           7253 non-null   object
 2   Year               7253 non-null   int64
 3   Kilometers_Driven  7253 non-null   int64
 4   Fuel_Type          7253 non-null   object
 5   Transmission       7253 non-null   object
 6   Owner_Type         7253 non-null   object
 7   Mileage            7251 non-null   object
```

```
 8   Engine             7207 non-null   object
 9   Power              7207 non-null   object
 10  Seats              7200 non-null   float64
 11  New_Price          1006 non-null   object
 12  Price              6019 non-null   float64
dtypes: float64(2), int64(2), object(9)
memory usage: 736.8+ KB
```

FEATURES ENGINERING

```
# Handle missing values
df['Mileage'] = df['Mileage'].fillna(df['Mileage'].mode()[0])
df['Engine'] = df['Engine'].fillna(df['Engine'].mode()[0])
df['Power'] = df['Power'].fillna(df['Power'].mode()[0])
df['Seats'] = df['Seats'].fillna(df['Seats'].mode()[0])
df.drop(columns=['New_Price'], inplace=True)  # Drop column with
excessive missing values
df = df.dropna(subset=['Price']).reset_index(drop=True)
```

Creating Features

```
from datetime import date
date.today().year
df['Car_Age']=date.today().year-df['Year']
df.head()
```

```
                                  Name    Location  Year
Kilometers_Driven  \
0            Maruti Wagon R LXI CNG      Mumbai  2010
72000
1  Hyundai Creta 1.6 CRDi SX Option        Pune  2015
41000
2                      Honda Jazz V     Chennai  2011
46000
3                 Maruti Ertiga VDI     Chennai  2012
87000
4   Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013
40670

  Fuel_Type Transmission Owner_Type     Mileage   Engine      Power
Seats  \
0       CNG       Manual      First  26.6 km/kg   998 CC  58.16 bhp
5.0
1    Diesel       Manual      First  19.67 kmpl  1582 CC  126.2 bhp
5.0
2    Petrol       Manual      First   18.2 kmpl  1199 CC   88.7 bhp
5.0
3    Diesel       Manual      First  20.77 kmpl  1248 CC  88.76 bhp
7.0
4    Diesel    Automatic     Second   15.2 kmpl  1968 CC  140.8 bhp
```

```
5.0
```

```
   Price  Car_Age    Brand      Model
0   1.75       15   Maruti     WagonR
1  12.50       10  Hyundai   Creta1.6
2   4.50       14    Honda      JazzV
3   6.00       13   Maruti  ErtigaVDI
4  17.74       12     Audi      A4New
```

```
# Extract numerical values from 'Mileage', 'Engine', and 'Power'
df['Mileage'] = df['Mileage'].str.extract('(\d+\.?\d*)').astype(float)
df['Engine'] = df['Engine'].str.extract('(\d+\.?\d*)').astype(float)
df['Power'] = df['Power'].str.extract('(\d+\.?\d*)').astype(float)

df['Price_Per_Kilometer'] = df['Price'] / df['Kilometers_Driven']

df.head(10)
```

```
                                     Name    Location  Year
Kilometers_Driven  \
0               Maruti Wagon R LXI CNG      Mumbai  2010
72000
1     Hyundai Creta 1.6 CRDi SX Option        Pune  2015
41000
2                         Honda Jazz V     Chennai  2011
46000
3                     Maruti Ertiga VDI     Chennai  2012
87000
4      Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013
40670
5      Hyundai EON LPG Era Plus Option   Hyderabad  2012
75000
6               Nissan Micra Diesel XV      Jaipur  2013
86999
7    Toyota Innova Crysta 2.8 GX AT 8S      Mumbai  2016
36000
8  Volkswagen Vento Diesel Comfortline        Pune  2013
64430
9       Tata Indica Vista Quadrajet LS     Chennai  2012
65932

  Fuel_Type Transmission Owner_Type  Mileage  Engine   Power  Seats
Price  \
0       CNG       Manual      First    26.60   998.0   58.16    5.0
1.75
1    Diesel       Manual      First    19.67  1582.0  126.20    5.0
12.50
2    Petrol       Manual      First    18.20  1199.0   88.70    5.0
4.50
3    Diesel       Manual      First    20.77  1248.0   88.76    7.0
```

```
6.00
4    Diesel    Automatic     Second    15.20  1968.0  140.80    5.0
17.74
5       LPG       Manual      First    21.10   814.0   55.20    5.0
2.35
6    Diesel       Manual      First    23.08  1461.0   63.10    5.0
3.50
7    Diesel    Automatic      First    11.36  2755.0  171.50    8.0
17.50
8    Diesel       Manual      First    20.54  1598.0  103.60    5.0
5.20
9    Diesel       Manual     Second    22.30  1248.0   74.00    5.0
1.95
```

	Car_Age	Brand	Model	Price_Per_Kilometer
0	15	Maruti	WagonR	0.000024
1	10	Hyundai	Creta1.6	0.000305
2	14	Honda	JazzV	0.000098
3	13	Maruti	ErtigaVDI	0.000069
4	12	Audi	A4New	0.000436
5	13	Hyundai	EONLPG	0.000031
6	12	Nissan	MicraDiesel	0.000040
7	9	Toyota	InnovaCrysta	0.000486
8	12	Volkswagen	VentoDiesel	0.000081
9	13	Tata	IndicaVista	0.000030

Let's split the name and introduce new variables "Brand" and "Model

```
df['Brand'] = df.Name.str.split().str.get(0)

df['Model'] = df.Name.str.split().str.get(1) +
df.Name.str.split().str.get(2)

df[['Name','Brand','Model']]
```

	Name	Brand	Model
0	Maruti Wagon R LXI CNG	Maruti	WagonR
1	Hyundai Creta 1.6 CRDi SX Option	Hyundai	Creta1.6
2	Honda Jazz V	Honda	JazzV
3	Maruti Ertiga VDI	Maruti	ErtigaVDI
4	Audi A4 New 2.0 TDI Multitronic	Audi	A4New
...	...	...	...
6014	Maruti Swift VDI	Maruti	SwiftVDI
6015	Hyundai Xcent 1.1 CRDi S	Hyundai	Xcent1.1
6016	Mahindra Xylo D4 BSIV	Mahindra	XyloD4
6017	Maruti Wagon R VXI	Maruti	WagonR
6018	Chevrolet Beat Diesel	Chevrolet	BeatDiesel

```
[6019 rows x 3 columns]
```

Data Cleaning/Wrangling

```
print(df.Brand.unique())
print(df.Brand.nunique())
```

```
['Maruti' 'Hyundai' 'Honda' 'Audi' 'Nissan' 'Toyota' 'Volkswagen'
'Tata'
 'Land' 'Mitsubishi' 'Renault' 'Mercedes-Benz' 'BMW' 'Mahindra' 'Ford'
 'Porsche' 'Datsun' 'Jaguar' 'Volvo' 'Chevrolet' 'Skoda' 'Mini' 'Fiat'
 'Jeep' 'Smart' 'Ambassador' 'Isuzu' 'ISUZU' 'Force' 'Bentley'
 'Lamborghini']
31
```

```
searchfor = ['Isuzu' ,'ISUZU','Mini','Land']
df[df.Brand.str.contains('|'.join(searchfor))].head(5)
```

```
                                     Name    Location  Year
Kilometers_Driven  \
13      Land Rover Range Rover 2.2L Pure       Delhi  2014
72000
14        Land Rover Freelander 2 TD4 SE        Pune  2012
85000
176             Mini Countryman Cooper D      Jaipur  2017
8525
191  Land Rover Range Rover 2.2L Dynamic  Coimbatore  2018
36091
228            Mini Cooper Convertible S       Kochi  2017
26327

    Fuel_Type Transmission Owner_Type  Mileage  Engine   Power  Seats
Price  \
13     Diesel    Automatic      First    12.70  2179.0  187.70    5.0
27.00
14     Diesel    Automatic     Second     0.00  2179.0  115.00    5.0
17.50
176    Diesel    Automatic     Second    16.60  1998.0  112.00    5.0
23.00
191    Diesel    Automatic      First    12.70  2179.0  187.70    5.0
55.76
228    Petrol    Automatic      First    16.82  1998.0  189.08    4.0
35.67

     Car_Age Brand              Model  Price_Per_Kilometer
13        11  Land         RoverRange             0.000375
14        13  Land    RoverFreelander             0.000206
176        8  Mini   CountrymanCooper             0.002698
```

```
191         7  Land         RoverRange              0.001545
228         8  Mini  CooperConvertible              0.001355
```

```
df["Brand"].replace({"ISUZU": "Isuzu", "Mini": "Mini
Cooper","Land":"Land Rover"}, inplace=True)    #replace values
```

```
df.head()
```

```
                                Name    Location  Year
Kilometers_Driven  \
0            Maruti Wagon R LXI CNG      Mumbai  2010
72000
1  Hyundai Creta 1.6 CRDi SX Option        Pune  2015
41000
2                      Honda Jazz V     Chennai  2011
46000
3                 Maruti Ertiga VDI     Chennai  2012
87000
4   Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013
40670

  Fuel_Type Transmission Owner_Type  Mileage  Engine   Power  Seats
Price  \
0       CNG       Manual      First    26.60   998.0   58.16    5.0
1.75
1    Diesel       Manual      First    19.67  1582.0  126.20    5.0
12.50
2    Petrol       Manual      First    18.20  1199.0   88.70    5.0
4.50
3    Diesel       Manual      First    20.77  1248.0   88.76    7.0
6.00
4    Diesel    Automatic     Second    15.20  1968.0  140.80    5.0
17.74

   Car_Age    Brand      Model  Price_Per_Kilometer
0       15   Maruti     WagonR             0.000024
1       10  Hyundai   Creta1.6             0.000305
2       14    Honda      JazzV             0.000098
3       13   Maruti  ErtigaVDI             0.000069
4       12     Audi      A4New             0.000436
```

EDA in PYTHON

1.Statistics Summary

```
df.describe().T                                                    #
statistics summary of data belonging to numerical datatype such as
int, float
```

```
                        count          mean           std
min  \
Year                   6019.0   2013.358199      3.269742  1998.000000

Kilometers_Driven      6019.0  58738.380296  91268.843206   171.000000

Mileage                6019.0     18.134584      4.581574     0.000000

Engine                 6019.0   1618.738827    600.445858    72.000000

Power                  5912.0    113.014026     53.797403    34.200000

Seats                  6019.0      5.276790      0.806346     0.000000

Price                  6019.0      9.479468     11.187917     0.440000

Car_Age                6019.0     11.641801      3.269742     6.000000

Price_Per_Kilometer    6019.0      0.000360      0.001323     0.000003


                                25%           50%           75%
max
Year                    2011.000000   2014.000000   2016.000000
2.019000e+03
Kilometers_Driven      34000.000000  53000.000000  73000.000000
6.500000e+06
Mileage                   15.170000     18.150000     21.100000
3.354000e+01
Engine                  1197.000000   1493.000000   1969.000000
5.998000e+03
Power                     75.000000     94.000000    138.100000
5.600000e+02
Seats                      5.000000      5.000000      5.000000
1.000000e+01
Price                      3.500000      5.640000      9.950000
1.600000e+02
Car_Age                    9.000000     11.000000     14.000000
2.700000e+01
Price_Per_Kilometer        0.000058      0.000115      0.000274
4.375000e-02

df.describe(include='all').T                                    #
include object, category etc

                     count unique                      top  freq  \
Name                  6019   1876  Mahindra XUV500 W8 2WD    49
Location              6019     11                  Mumbai   790
Year                 6019.0    NaN                     NaN   NaN
Kilometers_Driven    6019.0    NaN                     NaN   NaN
Fuel_Type             6019      5                  Diesel  3205
```

Transmission	6019	2		Manual	4299
Owner_Type	6019	4		First	4929
Mileage	6019.0	NaN		NaN	NaN
Engine	6019.0	NaN		NaN	NaN
Power	5912.0	NaN		NaN	NaN
Seats	6019.0	NaN		NaN	NaN
Price	6019.0	NaN		NaN	NaN
Car_Age	6019.0	NaN		NaN	NaN
Brand	6019	30		Maruti	1211
Model	6019	689		WagonR	154
Price_Per_Kilometer	6019.0	NaN		NaN	NaN

	mean	std	min	25%	50% \
Name	NaN	NaN	NaN	NaN	NaN
Location	NaN	NaN	NaN	NaN	NaN
Year	2013.358199	3.269742	1998.0	2011.0	2014.0
Kilometers_Driven	58738.380296	91268.843206	171.0	34000.0	53000.0
Fuel_Type	NaN	NaN	NaN	NaN	NaN
Transmission	NaN	NaN	NaN	NaN	NaN
Owner_Type	NaN	NaN	NaN	NaN	NaN
Mileage	18.134584	4.581574	0.0	15.17	18.15
Engine	1618.738827	600.445858	72.0	1197.0	1493.0
Power	113.014026	53.797403	34.2	75.0	94.0
Seats	5.27679	0.806346	0.0	5.0	5.0
Price	9.479468	11.187917	0.44	3.5	5.64
Car_Age	11.641801	3.269742	6.0	9.0	11.0
Brand	NaN	NaN	NaN	NaN	NaN
Model	NaN	NaN	NaN	NaN	NaN
Price_Per_Kilometer	0.00036	0.001323	0.000003	0.000058	0.000115

	75%	max
Name	NaN	NaN

```
Location                    NaN        NaN
Year                     2016.0     2019.0
Kilometers_Driven       73000.0  6500000.0
Fuel_Type                   NaN        NaN
Transmission                NaN        NaN
Owner_Type                  NaN        NaN
Mileage                    21.1      33.54
Engine                   1969.0     5998.0
Power                     138.1      560.0
Seats                       5.0       10.0
Price                      9.95      160.0
Car_Age                    14.0       27.0
Brand                       NaN        NaN
Model                       NaN        NaN
Price_Per_Kilometer    0.000274    0.04375
```

lets separate Numerical and categorical variables for easy analysis

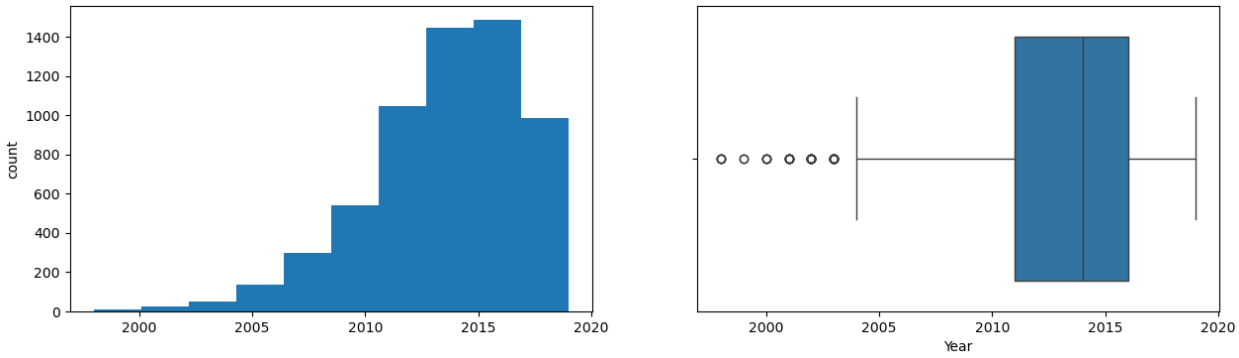
```
cat_cols=df.select_dtypes(include=['object']).columns
num_cols = df.select_dtypes(include=np.number).columns.tolist()
print("Categorical Variables:")
print(cat_cols)
print("Numerical Variables:")
print(num_cols)
```

```
Categorical Variables:
Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type',
'Brand',
       'Model'],
      dtype='object')
Numerical Variables:
['Year', 'Kilometers_Driven', 'Mileage', 'Engine', 'Power', 'Seats',
'Price', 'Car_Age', 'Price_Per_Kilometer']
```

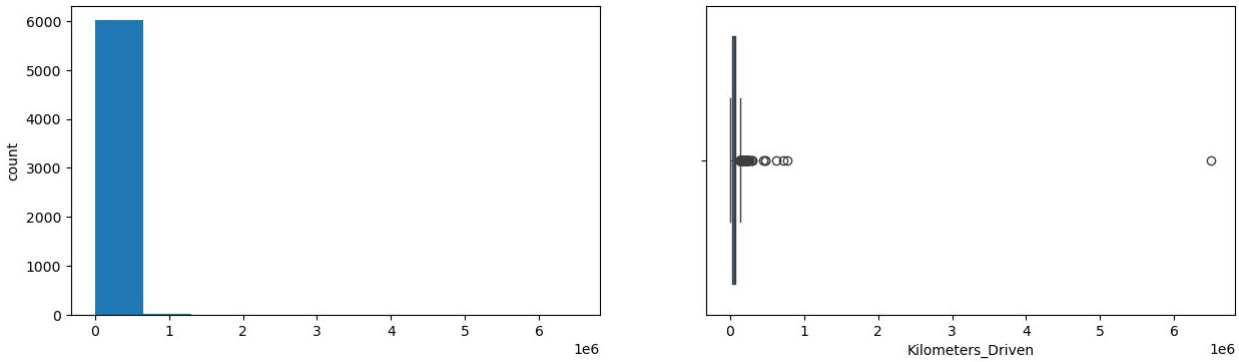
Univariate Analysis

```
for col in num_cols:
    print(col)
    print('Skew :', round(df[col].skew(), 2))
    plt.figure(figsize = (15, 4))
    plt.subplot(1, 2, 1)
    df[col].hist(grid=False)
    plt.ylabel('count')
    plt.subplot(1, 2, 2)
    sns.boxplot(x=df[col])
    plt.show()
```

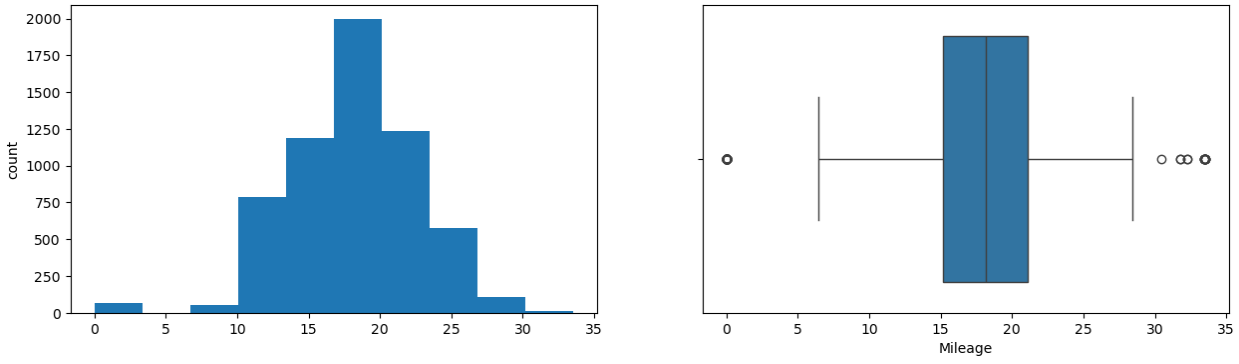
```
Year
Skew : -0.85
```



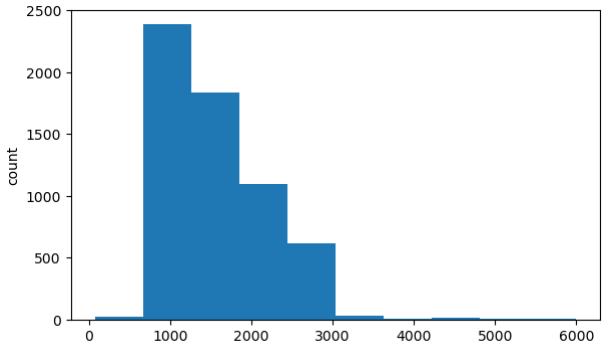

```
Kilometers_Driven
Skew : 58.72
```

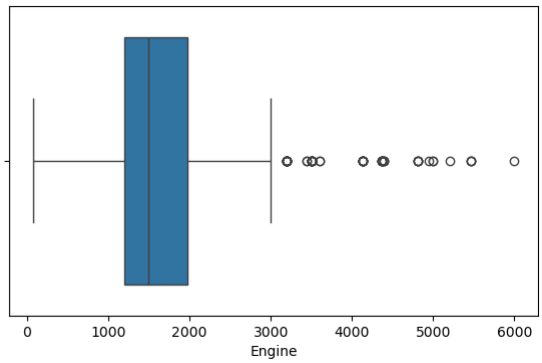



```
Mileage
Skew : -0.43
```

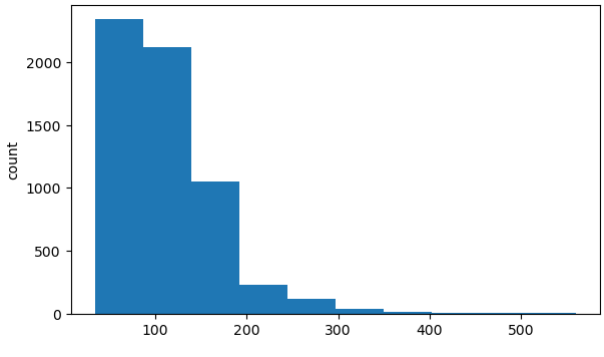


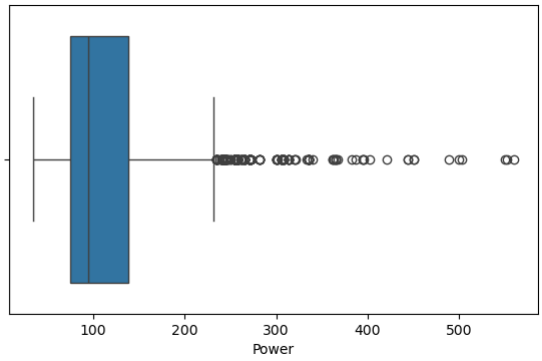

```
Engine
Skew : 1.43
```



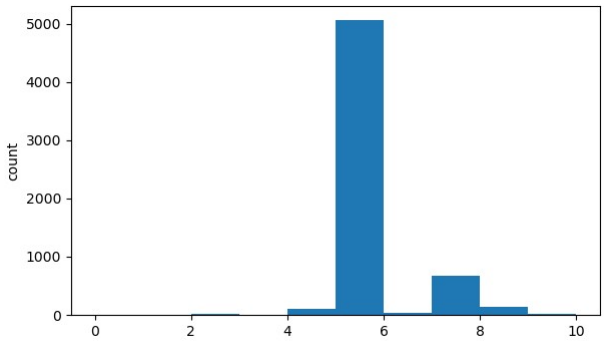


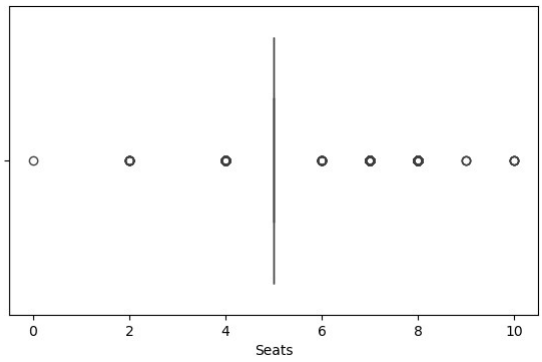

```
Power
Skew : 1.92
```



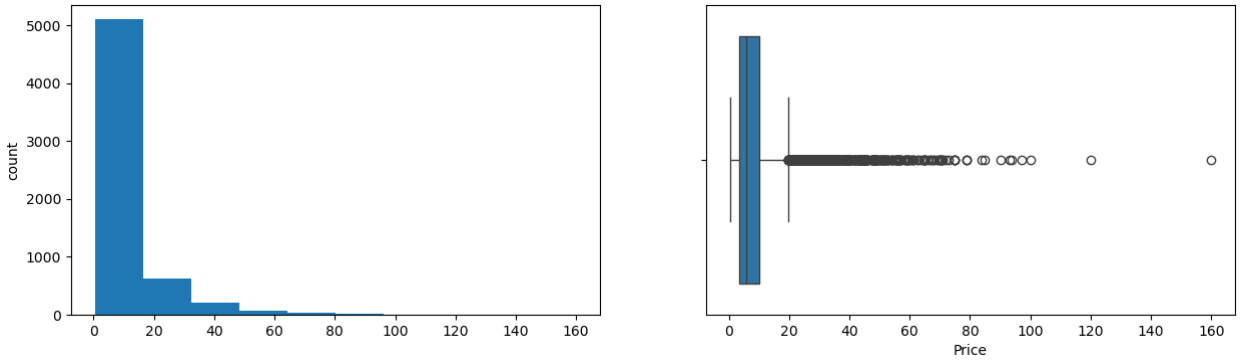



```
Seats
Skew : 1.85
```

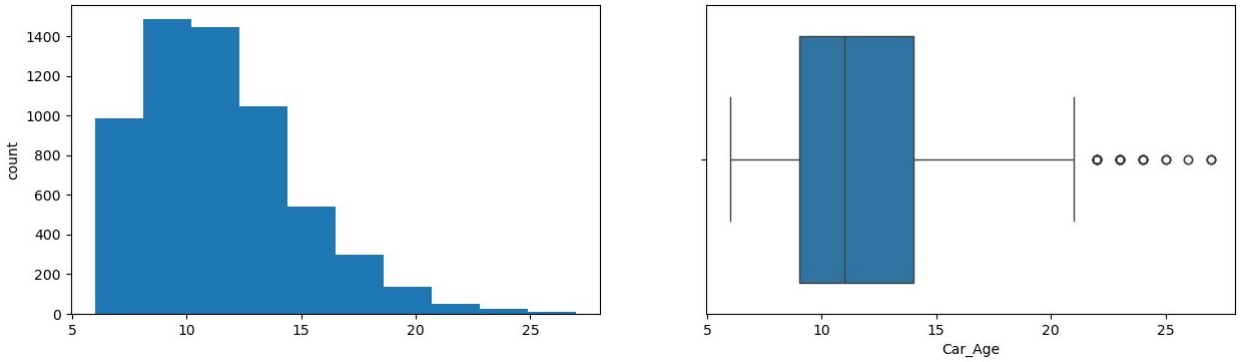




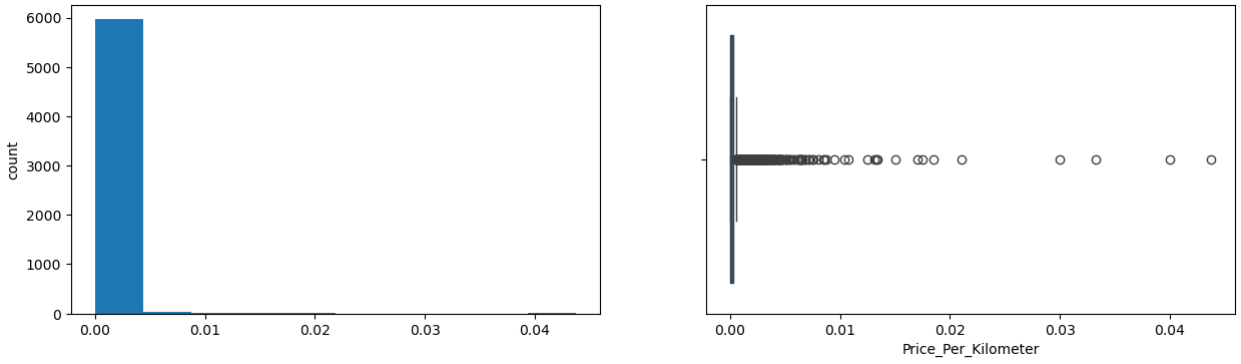

```
Price
Skew : 3.34
```




```
Car_Age
Skew : 0.85
```



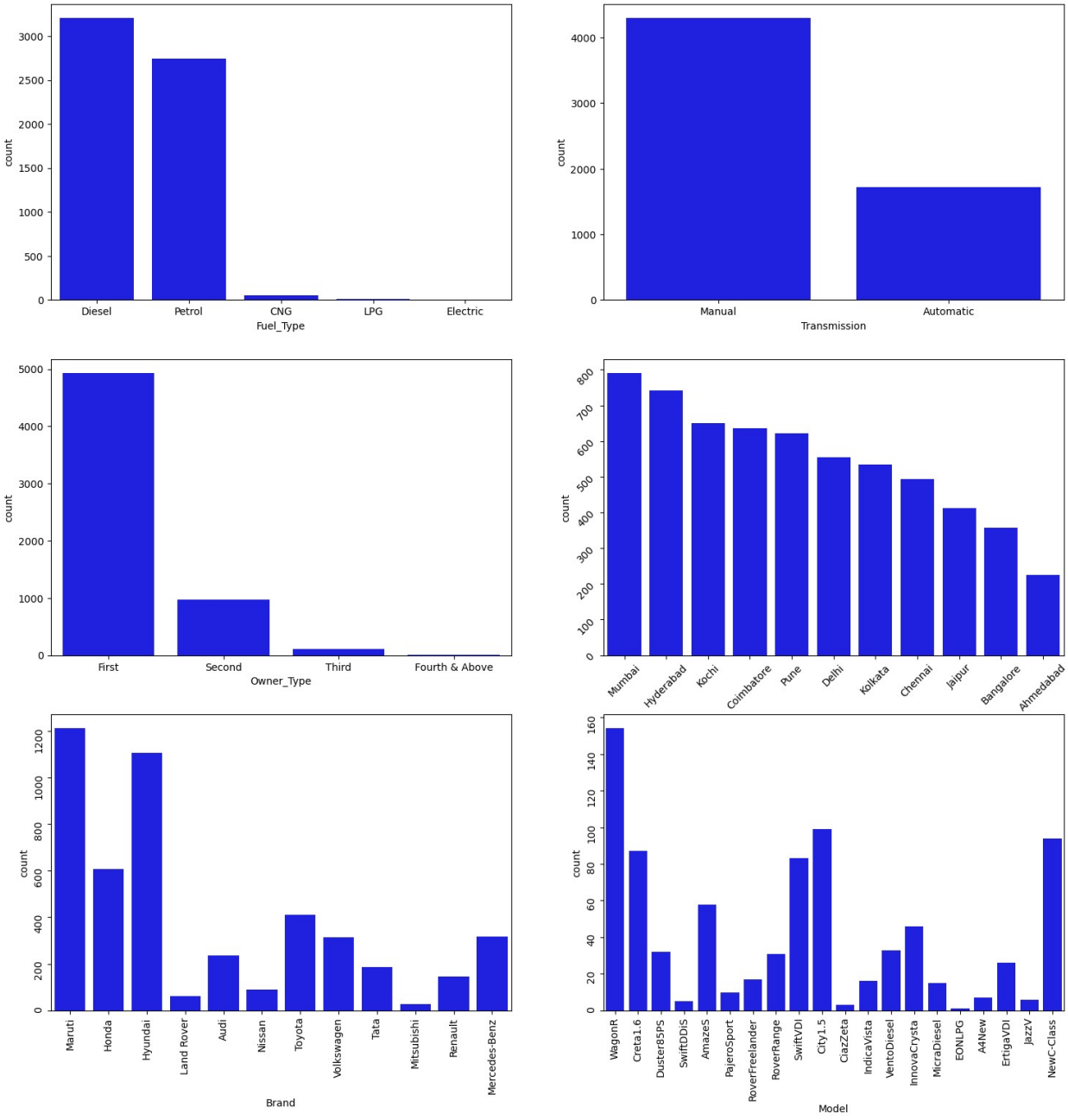

```
Price_Per_Kilometer
Skew : 18.28
```




```
##count plot for categorical data

fig, axes = plt.subplots(3, 2, figsize = (18, 18))
fig.suptitle('Bar plot for all categorical variables in the dataset')
sns.countplot(ax = axes[0, 0], x = 'Fuel_Type', data = df, color =
'blue',
              order = df['Fuel_Type'].value_counts().index);
```

```
sns.countplot(ax = axes[0, 1], x = 'Transmission', data = df, color =
'blue',
              order =df['Transmission'].value_counts().index);
sns.countplot(ax = axes[1, 0], x = 'Owner_Type', data = df, color =
'blue',
              order = df['Owner_Type'].value_counts().index);
sns.countplot(ax = axes[1, 1], x = 'Location', data = df, color =
'blue',
              order = df['Location'].value_counts().index);
sns.countplot(ax = axes[2, 0], x = 'Brand', data = df, color = 'blue',

              order = df['Brand'].head(20).value_counts().index);
sns.countplot(ax = axes[2, 1], x = 'Model', data = df, color = 'blue',

              order = df['Model'].head(20).value_counts().index);
axes[1][1].tick_params(labelrotation=45);
axes[2][0].tick_params(labelrotation=90);
axes[2][1].tick_params(labelrotation=90);
```

Bar plot for all categorical variables in the dataset



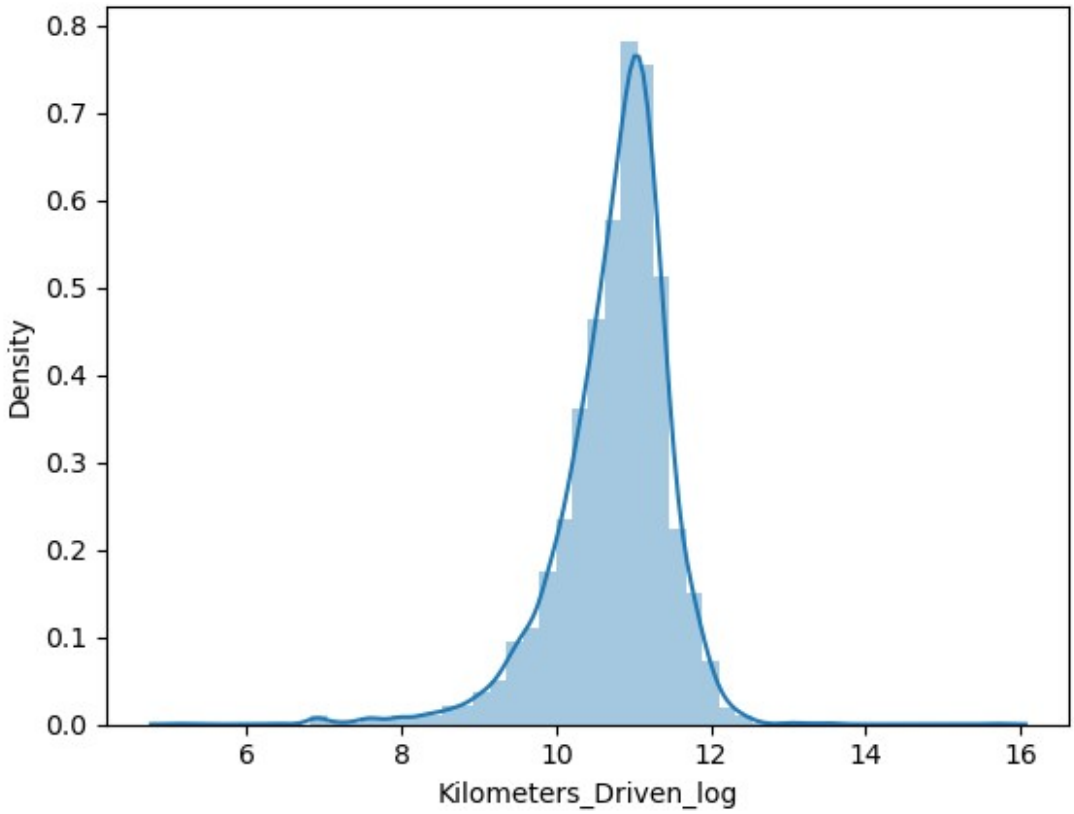
Data Transformation

```
# Function for log transformation of the column
def log_transform(data,col):
    for colname in col:
        if (df[colname] == 1.0).all():
```

```
            df[colname + '_log'] = np.log(df[colname]+1)
        else:
            df[colname + '_log'] = np.log(df[colname])
    df.info()

log_transform(df,['Kilometers_Driven','Price'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Name                   6019 non-null   object 
 1   Location               6019 non-null   object 
 2   Year                   6019 non-null   int64  
 3   Kilometers_Driven      6019 non-null   int64  
 4   Fuel_Type              6019 non-null   object 
 5   Transmission           6019 non-null   object 
 6   Owner_Type             6019 non-null   object 
 7   Mileage                6019 non-null   float64
 8   Engine                 6019 non-null   float64
 9   Power                  5912 non-null   float64
 10  Seats                  6019 non-null   float64
 11  Price                  6019 non-null   float64
 12  Car_Age                6019 non-null   int64  
 13  Brand                  6019 non-null   object 
 14  Model                  6019 non-null   object 
 15  Price_Per_Kilometer    6019 non-null   float64
 16  Kilometers_Driven_log  6019 non-null   float64
 17  Price_log              6019 non-null   float64
dtypes: float64(8), int64(3), object(7)
memory usage: 846.6+ KB
```

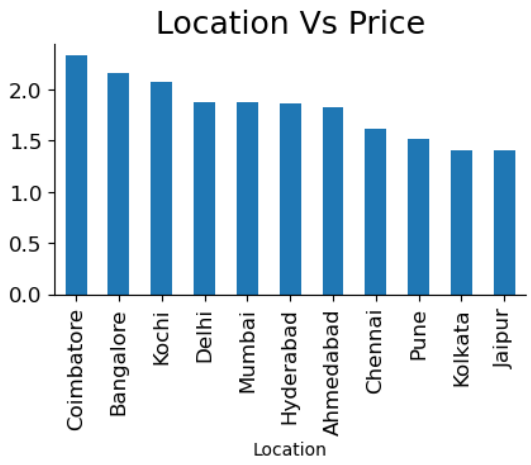
```
#Log transformation of the feature 'Kilometers_Driven'
sns.distplot(df["Kilometers_Driven_log"],
axlabel="Kilometers_Driven_log");
```

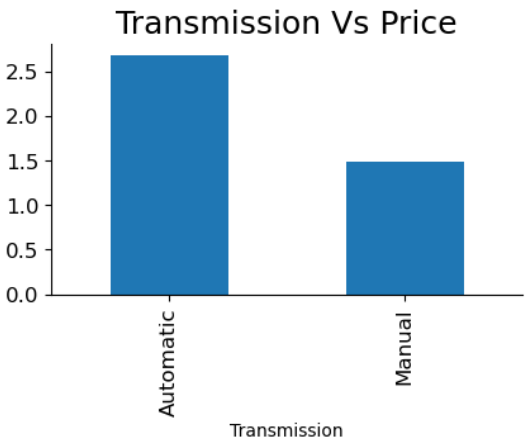


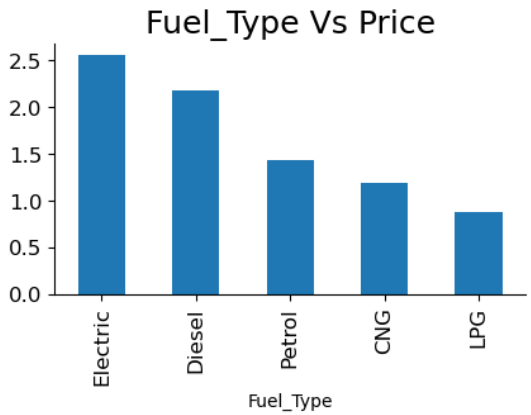
relationship between Categorical variables and continuous variables

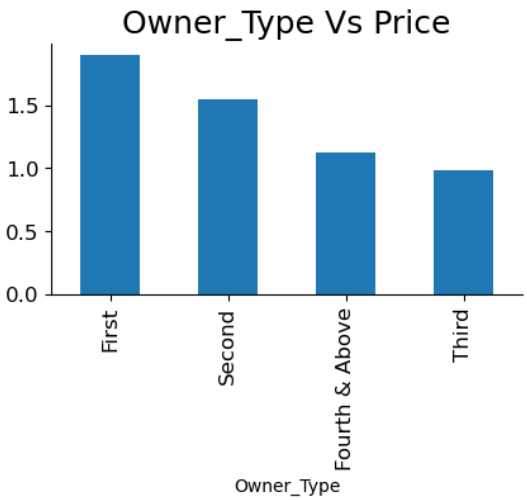
```
fig, axarr = plt.subplots(4, 2, figsize=(12, 18))
df.groupby('Location')
['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[0]
[0], fontsize=12)
axarr[0][0].set_title("Location Vs Price", fontsize=18)
df.groupby('Transmission')
['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[0]
[1], fontsize=12)
axarr[0][1].set_title("Transmission Vs Price", fontsize=18)
df.groupby('Fuel_Type')
['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[1]
[0], fontsize=12)
axarr[1][0].set_title("Fuel_Type Vs Price", fontsize=18)
df.groupby('Owner_Type')
['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[1]
[1], fontsize=12)
axarr[1][1].set_title("Owner_Type Vs Price", fontsize=18)
df.groupby('Brand')
['Price_log'].mean().sort_values(ascending=False).head(10).plot.bar(ax
=axarr[2][0], fontsize=12)
axarr[2][0].set_title("Brand Vs Price", fontsize=18)
```

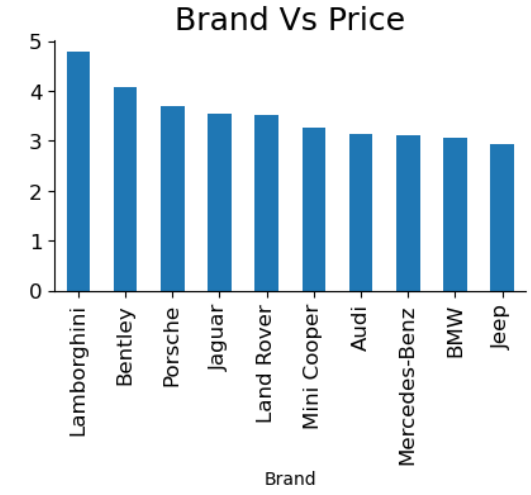
```
df.groupby('Model')
['Price_log'].mean().sort_values(ascending=False).head(10).plot.bar(ax
=axarr[2][1], fontsize=12)
axarr[2][1].set_title("Model Vs Price", fontsize=18)
df.groupby('Seats')
['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[3]
[0], fontsize=12)
axarr[3][0].set_title("Seats Vs Price", fontsize=18)
df.groupby('Car_Age')
['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[3]
[1], fontsize=12)
axarr[3][1].set_title("Car_Age Vs Price", fontsize=18)
plt.subplots_adjust(hspace=1.0)
plt.subplots_adjust(wspace=.5)
sns.despine()
```

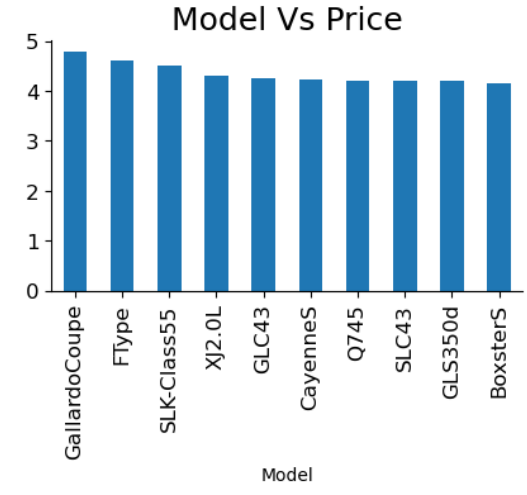

Location Vs Price
0.0
0.5
1.0
1.5
2.0
Coimbatore
Bangalore
Kochi
Delhi
Mumbai
Hyderabad
Ahmedabad
Chennai
Pune
Kolkata
Jaipur
Location

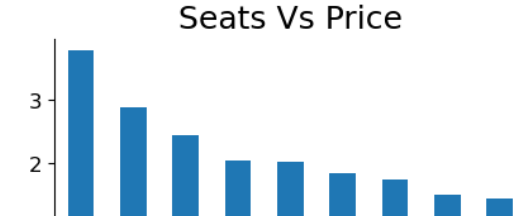

Transmission Vs Price
0.0
0.5
1.0
1.5
2.0
2.5
Automatic
Manual
Transmission

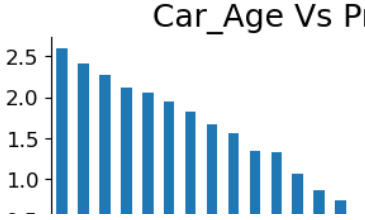

Fuel_Type Vs Price
0.0
0.5
1.0
1.5
2.0
2.5
Electric
Diesel
Petrol
CNG
LPG
Fuel_Type


Owner_Type Vs Price
0.0
0.5
1.0
1.5
First
Second
Fourth & Above
Third
Owner_Type

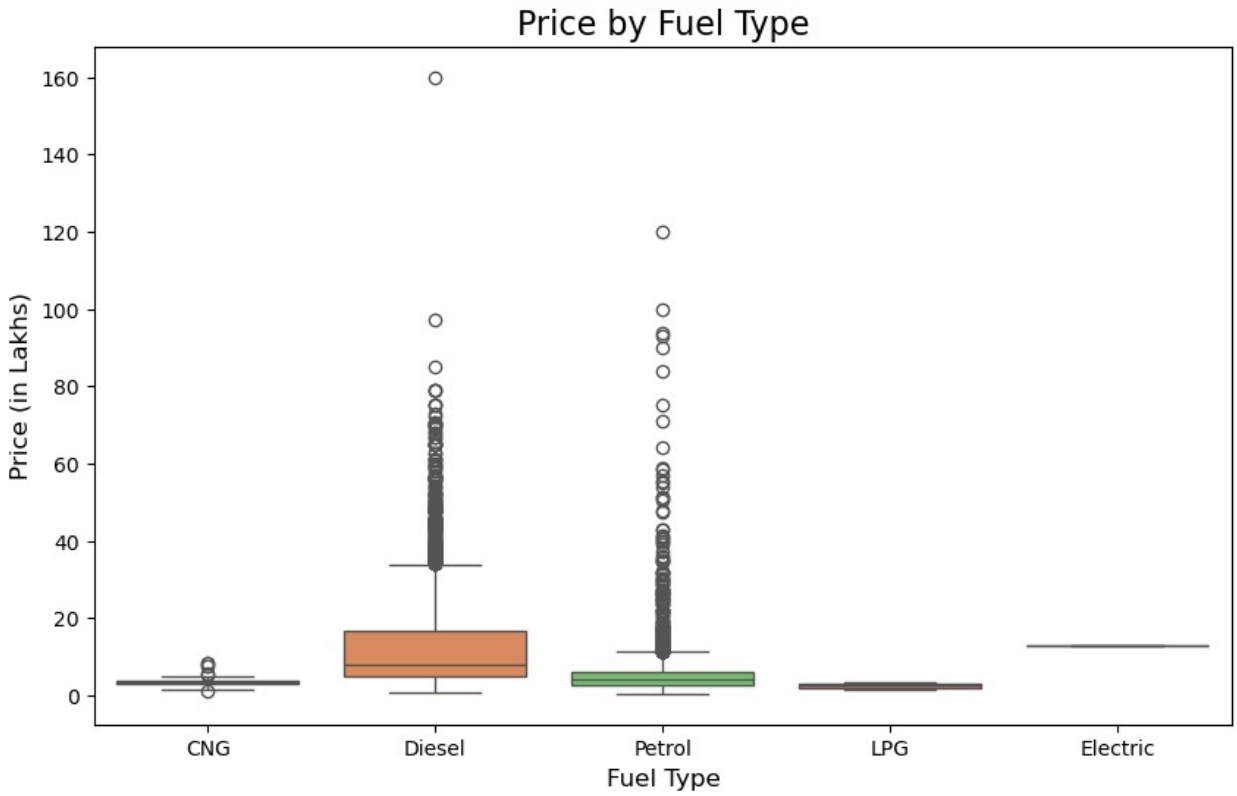

Brand Vs Price
0
1
2
3
4
5
Lamborghini
Bentley
Porsche
Jaguar
Land Rover
Mini Cooper
Audi
Mercedes-Benz
BMW
Jeep
Brand


Model Vs Price
0
1
2
3
4
5
GallardoCoupe
FType
SLK-Class55
XJ2.0L
GLC43
CayenneS
Q745
SLC43
GLS350d
BoxsterS
Model

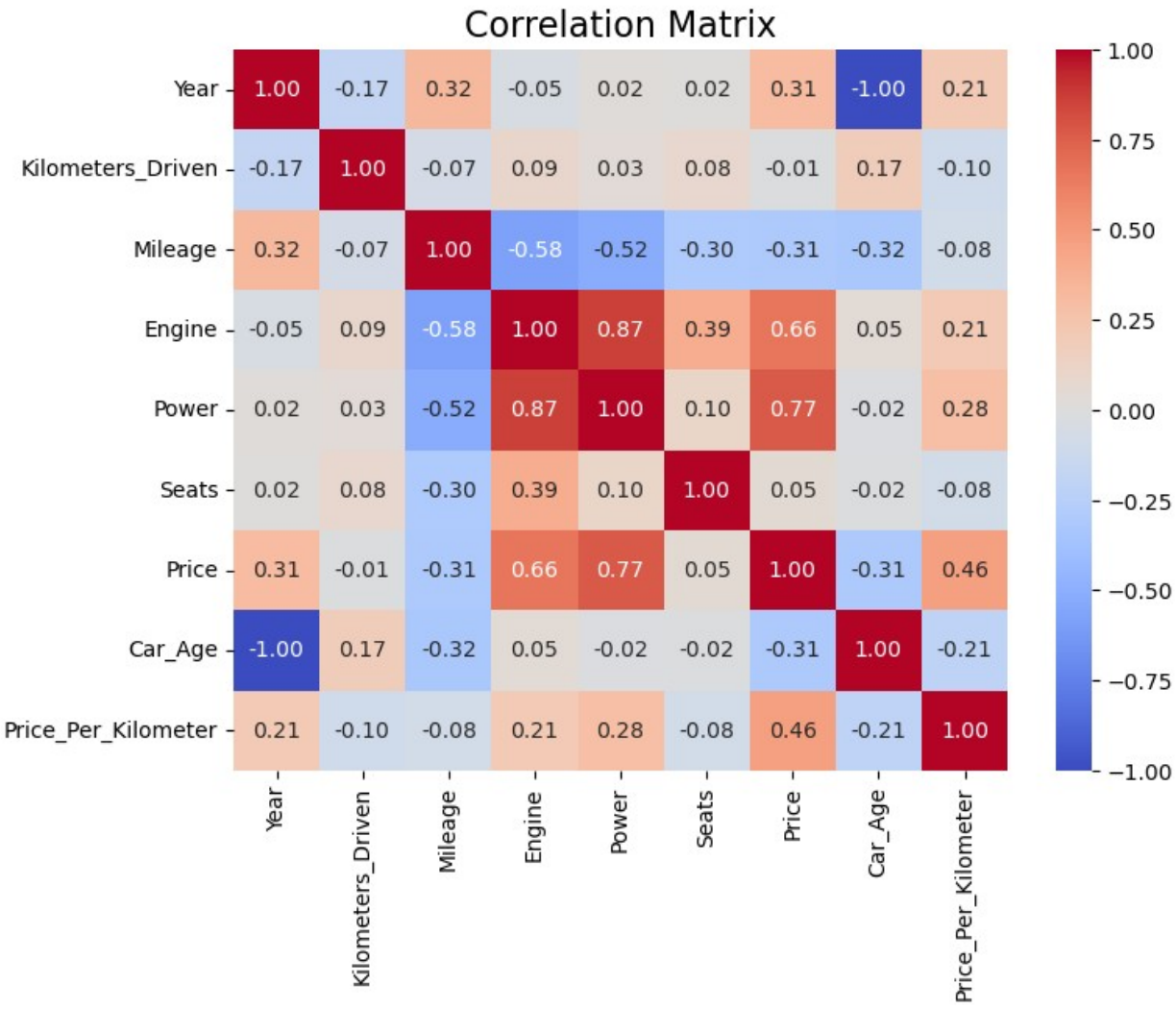

Seats Vs Price
2
3


Car_Age Vs Price
1.0
1.5
2.0
2.5

```
# Boxplot for Price vs Fuel_Type
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Fuel_Type', y='Price', palette='muted')
plt.title('Price by Fuel Type', fontsize=16)
plt.xlabel('Fuel Type', fontsize=12)
plt.ylabel('Price (in Lakhs)', fontsize=12)
plt.show()
```

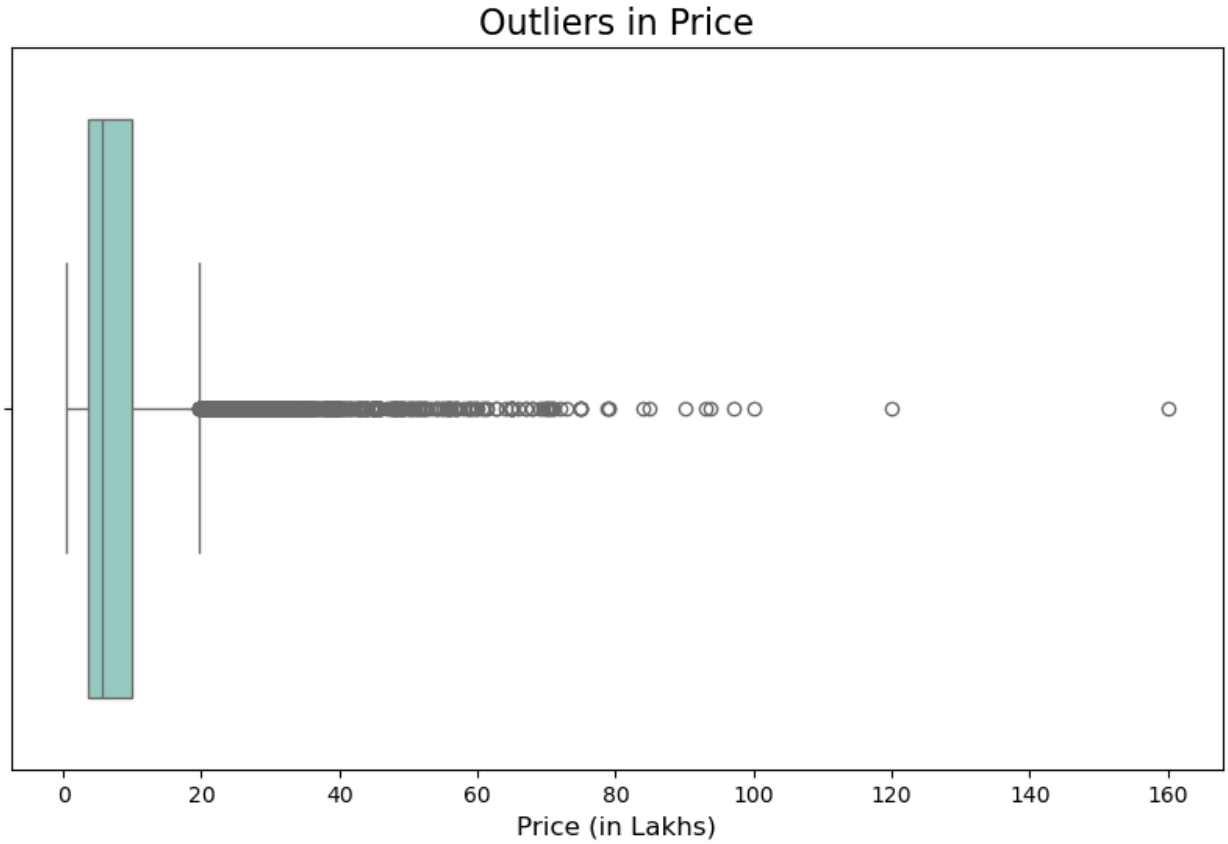


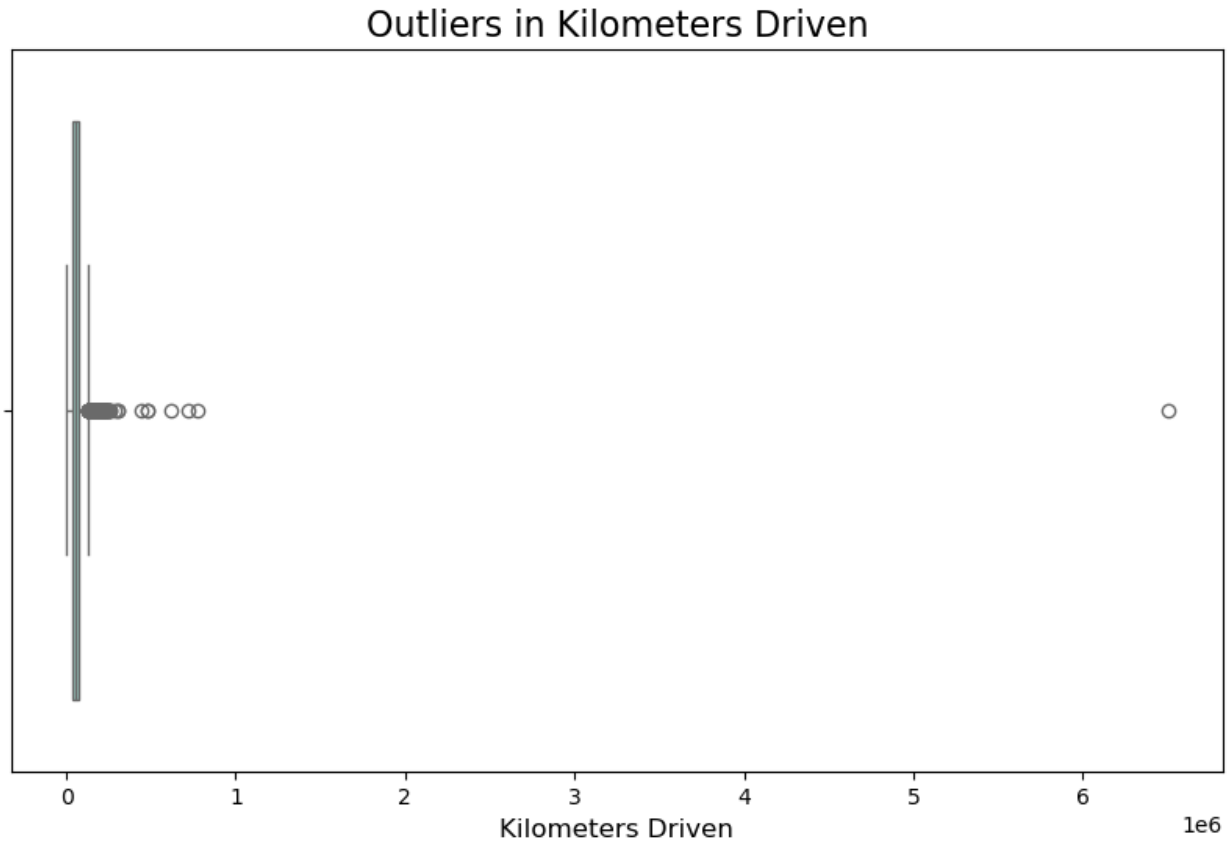

```
### coorelation matrix

# Correlation Heatmap
numerical_cols = ['Car_Age', 'Kilometers_Driven', 'Mileage', 'Engine',
'Power', 'Price']
plt.figure(figsize=(8, 6))
sns.heatmap(df[num_cols].corr(), annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix', fontsize=16)
plt.show()
```



Outlier Detection

```
# Boxplot for Price
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Price', palette='Set3')
plt.title('Outliers in Price', fontsize=16)
plt.xlabel('Price (in Lakhs)', fontsize=12)
plt.show()

# Boxplot for Kilometers Driven
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Kilometers_Driven', palette='Set3')
plt.title('Outliers in Kilometers Driven', fontsize=16)
plt.xlabel('Kilometers Driven', fontsize=12)
plt.show()
```


Outliers in Price
0
20
40
60
80
100
120
140
160
Price (in Lakhs)



Save Visualizations or Reports

```
# Save plots to files
plt.savefig('price_distribution.png')  # Example for saving the price distribution plot

# Save the dataset
output_path = 'cleaned_used_cars_data_with_EDA.csv'
df.to_csv(output_path, index=False)
print(f"Data saved at {output_path}")
```

```
Data saved at cleaned_used_cars_data_with_EDA.csv
```

```
<Figure size 640x480 with 0 Axes>
```