Data Cleaning and Exploratory Data Analysis

In [1]:
```
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')
```

In [3]:
```
data = pd.read_csv("used_cars_data.csv")
```

In [4]:
```
data.head()
```

Out[4]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp

In [5]:
```
data.tail()
```

Out[5]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Po
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	First	20.54 kmpl	1598 CC	1(
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	First	17.21 kmpl	1197 CC	1(
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	First	23.08 kmpl	1461 CC	(
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Third	17.2 kmpl	1197 CC	1(
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	First	10.0 kmpl	2148 CC	

In [6]:
```
data.info()
```

```
<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

In [8]:
```
data.nunique()
```

Out[8]:
```
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

In [9]:
```
data.isnull().sum() #to get the number of missing records in each column
```

Out[9]:
```
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
```

In [10]:
```
(data.isnull().sum()/(len(data)))*100 #percentage of missing values in each column
```

Out[10]:
```
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

In [11]:
```
# Remove S.No. column from data
data = data.drop(['S.No.'], axis = 1)
data.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
```

Data Cleaning/Wrangling

In [13]:
```
print(data.Name.unique())
print(data.Name.nunique())
```

```
['Maruti Wagon R LXI CNG' 'Hyundai Creta 1.6 CRDi SX Option'
 'Honda Jazz V' ... 'Ford EcoSport 1.5 Petrol Ambiente'
 'Jeep Compass 1.4 Sport' 'Hyundai Elite i20 Magna Plus']
2041
```

In [15]:
```
searchfor = ['Isuzu' ,'ISUZU','Mini','Land']
data[data.Name.str.contains('|'.join(searchfor))].head(5)
```

Out[15]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	S
13	Land Rover Range Rover 2.2L Pure	Delhi	2014	72000	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	
14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	Second	0.0 kmpl	2179 CC	115 bhp	
176	Mini Countryman Cooper D	Jaipur	2017	8525	Diesel	Automatic	Second	16.6 kmpl	1998 CC	112 bhp	
191	Land Rover Range Rover 2.2L Dynamic	Coimbatore	2018	36091	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	
228	Mini Cooper Convertible S	Kochi	2017	26327	Petrol	Automatic	First	16.82 kmpl	1998 CC	189.08 bhp	

In [16]:
```
data["Name"].replace({"ISUZU": "Isuzu", "Mini": "Mini Cooper","Land":"Land Rover"}, inplace=True)
```

Exploratory Data Analysis

In [17]:
```
data.describe()
```

Out[17]:

	Year	Kilometers_Driven	Seats	Price
count	7253.000000	7.253000e+03	7200.000000	6019.000000
mean	2013.365366	5.869906e+04	5.279722	9.479468
std	3.254421	8.442772e+04	0.811660	11.187917
min	1996.000000	1.710000e+02	0.000000	0.440000
25%	2011.000000	3.400000e+04	5.000000	3.500000
50%	2014.000000	5.341600e+04	5.000000	5.640000
75%	2016.000000	7.300000e+04	5.000000	9.950000
max	2019.000000	6.500000e+06	10.000000	160.000000

In [18]:
```
data.describe(include='all').T
```

Out[18]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	n
Name	7253	2041	Mahindra XUV500 W8 2WD	55	NaN	NaN	NaN	NaN	NaN	NaN	N
Location	7253	11	Mumbai	949	NaN	NaN	NaN	NaN	NaN	NaN	N
Year	7253.0	NaN	NaN	NaN	2013.365366	3.254421	1996.0	2011.0	2014.0	2016.0	201
Kilometers_Driven	7253.0	NaN	NaN	NaN	58699.063146	84427.720583	171.0	34000.0	53416.0	73000.0	650000
Fuel_Type	7253	5	Diesel	3852	NaN	NaN	NaN	NaN	NaN	NaN	N
Transmission	7253	2	Manual	5204	NaN	NaN	NaN	NaN	NaN	NaN	N
Owner_Type	7253	4	First	5952	NaN	NaN	NaN	NaN	NaN	NaN	N
Mileage	7251	450	17.0 kmpl	207	NaN	NaN	NaN	NaN	NaN	NaN	N
Engine	7207	150	1197 CC	732	NaN	NaN	NaN	NaN	NaN	NaN	N
Power	7207	386	74 bhp	280	NaN	NaN	NaN	NaN	NaN	NaN	N
Seats	7200.0	NaN	NaN	NaN	5.279722	0.81166	0.0	5.0	5.0	5.0	1
New_Price	1006	625	63.71 Lakh	6	NaN	NaN	NaN	NaN	NaN	NaN	N
Price	6019.0	NaN	NaN	NaN	9.479468	11.187917	0.44	3.5	5.64	9.95	16

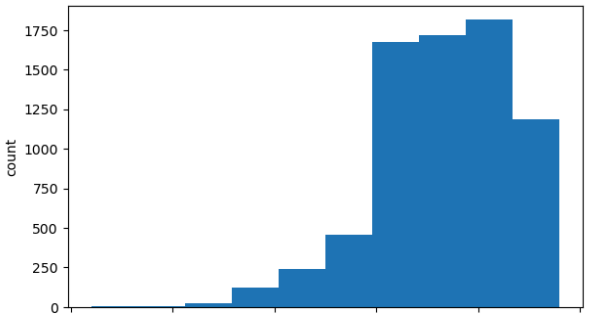
In [19]:
```
#separate Numerical and categorical variables for easy analysis
cat_cols=data.select_dtypes(include=['object']).columns
num_cols = data.select_dtypes(include=np.number).columns.tolist()
print("Categorical Variables:")
print(cat_cols)
print("Numerical Variables:")
print(num_cols)
```

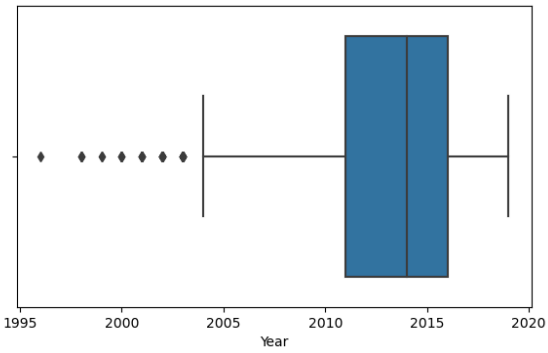
```
Categorical Variables:
Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type',
       'Mileage', 'Engine', 'Power', 'New_Price'],
      dtype='object')
Numerical Variables:
['Year', 'Kilometers_Driven', 'Seats', 'Price']
```

Univariate Analysis

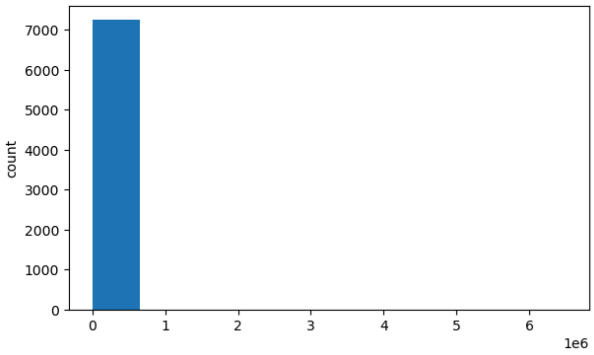
In [20]:
```
#histogram and box plot is used to show the pattern of the variables, as some variables have skewn
for col in num_cols:
    print(col)
    print('Skew :', round(data[col].skew(), 2))
    plt.figure(figsize = (15, 4))
    plt.subplot(1, 2, 1)
    data[col].hist(grid=False)
    plt.ylabel('count')
    plt.subplot(1, 2, 2)
    sns.boxplot(x=data[col])
    plt.show()
```

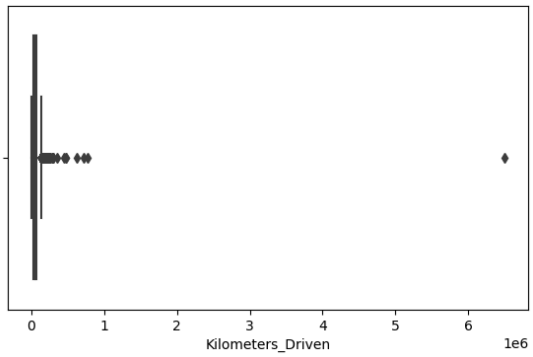
Year
Skew : -0.84



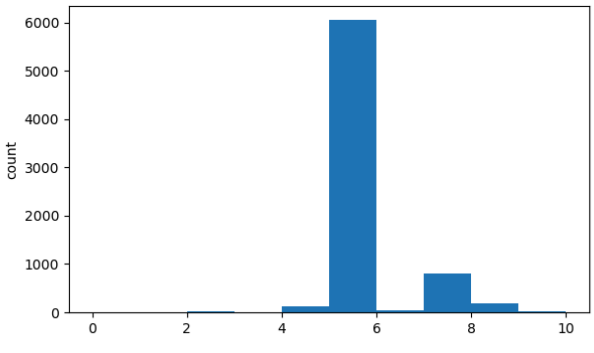


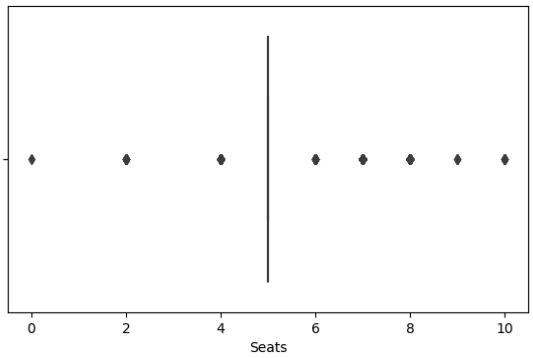
Kilometers_Driven
Skew : 61.58



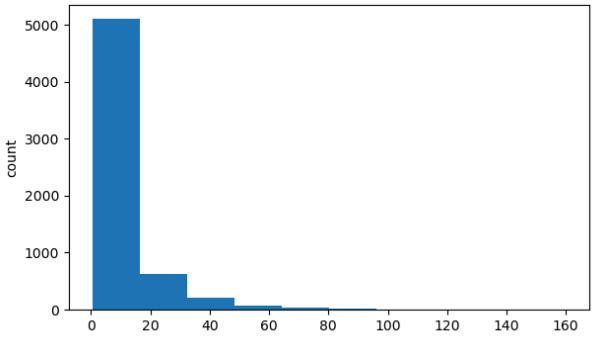


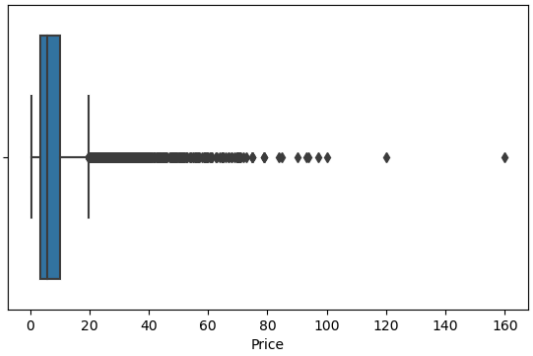
Seats
Skew : 1.9



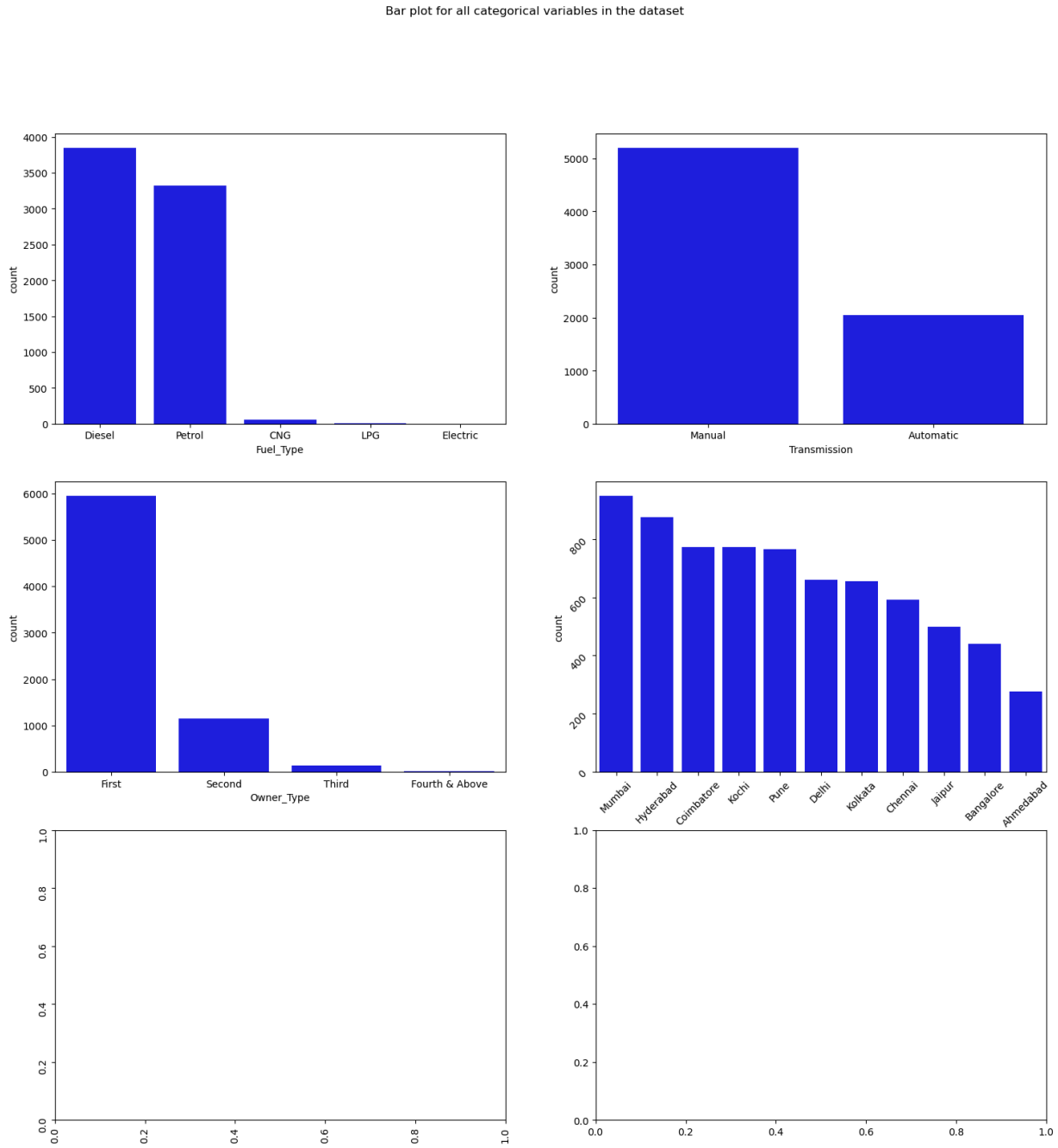


Price
Skew : 3.34


count
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
120
140
160


0
20
40
60
80
100
120
140
160
Price

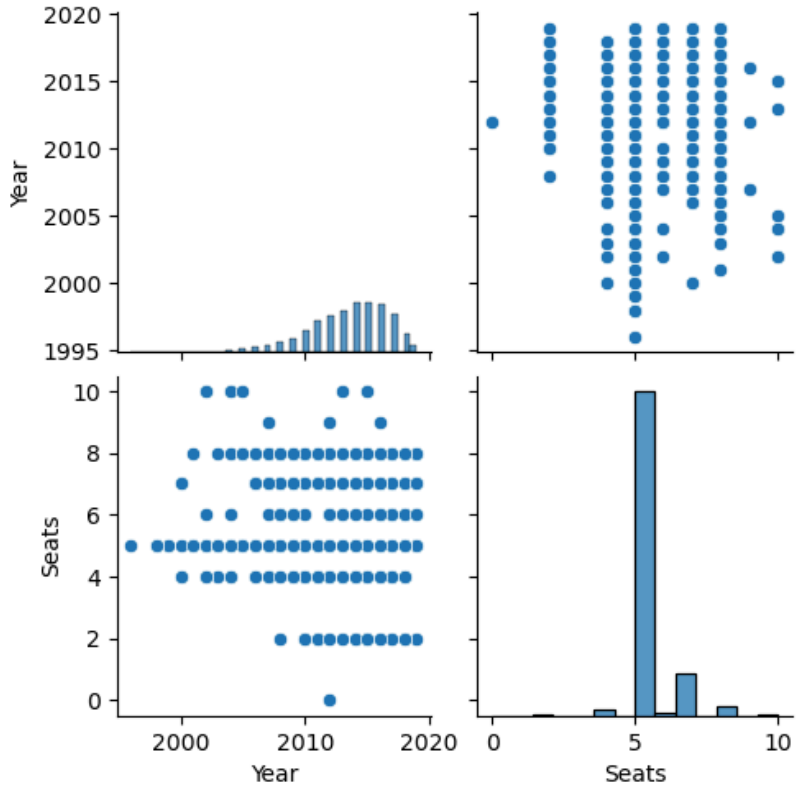
In [27]:
```
#categorical variables are being visualized using a count plot.
#Categorical variables provide the pattern of factors influencing car price

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

axes[1][1].tick_params(labelrotation=45);
axes[2][0].tick_params(labelrotation=90);
```



Bivariate Analysis

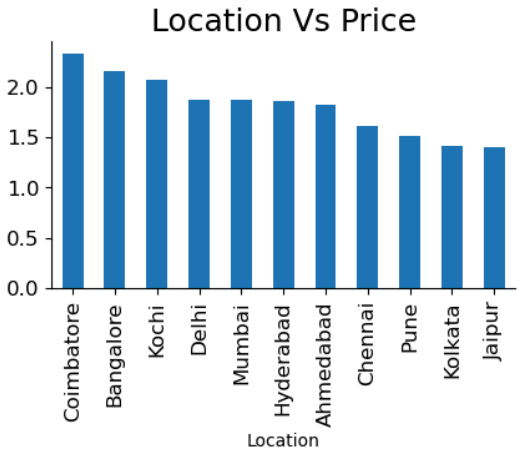
In [28]:
```
# pair plot has been used to show the relationship between two Categorical variables.
plt.figure(figsize=(13,17))
sns.pairplot(data=data.drop(['Kilometers_Driven','Price'],axis=1))
plt.show()
```

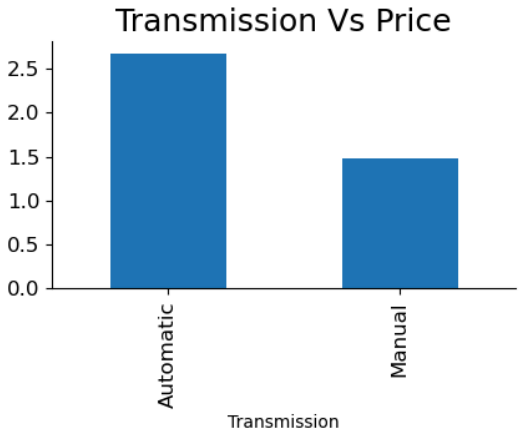
```
<Figure size 1300x1700 with 0 Axes>
```

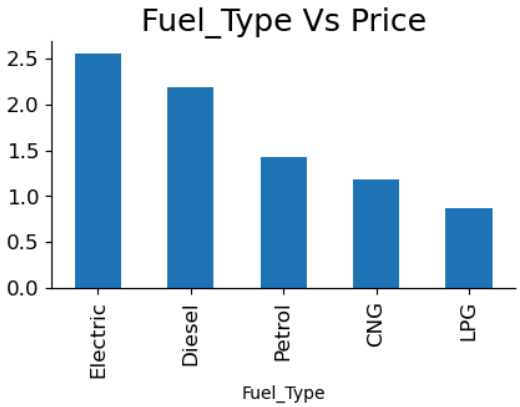


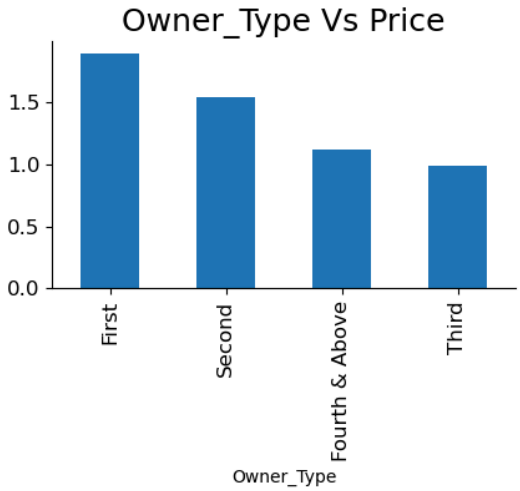
In [32]:

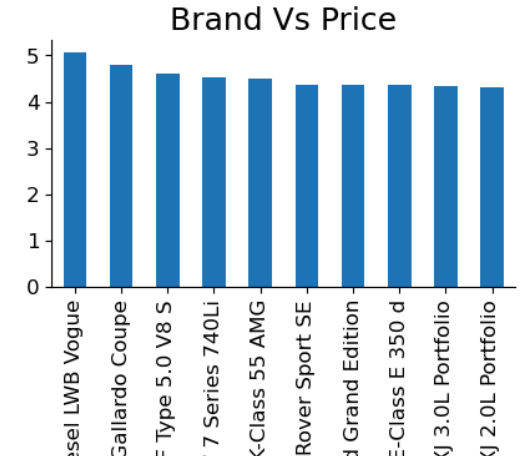
```
# Preprocessing: Assuming 'Price' is the column for price, let's convert it to logarithmic scale
data['Price_log'] = data['Price'].apply(lambda x: np.log(x))

# Plotting
fig, axarr = plt.subplots(4, 2, figsize=(12, 18))

data.groupby('Location')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[0][0],
axarr[0][0].set_title("Location Vs Price", fontsize=18)

data.groupby('Transmission')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[0]
axarr[0][1].set_title("Transmission Vs Price", fontsize=18)

data.groupby('Fuel_Type')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[1][0]
axarr[1][0].set_title("Fuel_Type Vs Price", fontsize=18)

data.groupby('Owner_Type')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[1][1
axarr[1][1].set_title("Owner_Type Vs Price", fontsize=18)

data.groupby('Name')['Price_log'].mean().sort_values(ascending=False).head(10).plot.bar(ax=axarr[2
axarr[2][0].set_title("Brand Vs Price", fontsize=18)

data.groupby('Engine')['Price_log'].mean().sort_values(ascending=False).head(10).plot.bar(ax=axarr
axarr[2][1].set_title("Model Vs Price", fontsize=18)

data.groupby('Seats')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[3][0], fo
axarr[3][0].set_title("Seats Vs Price", fontsize=18)

data.groupby('Year')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[3][1], fon
axarr[3][1].set_title("Year Vs Price", fontsize=18)

plt.subplots_adjust(hspace=1.0)
plt.subplots_adjust(wspace=.5)
sns.despine()

plt.show()
```

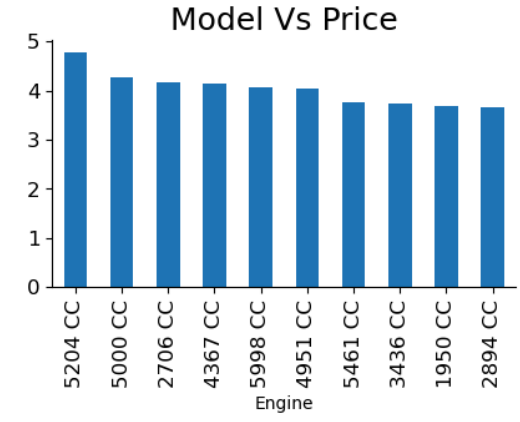


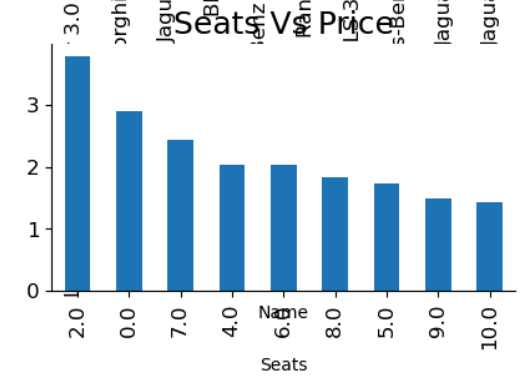


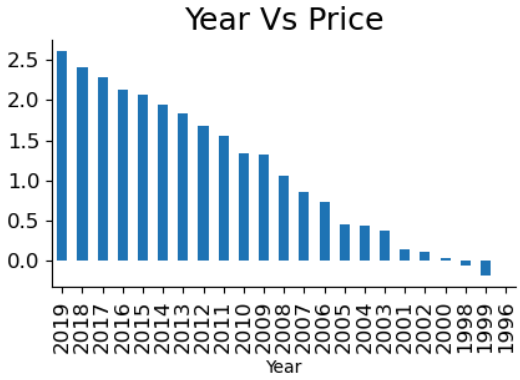










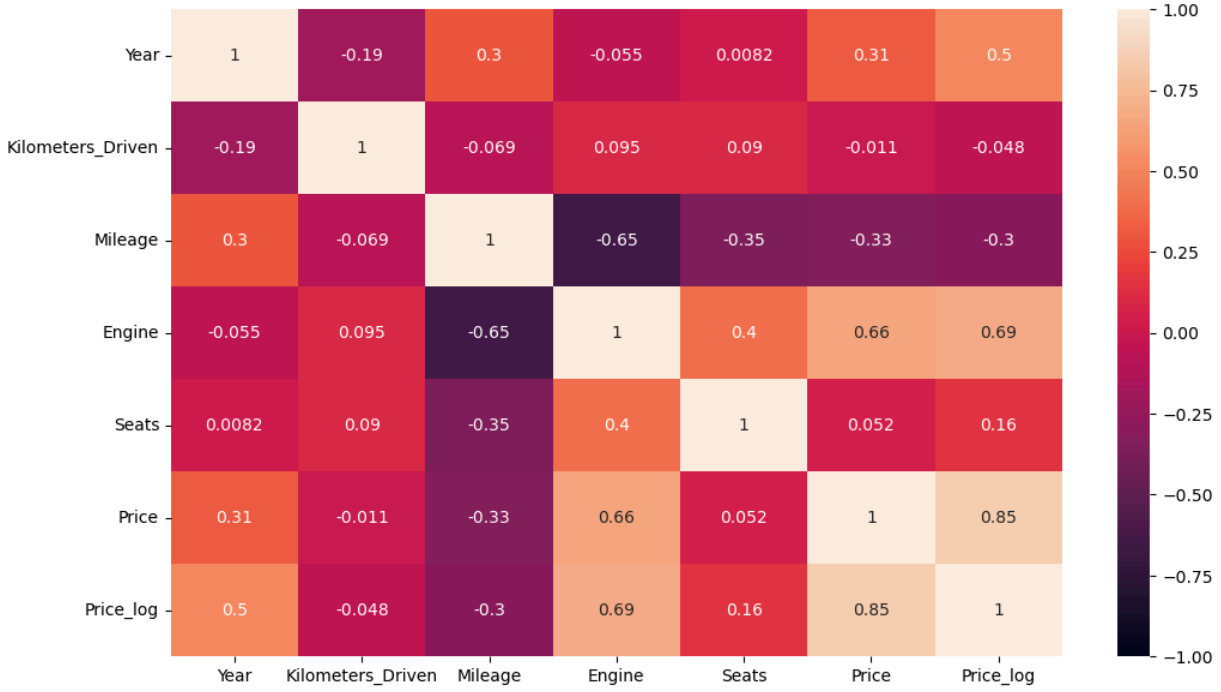


Multivariate Analysis

In [55]:
```
data.loc[data["Mileage"]==0.0,'Mileage']=np.nan
data.Mileage.isnull().sum()
```

Out[55]:
```
83
```

In [65]:
```
plt.figure(figsize=(12, 7))
sns.heatmap(data.drop(['Name','Location','Fuel_Type','Transmission','Owner_Type','Power','New_Pric
plt.show()
```



In [ ]: