Ques-1: Perform EDA

In [1]:
```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```python
df=pd.read_csv('Cars.csv')
```

In [3]:
```python
df
```

Out[3]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Colour	Seats	No. of Doors	New_Price	Price
0	Mahindra Scorpio	Pune	2012.0	99000.0	Diesel	Manual	Third	12.05 kmpl	2179 CC	120 bhp	Black/Silver	8.0	5.0	NaN	6.00
1	Maruti Baleno	Kochi	2018.0	18678.0	Petrol	Manual	First	21.1 kmpl	998 CC	100 bhp	Others	5.0	4.0	NaN	8.32
2	Mahindra Xylo	Bangalore	2013.0	197000.0	Diesel	Manual	First	11.68 kmpl	2498 CC	112 bhp	White	7.0	5.0	NaN	4.00
3	Hyundai Grand	Delhi	2014.0	45000.0	Diesel	Manual	First	24.0 kmpl	1120 CC	70 bhp	White	5.0	4.0	NaN	3.49
4	Toyota Innova	Delhi	2011.0	65000.0	Diesel	Manual	First	12.8 kmpl	2494 CC	102 bhp	Others	8.0	5.0	NaN	6.40
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5956	Honda Civic	Pune	2011.0	47000.0	Petrol	Automatic	Second	13.9 kmpl	1799 CC	130.3 bhp	Others	5.0	4.0	NaN	4.50
5957	Hyundai i20	Delhi	2013.0	63777.0	Petrol	Manual	First	18.5 kmpl	1197 CC	82.9 bhp	Black/Silver	5.0	4.0	NaN	3.67
5958	Maruti Swift	Coimbatore	2016.0	37806.0	Petrol	Manual	First	20.4 kmpl	1197 CC	81.80 bhp	Black/Silver	5.0	4.0	NaN	5.20
5959	Mercedes-Benz SLK-Class	Coimbatore	2016.0	22732.0	Petrol	Automatic	First	18.1 kmpl	3498 CC	306 bhp	Black/Silver	2.0	2.0	NaN	55.54
5960	Hyundai i10	Kolkata	2016.0	7000.0	Petrol	Manual	First	20.36 kmpl	1197 CC	78.9 bhp	White	5.0	4.0	NaN	3.60

5961 rows × 15 columns

In [4]:
```python
df.head()
```

Out[4]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Colour	Seats	No. of Doors	New_Price	Price
0	Mahindra Scorpio	Pune	2012.0	99000.0	Diesel	Manual	Third	12.05 kmpl	2179 CC	120 bhp	Black/Silver	8.0	5.0	NaN	6.00
1	Maruti Baleno	Kochi	2018.0	18678.0	Petrol	Manual	First	21.1 kmpl	998 CC	100 bhp	Others	5.0	4.0	NaN	8.32
2	Mahindra Xylo	Bangalore	2013.0	197000.0	Diesel	Manual	First	11.68 kmpl	2498 CC	112 bhp	White	7.0	5.0	NaN	4.00
3	Hyundai Grand	Delhi	2014.0	45000.0	Diesel	Manual	First	24.0 kmpl	1120 CC	70 bhp	White	5.0	4.0	NaN	3.49
4	Toyota Innova	Delhi	2011.0	65000.0	Diesel	Manual	First	12.8 kmpl	2494 CC	102 bhp	Others	8.0	5.0	NaN	6.40

In [5]:
```python
df.tail()
```

Out[5]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Colour	Seats	No. of Doors	New_Price	Price
5956	Honda Civic	Pune	2011.0	47000.0	Petrol	Automatic	Second	13.9 kmpl	1799 CC	130.3 bhp	Others	5.0	4.0	NaN	4.50
5957	Hyundai i20	Delhi	2013.0	63777.0	Petrol	Manual	First	18.5 kmpl	1197 CC	82.9 bhp	Black/Silver	5.0	4.0	NaN	3.67
5958	Maruti Swift	Coimbatore	2016.0	37806.0	Petrol	Manual	First	20.4 kmpl	1197 CC	81.80 bhp	Black/Silver	5.0	4.0	NaN	5.20
5959	Mercedes-Benz SLK-Class	Coimbatore	2016.0	22732.0	Petrol	Automatic	First	18.1 kmpl	3498 CC	306 bhp	Black/Silver	2.0	2.0	NaN	55.54
5960	Hyundai i10	Kolkata	2016.0	7000.0	Petrol	Manual	First	20.36 kmpl	1197 CC	78.9 bhp	White	5.0	4.0	NaN	3.60

In [6]:
```python
for col in df.columns:
    print(col)
```

```
Name
Location
Year
Kilometers_Driven
Fuel_Type
Transmission
Owner_Type
Mileage
Engine
Power
Colour
Seats
No. of Doors
New_Price
Price
```

In [7]:
```python
df.shape
```

Out[7]:
```
(5961, 15)
```

In [8]:
```python
df['Location'].value_counts()
```

Out[8]:
```
Mumbai        781
Hyderabad     739
Kochi         646
Coimbatore    630
Pune          611
Delhi         549
Kolkata       526
Chennai       489
Jaipur        406
Bangalore     351
Ahmedabad     222
Name: Location, dtype: int64
```

In [9]:
```python
df['Year'].value_counts()
```

Out[9]:
```
2014.0    793
2016.0    740
2015.0    736
2013.0    642
2017.0    586
2012.0    573
2011.0    461
2010.0    338
2018.0    298
2009.0    196
2008.0    170
2007.0    123
2019.0    101
2006.0     75
2005.0     55
2004.0     28
2002.0     14
2003.0     13
2001.0      7
2000.0      4
1998.0      4
1999.0      2
Name: Year, dtype: int64
```

In [10]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5961 entries, 0 to 5960
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               5961 non-null   object 
 1   Location           5950 non-null   object 
 2   Year               5959 non-null   float64
 3   Kilometers_Driven  5953 non-null   float64
 4   Fuel_Type          5961 non-null   object 
 5   Transmission       5934 non-null   object 
 6   Owner_Type         5946 non-null   object 
 7   Mileage            5959 non-null   object 
 8   Engine             5944 non-null   object 
 9   Power              5929 non-null   object 
 10  Colour             5950 non-null   object 
 11  Seats              5956 non-null   float64
 12  No. of Doors       5960 non-null   float64
 13  New_Price          824 non-null    object 
 14  Price              5961 non-null   float64
dtypes: float64(5), object(10)
memory usage: 698.7+ KB
```

In [11]:
```python
df.nunique()
```

Out[11]:
```
Name                  212
Location               11
Year                   22
Kilometers_Driven    3068
Fuel_Type               5
Transmission            2
Owner_Type              4
Mileage               439
Engine                143
Power                 369
Colour                  3
Seats                   8
No. of Doors            3
New_Price             540
Price                1369
dtype: int64
```

In [12]:
```python
df.isnull().sum()
```

Out[12]:
```
Name                    0
Location               11
Year                    2
Kilometers_Driven       8
Fuel_Type               0
Transmission           27
Owner_Type             15
Mileage                 2
Engine                 17
Power                  32
Colour                 11
Seats                   5
No. of Doors            1
New_Price            5137
Price                   0
dtype: int64
```

In [13]:
```python
(df.isnull().sum()/(len(df)))*100
```

Out[13]:
```
Name                  0.000000
Location              0.184533
Year                  0.033551
Kilometers_Driven     0.134206
Fuel_Type             0.000000
Transmission          0.452944
Owner_Type            0.251636
Mileage               0.033551
Engine                0.285187
Power                 0.536823
Colour                0.184533
Seats                 0.083879
No. of Doors          0.016776
New_Price            86.176816
Price                 0.000000
dtype: float64
```

In [14]:
```python
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']]
```

Out[14]:

	Name	Brand	Model
0	Mahindra Scorpio	Mahindra	NaN
1	Maruti Baleno	Maruti	NaN
2	Mahindra Xylo	Mahindra	NaN
3	Hyundai Grand	Hyundai	NaN
4	Toyota Innova	Toyota	NaN
...	...	...	...
5956	Honda Civic	Honda	NaN
5957	Hyundai i20	Hyundai	NaN
5958	Maruti Swift	Maruti	NaN
5959	Mercedes-Benz SLK-Class	Mercedes-Benz	NaN
5960	Hyundai i10	Hyundai	NaN

5961 rows × 3 columns

In [15]:
```python
print(df.Brand.unique())
print(df.Brand.nunique())
```

```
['Mahindra' 'Maruti' 'Hyundai' 'Toyota' 'Honda' 'Chevrolet' 'Audi' 'Skoda'
 'Renault' 'Land' 'BMW' 'ISUZU' 'Jaguar' 'Mercedes-Benz' 'Volkswagen'
 'Tata' 'Mitsubishi' 'Ford' 'Nissan' 'Volvo' 'Fiat' 'Porsche' 'Mini'
 'Datsun' 'Jeep' 'Force' 'Isuzu' 'Smart' 'Lamborghini' 'Bentley']
30
```

In [16]:
```python
searchfor = ['Isuzu','ISUZU','Mini','Land']
df[df.Brand.str.contains('|'.join(searchfor))].head(5)
```

Out[16]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Colour	Seats	No. of Doors	New_Price	Price	Brand	Model
19	Land Rover Freelander	Hyderabad	2012.0	139000.0	Diesel	Automatic	First	0.0 kmpl	2179 CC	115 bhp	Black/Silver	5.0	4.0	NaN	16.75	Land	RoverFreelander
28	ISUZU D-MAX	Jaipur	2017.0	25000.0	Diesel	Manual	First	12.4 kmpl	2499 CC	134 bhp	Others	5.0	4.0	NaN	8.00	ISUZU	NaN
97	Land Rover Range	Delhi	2015.0	37000.0	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	White	5.0	4.0	NaN	36.00	Land	RoverRange
108	Land Rover Range	Coimbatore	2017.0	49275.0	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	Others	5.0	4.0	NaN	45.64	Land	RoverRange
113	Land Rover Range	Coimbatore	2013.0	75995.0	Diesel	Automatic	Second	11.49 kmpl	4367 CC	335.3 bhp	White	5.0	4.0	NaN	65.81	Land	RoverRange

In [17]:
```python
df['Brand'].replace({'ISUZU':'Isuzu','Mini':'Mini Cooper','Land':'Land Rover'},inplace=True)
```

In [18]:
```python
df.describe().T
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
Year	5959.0	2013.389159	3.243051	1998.00	2011.5	2014.00	2016.0	2019.0
Kilometers_Driven	5953.0	58711.100118	91712.207172	171.00	33931.0	53000.00	73000.0	6500000.0
Seats	5956.0	5.269140	0.789048	2.00	5.0	5.00	5.0	10.0
No. of Doors	5960.0	4.114933	0.344757	2.00	4.0	4.00	4.0	5.0
Price	5961.0	9.528103	11.214382	0.44	3.5	5.66	10.0	160.0

In [19]:
```python
df.describe(include='all').T
```

Out[19]:

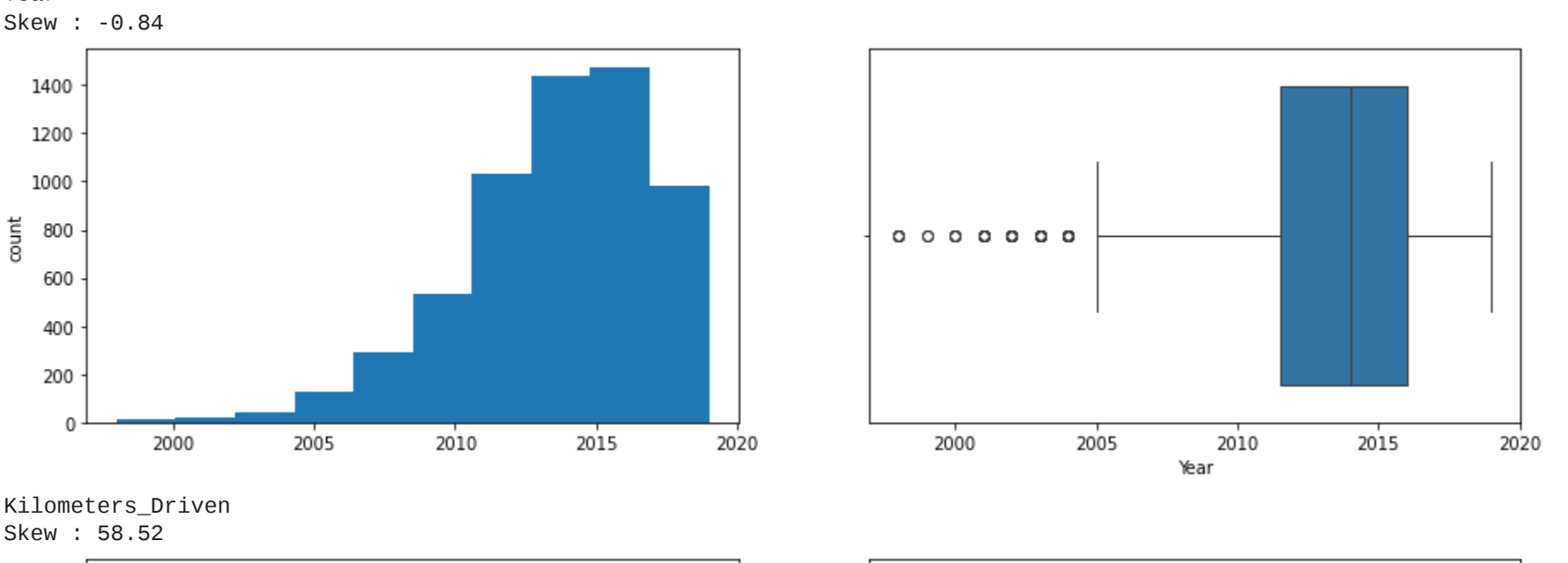
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Name	5961	212	Maruti Swift	343	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Location	5950	11	Mumbai	781	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Year	5959.0	NaN	NaN	NaN	2013.389159	3.243051	1998.0	2011.5	2014.0	2016.0	2019.0
Kilometers_Driven	5953.0	NaN	NaN	NaN	58711.100118	91712.207172	171.0	33931.0	53000.0	73000.0	6500000.0
Fuel_Type	5961	5	Diesel	3188	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Transmission	5934	2	Manual	4225	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Owner_Type	5946	4	First	4875	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Mileage	5959	439	18.9 kmpl	172	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Engine	5944	143	1197 CC	606	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Power	5929	369	74 bhp	233	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Colour	5950	3	White	2115	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Seats	5956.0	NaN	NaN	NaN	5.26914	0.789048	2.0	5.0	5.0	5.0	10.0
No. of Doors	5960.0	NaN	NaN	NaN	4.114933	0.344757	2.0	4.0	4.0	4.0	5.0
New_Price	824	540	4.78 Lakh	6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	5961.0	NaN	NaN	NaN	9.528103	11.214382	0.44	3.5	5.66	10.0	160.0
Brand	5961	29	Maruti	1189	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Model	57	3	RoverRange	28	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [20]:
```python
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)
```

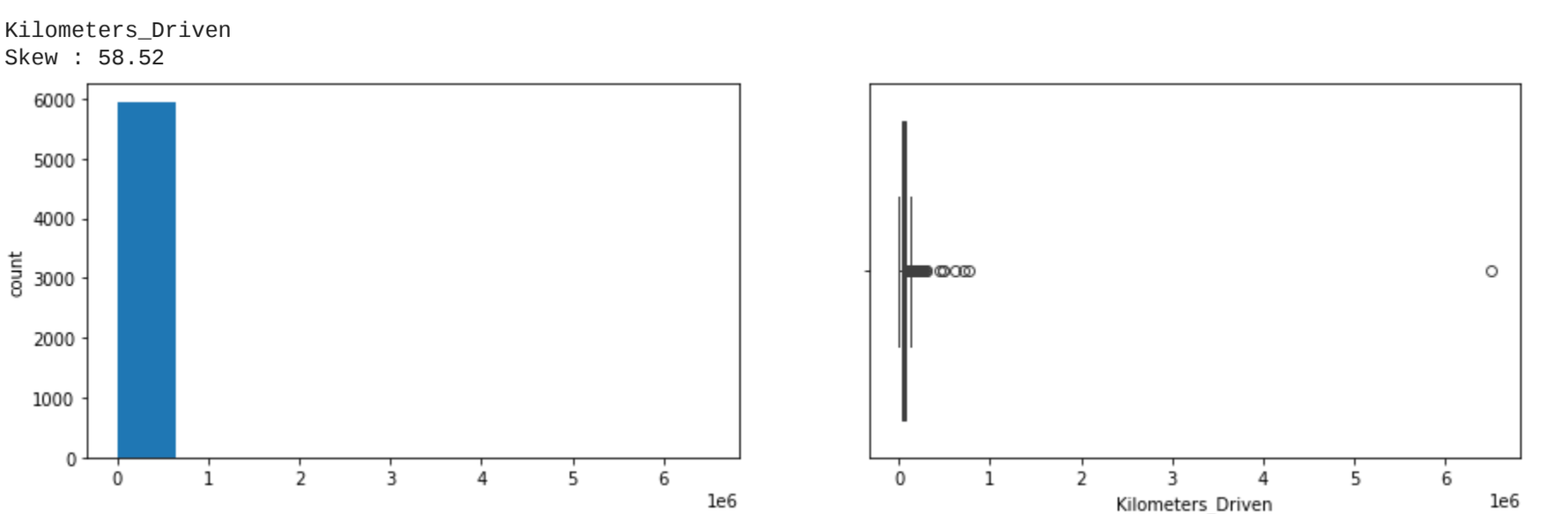
```
Categorical Variables:
Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type',
       'Mileage', 'Engine', 'Power', 'Colour', 'New_Price', 'Brand', 'Model'],
      dtype='object')
Numerical Variables:
['Year', 'Kilometers_Driven', 'Seats', 'No. of Doors', 'Price']
```

In [21]:
```python
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()
```

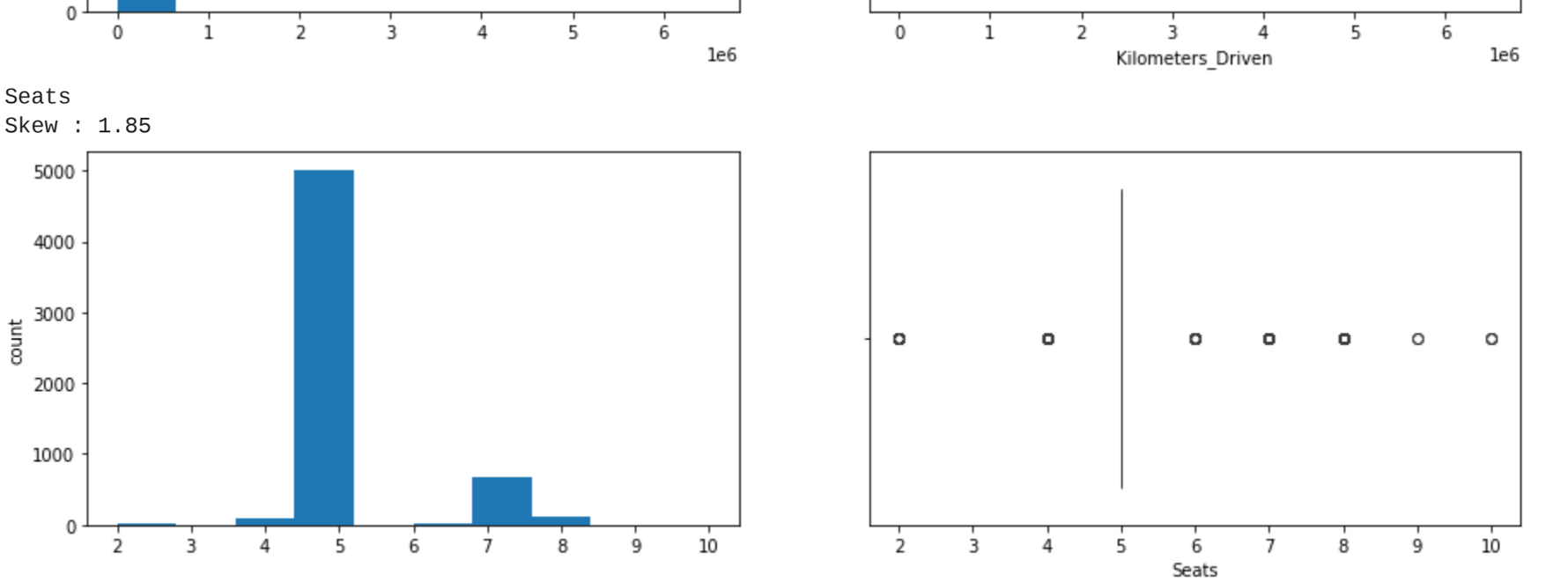
```
Year
Skew : -0.84
```



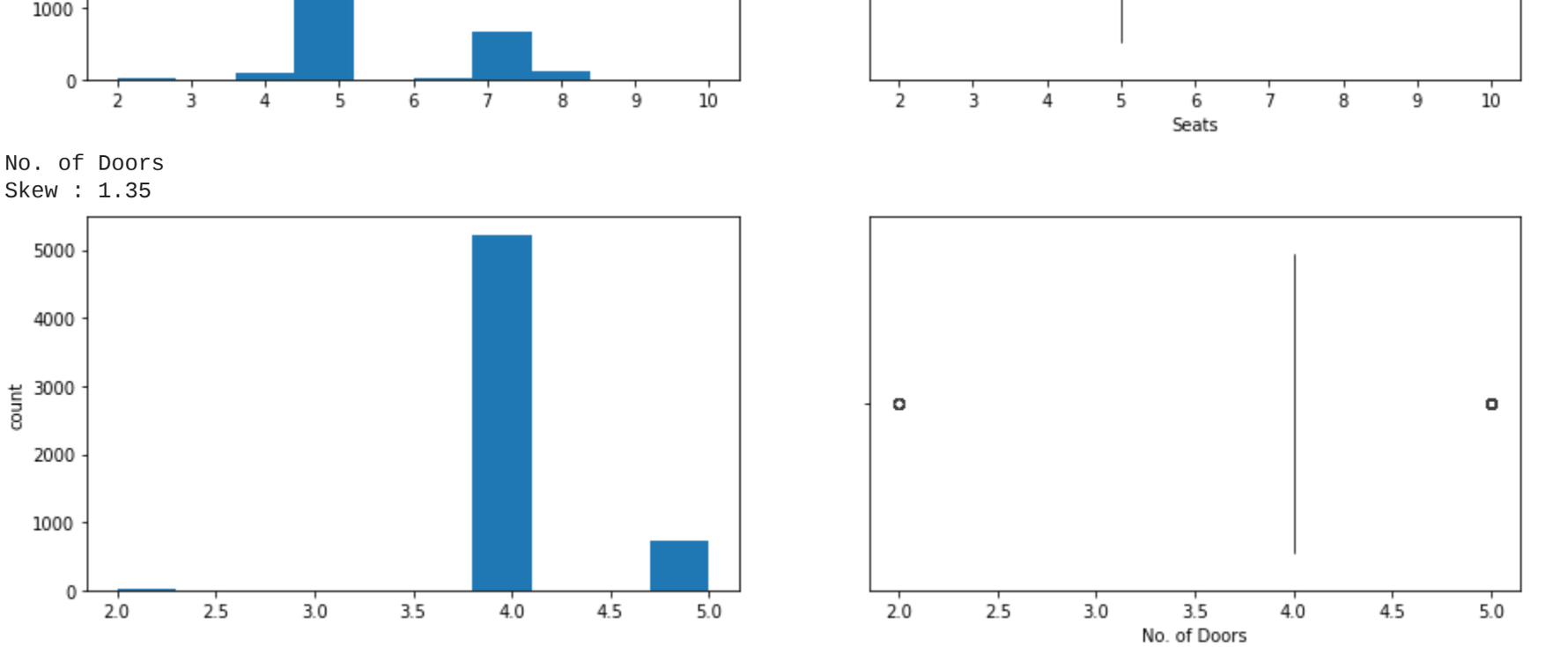

```
Kilometers_Driven
Skew : 58.52
```



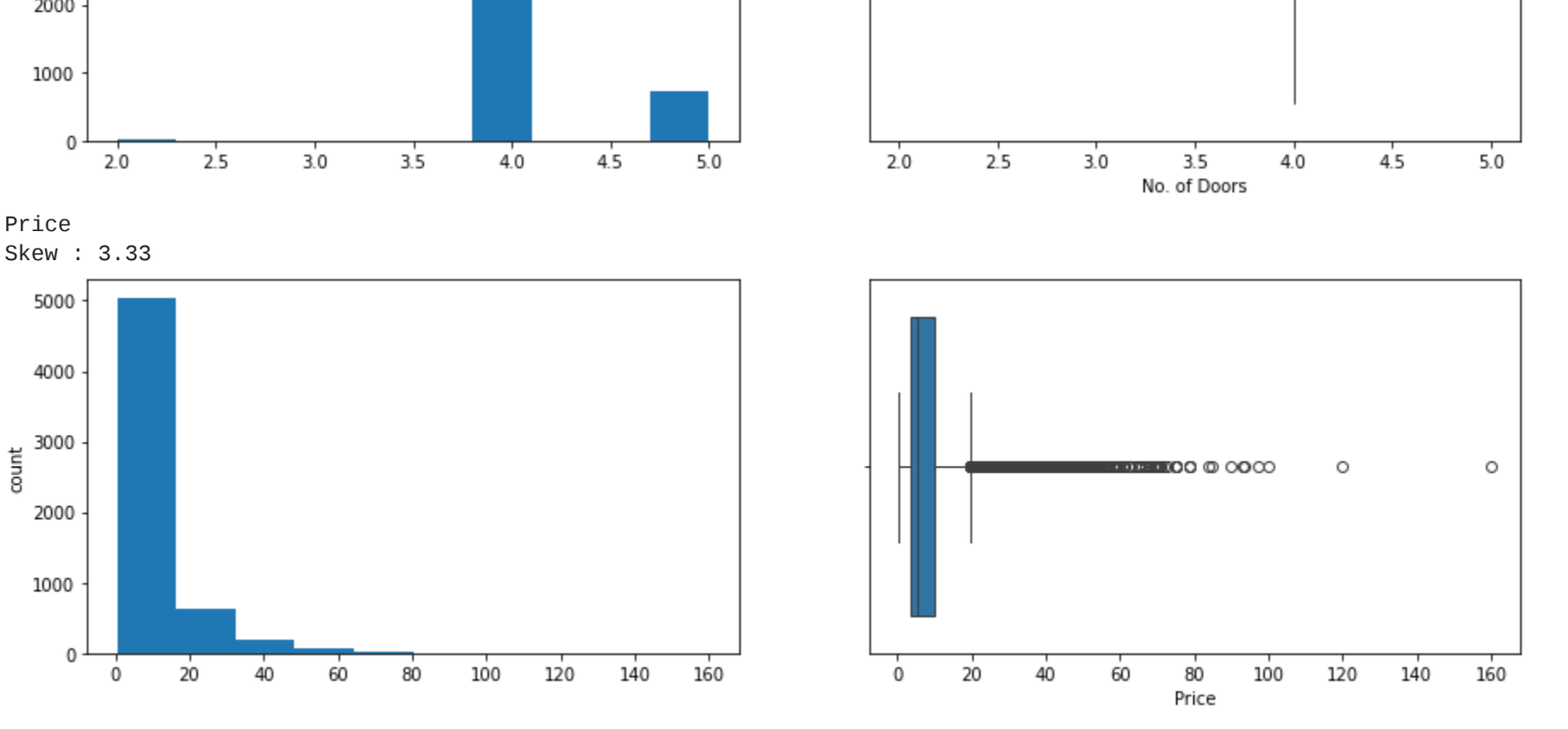

```
Seats
Skew : 1.85
```



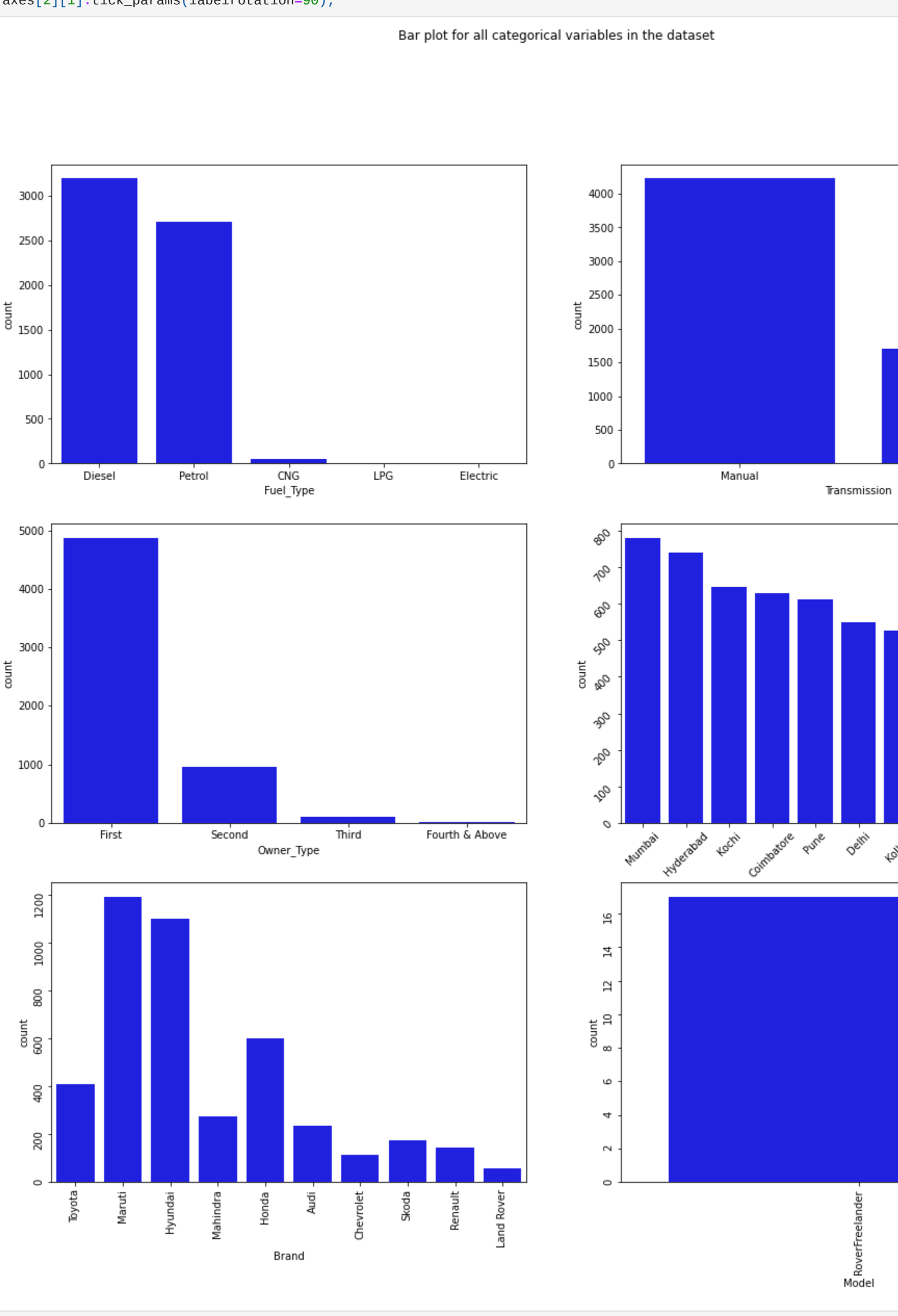

```
No. of Doors
Skew : 1.35
```




```
Price
Skew : 3.33
```



In [22]:
```python
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);
```



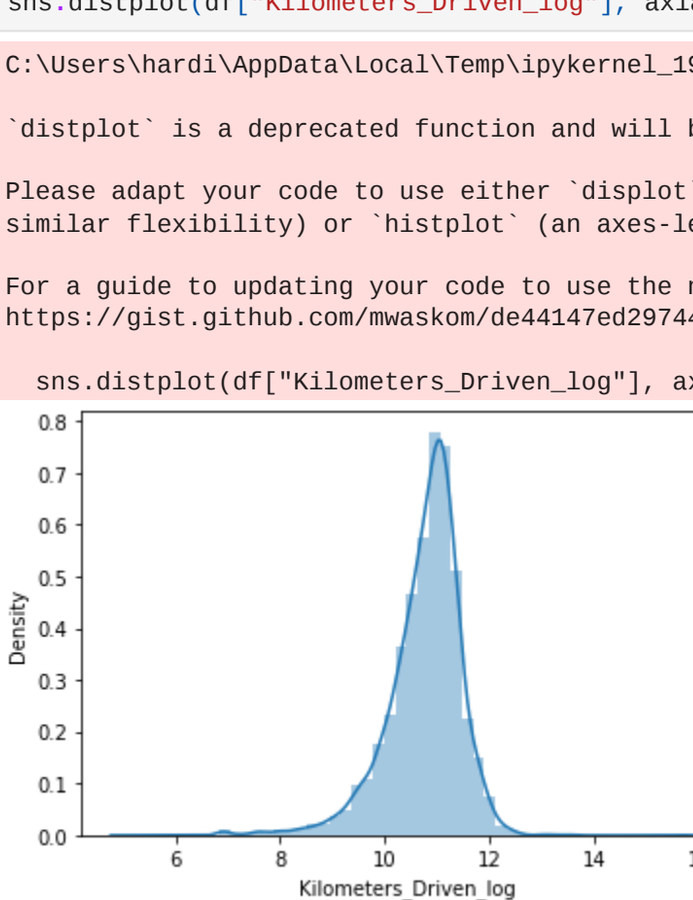
In [23]:
```python
# Function for log transformation of the column
def log_transform(data,col):
    for colname in col:
        if (df[colname] == 1.0).all():
            df[colname + '_log'] = np.log(df[colname]+1)
        else:
            df[colname + '_log'] = np.log(df[colname])
    df.info()
```

In [24]:
```python
log_transform(df,['Kilometers_Driven','Price'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5961 entries, 0 to 5960
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Name                   5961 non-null   object 
 1   Location               5950 non-null   object 
 2   Year                   5959 non-null   float64
 3   Kilometers_Driven      5953 non-null   float64
 4   Fuel_Type              5961 non-null   object 
 5   Transmission           5934 non-null   object 
 6   Owner_Type             5946 non-null   object 
 7   Mileage                5959 non-null   object 
 8   Engine                 5944 non-null   object 
 9   Power                  5929 non-null   object 
 10  Colour                 5950 non-null   object 
 11  Seats                  5956 non-null   float64
 12  No. of Doors           5960 non-null   float64
 13  New_Price              824 non-null    object 
 14  Price                  5961 non-null   float64
 15  Brand                  5961 non-null   object 
 16  Model                  57 non-null     object 
 17  Kilometers_Driven_log  5953 non-null   float64
 18  Price_log              5961 non-null   float64
dtypes: float64(7), object(12)
memory usage: 885.0+ KB
```

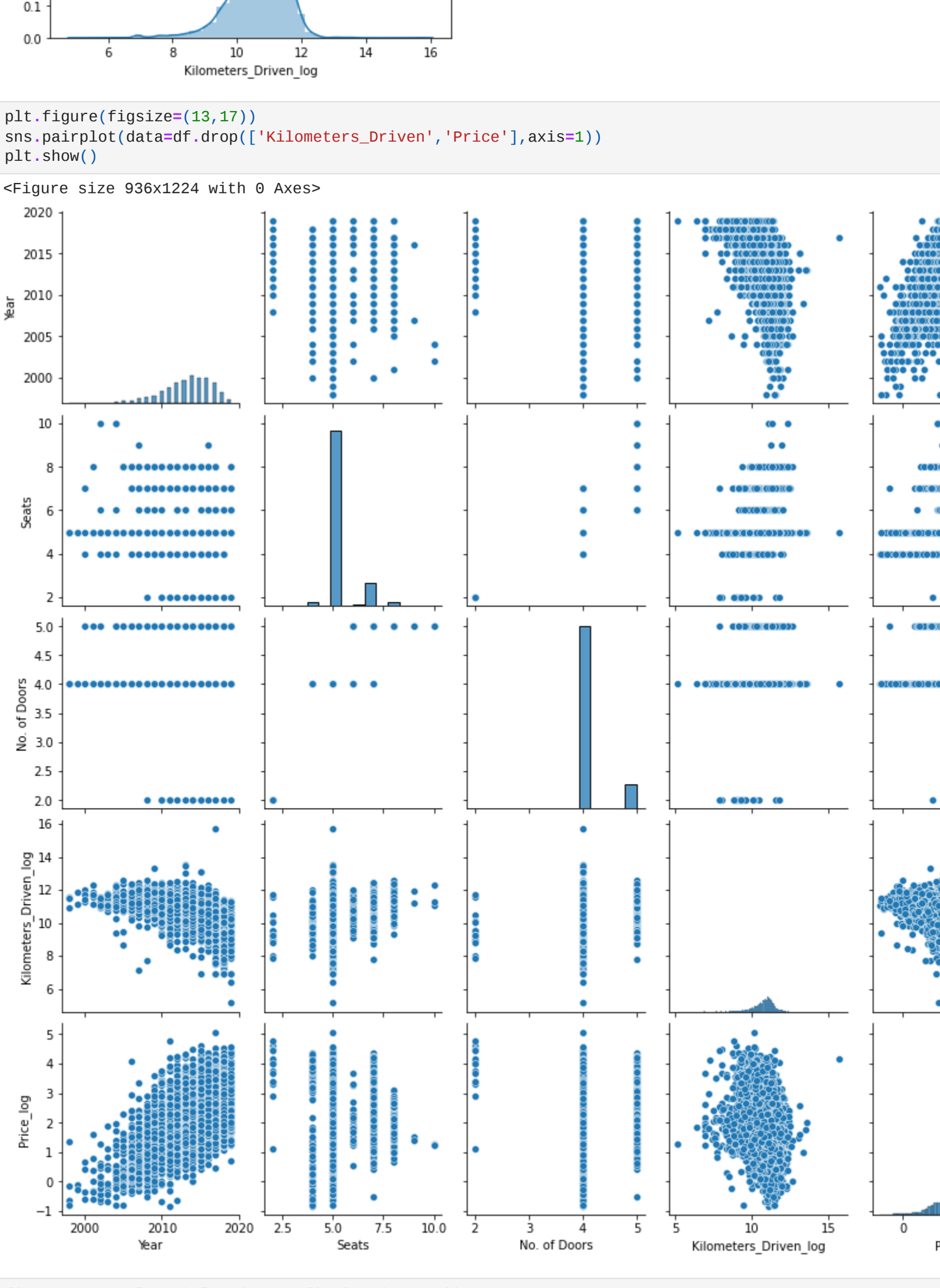
In [25]:
```python
#Log transformation of the feature 'Kilometers_Driven'
sns.distplot(df["Kilometers_Driven_log"], axlabel="Kilometers_Driven_log");
```

```
C:\Users\hardi\AppData\Local\Temp\ipykernel_19180\681597022.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Kilometers_Driven_log"], axlabel="Kilometers_Driven_log");
```

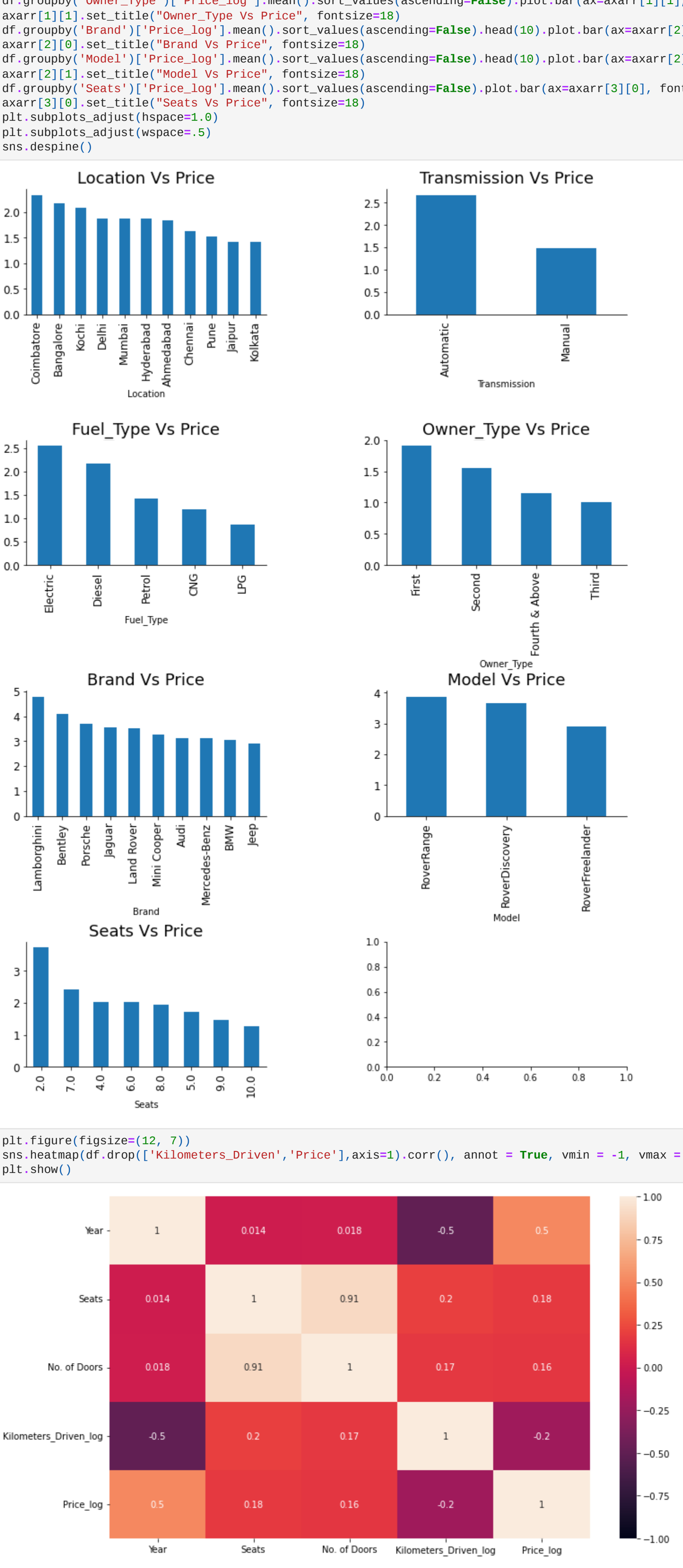


In [26]:
```python
plt.figure(figsize=(13,17))
sns.pairplot(data=df.drop(['Kilometers_Driven','Price'],axis=1))
plt.show()
```

```
<Figure size 936x1224 with 0 Axes>
```



In [27]:
```python
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)
plt.subplots_adjust(hspace=1.0)
plt.subplots_adjust(wspace=.5)
sns.despine()
```



In [28]:
```python
plt.figure(figsize=(12, 7))
sns.heatmap(df.drop(['Kilometers_Driven','Price'],axis=1).corr(), annot = True, vmin = -1, vmax = 1)
plt.show()
```

In [29]:
```python
df.loc[df["Mileage"]==0.0,'Mileage']=np.nan
df.Mileage.isnull().sum()
```

Out[29]:
```
2
```

In [30]:
```python
df['Brand'].value_counts()[0:10]
```

Out[30]:
```
Maruti           1189
Hyundai          1100
Honda             601
Toyota            410
Mercedes-Benz     318
Volkswagen        315
Ford              298
Mahindra          272
BMW               267
Audi              236
Name: Brand, dtype: int64
```

In [31]:
```python
df['Model'].value_counts()[0:10]
```

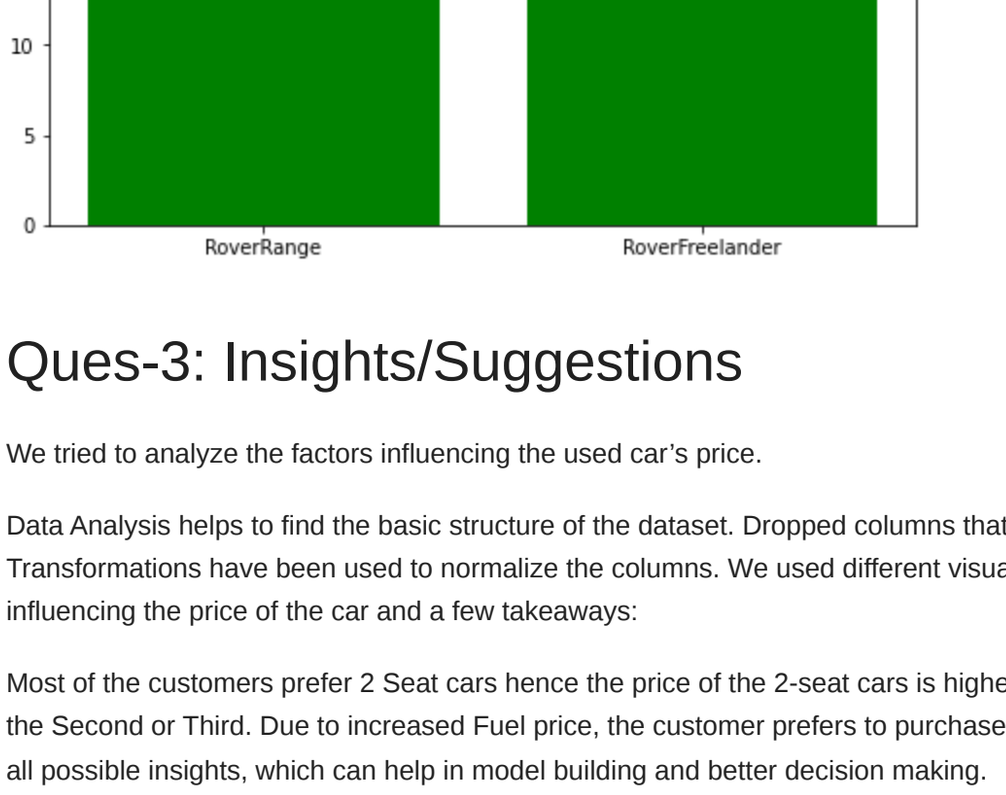
Out[31]:
```
RoverRange         28
RoverFreelander    17
RoverDiscovery     12
Name: Model, dtype: int64
```

In [32]:
```python
list(df['Model'].value_counts()[0:2].keys())
```

Out[32]:
```
['RoverRange', 'RoverFreelander']
```

Ques-2: Comparison two model

In [33]:
```python
plt.figure(figsize=(8,5))
plt.bar(list(df['Model'].value_counts()[0:2].keys()),list(df['Model'].value_counts()[0:2]),color='g')
plt.show()
```



Ques-3: Insights/Suggestions

We tried to analyze the factors influencing the used car's price.

Data Analysis helps to find the basic structure of the dataset. Dropped columns that are not adding value to our analysis. Performed Feature Engineering by adding some columns which contribute to our analysis. Data Transformations have been used to normalize the columns. We used different visualizations for EDA like Univariate, Bi-Variate, and Multivariate Analysis. Through EDA, we got useful insights, and below are the factors influencing the price of the car and a few takeaways:

Most of the customers prefer 2 Seat cars hence the price of the 2-seat cars is higher than other cars. The price of the car decreases as the Age of the car increases. Customers prefer to purchase the First owner rather than the Second or Third. Due to increased Fuel price, the customer prefers to purchase an Electric vehicle. Automatic Transmission is easier than Manual. This way, we perform EDA on the datasets to explore the data and extract all possible insights, which can help in model building and better decision making.

However, this was only an overview of how EDA works; you can go deeper into it and attempt the stages on larger datasets.

If the EDA process is clear and precise, our model will work better and gives higher accuracy!

In [ ]: