In [2]:
```
import itertools
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import date

# columns=['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleratic

sns.set() # Default settings for Seaborn
```

Importing Data and EDA

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

In [4]:
```
df.head(5)
```

Out[4]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second

In [5]:
```
df
```

Out[5]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	Fi
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	Fi
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	Fi
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	Fi
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Seco
...	...	...	...	...	...	...	...	
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	Fi
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	Fi
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	Fi
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Th
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	Fi

7253 rows × 14 columns

In [6]:
```
df.tail()
```

Out[6]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Typ
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	Fir
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	Fir
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	Fir
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Thi
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	Fir

In [7]:
```
# Analysis on checking for duplicates
df.nunique()
```

Out[7]:
```
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
```

In [8]:
```
# Checking for null values
df.isnull().sum()
```

Out[8]:
```
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 [9]:
```
# The above calculations can also be done percentage wise
(df.isnull().sum()/len(df)) * 100
```

Out[9]:
```
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 Cleaning / Reduction

In [10]:
```
# In the dataset, we can see that we have a column called "S.No", which is not very
df.drop(columns = ['S.No.'], axis = 1)
```

Out[10]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mil
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	k
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	
...	...	...	...	...	...	...	...	
7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	First	
7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	First	
7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	First	
7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Third	
7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	First	

7253 rows × 13 columns

Feature Engineering

In [11]:
```
# If we take a close look on data, we will see that the data has a column called "Y
# But, if we have a column called 'car_age', that will be quite useful.

df['Car_Age'] = date.today().year - df['Year']
```

In [12]:
```
df.head(5)
```

Out[12]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second

In [13]:
```
# The column 'Name' is not very useful, but if we extract 'Model Name' and 'Brand N

df['Brand_Name'] = df.Name.str.split().str.get(0)
df['Model_Name'] = df.Name.str.split().str.get(1) + df.Name.str.split().str.get(2)
```

In [14]:
```
df.head(5)
```

Out[14]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second

In [15]:
```
df[['Location', 'Owner_Type', 'Engine', 'Price']]
```

Out[15]:

	Location	Owner_Type	Engine	Price
0	Mumbai	First	998 CC	1.75
1	Pune	First	1582 CC	12.50
2	Chennai	First	1199 CC	4.50
3	Chennai	First	1248 CC	6.00
4	Coimbatore	Second	1968 CC	17.74
...	...	...	...	...
7248	Hyderabad	First	1598 CC	NaN
7249	Mumbai	First	1197 CC	NaN
7250	Kolkata	First	1461 CC	NaN
7251	Pune	Third	1197 CC	NaN
7252	Kochi	First	2148 CC	NaN

7253 rows × 4 columns

Data cleaning and Wrangling

In [16]:
```
# In the 'Brand_Name' column, we can see that some entries have inconsistency among

print(df.Brand_Name.unique())
print(df.Brand_Name.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' 'Hindustan' 'OpelCorsa']
33
```

In [17]:
```
# Looking in the 'Brand_Name' column, we came to know that 'Isuzu', 'ISUZU', 'Mini'

search_for = ['Isuzu', 'ISUZU', 'Mini', 'Land']

df[df.Brand_Name.str.contains('|'.join(search_for))]
```

Out[17]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
13	13	Land Rover Range Rover 2.2L Pure	Delhi	2014	72000	Diesel	Automatic	F
14	14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	Secc
176	176	Mini Countryman Cooper D	Jaipur	2017	8525	Diesel	Automatic	Secc
191	191	Land Rover Range Rover 2.2L Dynamic	Coimbatore	2018	36091	Diesel	Automatic	F
228	228	Mini Cooper Convertible S	Kochi	2017	26327	Petrol	Automatic	F
...	...	...	...	...	...	...	...	
6919	6919	ISUZU D-MAX V-Cross 4X4	Jaipur	2017	290000	Diesel	Manual	F
7132	7132	Mini Clubman Cooper S	Pune	2017	2890	Petrol	Manual	F
7157	7157	Land Rover Range Rover 2.2L Pure	Hyderabad	2015	49000	Diesel	Automatic	Secc
7160	7160	Mini Cooper Countryman D	Hyderabad	2013	50000	Diesel	Automatic	F
7198	7198	Land Rover Discovery 4 TDV6 Auto Diesel	Hyderabad	2012	147202	Diesel	Automatic	F

103 rows × 17 columns

In [18]:
```
# Replacing the above anomalies

df['Brand_Name'].replace({"ISUZU" : 'Isuzu', "Mini" : "Mini Cooper", "Land" : "Land
```

In [19]:
```
df
```

Out[19]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	Fi
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	Fi
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	Fi
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	Fi
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Seco
...	...	...	...	...	...	...	...	
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	Fi
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	Fi
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	Fi
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Th
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	Fi

7253 rows × 17 columns

EDA (Exploratory Data Analysis)

In [20]:
```
# Getting a high level view of data

df.describe()
```

Out[20]:

	S.No.	Year	Kilometers_Driven	Seats	Price	Car_Age
count	7253.000000	7253.000000	7.253000e+03	7200.000000	6019.000000	7253.000000
mean	3626.000000	2013.365366	5.869906e+04	5.279722	9.479468	10.634634
std	2093.905084	3.254421	8.442772e+04	0.811660	11.187917	3.254421
min	0.000000	1996.000000	1.710000e+02	0.000000	0.440000	5.000000
25%	1813.000000	2011.000000	3.400000e+04	5.000000	3.500000	8.000000
50%	3626.000000	2014.000000	5.341600e+04	5.000000	5.640000	10.000000
75%	5439.000000	2016.000000	7.300000e+04	5.000000	9.950000	13.000000
max	7252.000000	2019.000000	6.500000e+06	10.000000	160.000000	28.000000

In [21]:
```
df.describe(include = 'all')
```

Out[21]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission
count	7253.000000	7253	7253	7253.000000	7.253000e+03	7253	7253
unique	NaN	2041	11	NaN	NaN	5	2
top	NaN	Mahindra XUV500 W8 2WD	Mumbai	NaN	NaN	Diesel	Manual
freq	NaN	55	949	NaN	NaN	3852	5204
mean	3626.000000	NaN	NaN	2013.365366	5.869906e+04	NaN	NaN
std	2093.905084	NaN	NaN	3.254421	8.442772e+04	NaN	NaN
min	0.000000	NaN	NaN	1996.000000	1.710000e+02	NaN	NaN
25%	1813.000000	NaN	NaN	2011.000000	3.400000e+04	NaN	NaN
50%	3626.000000	NaN	NaN	2014.000000	5.341600e+04	NaN	NaN
75%	5439.000000	NaN	NaN	2016.000000	7.300000e+04	NaN	NaN
max	7252.000000	NaN	NaN	2019.000000	6.500000e+06	NaN	NaN

In [22]:
```
# Separating Categorical and Numerical Data

cols_data = df.select_dtypes(include = ['object']).columns
numeric_data = df.select_dtypes(include =np.number).columns.tolist()

print("Categorical data  ", cols_data)
print("Numerical data  ", numeric_data)
```

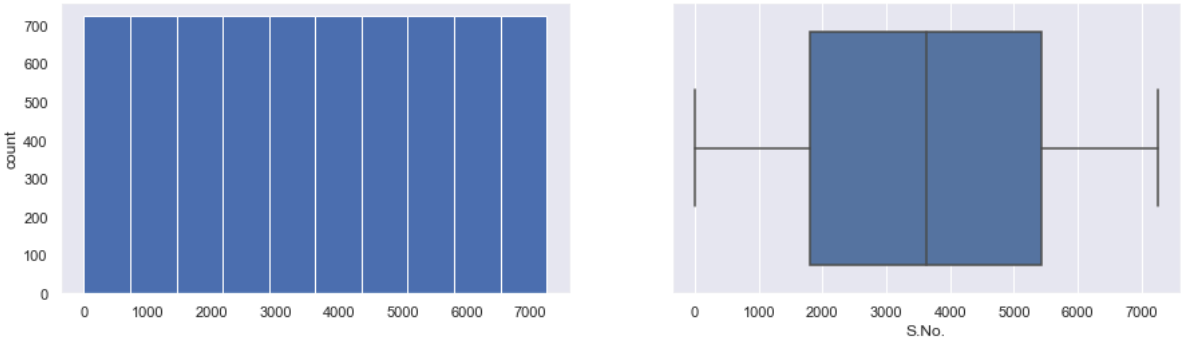
```
Categorical data   Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_
Type',
       'Mileage', 'Engine', 'Power', 'New_Price', 'Brand_Name', 'Model_Name'],
      dtype='object')
Numerical data   ['S.No.', 'Year', 'Kilometers_Driven', 'Seats', 'Price', 'Car_Ag
e']
```

Univariate Analysis

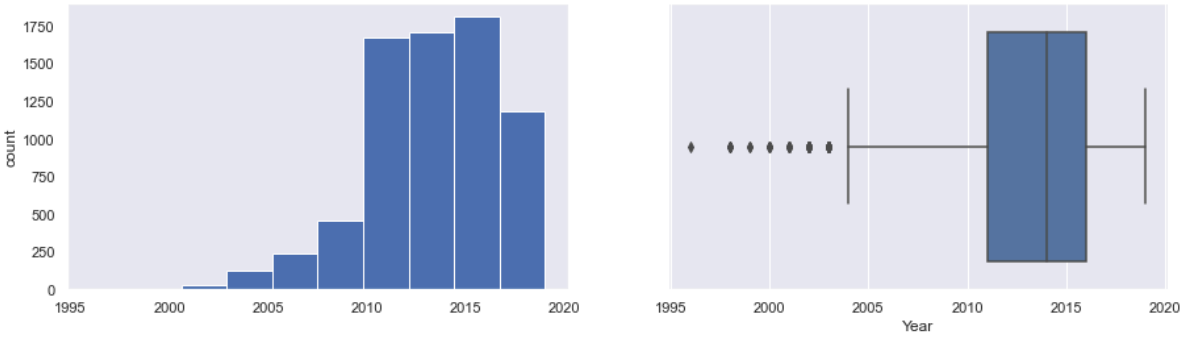
In [23]:
```
# Now, we will perform Unviariate analysis separately on Numerical and Categorical

for col in numeric_data:
    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()
```

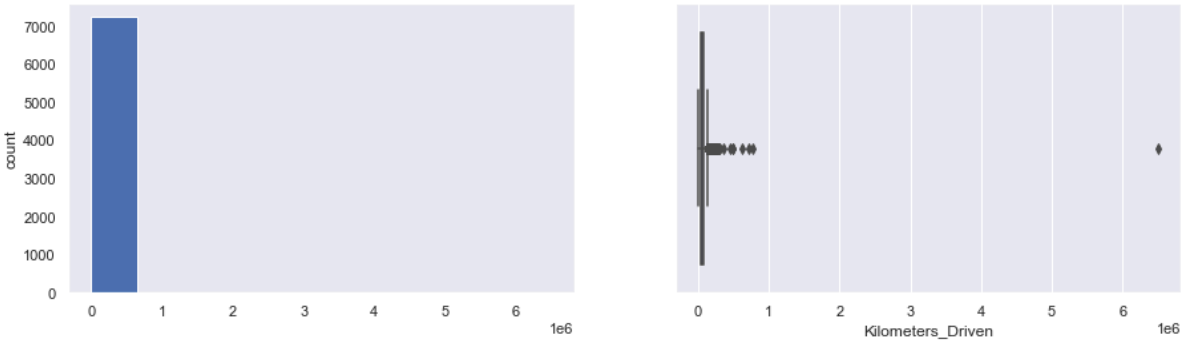
```
S.No.
Skew : 0.0
```



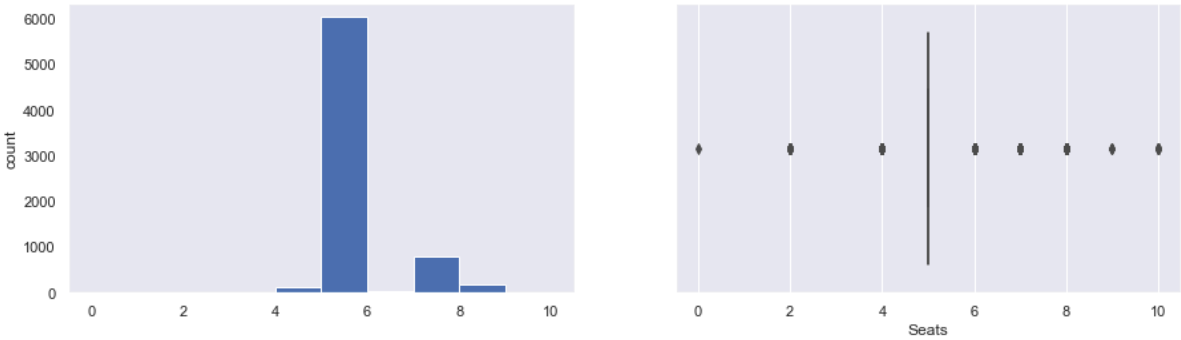

```
Year
Skew : -0.84
```



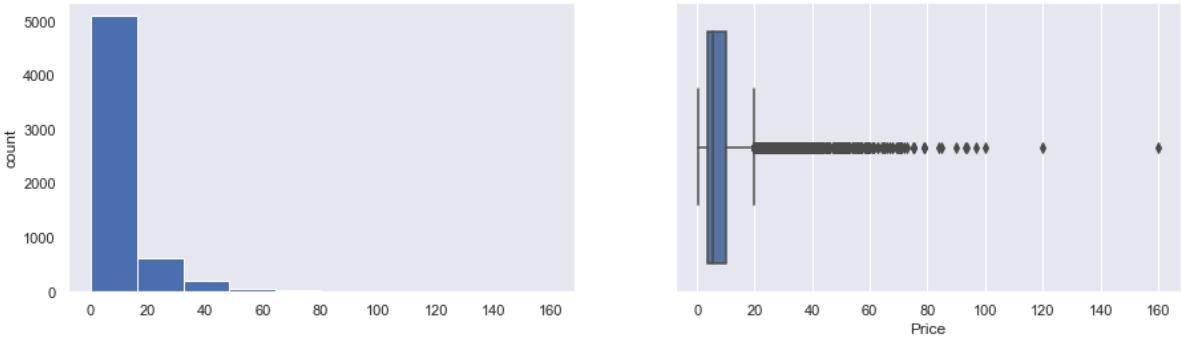

```
Kilometers_Driven
Skew : 61.58
```



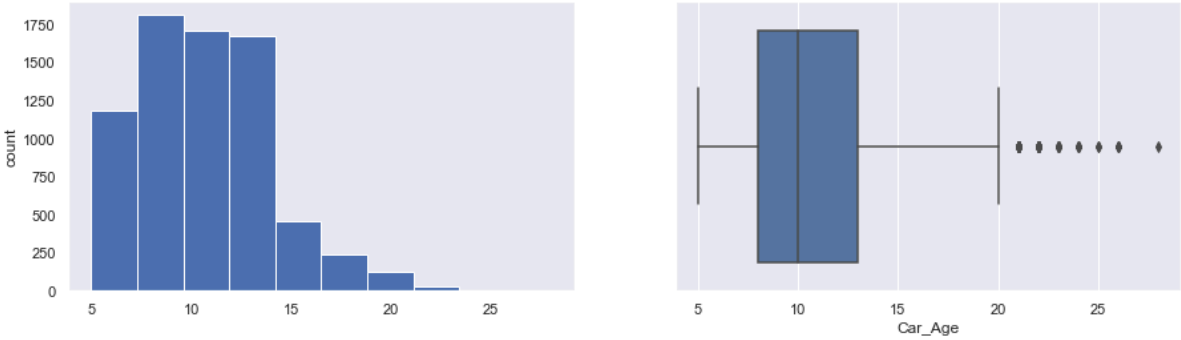

```
Seats
Skew : 1.9
```



Price
Skew : 3.34

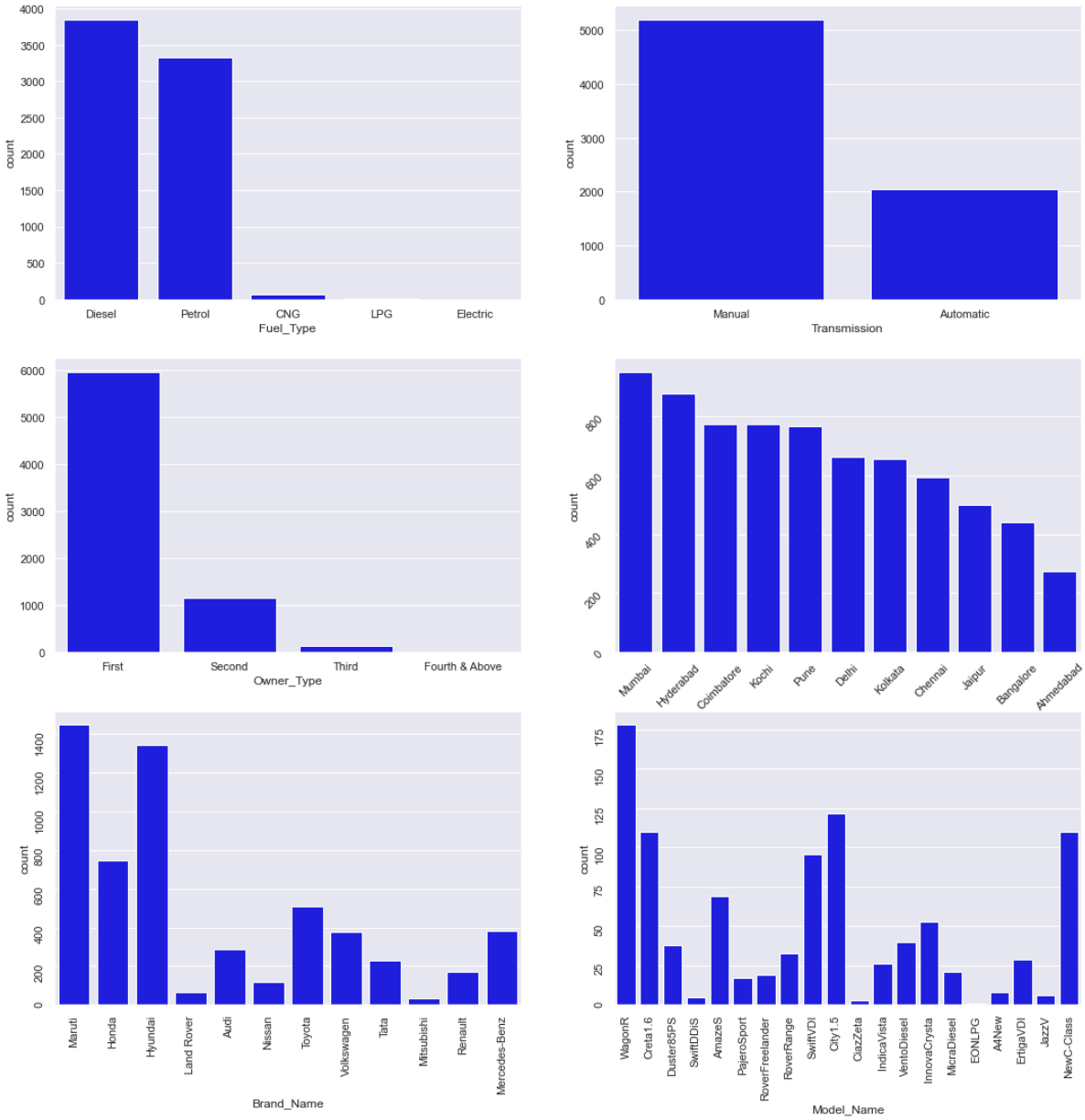


Car_Age
Skew : 0.84



In [24]:
```
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_Name', data = df, color = 'blue',
              order = df['Brand_Name'].head(20).value_counts().index);
sns.countplot(ax = axes[2, 1], x = 'Model_Name', data = df, color = 'blue',
              order = df['Model_Name'].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



A lot of insights can be generated by above graphs ->

1.) In the 'Owner_Type' chart, we can see that major sales of cars were done as First hand cars.

2.) Manual transmission cars are more soughted than Automatic cars, etc

In [25]:
```
# Variables with higher skewness must be Log Transformed to come up to other variab
# Function for Log Transformation

def Log_Transformation(data, cols):
    for colname in cols:
        if (df[colname] == 1.0).all():
            df[colname + '_log'] = np.log(df[colname] + 1)
        else :
            df[colname + '_log'] = np.log(df[colname])
```

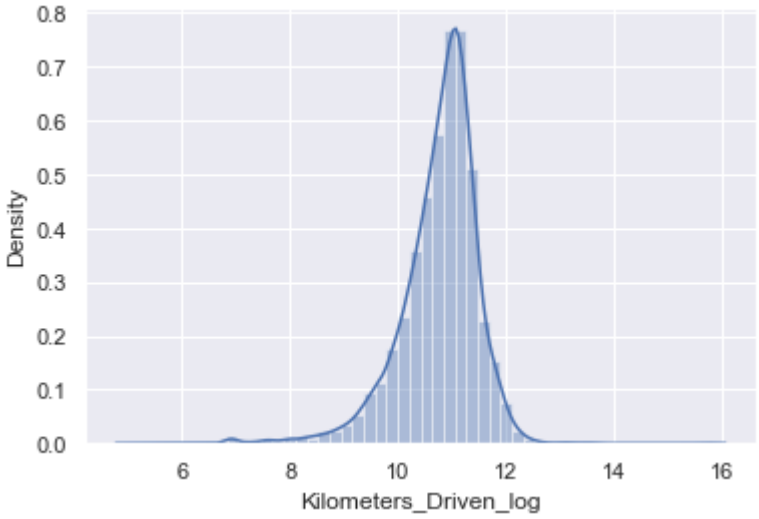
In [26]:
```
Log_Transformation(df, ['Kilometers_Driven', 'Price'])
```

In [27]:
```
# Graphical representation of Log Transformation

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

```
D:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distp
lot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

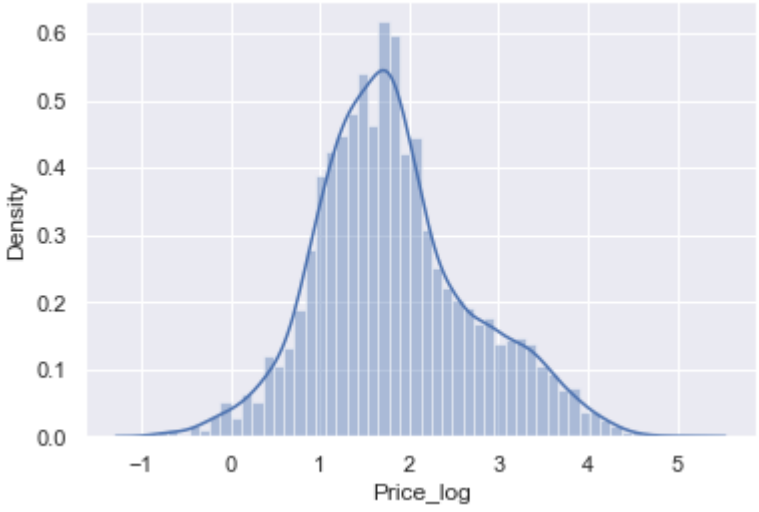
Out[27]:
```
<AxesSubplot:xlabel='Kilometers_Driven_log', ylabel='Density'>
```



In [28]:
```
sns.distplot(df['Price_log'], axlabel = "Price_log")
```

```
D:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distp
lot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[28]:
```
<AxesSubplot:xlabel='Price_log', ylabel='Density'>
```

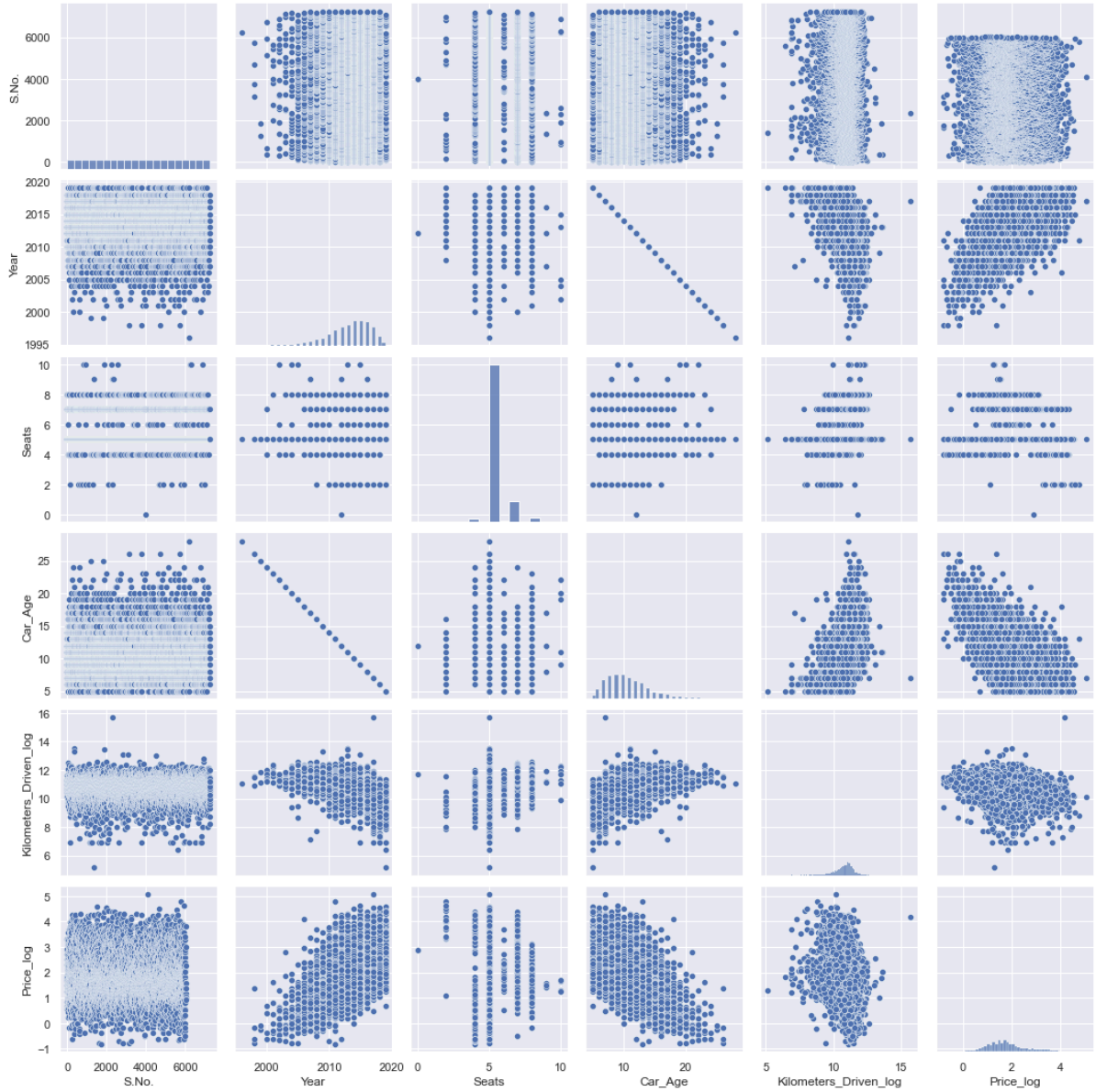


EDA BiVariate Analysis

In [31]:
```
# This analysis is mainly used to check the correlation between 2 or more variables
# This analysis is also useful if you have a classifier as your output.
```

```
plt.figure(figsize=(13,17))
sns.pairplot(data = df.drop(["Kilometers_Driven", "Price"], axis = 1))
plt.show()
```

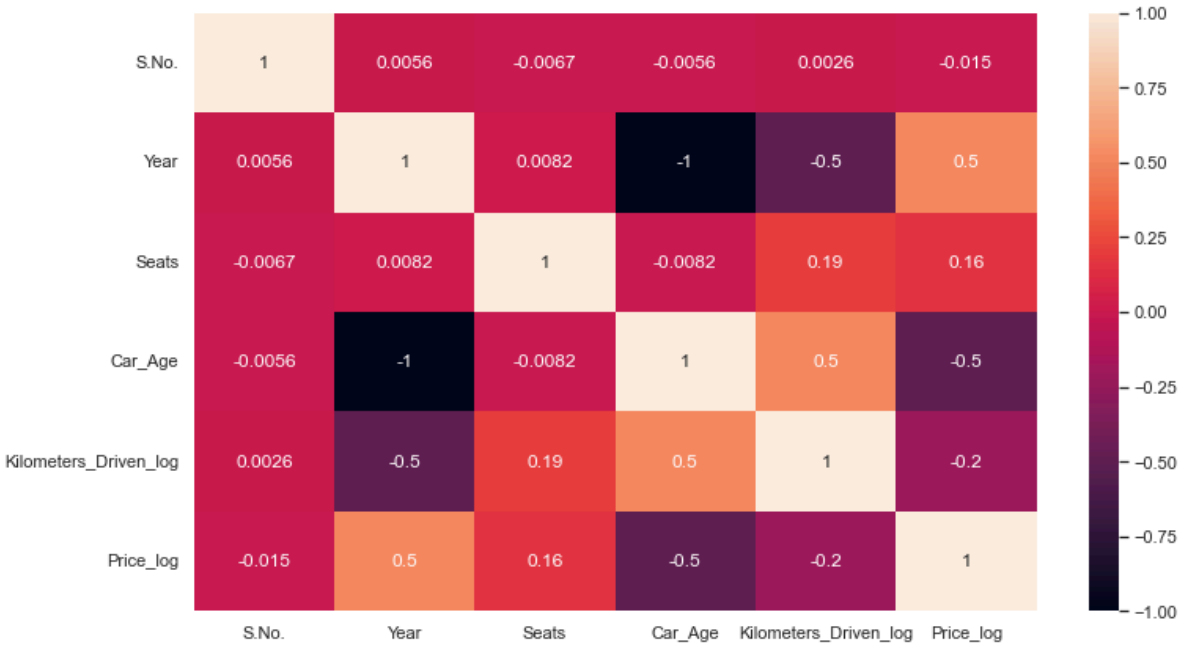
```
<Figure size 936x1224 with 0 Axes>
```



EDA Multivariate Analysis

In [32]:
```
# Above pairplots can be difficult to understand for some people to understand.
# So, we can use a "Heat Map" to clearly show the correlation between the variables

plt.figure(figsize=(12, 7))
sns.heatmap(data = df.drop(["Kilometers_Driven", "Price"], axis = 1).corr(), annot
plt.show()
```



From the above heatmap, we can infer the following ->

1.) Year and Price_log have a good correlation with them, infering that Year will put a good effect on Price_log

2.) Kilometers_Driven_log and Seats have a fair correlation with them, infering that more the car gets used, seats condition will get efffected.

etc....

Imputing missing values

In [ ]:
```
# Some columns, like mileage, seats, etc have missing values, like NA, etc.
# For imputing, we have some basic technniques
# We can fill missing values with the mean/median of the respective column, etc
# We can also use the domain knowledge to fill missing values too
# And lastly, some assumptions can also be made to fill the missing data
```

In [42]:
```
# Filling missing values with Null

df.loc[df["Mileage"]==0.0,'Mileage']=np.nan
df.Mileage.isnull().sum()

df["Mileage"].fillna(value=np.nan,inplace=True)
```

In [43]:
```
df.head(5)
```

Out[43]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second

In [55]:
```
# Imputing seats, engine, power columns too

#df['Seats'] = df.groupby(['Brand_Name', 'Model_Name'])['Seats'].apply(lambda x : x
#df['Engine'] = df.groupby(['Brand_Name', 'Model_Name'])['Engine'].apply(lambda x :
#df['Power'] = df.groupby(['Brand_Name', 'Model_Name'])['Power'].apply(lambda x : x

#df['Engine'] = pd.to_numeric(df['Prices'], errors='coerce')
df['Engine'] = pd.to_numeric(df['Engine'], errors = 'coerce')
df['Engine']=df.groupby(['Brand_Name','Model_Name'])['Engine'].apply(lambda x:x.fil

df['Seats'] = pd.to_numeric(df['Seats'], errors = 'coerce')
df['Seats'] = df.groupby(['Brand_Name', 'Model_Name'])['Seats'].apply(lambda x:x.fi

df['Power'] = pd.to_numeric(df['Power'], errors = 'coerce')
df['Power'] = df.groupby(['Brand_Name', 'Model_Name'])['Power'].apply(lambda x:x.fi
```

In [56]:
```
df
```

Out[56]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	Fi
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	Fi
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	Fi
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	Fi
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Seco
...	...	...	...	...	...	...	...	
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	Fi
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	Fi
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	Fi
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Th
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	Fi

7253 rows × 19 columns

Thank You