Importing necessary libraries

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

Importing the dataset

```
df=pd.read_csv('cardekho.csv')
df.head()
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage(km/ltr/kg
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.1
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1

```
df.shape
```

```
(8128, 12)
```

```
original_description = df.describe()
print(original_description)
```

```
              year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         7907.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.418783
std       4.044249   8.062534e+05  5.655055e+04            4.037145
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.780000
50%    2015.000000   4.500000e+05  6.000000e+04           19.300000
75%    2017.000000   6.750000e+05  9.800000e+04           22.320000
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  7907.000000  7912.000000  7907.000000
mean   1458.625016    91.517919     5.416719
std     503.916303    35.822499     0.959588
min     624.000000     0.000000     2.000000
25%    1197.000000    68.050000     5.000000
```

```
50%    1248.000000    82.000000     5.000000
75%    1582.000000   102.000000     5.000000
max    3604.000000   400.000000    14.000000
```

Seperating Dependent and Independent Variable

```
# Independent Variable
x=df.iloc[:,[i for i in range(df.shape[1]) if i!=2]].values #-->features
y=df.iloc[:,2].values #target--->selling price

print(x)
print(y)
```

```
[['Maruti Swift Dzire VDI' 2014 145500 ... 1248.0 74.0 5.0]
 ['Skoda Rapid 1.5 TDI Ambition' 2014 120000 ... 1498.0 103.52 5.0]
 ['Honda City 2017-2020 EXi' 2006 140000 ... 1497.0 78.0 5.0]
 ...
 ['Maruti Swift Dzire ZDi' 2009 120000 ... 1248.0 73.9 5.0]
 ['Tata Indigo CR4' 2013 25000 ... 1396.0 70.0 5.0]
 ['Tata Indigo CR4' 2013 25000 ... 1396.0 70.0 5.0]]
[450000 370000 158000 ... 382000 290000 290000]
```

**DATA CLEANING **

Missing Data (Unclean Data): mileage(km/ltr/kg) has missing values (count is 7907, while other columns have 8128 entries). This indicates that some mileage data is missing. engine and seats also have missing data (count is 7907, not matching the full dataset size of 8128). Cleaning Suggestion:

Handle missing values in mileage(km/ltr/kg), engine, and seats. You could either: Impute missing values with the mean/median (if data is missing randomly). Drop rows with missing values (if they are not significant in terms of size). Use a model to predict missing values based on other features.

2. Noise and Outliers: max values: The selling_price has a max value of 10,000,000 which is significantly higher than typical values. The km_driven has a max value of 2,360,457 which could be an outlier. mileage(km/ltr/kg) has values as high as 42 km/ltr/kg, which is quite unusual, as most cars usually have mileage below 30. Cleaning Suggestion:

Investigate extreme values further. If they seem to be data entry errors, you could cap or remove outliers. For example, values above a certain threshold (like a very high price or mileage) can be treated as outliers and removed or adjusted.

3. Inconsistent or Impossible Values: mileage(km/ltr/kg) has 0 values: Mileage cannot logically be zero for any valid vehicle. This could be data entry errors or misreported mileage values. max_power has 0 values: A power of 0 seems incorrect for any working vehicle. Cleaning Suggestion:

Investigate entries with 0 or invalid values and either replace them with a reasonable value (e.g., the mean or median) or remove the rows. For mileage and max_power, consider replacing the zeros with mean values or removing the rows where they appear.

5. Data Range Analysis: Engine Size: The engine sizes vary from 624 to 3604. While these values seem reasonable, extreme or incorrect values may need inspection. Seats: The seats feature has a max value of 14. Vehicles typically have fewer than 10 seats, so values like 14 might be worth investigating. Cleaning Suggestion:

Verify if the seats values greater than 10 are correct. They could indicate a vehicle type like a bus, but if it's unexpected, the values may need adjustment. Ensure that all engine values are within an acceptable range for vehicles.

```
df = df.drop(columns=['name'])
```

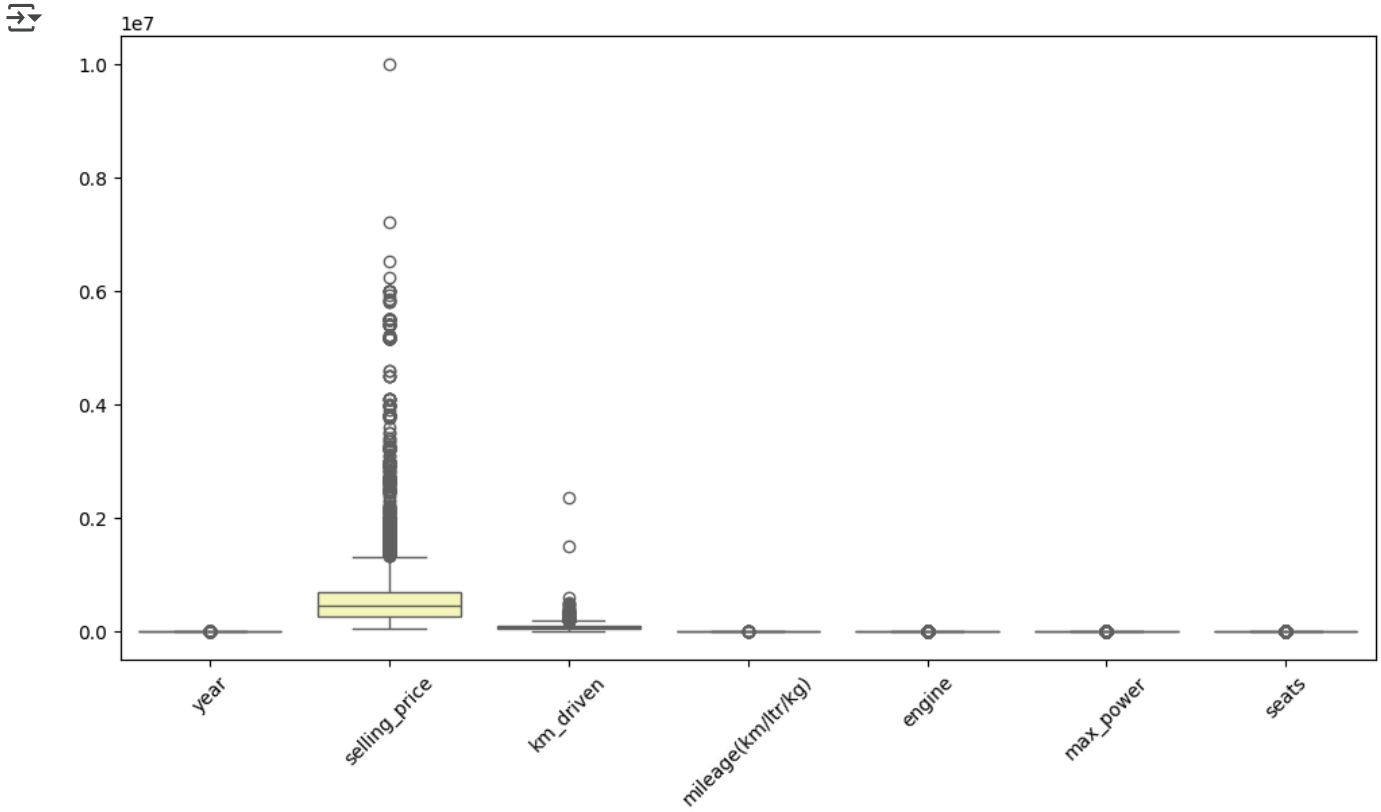
```
# check null
null_values=df.isnull().sum()
print(null_values)
```

```
year                    0
selling_price           0
km_driven               0
fuel                    0
seller_type             0
transmission            0
owner                   0
mileage(km/ltr/kg)    221
engine                221
max_power             216
seats                 221
dtype: int64
```

Start coding or generate with AI.

```
plt.figure(figsize=(12, 6))  # Increase figure size
sns.boxplot(data=df, palette='Set3')
plt.xticks(rotation=45)  # Rotate x-axis labels

plt.show()
```




```
# Convert 'engine' to numeric (ignore non-numeric values)
df['engine'] = pd.to_numeric(df['engine'], errors='coerce')
```

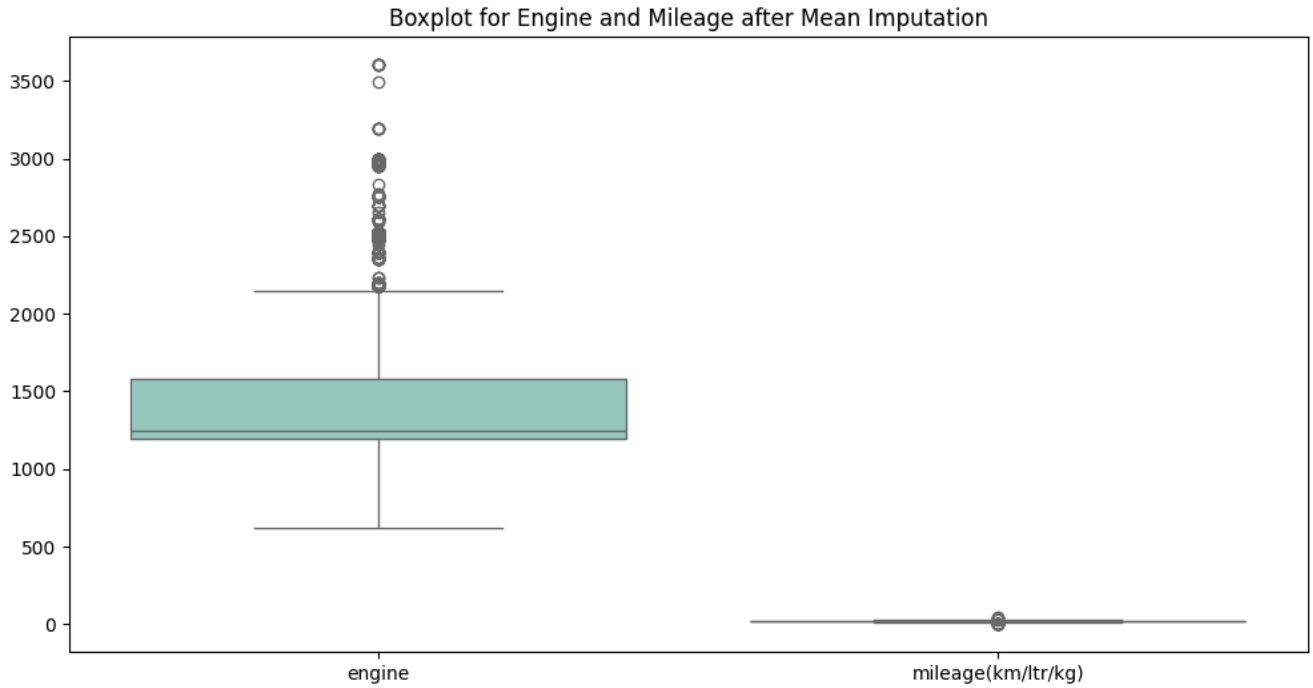
```
df_mean = df.copy()
df_mean[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(df[['engine', 'mile
```

```
print(df_mean.describe())

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_mean[['engine', 'mileage(km/ltr/kg)']], palette='Set3')  # Select only numeric columns
plt.title("Boxplot for Engine and Mileage after Mean Imputation")
plt.show()
```

```
               year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.418783
std       4.044249   8.062534e+05  5.655055e+04            3.981875
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.800000
50%    2015.000000   4.500000e+05  6.000000e+04           19.418783
75%    2017.000000   6.750000e+05  9.800000e+04           22.277500
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.000000  7912.000000  7907.000000
mean   1458.625016    91.517919     5.416719
std     497.017504    35.822499     0.959588
min     624.000000     0.000000     2.000000
25%    1197.000000    68.050000     5.000000
50%    1248.000000    82.000000     5.000000
75%    1582.000000   102.000000     5.000000
max    3604.000000   400.000000    14.000000
```



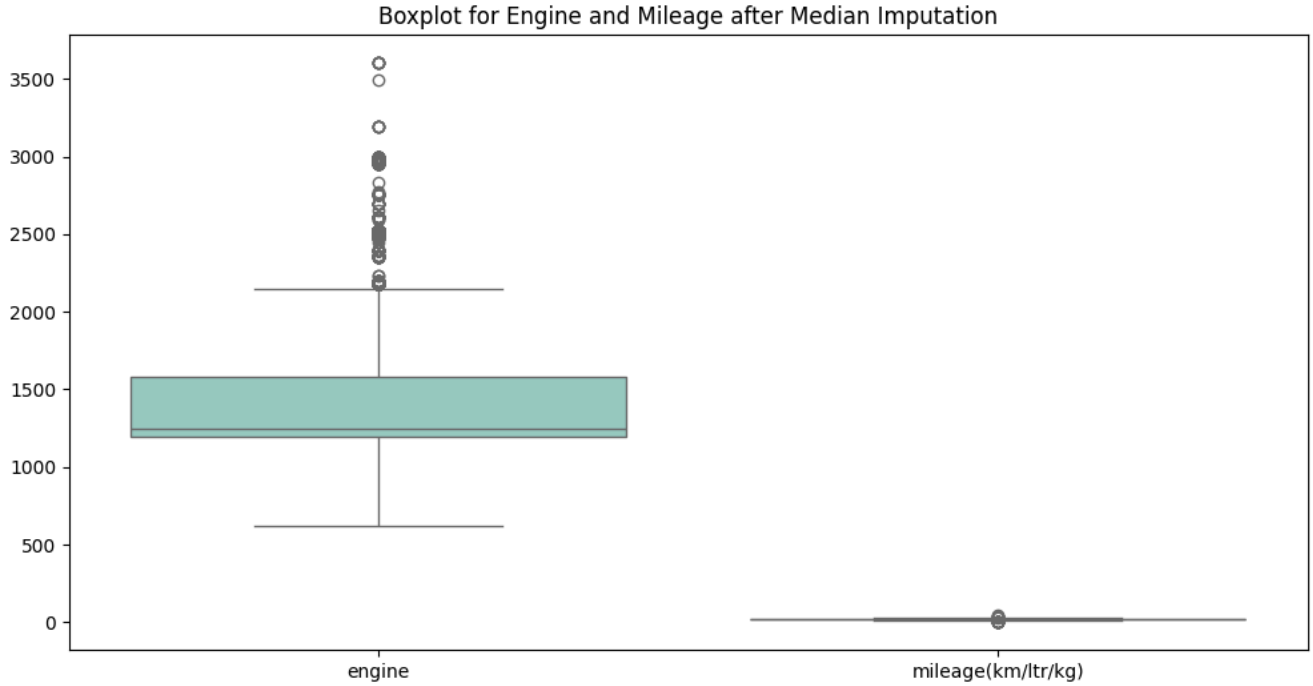

```
df_median = df.copy()
df_median[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(df[['engine', 'mi

print(df_median.describe())

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_median[['engine', 'mileage(km/ltr/kg)']], palette='Set3')  # Select only numeric column
plt.title("Boxplot for Engine and Mileage after Median Imputation")
plt.show()
```

```
              year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.415554
std       4.044249   8.062534e+05  5.655055e+04            3.981922
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.800000
50%    2015.000000   4.500000e+05  6.000000e+04           19.300000
75%    2017.000000   6.750000e+05  9.800000e+04           22.277500
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.00000  7912.000000  7907.000000
mean   1452.89813    91.517919     5.416719
std     498.19672    35.822499     0.959588
min     624.00000     0.000000     2.000000
25%    1197.00000    68.050000     5.000000
50%    1248.00000    82.000000     5.000000
75%    1582.00000   102.000000     5.000000
max    3604.00000   400.000000    14.000000
```



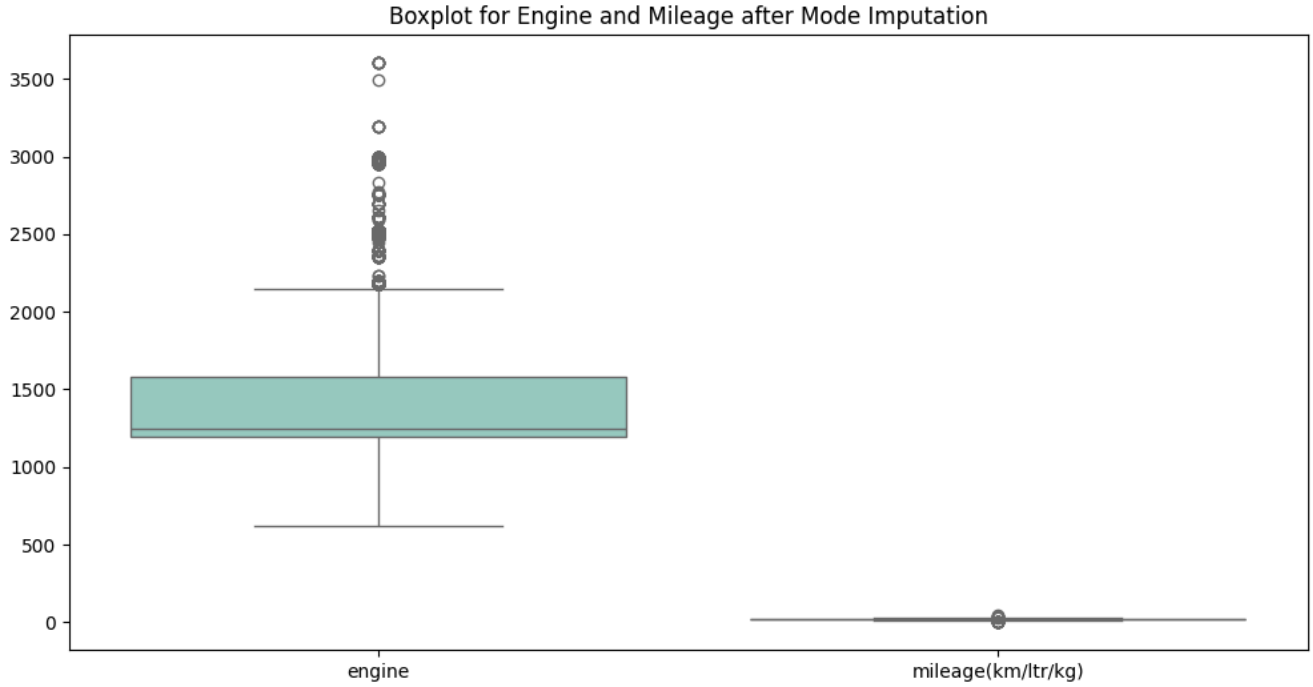

```
df_mode = df.copy()
df_mode[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(df[['engine', 'mile

print(df_mode.describe())

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_mode[['engine', 'mileage(km/ltr/kg)']], palette='Set3')  # Select only numeric columns
plt.title("Boxplot for Engine and Mileage after Mode Imputation")
plt.show()
```

```
              year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.404678
std       4.044249   8.062534e+05  5.655055e+04            3.982769
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.800000
50%    2015.000000   4.500000e+05  6.000000e+04           19.100000
75%    2017.000000   6.750000e+05  9.800000e+04           22.277500
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.00000  7912.000000  7907.000000
mean   1452.89813    91.517919     5.416719
std     498.19672    35.822499     0.959588
min     624.00000     0.000000     2.000000
25%    1197.00000    68.050000     5.000000
50%    1248.00000    82.000000     5.000000
75%    1582.00000   102.000000     5.000000
max    3604.00000   400.000000    14.000000
```



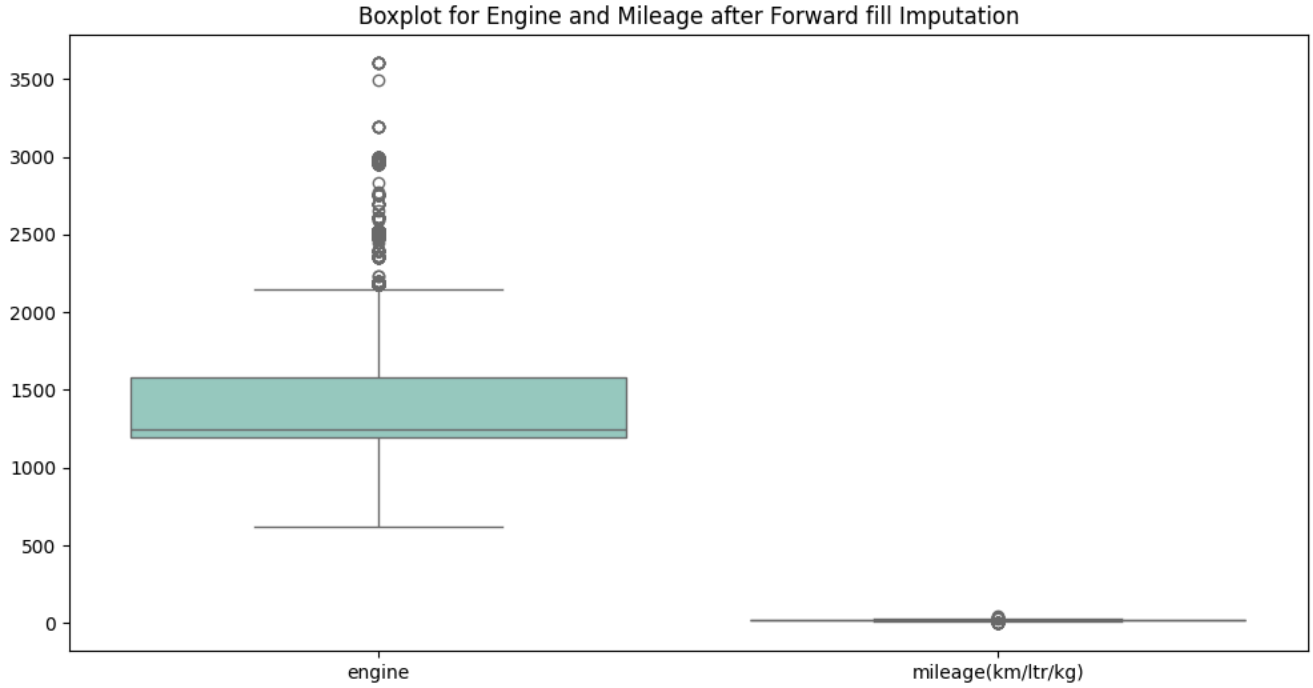

```
df_ffill = df.copy()
df_ffill[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(method='ffill')

print(df_ffill.describe())

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_ffill[['engine', 'mileage(km/ltr/kg)']], palette='Set3')  # Select only numeric columns
plt.title("Boxplot for Engine and Mileage after Forward fill Imputation")
plt.show()
```

```
<ipython-input-207-0d173e6eb424>:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and wil
  df_ffill[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(method='ffil
              year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.404732
std       4.044249   8.062534e+05  5.655055e+04            4.048404
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.780000
50%    2015.000000   4.500000e+05  6.000000e+04           19.300000
75%    2017.000000   6.750000e+05  9.800000e+04           22.320000
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.000000  7912.000000  7907.000000
mean   1458.028666    91.517919     5.416719
std     504.405870    35.822499     0.959588
min     624.000000     0.000000     2.000000
25%    1197.000000    68.050000     5.000000
50%    1248.000000    82.000000     5.000000
75%    1582.000000   102.000000     5.000000
max    3604.000000   400.000000    14.000000
```



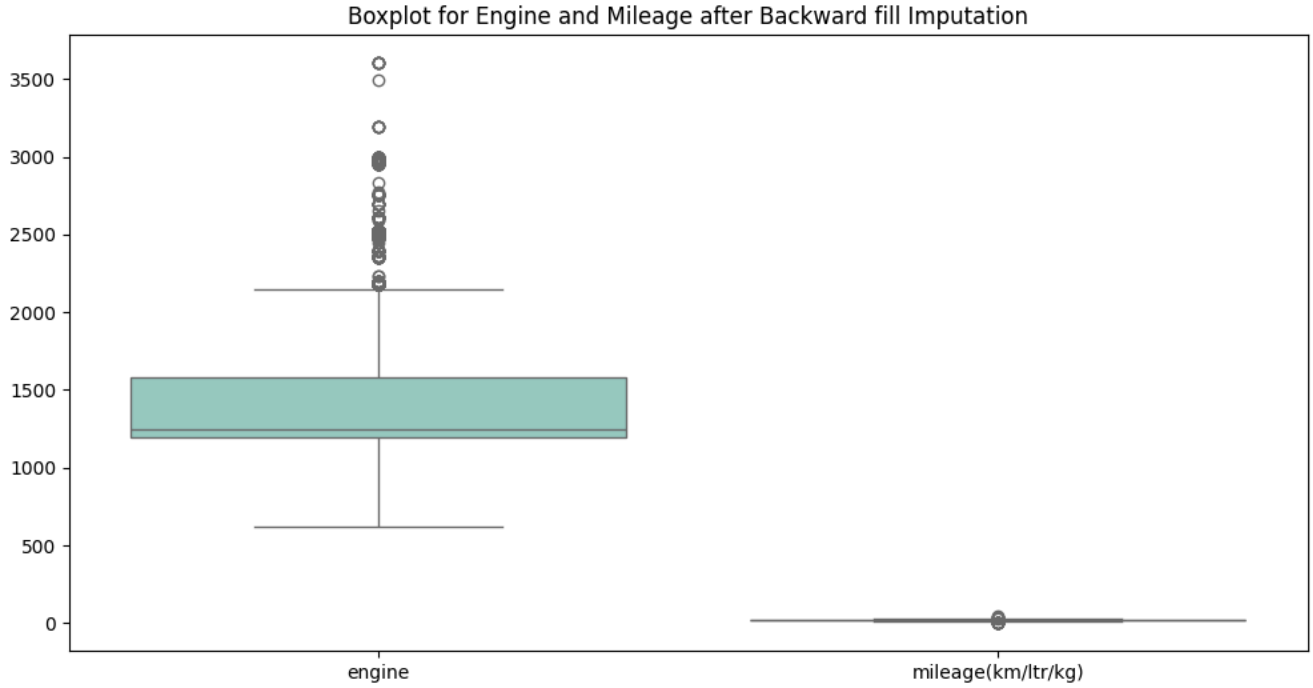

```
df_bfill = df.copy()
df_bfill[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(method='bfill')

print(df_bfill.describe())

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_bfill[['engine', 'mileage(km/ltr/kg)']], palette='Set3')  #
plt.title("Boxplot for Engine and Mileage after Backward fill Imputation")
plt.show()
```

```
<ipython-input-208-3f267383c48f>:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and wil
  df_bfill[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].fillna(method='bfil
              year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.426807
std       4.044249   8.062534e+05  5.655055e+04            4.032866
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.780000
50%    2015.000000   4.500000e+05  6.000000e+04           19.330000
75%    2017.000000   6.750000e+05  9.800000e+04           22.320000
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.000000  7912.000000  7907.000000
mean   1457.017224    91.517919     5.416719
std     502.914414    35.822499     0.959588
min     624.000000     0.000000     2.000000
25%    1197.000000    68.050000     5.000000
50%    1248.000000    82.000000     5.000000
75%    1582.000000   102.000000     5.000000
max    3604.000000   400.000000    14.000000
```

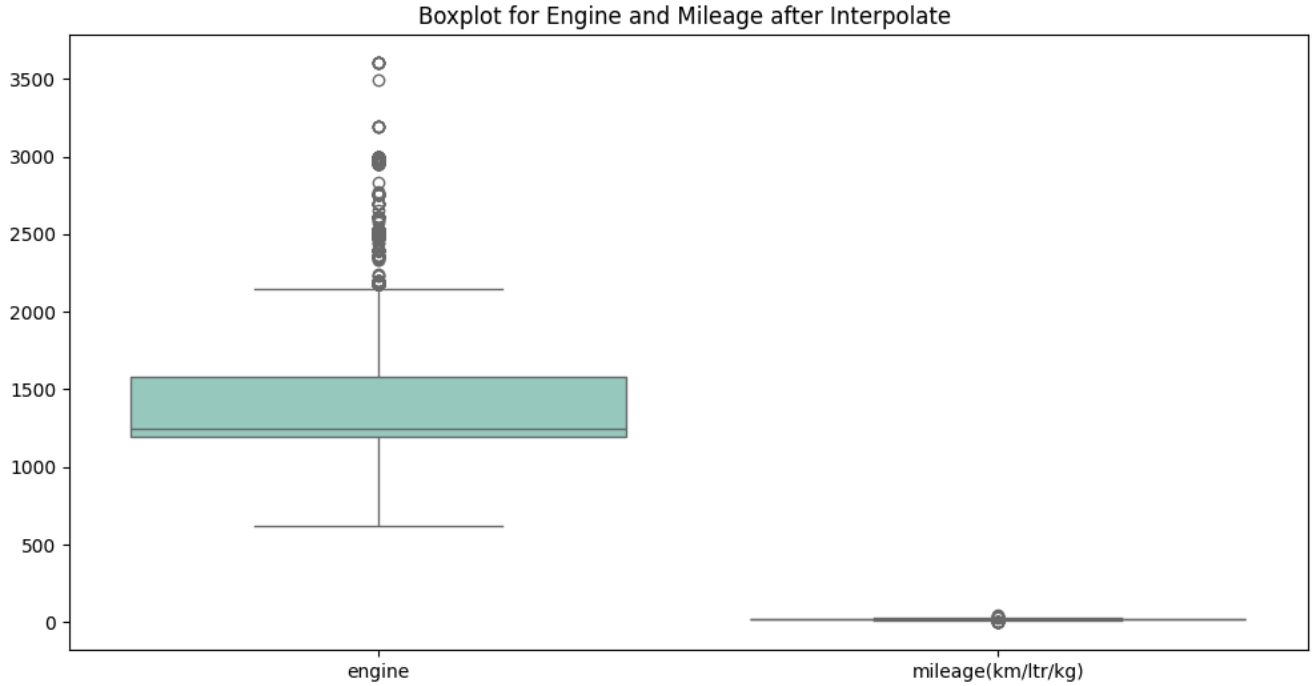



```
# Interpolation
df_int = df.copy()
df_int[['engine', 'mileage(km/ltr/kg)']] = df[['engine', 'mileage(km/ltr/kg)']].interpolate()

print(df_int.describe())
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_int[['engine', 'mileage(km/ltr/kg)']], palette='Set3')  #
plt.title("Boxplot for Engine and Mileage after Interpolate ")
plt.show()
```

```
              year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.415770
std       4.044249   8.062534e+05  5.655055e+04            4.010806
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.780000
50%    2015.000000   4.500000e+05  6.000000e+04           19.300000
75%    2017.000000   6.750000e+05  9.800000e+04           22.320000
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.000000  7912.000000  7907.000000
mean   1457.522945    91.517919     5.416719
std     500.293912    35.822499     0.959588
min     624.000000     0.000000     2.000000
25%    1197.000000    68.050000     5.000000
50%    1248.000000    82.000000     5.000000
75%    1582.000000   102.000000     5.000000
max    3604.000000   400.000000    14.000000
```



1. Engine (Numerical - Discrete) Best Method: Median Imputation Since engine capacity values (e.g., 1000cc, 1200cc, 1500cc) are discrete and often have a skewed distribution, using the median ensures that extreme values (luxury or high-performance engines) do not distort the results. Mean imputation can be misleading if the dataset has high-end sports cars and economy cars mixed.
2. Mileage (Numerical - Continuous) Best Method: Median Imputation Mileage can vary significantly based on car type (sedan, SUV, hatchback), fuel type, and engine size. If extreme values (e.g., electric cars with very high efficiency or sports cars with low mileage) exist, using the median prevents distortions. If the dataset is normally distributed without outliers, mean imputation could also work, but median is generally safer.

```
print(df.isnull().sum())
```

```
year                   0
selling_price          0
km_driven              0
fuel                   0
seller_type            0
transmission           0
owner                  0
mileage(km/ltr/kg)   221
engine               221
```

```
max_power             216
seats                 221
dtype: int64
```

```
# Filling missing values with median
df['engine'].fillna(df['engine'].median(), inplace=True)
df['mileage(km/ltr/kg)'].fillna(df['mileage(km/ltr/kg)'].median(), inplace=True)
```

Show hidden output

```
print(df.describe())
```

```
               year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  8128.000000   8.128000e+03  8.128000e+03         8128.000000
mean   2013.804011   6.382718e+05  6.981951e+04           19.415554
std       4.044249   8.062534e+05  5.655055e+04            3.981922
min    1983.000000   2.999900e+04  1.000000e+00            0.000000
25%    2011.000000   2.549990e+05  3.500000e+04           16.800000
50%    2015.000000   4.500000e+05  6.000000e+04           19.300000
75%    2017.000000   6.750000e+05  9.800000e+04           22.277500
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

            engine    max_power        seats
count  8128.00000  7912.000000  7907.000000
mean   1452.89813    91.517919     5.416719
std     498.19672    35.822499     0.959588
min     624.00000     0.000000     2.000000
25%    1197.00000    68.050000     5.000000
50%    1248.00000    82.000000     5.000000
75%    1582.00000   102.000000     5.000000
max    3604.00000   400.000000    14.000000
```

```
# dropping engine and seats

df.dropna(subset=['seats','max_power'], inplace=True)
```

```
print(df.isnull().sum())
```

```
year                  0
selling_price         0
km_driven             0
fuel                  0
seller_type           0
transmission          0
owner                 0
mileage(km/ltr/kg)    0
engine                0
max_power             0
seats                 0
dtype: int64
```

dataset description without missing values

```
# Independent Variable
x=df.iloc[:,[i for i in range(df.shape[1]) if i!=2]].values #-->features
y=df.iloc[:,2].values #target--->selling price

print(x)
print(y)
```

```
[[2014 450000 'Diesel' ... 1248.0 74.0 5.0]
 [2014 370000 'Diesel' ... 1498.0 103.52 5.0]
 [2006 158000 'Petrol' ... 1497.0 78.0 5.0]
 ...
```

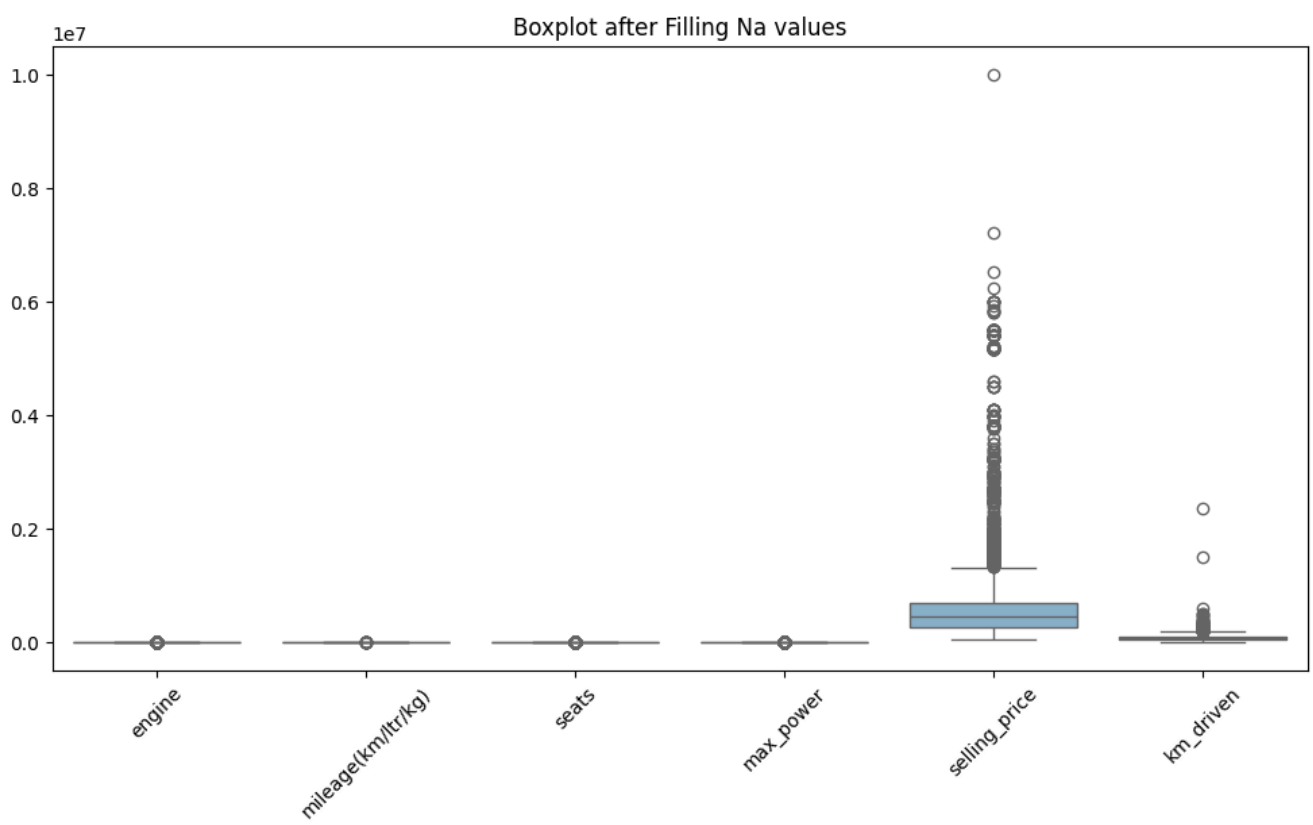
```
 [2009 382000 'Diesel' ... 1248.0 73.9 5.0]
 [2013 290000 'Diesel' ... 1396.0 70.0 5.0]
 [2013 290000 'Diesel' ... 1396.0 70.0 5.0]]
[145500 120000 140000 ... 120000  25000  25000]
```

```
print(df.describe())
```

```
               year  selling_price     km_driven  mileage(km/ltr/kg)  \
count  7906.000000   7.906000e+03  7.906000e+03         7906.000000
mean   2013.983936   6.498137e+05  6.918866e+04           19.419861
std       3.863695   8.135827e+05  5.679230e+04            4.036263
min    1994.000000   2.999900e+04  1.000000e+00            0.000000
25%    2012.000000   2.700000e+05  3.500000e+04           16.780000
50%    2015.000000   4.500000e+05  6.000000e+04           19.300000
75%    2017.000000   6.900000e+05  9.542500e+04           22.320000
max    2020.000000   1.000000e+07  2.360457e+06           42.000000

             engine    max_power        seats
count  7906.000000  7906.000000  7906.000000
mean   1458.708829    91.587374     5.416393
std     503.893057    35.747216     0.959208
min     624.000000    32.800000     2.000000
25%    1197.000000    68.050000     5.000000
50%    1248.000000    82.000000     5.000000
75%    1582.000000   102.000000     5.000000
max    3604.000000   400.000000    14.000000
```

```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_mean[['engine', 'mileage(km/ltr/kg)','seats','max_power','selling_price','km_driven']],
plt.title("Boxplot after Filling Na values")
plt.xticks(rotation=45)
plt.show()
```



INFERENCE :

1. The original dataset has no unclean, noisy data.

2. It has a categorical attribute : Transmission , owner, seller_type , fuel
3. The data is not on the same scale.

DATA TRANSFORMATION : One Hot Encoding

```
#Before applying
styled_df = df.style.background_gradient(cmap="coolwarm")

# Display in Jupyter Notebook
styled_df
```

1	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.140000
2	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.700000
3	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.000000
4	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.100000
5	2017	440000	45000	Petrol	Individual	Manual	First Owner	20.140000
6	2007	96000	175000	LPG	Individual	Manual	First Owner	17.300000
7	2001	45000	5000	Petrol	Individual	Manual	Second Owner	16.100000
8	2011	350000	90000	Diesel	Individual	Manual	First Owner	23.590000
9	2013	200000	169000	Diesel	Individual	Manual	First Owner	20.000000
10	2014	500000	68000	Diesel	Individual	Manual	Second Owner	19.010000
11	2005	92000	100000	Petrol	Individual	Manual	Second Owner	17.300000
12	2009	280000	140000	Diesel	Individual	Manual	Second Owner	19.300000
14	2009	180000	90000	Petrol	Individual	Manual	Second Owner	18.900000
15	2016	400000	40000	Petrol	Individual	Manual	First Owner	18.150000
16	2016	778000	70000	Diesel	Individual	Manual	Second Owner	24.520000
17	2012	500000	53000	Diesel	Individual	Manual	Second Owner	23.000000
18	2002	150000	80000	Petrol	Individual	Manual	Second Owner	19.700000
19	2016	680000	100000	Diesel	Individual	Manual	First Owner	22.540000
20	2011	174000	100000	Diesel	Individual	Manual	Second Owner	21.000000
21	2017	950000	50000	Diesel	Individual	Manual	First Owner	25.500000
22	2015	525000	40000	Diesel	Individual	Manual	First Owner	26.590000
23	2012	600000	72000	Diesel	Individual	Manual	First Owner	21.500000
24	2018	500000	35000	Petrol	Individual	Manual	First Owner	20.300000
25	2016	575000	45000	Petrol	Individual	Manual	First Owner	21.400000

26	2017	275000	28000	Petrol	Individual	Manual	First Owner	24.700000
27	2013	300000	70000	Diesel	Individual	Manual	First Owner	18.200000
28	2009	220000	120000	Petrol	Individual	Manual	First Owner	18.900000
29	2018	254999	25000	Petrol	Individual	Manual	First Owner	16.800000
30	2017	670000	70000	Diesel	Individual	Manual	First Owner	24.300000
32	2012	150000	35000	Petrol	Individual	Manual	Second Owner	14.000000
33	2018	730000	2388	Petrol	Individual	Manual	First Owner	18.600000
34	2017	650000	16200	Diesel	Individual	Manual	First Owner	24.300000
35	2019	330000	10000	CNG	Individual	Manual	Second Owner	33.440000
36	2019	366000	15000	Petrol	Individual	Manual	First Owner	23.950000
37	2019	1149000	5000	Petrol	Individual	Manual	First Owner	17.000000
38	2016	150000	42000	Petrol	Individual	Manual	First Owner	20.630000
39	2011	425000	60000	Diesel	Individual	Manual	Second Owner	13.930000
40	2012	150000	76000	Petrol	Individual	Manual	First Owner	16.100000
41	2019	2100000	5000	Petrol	Individual	Automatic	First Owner	16.000000
42	2018	925000	28900	Petrol	Dealer	Manual	First Owner	17.800000
43	2013	425000	86300	Petrol	Dealer	Manual	First Owner	16.800000
44	2018	675000	23300	Petrol	Dealer	Automatic	First Owner	18.500000
45	2018	819999	32600	Diesel	Dealer	Manual	First Owner	24.300000
46	2018	390000	10300	Petrol	Dealer	Manual	First Owner	23.950000
47	2014	1500000	77000	Diesel	Dealer	Manual	First Owner	12.550000
48	2013	700000	99000	Diesel	Dealer	Manual	First Owner	12.990000
49	2014	1450000	27800	Diesel	Dealer	Automatic	Second Owner	14.800000
50	2015	425000	49800	Diesel	Dealer	Manual	First Owner	24.700000
51	2013	1090000	151000	Diesel	Dealer	Manual	First Owner	13.500000
52	2015	600000	54700	Diesel	Dealer	Manual	First	26.000000

52	2015	600000	54700	Diesel	Dealer	Manual	Owner	20.000000
53	2018	850000	64000	Petrol	Dealer	Manual	First Owner	20.650000
54	2015	525000	63000	Diesel	Dealer	Manual	First Owner	27.300000
55	2016	1650000	127700	Diesel	Dealer	Automatic	Second Owner	11.360000
56	2016	950000	99000	Diesel	Dealer	Manual	First Owner	12.990000
57	2013	1750000	33900	Diesel	Dealer	Automatic	Second Owner	17.680000
58	2018	1590000	25000	Petrol	Dealer	Automatic	First Owner	14.280000
59	2013	1689999	50000	Diesel	Dealer	Automatic	First Owner	18.530000
60	2013	1425000	59000	Diesel	Dealer	Automatic	First Owner	14.840000
61	2011	265000	120000	Diesel	Individual	Manual	First Owner	21.120000
62	2011	190000	110000	Petrol	Individual	Manual	Third Owner	20.360000
63	2013	425000	60000	Diesel	Individual	Manual	First Owner	21.270000
64	2015	630000	147000	Diesel	Individual	Manual	Second Owner	26.590000
65	2017	600000	25000	Petrol	Individual	Manual	Third Owner	18.160000
66	2019	650000	30000	Petrol	Individual	Manual	First Owner	21.400000
67	2016	540000	40000	Diesel	Individual	Manual	First Owner	22.000000
68	2015	630000	135000	Diesel	Individual	Manual	First Owner	25.100000
69	2018	448000	30000	Petrol	Individual	Automatic	First Owner	20.510000
70	2017	500000	80000	Diesel	Individual	Manual	First Owner	21.660000
71	2016	745000	70000	Diesel	Individual	Manual	First Owner	24.300000
72	2019	1025000	9850	Diesel	Individual	Automatic	First Owner	24.300000
73	2011	235000	60000	Petrol	Individual	Manual	Second Owner	20.360000
74	2016	630000	70000	Diesel	Individual	Manual	First Owner	25.200000
75	2015	1700000	78000	Diesel	Individual	Manual	Second Owner	12.550000
76	2013	450000	120000	Diesel	Individual	Manual	Second Owner	22.900000
77	2012	450000	120000	Petrol	Individual	Manual	Second Owner	16.020000

79	2013	450000	170000	Diesel	Individual	Manual	First Owner	20.540000
80	2016	1200000	140000	Diesel	Individual	Manual	First Owner	12.990000
81	2015	610000	90000	Diesel	Individual	Manual	First Owner	22.770000
82	2016	2500000	30000	Petrol	Individual	Automatic	Second Owner	15.710000
83	2017	484999	10000	Petrol	Dealer	Manual	First Owner	23.100000
84	2016	275000	49000	Petrol	Dealer	Manual	First Owner	19.020000
85	2016	315000	32000	Petrol	Dealer	Manual	First Owner	24.700000
86	2016	275000	38000	Petrol	Dealer	Manual	First Owner	24.700000
88	2012	290000	28000	Petrol	Dealer	Manual	First Owner	19.810000
89	2017	280000	32000	Petrol	Dealer	Manual	First Owner	14.000000
90	2010	225000	44000	LPG	Dealer	Manual	First Owner	26.200000
91	2015	455000	42000	Petrol	Dealer	Manual	First Owner	16.470000
92	2015	351000	45000	Petrol	Dealer	Manual	Second Owner	19.810000
93	2012	535000	12000	Petrol	Dealer	Manual	Second Owner	15.040000
94	2009	175000	55500	Petrol	Dealer	Manual	First Owner	18.200000
95	2013	525000	61500	Petrol	Dealer	Manual	First Owner	18.500000
96	2016	600000	150000	Diesel	Individual	Manual	First Owner	26.590000
97	2016	565000	72000	Petrol	Dealer	Automatic	First Owner	19.100000
98	2008	120000	68000	Petrol	Dealer	Manual	Third Owner	19.700000
99	2017	725000	110000	Diesel	Individual	Manual	First Owner	22.540000
100	2009	185000	77000	Petrol	Dealer	Manual	Second Owner	21.790000
101	2010	200000	100000	Diesel	Individual	Manual	Second Owner	18.800000
102	2019	615000	10000	Petrol	Individual	Manual	First Owner	21.210000
103	2016	270000	100000	Petrol	Individual	Manual	Third Owner	15.370000
104	2018	610000	35000	Petrol	Individual	Manual	First Owner	21.400000
105	2014	625000	70000	Diesel	Individual	Automatic	Second Owner	11.790000