```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv('/content/drive/MyDrive/python-Saylani/quikr_car.csv')
df.head()
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel

```
df.shape
```

```
(892, 6)
```

```
df.describe()
```

	name	company	year	Price	kms_driven	fuel_type
count	892	892	892	892	840	837
unique	525	48	61	274	258	3
top	Honda City	Maruti	2015	Ask For Price	45,000 kms	Petrol
freq	13	235	117	35	30	440

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        892 non-null    object
 1   company     892 non-null    object
 2   year        892 non-null    object
 3   Price       892 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
dtypes: object(6)
memory usage: 41.9+ KB
```

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

	0
name	0
company	0
year	0
Price	0
kms_driven	52
fuel_type	55

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(94)
```

```
df.fuel_type.unique()
```

```
array(['Petrol', 'Diesel', nan, 'LPG'], dtype=object)
```

```
df.fuel_type.value_counts()
```

```
           count
fuel_type
Petrol       440
Diesel       395
LPG            2

dtype: int64
```

Data Postmortam

- Inconsistent names
- many of brands are not company name , some company names are in small case and also in big case but same company name
- Year is in the object format and non year values
- price has commas values and in string format
- kms_driven has object data type and commas with last value 'kms'
- fuel_type has nan values

Data Cleaning

```
df.year.unique()
```

```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '...', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
       'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
       '2 bs', 'arry', 'Eon', 'o...', 'ture', 'emi', 'car', 'able',
'no.',
       'd...', 'SALE', 'digo', 'sell', 'd Ex', 'n...', 'e...', 'D...',
       ', Ac', 'go .', 'k...', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
       't xe', 'EV2', 'r...', 'zest'], dtype=object)
```

```
# year column non year values
df=df[df['year'].str.isnumeric()]
```

```
df.year.unique()
```

```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '2003', '2004', '1995', '2002', '2001'], dtype=object)
```

```
df.company.unique()
```

```
array(['Hyundai', 'Mahindra', 'Maruti', 'Ford', 'Skoda', 'Audi',
'Toyota',
       'Renault', 'Honda', 'Datsun', 'Mitsubishi', 'Tata', 'Volkswagen',
       'I', 'Chevrolet', 'Mini', 'BMW', 'Nissan', 'Hindustan', 'Fiat',
       'Force', 'Mercedes', 'Land', 'Jaguar', 'Jeep', 'Volvo'],
      dtype=object)
```

```
df.shape
```

```
(842, 6)
```

```
# year is in object format
df['year']=df['year'].astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 842 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        842 non-null    object
 1   company     842 non-null    object
 2   year        842 non-null    int64 
 3   Price       842 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
dtypes: int64(1), object(5)
memory usage: 46.0+ KB
```

```
df.Price.unique()
```

```
array(['80,000', '4,25,000', 'Ask For Price', '3,25,000', '5,75,000',
       '1,75,000', '1,90,000', '8,30,000', '2,50,000', '1,82,000',
       '3,15,000', '4,15,000', '3,20,000', '10,00,000', '5,00,000',
       '3,50,000', '1,60,000', '3,10,000', '75,000', '1,00,000',
       '2,90,000', '95,000', '1,80,000', '3,85,000', '1,05,000',
       '6,50,000', '6,89,999', '4,48,000', '5,49,000', '5,01,000',
       '4,89,999', '2,80,000', '3,49,999', '2,84,999', '3,45,000',
       '4,99,999', '2,35,000', '2,49,999', '14,75,000', '3,95,000',
       '2,20,000', '1,70,000', '85,000', '2,00,000', '5,70,000',
       '1,10,000', '4,48,999', '18,91,111', '1,59,500', '3,44,999',
       '4,49,999', '8,65,000', '6,99,000', '3,75,000', '2,24,999',
       '12,00,000', '1,95,000', '3,51,000', '2,40,000', '90,000',
       '1,55,000', '6,00,000', '1,89,500', '2,10,000', '3,90,000',
       '1,35,000', '16,00,000', '7,01,000', '2,65,000', '5,25,000',
       '3,72,000', '6,35,000', '5,50,000', '4,85,000', '3,29,500',
       '2,51,111', '5,69,999', '69,999', '2,99,999', '3,99,999',
       '4,50,000', '2,70,000', '1,58,400', '1,79,000', '1,25,000',
       '2,99,000', '1,50,000', '2,75,000', '2,85,000', '3,40,000',
       '70,000', '2,89,999', '8,49,999', '7,49,999', '2,74,999',
       '9,84,999', '5,99,999', '2,44,999', '4,74,999', '2,45,000',
       '1,69,500', '3,70,000', '1,68,000', '1,45,000', '98,500',
       '2,09,000', '1,85,000', '9,00,000', '6,99,999', '1,99,999',
       '5,44,999', '1,99,000', '5,40,000', '49,000', '7,00,000',
'55,000',
       '8,95,000', '3,55,000', '5,65,000', '3,65,000', '40,000',
       '4,00,000', '3,30,000', '5,80,000', '3,79,000', '2,19,000',
       '5,19,000', '7,30,000', '20,00,000', '21,00,000', '14,00,000',
       '3,11,000', '8,55,000', '5,35,000', '1,78,000', '3,00,000',
       '2,55,000', '5,49,999', '3,80,000', '57,000', '4,10,000',
       '2,25,000', '1,20,000', '59,000', '5,99,000', '6,75,000',
'72,500',
       '6,10,000', '2,30,000', '5,20,000', '5,24,999', '4,24,999',
       '6,44,999', '5,84,999', '7,99,999', '4,44,999', '6,49,999',
       '9,44,999', '5,74,999', '3,74,999', '1,30,000', '4,01,000',
       '13,50,000', '1,74,999', '2,39,999', '99,999', '3,24,999',
       '10,74,999', '11,30,000', '1,49,000', '7,70,000', '30,000',
       '3,35,000', '3,99,000', '65,000', '1,69,999', '1,65,000',
       '5,60,000', '9,50,000', '7,15,000', '45,000', '9,40,000',
       '1,55,555', '15,00,000', '4,95,000', '8,00,000', '12,99,000',
       '5,30,000', '14,99,000', '32,000', '4,05,000', '7,60,000',
       '7,50,000', '4,19,000', '1,40,000', '15,40,000', '1,23,000',
       '4,98,000', '4,80,000', '4,88,000', '15,25,000', '5,48,900',
       '7,25,000', '99,000', '52,000', '28,00,000', '4,99,000',
       '3,81,000', '2,78,000', '6,90,000', '2,60,000', '90,001',
       '1,15,000', '15,99,000', '1,59,000', '51,999', '2,15,000',
       '35,000', '11,50,000', '2,69,000', '60,000', '4,30,000',
       '85,00,003', '4,01,919', '4,90,000', '4,24,000', '2,05,000',
       '5,49,900', '4,35,000', '1,89,700', '3,89,700', '3,60,000',
       '2,95,000', '1,14,990', '10,65,000', '4,70,000', '48,000',
       '1,88,000', '4,65,000', '1,79,999', '21,90,000', '23,90,000',
       '10,75,000', '4,75,000', '10,25,000', '6,15,000', '19,00,000',
       '14,90,000', '15,10,000', '18,50,000', '7,90,000', '17,25,000',
       '12,25,000', '68,000', '9,70,000', '31,00,000', '8,99,000',
       '88,000', '53,000', '5,68,500', '71,000', '5,90,000',
'7,95,000',
       '42,000', '1,89,000', '1,62,000', '35,999', '29,00,000',
'39,999',
```

```
# price has 'ask for price'
df=df[df['Price']!='Ask For Price']
```

```
df.shape
```

```
(819, 6)
```

```
# price has commas and in object format
df['Price']=df['Price'].str.replace(',','').astype(int)
df['Price'].unique()
```

```
array([  80000,  425000,  325000,  575000,  175000,  190000,  830000,
        250000,  182000,  315000,  415000,  320000, 1000000,  500000,
        350000,  160000,  310000,   75000,  100000,  290000,   95000,
        180000,  385000,  105000,  650000,  689999,  448000,  549000,
        501000,  489999,  280000,  349999,  284999,  345000,  499999,
        235000,  249999, 1475000,  395000,  220000,  170000,   85000,
        200000,  570000,  110000,  448999, 1891111,  159500,  344999,
        449999,  865000,  699000,  375000,  224999, 1200000,  195000,
        351000,  240000,   90000,  155000,  600000,  189500,  210000,
        390000,  135000, 1600000,  701000,  265000,  525000,  372000,
        635000,  550000,  485000,  329500,  251111,  569999,   69999,
        299999,  399999,  450000,  270000,  158400,  179000,  125000,
        299000,  150000,  275000,  285000,  340000,   70000,  289999,
        849999,  749999,  274999,  984999,  599999,  244999,  474999,
        245000,  169500,  370000,  168000,  145000,   98500,  209000,
        185000,  900000,  699999,  199999,  544999,  199000,  540000,
         49000,  700000,   55000,  895000,  355000,  565000,  365000,
         40000,  400000,  330000,  580000,  379000,  219000,  519000,
        730000, 2000000, 2100000, 1400000,  311000,  855000,  535000,
        178000,  300000,  255000,  549999,  380000,   57000,  410000,
        225000,  120000,   59000,  599000,  675000,   72500,  610000,
        230000,  520000,  524999,  424999,  644999,  584999,  799999,
        444999,  649999,  944999,  574999,  374999,  130000,  401000,
       1350000,  174999,  239999,   99999,  324999, 1074999, 1130000,
        149000,  770000,   30000,  335000,  399000,   65000,  169999,
        165000,  560000,  950000,  715000,   45000,  940000,  155555,
       1500000,  495000,  800000, 1299000,  530000, 1499000,   32000,
        405000,  760000,  750000,  419000,  140000, 1540000,  123000,
        498000,  480000,  488000, 1525000,  548900,  725000,   99000,
         52000, 2800000,  499000,  381000,  278000,  690000,  260000,
         90001,  115000, 1599000,  159000,   51999,  215000,   35000,
       1150000,  269000,   60000,  430000, 8500003,  401919,  490000,
        424000,  205000,  549900,  435000,  189700,  389700,  360000,
        295000,  114990, 1065000,  470000,   48000,  188000,  465000,
        179999, 2190000, 2390000, 1075000,  475000, 1025000,  615000,
       1900000, 1490000, 1510000, 1850000,  790000, 1725000, 1225000,
         68000,  970000, 3100000,  899000,   88000,   53000,  568500,
         71000,  590000,  795000,   42000,  189000,  162000,   35999,
       2900000,   39999,   50500,  510000,  860000,  500001])
```

```
df['kms_driven'].unique()
```

```
array(['45,000 kms', '40 kms', '28,000 kms', '36,000 kms', '41,000
kms',
       '25,000 kms', '24,530 kms', '60,000 kms', '30,000 kms',
       '32,000 kms', '48,660 kms', '4,000 kms', '16,934 kms',
       '43,000 kms', '35,550 kms', '39,522 kms', '39,000 kms',
       '55,000 kms', '72,000 kms', '15,975 kms', '70,000 kms',
       '23,452 kms', '35,522 kms', '48,508 kms', '15,487 kms',
       '82,000 kms', '20,000 kms', '68,000 kms', '38,000 kms',
       '27,000 kms', '33,000 kms', '46,000 kms', '16,000 kms',
       '47,000 kms', '35,000 kms', '30,874 kms', '15,000 kms',
       '29,685 kms', '1,30,000 kms', '19,000 kms', '54,000 kms',
       '13,000 kms', '38,200 kms', '22,000 kms', '50,000 kms',
       '13,500 kms', '3,600 kms', '45,863 kms', '60,500 kms',
       '12,500 kms', '18,000 kms', '13,349 kms', '29,000 kms',
       '44,000 kms', '42,000 kms', '14,000 kms', '49,000 kms',
       '36,200 kms', '51,000 kms', '1,04,000 kms', '33,333 kms',
       '33,600 kms', '5,600 kms', '7,500 kms', '26,000 kms', '24,330
kms',
       '65,480 kms', '2,00,000 kms', '59,000 kms', '99,000 kms',
       '2,800 kms', '21,000 kms', '11,000 kms', '66,000 kms', '3,000
kms',
       '7,000 kms', '38,500 kms', '37,200 kms', '43,200 kms',
       '24,800 kms', '45,872 kms', '40,000 kms', '11,400 kms',
       '97,200 kms', '52,000 kms', '31,000 kms', '1,75,430 kms',
       '37,000 kms', '65,000 kms', '3,350 kms', '75,000 kms',
       '62,000 kms', '73,000 kms', '2,200 kms', '54,870 kms',
       '34,580 kms', '97,000 kms', '60 kms', '80,200 kms', '3,200 kms',
       '0,000 kms', '5,000 kms', '588 kms', '71,200 kms', '1,75,400
kms',
       '9,300 kms', '56,758 kms', '10,000 kms', '56,450 kms',
       '56,000 kms', '32,700 kms', '9,000 kms', '73 kms', '1,60,000
kms',
       '58,559 kms', '57,000 kms', '1,70,000 kms', '80,000 kms',
       '6,821 kms', '23,000 kms', '34,000 kms', '1,800 kms',
       '4,00,000 kms', '48,000 kms', '90,000 kms', '12,000 kms',
       '69,900 kms', '1,66,000 kms', '122 kms', '0 kms', '36,469 kms',
       '7,800 kms', '24,695 kms', '15,141 kms', '59,910 kms',
       '1,00,000 kms', '4,500 kms', '1,29,000 kms', '300 kms',
       '1,31,000 kms', '1,11,111 kms', '59,466 kms', '25,500 kms',
       '44,005 kms', '2,110 kms', '43,222 kms', '1,00,200 kms', '65
kms',
       '1,40,000 kms', '1,03,553 kms', '58,000 kms', '1,20,000 kms',
       '49,800 kms', '100 kms', '81,876 kms', '6,020 kms', '55,700
kms',
       '18,500 kms', '53,000 kms', '35,500 kms', '22,134 kms',
       '1,000 kms', '8,500 kms', '87,000 kms', '6,000 kms', '8,000
kms',
       '55,800 kms', '56,400 kms', '72,160 kms', '11,500 kms',
       '1,33,000 kms', '2,000 kms', '88,000 kms', '65,422 kms',
       '1,17,000 kms', '1,50,000 kms', '10,750 kms', '6,800 kms',
       '9,800 kms', '57,923 kms', '30,201 kms', '6,200 kms', '37,518
kms',
       '24,652 kms', '383 kms', '95,000 kms', '3,528 kms', '52,500
kms',
       '47,900 kms', '52,800 kms', '1,95,000 kms', '48,008 kms',
       '48,247 kms', '9,400 kms', '64,000 kms', '2,137 kms', '10,544
kms',
       '1,47,000 kms', '90,001 kms', '48,006 kms', '74,000 kms',
```

```
# kms_driven has object values with kms at last
df['kms_driven']=df['kms_driven'].str.split(' ').str.get(0).str.replace(',',''
```

```
df=df[df['kms_driven']!='Petrol']
```

```
df['kms_driven']=df['kms_driven'].astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 817 entries, 0 to 889
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        817 non-null    object
 1   company     817 non-null    object
 2   year        817 non-null    int64 
 3   Price       817 non-null    int64 
 4   kms_driven  817 non-null    int64 
 5   fuel_type   816 non-null    object
dtypes: int64(3), object(3)
memory usage: 44.7+ KB
```

```
# fuel_type has nan values
df=df[~df['fuel_type'].isna()]
```

```
df['name'].unique()
```

```
array(['Hyundai Santro Xing XO eRLX Euro III', 'Mahindra Jeep CL550
MDI',
       'Hyundai Grand i10 Magna 1.2 Kappa VTVT',
       'Ford EcoSport Titanium 1.5L TDCi', 'Ford Figo', 'Hyundai Eon',
       'Ford EcoSport Ambiente 1.5L TDCi',
       'Maruti Suzuki Alto K10 VXi AMT', 'Skoda Fabia Classic 1.2 MPI',
       'Maruti Suzuki Stingray VXi', 'Hyundai Elite i20 Magna 1.2',
       'Mahindra Scorpio SLE BS IV', 'Audi A8', 'Audi Q7',
       'Mahindra Scorpio S10', 'Maruti Suzuki Alto 800',
       'Maruti Suzuki Alto 800 Vxi', 'Hyundai i20 Sportz 1.2',
       'Maruti Suzuki Alto 800 Lx', 'Maruti Suzuki Vitara Brezza ZDi',
       'Maruti Suzuki Alto LX', 'Mahindra Bolero DI',
       'Maruti Suzuki Swift Dzire ZDi', 'Mahindra Scorpio S10 4WD',
       'Maruti Suzuki Swift Vdi BSIII',
       'Maruti Suzuki Wagon R VXi BS III',
       'Maruti Suzuki Wagon R VXi Minor',
       'Toyota Innova 2.0 G 8 STR BS IV', 'Renault Lodgy 85 PS RXL',
       'Skoda Yeti Ambition 2.0 TDI CR 4x2',
       'Maruti Suzuki Baleno Delta 1.2',
       'Renault Duster 110 PS RxZ Diesel Plus',
       'Renault Duster 85 PS RxE Diesel', 'Honda City 1.5 S MT',
       'Maruti Suzuki Dzire', 'Honda Amaze', 'Honda Amaze 1.5 SX i
DTEC',
       'Honda City', 'Datsun Redi GO S', 'Maruti Suzuki SX4 ZXI MT',
       'Mitsubishi Pajero Sport Limited Edition',
       'Maruti Suzuki Swift VXi 1.2 ABS BS IV', 'Honda City ZX CVT',
       'Maruti Suzuki Wagon R LX BS IV', 'Tata Indigo eCS LS CR4 BS
IV',
       'Volkswagen Polo Highline Exquisite P', 'Chevrolet Spark LS
1.0',
       'Renault Duster 110PS Diesel RxZ', 'Mini Cooper S 1.6',
       'Skoda Fabia 1.2L Diesel Ambiente', 'Renault Duster',
       'Mahindra Scorpio S4', 'Mahindra Scorpio VLX 2WD BS IV',
       'Mahindra Quanto C8', 'Ford EcoSport', 'Honda Brio',
       'Volkswagen Vento Highline Plus 1.5 Diesel AT',
       'Hyundai i20 Magna', 'Toyota Corolla Altis Diesel D4DG',
       'Hyundai Verna Transform SX VTVT',
       'Toyota Corolla Altis Petrol Ltd', 'Honda City 1.5 EXi New',
       'Skoda Fabia 1.2L Diesel Elegance', 'BMW 3 Series 320i',
       'Maruti Suzuki A Star Lxi', 'Toyota Etios GD',
       'Ford Figo Diesel EXI Option',
       'Maruti Suzuki Swift Dzire VXi 1.2 BS IV',
       'Chevrolet Beat LT Diesel', 'BMW 7 Series 740Li Sedan',
       'Mahindra XUV500 W8 AWD 2013', 'Hyundai i10 Magna 1.2',
       'Hyundai Verna Fluidic New', 'Maruti Suzuki Swift VXi 1.2 BS
IV',
       'Maruti Suzuki Ertiga ZXI Plus', 'Maruti Suzuki Ertiga Vxi',
       'Maruti Suzuki Ertiga VDi', 'Maruti Suzuki Alto LXi BS III',
       'Hyundai Grand i10 Asta 1.1 CRDi', 'Honda Amaze 1.2 S i VTEC',
       'Hyundai i20 Asta 1.4 CRDI 6 Speed', 'Ford Figo Diesel EXI',
       'Maruti Suzuki Eeco 5 STR WITH AC HTR', 'Maruti Suzuki Ertiga
ZXi',
       'Maruti Suzuki Esteem LXi BS III', 'Maruti Suzuki Ritz VXI',
       'Maruti Suzuki Ritz LDi', 'Maruti Suzuki Dzire VDI',
       'Toyota Etios Liva G', 'Hyundai i20 Sportz 1.4 CRDI',
       'Chevrolet Spark', 'Nissan Micra XV', 'Maruti Suzuki Swift',
       'Honda Amaze 1.5 S i DTEC', 'Chevrolet Beat',
       'Honda City 1.5 V MT', 'Ford EcoSport Trend 1.5L TDCi',
```

```python
# name column have spanned data
df['name']=df['name'].str.split(' ').str.slice(0,3).str.join(' ')
```

```python
df.head()
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
3	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
4	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
6	Ford Figo	Ford	2012	175000	41000	Diesel

```python
df['company'].unique()
```

```
array(['Hyundai', 'Mahindra', 'Ford', 'Maruti', 'Skoda', 'Audi',
'Toyota',
       'Renault', 'Honda', 'Datsun', 'Mitsubishi', 'Tata', 'Volkswagen',
       'Chevrolet', 'Mini', 'BMW', 'Nissan', 'Hindustan', 'Fiat',
'Force',
       'Mercedes', 'Land', 'Jaguar', 'Jeep', 'Volvo'], dtype=object)
```

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

	0
name	0
company	0
year	0
Price	0
kms_driven	0
fuel_type	0

dtype: int64

```
df.head()
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
3	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
4	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
6	Ford Figo	Ford	2012	175000	41000	Diesel

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 816 entries, 0 to 889
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        816 non-null    object
 1   company     816 non-null    object
 2   year        816 non-null    int64 
 3   Price       816 non-null    int64 
 4   kms_driven  816 non-null    int64 
 5   fuel_type   816 non-null    object
dtypes: int64(3), object(3)
memory usage: 44.6+ KB
```

```
df.duplicated().sum()
```

```
np.int64(96)
```

```
df.drop_duplicates(inplace=True)
```

```
df.shape
```

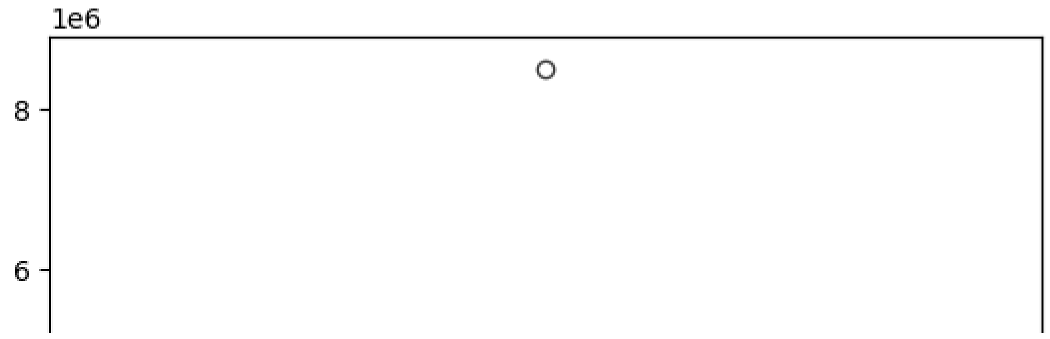
```
(720, 6)
```

```
df.describe()
```

	year	Price	kms_driven
count	720.000000	7.200000e+02	720.000000
mean	2012.325000	4.021574e+05	46787.397222
std	4.132363	4.781514e+05	34609.808464
min	1995.000000	3.000000e+04	0.000000
25%	2010.000000	1.672500e+05	27000.000000
50%	2013.000000	2.900000e+05	42000.000000
75%	2015.000000	4.899990e+05	59000.000000
max	2019.000000	8.500003e+06	400000.000000

```
sns.boxplot(df['Price'])
```

```
<Axes: ylabel='Price'>
```



```
df=df[df['Price']<5000000]
```

```
df.shape
```

```
(719, 6)
```

Data Seperation

```
x=df.drop(columns='Price')
y=df['Price']
```

```
x
```

	name	company	year	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	40	Diesel
3	Hyundai Grand i10	Hyundai	2014	28000	Petrol
4	Ford EcoSport Titanium	Ford	2014	36000	Diesel
6	Ford Figo	Ford	2012	41000	Diesel
...	...	...	...	...	...
883	Maruti Suzuki Ritz	Maruti	2011	50000	Petrol
885	Tata Indica V2	Tata	2009	30000	Diesel
886	Toyota Corolla Altis	Toyota	2009	132000	Petrol
888	Tata Zest XM	Tata	2018	27000	Diesel
889	Mahindra Quanto C8	Mahindra	2013	40000	Diesel

719 rows × 5 columns

```
y.shape
```

```
(719,)
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
x_train.shape,x_test.shape
```

```
((575, 5), (144, 5))
```

Feature Transformation

```
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
```

```
scalar=StandardScaler()
```

```
scalar.fit(x[['kms_driven']])
```

```
▾ StandardScaler
StandardScaler()
```

```
ohe=OneHotEncoder()
ohe.fit(x[['name','company','fuel_type']])
```

```
▾ OneHotEncoder
OneHotEncoder()
```