```
import pandas as pd

car=pd.read_csv('quikr_car.csv')

car.head()
```

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

   kms_driven fuel_type
0  45,000 kms    Petrol
1      40 kms    Diesel
2  22,000 kms    Petrol
3  28,000 kms    Petrol
4  36,000 kms    Diesel
```

```
car.shape
```

```
(892, 6)
```

```
car.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
```

There are missing values in two columns:- kms_driven and fuel_type. We need to clean the data.

```
# looking into our data
```

```
car['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)
```

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

```
array(['45,000 kms', '40 kms', '22,000 kms', '28,000 kms', '36,000
kms',
       '59,000 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', nan, '54,000 kms', '13,000 kms', '38,200 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', '28,028 kms', '2,00,000 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', '84,000 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', '24,000 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', '1,80,000 kms', '53,000 kms', '35,500 kms',
       '22,134 kms', '1,000 kms', '8,500 kms', '87,000 kms', '6,000
kms',
       '15,574 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', '5 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', '49,500
kms',
       '1,47,000 kms', '90,001 kms', '48,006 kms', '74,000 kms',
       '85,000 kms', '29,500 kms', '39,700 kms', '67,000 kms',
       '19,336 kms', '60,105 kms', '45,933 kms', '1,02,563 kms',
       '28,600 kms', '41,800 kms', '1,16,000 kms', '42,590 kms',
       '7,400 kms', '54,500 kms', '76,000 kms', '00 kms', '11,523
kms',
       '38,600 kms', '95,500 kms', '37,458 kms', '85,960 kms',
       '12,516 kms', '30,600 kms', '2,550 kms', '62,500 kms',
       '69,000 kms', '28,400 kms', '68,485 kms', '3,500 kms',
       '85,455 kms', '63,000 kms', '1,600 kms', '77,000 kms',
       '26,500 kms', '2,875 kms', '13,900 kms', '1,500 kms', '2,450
kms',
       '1,625 kms', '33,400 kms', '60,123 kms', '38,900 kms',
       '1,37,495 kms', '91,200 kms', '1,46,000 kms', '1,00,800 kms',
       '2,100 kms', '2,500 kms', '1,32,000 kms', 'Petrol'],
dtype=object)
```

```
car['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', '3,71,500', '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', '50,500', '5,10,000', '8,60,000',
       '5,00,001'], dtype=object)
```

```
car['fuel_type'].unique()
```

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

Quality of the data

- names are pretty inconsistent
- names have company names attached to it
- some names are spam like 'Maruti Ertiga showroom condition with' and 'Well mentained Tata Sumo'
- company: many of the names are not of any company like 'Used', 'URJENT', and so on.
- year has many non-year values
- year is in object. Change to integer
- Price has Ask for Price
- Price has commas in its prices and is in object
- kms_driven has object values with kms at last.
- It has nan values and two rows have 'Petrol' in them
- fuel_type has nan values

```
backup=car.copy()
```

Cleaning data

year has many non-year values

```
car=car[car['year'].str.isnumeric()]
```

year is in object. Change to integer

```
car['year']=car['year'].astype(int)
```

Price has 'Ask for Price',so remove that.

```
car=car[car['Price']!='Ask For Price']
```

Price has commas in its prices and is in object

```
car['Price']=car['Price'].str.replace(',','').astype(int)
```

kms_driven has object values with kms at last.

```
car['kms_driven']=car['kms_driven'].str.split().str.get(0).str.replace
(',','')

car['kms_driven']
```

```
0       45000
1          40
3       28000
4       36000
6       41000
```

```
        ...
886    132000
888     27000
889     40000
890    Petrol
891    Petrol
Name: kms_driven, Length: 819, dtype: object
```

It has nan values and two rows have 'Petrol' in them

```
car=car[car['kms_driven'].str.isnumeric()]
```

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

```
car['kms_driven']
```

```
0       45000
1          40
3       28000
4       36000
6       41000
        ...
883     50000
885     30000
886    132000
888     27000
889     40000
Name: kms_driven, Length: 817, dtype: int64
```

fuel_type has nan values

```
car=car[~car['fuel_type'].isna()]
```

```
car.shape
```

```
(816, 6)
```

```
car['name']
```

```
0        Hyundai Santro Xing XO eRLX Euro III
1                     Mahindra Jeep CL550 MDI
3      Hyundai Grand i10 Magna 1.2 Kappa VTVT
4            Ford EcoSport Titanium 1.5L TDCi
6                                  Ford Figo
                        ...
883                Maruti Suzuki Ritz VXI ABS
885                 Tata Indica V2 DLE BS III
886                      Toyota Corolla Altis
888                       Tata Zest XM Diesel
889                        Mahindra Quanto C8
Name: name, Length: 816, dtype: object
```

name and company had spammed data...but with the previous cleaning, those rows got removed.

Company does not need any cleaning now. Changing car names. Keeping only the first three words

```
car['name']=car['name'].str.split().str.slice(start=0,stop=3).str.join
('')
```

Resetting the index of the final cleaned data

```
car=car.reset_index(drop=True)
```

Cleaned data

```
car
```

```
                     name    company  year    Price  kms_driven
fuel_type
0       HyundaiSantroXing    Hyundai  2007    80000       45000
Petrol
1       MahindraJeepCL550   Mahindra  2006   425000          40
Diesel
2         HyundaiGrandi10    Hyundai  2014   325000       28000
Petrol
3    FordEcoSportTitanium       Ford  2014   575000       36000
Diesel
4                FordFigo       Ford  2012   175000       41000
Diesel
..                    ...        ...   ...      ...         ...        ..
.
811      MarutiSuzukiRitz     Maruti  2011   270000       50000
Petrol
812          TataIndicaV2       Tata  2009   110000       30000
Diesel
813    ToyotaCorollaAltis     Toyota  2009   300000      132000
Petrol
814            TataZestXM       Tata  2018   260000       27000
Diesel
815      MahindraQuantoC8   Mahindra  2013   390000       40000
Diesel

[816 rows x 6 columns]
```

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
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: 38.4+ KB
```

```
car.describe(include='all')
```

	name	company	year	Price	kms_driven
count	816	816	816.000000	8.160000e+02	816.000000
unique	254	25	NaN	NaN	NaN
top	MarutiSuzukiSwift	Maruti	NaN	NaN	NaN
freq	51	221	NaN	NaN	NaN
mean	NaN	NaN	2012.444853	4.117176e+05	46275.531863
std	NaN	NaN	4.002992	4.751844e+05	34297.428044
min	NaN	NaN	1995.000000	3.000000e+04	0.000000
25%	NaN	NaN	2010.000000	1.750000e+05	27000.000000
50%	NaN	NaN	2013.000000	2.999990e+05	41000.000000
75%	NaN	NaN	2015.000000	4.912500e+05	56818.500000
max	NaN	NaN	2019.000000	8.500003e+06	400000.000000

	fuel_type
count	816
unique	3
top	Petrol
freq	428
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

```
car.describe()
```

```
              year         Price     kms_driven
count   816.000000  8.160000e+02     816.000000
mean   2012.444853  4.117176e+05   46275.531863
std       4.002992  4.751844e+05   34297.428044
min    1995.000000  3.000000e+04       0.000000
25%    2010.000000  1.750000e+05   27000.000000
50%    2013.000000  2.999990e+05   41000.000000
75%    2015.000000  4.912500e+05   56818.500000
max    2019.000000  8.500003e+06  400000.000000
```

If we see the values of price, the minimum value is 3 into 10 raised to power 4, that e plus something is simply 10 raised to the power, so we can say that is 30000, and then we have 25th percentile, 50th percentile, and 75th percentile, but the maximum value is 85 into 10 to the power 6, which is approx 85 lakhs, which is impossible. It seems like an outlier...thus we need to remove it

```
car[car['Price']>6e6]
```

```
                 name   company  year    Price  kms_driven fuel_type
534  MahindraXUV500W6  Mahindra  2014  8500003       45000    Diesel
```

```
car[car['Price']>6e6].reset_index(drop=True)
```

```
               name   company  year    Price  kms_driven fuel_type
0  MahindraXUV500W6  Mahindra  2014  8500003       45000    Diesel
```

```
car
```

```
                     name   company  year   Price  kms_driven
fuel_type
0       HyundaiSantroXing   Hyundai  2007   80000       45000
Petrol
1       MahindraJeepCL550  Mahindra  2006  425000          40
Diesel
2         HyundaiGrandi10   Hyundai  2014  325000       28000
Petrol
3    FordEcoSportTitanium      Ford  2014  575000       36000
Diesel
4                FordFigo      Ford  2012  175000       41000
Diesel
..                    ...       ...   ...     ...         ...       ..
.
811      MarutiSuzukiRitz    Maruti  2011  270000       50000
Petrol
812          TataIndicaV2      Tata  2009  110000       30000
Diesel
813    ToyotaCorollaAltis    Toyota  2009  300000      132000
Petrol
814            TataZestXM      Tata  2018  260000       27000
Diesel
```

```
815      MahindraQuantoC8  Mahindra  2013  390000       40000
Diesel

[816 rows x 6 columns]
```

```
car.to_csv('Cleaned_Car_data.csv')
```

Model

```
X=car.drop('Price',axis=1)
y=car['Price']
```

```
print(X)
```

```
                     name   company  year  kms_driven fuel_type
0       HyundaiSantroXing   Hyundai  2007       45000    Petrol
1       MahindraJeepCL550  Mahindra  2006          40    Diesel
2         HyundaiGrandi10   Hyundai  2014       28000    Petrol
3    FordEcoSportTitanium      Ford  2014       36000    Diesel
4                FordFigo      Ford  2012       41000    Diesel
..                    ...       ...   ...         ...       ...
811      MarutiSuzukiRitz    Maruti  2011       50000    Petrol
812          TataIndicaV2      Tata  2009       30000    Diesel
813    ToyotaCorollaAltis    Toyota  2009      132000    Petrol
814            TataZestXM      Tata  2018       27000    Diesel
815      MahindraQuantoC8  Mahindra  2013       40000    Diesel

[816 rows x 5 columns]
```

```
print(y)
```

```
0       80000
1      425000
2      325000
3      575000
4      175000
        ...
811    270000
812    110000
813    300000
814    260000
815    390000
Name: Price, Length: 816, dtype: int64
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,rando
m_state=906)

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

```
OneHotEncoder()
```

```
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.cat
egories_),['name','company','fuel_type']),remainder='passthrough')

lr=LinearRegression()

pipe=make_pipeline(column_trans,lr)

pipe.fit(X_train,y_train)
```

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',

OneHotEncoder(categories=[array(['AudiA3Cabriolet', 'AudiA41.8',
'AudiA42.0', 'AudiA62.0', 'AudiA8',
       'AudiQ32.0', 'AudiQ52.0', 'AudiQ7', 'BMW3Series', 'BMW5Series',
       'BMW7Series', 'BMWX1', 'BMWX1sDrive20d', 'BMWX1xDrive20d',
       'ChevroletBeat', 'ChevroletBeat...

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

array(['Diesel', 'LPG', 'Petrol'], dtype=object)]),
                                                  ['name', 'company',
                                                   'fuel_type'])])),
                ('linearregression', LinearRegression())])
```

```
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.7768126284163006
```