In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
data=pd.read_csv(r"C:\Users\monim\Downloads\used_cars_data.csv")
data
```

Out[2]:

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

7253 rows × 14 columns

In [3]:

```
data.head()
```

Out[3]:

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

In [4]:

```
data.tail()
```

Out[4]:

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

In [5]:

```
data.describe()
```

Out[5]:

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

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   S.No.              7253 non-null   int64  
 1   Name               7253 non-null   object 
 2   Location           7253 non-null   object 
 3   Year               7253 non-null   int64  
 4   Kilometers_Driven  7253 non-null   int64  
 5   Fuel_Type          7253 non-null   object 
 6   Transmission       7253 non-null   object 
 7   Owner_Type         7253 non-null   object 
 8   Mileage            7251 non-null   object 
 9   Engine             7207 non-null   object 
 10  Power              7207 non-null   object 
 11  Seats              7200 non-null   float64
 12  New_Price          1006 non-null   object 
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

In [7]:

```
data.isna().any()
```

Out[7]:

```
S.No.                False
Name                 False
Location             False
Year                 False
Kilometers_Driven    False
Fuel_Type            False
Transmission         False
Owner_Type           False
Mileage               True
Engine                True
Power                 True
Seats                 True
New_Price             True
Price                 True
dtype: bool
```

In [8]:

```
data.columns
```

Out[8]:

```
Index(['S.No.', 'Name', 'Location', 'Year', 'Kilometers_Driven', 'Fuel_Typ
e',
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Seat
s',
       'New_Price', 'Price'],
      dtype='object')
```

In [9]:

```
data.isnull().sum()
```

Out[9]:

```
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 [11]:

```
data['Price'].value_counts()
```

Out[11]:

```
Price
4.50     88
5.50     84
3.50     82
4.25     73
3.25     71
         ..
11.62     1
43.60     1
19.05     1
3.94      1
7.43      1
Name: count, Length: 1373, dtype: int64
```

In [13]:

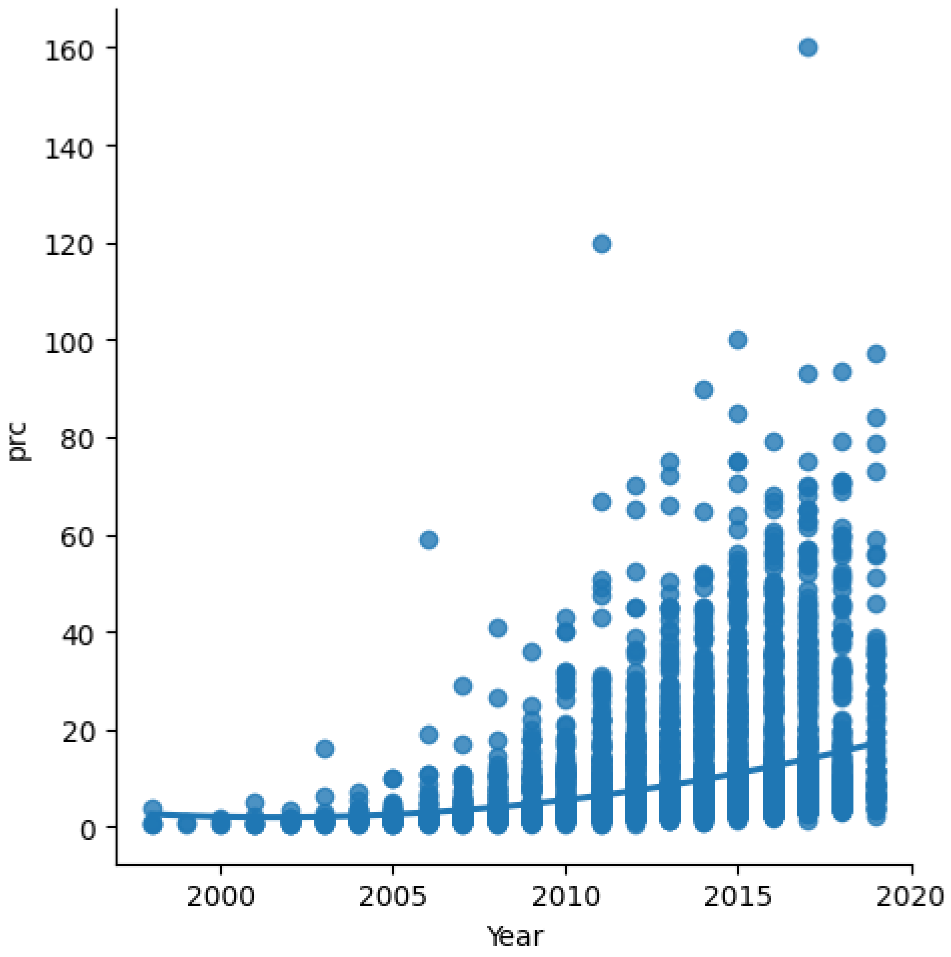
```
data=data[['Year','Price']]
data.columns=['Year','prc']
```

In [15]:

```
sns.lmplot(x='Year',y='prc',data=data,order=2,ci=None)
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x22f05fe2d40>
```



In [17]:

```
data.fillna(method='ffill',inplace=True)
```

```
C:\Users\monim\AppData\Local\Temp\ipykernel_17212\2063215332.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  data.fillna(method='ffill',inplace=True)
```

In [18]:

```
x = np.array(data['Year']).reshape(-1,1)
y = np.array(data['prc']).reshape(-1,1)
```

In [20]:

```
data.dropna(inplace=True)
```

```
C:\Users\monim\AppData\Local\Temp\ipykernel_17212\286435216.py:1: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  data.dropna(inplace=True)
```

In [21]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
regr = LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.0846921333432965
```

In [ ]: