In [1]:
```
import numpy as np
import pandas as pd
import matplotlib .pyplot as plt
import seaborn as sns
```

In [2]:
```
car= pd.read_csv(r"C:\Users\786\Downloads\car data.csv")
car
```

Out[2]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	M
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	M
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	M
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	M
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	Diesel	Dealer	M
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	M
298	city	2009	3.35	11.00	87934	Petrol	Dealer	M
299	city	2017	11.50	12.50	9000	Diesel	Dealer	M
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 9 columns

In [3]:
```
car.head()
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmiss
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Mar
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Mar
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Mar
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Mar
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Mar

In [4]:
```
car.describe()
```

Out[4]:

	Year	Selling_Price	Present_Price	Driven_kms	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.642584	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [5]:
```
car.tail()
```

Out[5]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
296	city	2016	9.50	11.6	33988	Diesel	Dealer	M
297	brio	2015	4.00	5.9	60000	Petrol	Dealer	M
298	city	2009	3.35	11.0	87934	Petrol	Dealer	M
299	city	2017	11.50	12.5	9000	Diesel	Dealer	M
300	brio	2016	5.30	5.9	5464	Petrol	Dealer	M

In [6]:
```
car.index
```

Out[6]:
```
RangeIndex(start=0, stop=301, step=1)
```

In [7]:
```
car.tail()
```

Out[7]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
296	city	2016	9.50	11.6	33988	Diesel	Dealer	M
297	brio	2015	4.00	5.9	60000	Petrol	Dealer	M
298	city	2009	3.35	11.0	87934	Petrol	Dealer	M
299	city	2017	11.50	12.5	9000	Diesel	Dealer	M
300	brio	2016	5.30	5.9	5464	Petrol	Dealer	M

In [8]:
```
car.isnull()
```

Out[8]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
296	False	False	False	False	False	False	False	
297	False	False	False	False	False	False	False	
298	False	False	False	False	False	False	False	
299	False	False	False	False	False	False	False	
300	False	False	False	False	False	False	False	

301 rows × 9 columns

In [9]:
```
car.isnull().sum()
```

Out[9]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Driven_kms       0
Fuel_Type        0
Selling_type     0
Transmission     0
Owner            0
dtype: int64
```

In [10]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Selling_type   301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

In [11]:
```
car.nunique()
```

Out[11]:
```
Car_Name          98
Year              16
Selling_Price    156
Present_Price    148
Driven_kms       206
Fuel_Type          3
Selling_type       2
Transmission       2
Owner              3
dtype: int64
```

In [12]:
```
car["Owner"].unique()
```

Out[12]:
```
array([0, 1, 3], dtype=int64)
```

In [13]:
```
car["Owner"].describe()
```

Out[13]:
```
count    301.000000
mean       0.043189
std        0.247915
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        3.000000
Name: Owner, dtype: float64
```

In [14]:
```
car["Year"].describe()
```

Out[14]:
```
count     301.000000
mean     2013.627907
std         2.891554
min      2003.000000
25%      2012.000000
50%      2014.000000
75%      2016.000000
max      2018.000000
Name: Year, dtype: float64
```

In [15]:
```
car["Selling_Price"].describe()
```

Out[15]:
```
count    301.000000
mean       4.661296
std        5.082812
min        0.100000
25%        0.900000
50%        3.600000
75%        6.000000
max       35.000000
Name: Selling_Price, dtype: float64
```

In [16]:
```
car["Selling_Price"].unique()
```

Out[16]:
```
array([ 3.35,  4.75,  7.25,  2.85,  4.6 ,  9.25,  6.75,  6.5 ,  8.75,
        7.45,  6.85,  7.5 ,  6.1 ,  2.25,  7.75,  3.25,  2.65,  4.9 ,
        4.4 ,  2.5 ,  2.9 ,  3.  ,  4.15,  6.  ,  1.95,  3.1 ,  2.35,
        4.95,  5.5 ,  2.95,  4.65,  0.35,  5.85,  2.55,  1.25,  1.05,
        5.8 , 14.9 , 23.  , 18.  , 16.  ,  2.75,  3.6 ,  4.5 ,  4.1 ,
       19.99,  6.95, 18.75, 23.5 , 33.  , 19.75,  4.35, 14.25,  3.95,
        1.5 ,  5.25, 14.5 , 14.73, 12.5 ,  3.49, 35.  ,  5.9 ,  3.45,
        3.8 , 11.25,  3.51,  4.  , 20.75, 17.  ,  7.05,  9.65,  1.75,
        1.7 ,  1.65,  1.45,  1.35,  1.2 ,  1.15,  1.11,  1.1 ,  1.  ,
        0.95,  0.9 ,  0.75,  0.8 ,  0.78,  0.72,  0.65,  0.6 ,  0.55,
        0.52,  0.51,  0.5 ,  0.48,  0.45,  0.42,  0.4 ,  0.38,  0.31,
        0.3 ,  0.27,  0.25,  0.2 ,  0.18,  0.17,  0.16,  0.15,  0.12,
        0.1 ,  5.75,  5.15,  7.9 ,  4.85, 11.75,  3.15,  6.45,  3.5 ,
        8.25,  5.11,  2.7 ,  6.15, 11.45,  3.9 ,  9.1 ,  4.8 ,  2.  ,
        5.35,  6.25,  5.95,  5.2 ,  3.75, 12.9 ,  5.  ,  5.4 ,  7.2 ,
       10.25,  8.5 ,  8.4 ,  9.15,  6.6 ,  3.65,  8.35,  6.7 ,  5.3 ,
       10.9 ,  8.65,  9.7 ,  2.1 ,  8.99,  7.4 ,  5.65, 10.11,  6.4 ,
        8.55,  9.5 , 11.5 ])
```

In [17]:
```
car["Fuel_Type"].describe()
```

Out[17]:
```
count        301
unique         3
top       Petrol
freq         239
Name: Fuel_Type, dtype: object
```

In [18]:
```
car["Fuel_Type"].unique()
```

Out[18]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [19]:
```
car["Selling_Price"].unique()
```

Out[19]:
```
array([ 3.35,  4.75,  7.25,  2.85,  4.6 ,  9.25,  6.75,  6.5 ,  8.75,
        7.45,  6.85,  7.5 ,  6.1 ,  2.25,  7.75,  3.25,  2.65,  4.9 ,
        4.4 ,  2.5 ,  2.9 ,  3.  ,  4.15,  6.  ,  1.95,  3.1 ,  2.35,
        4.95,  5.5 ,  2.95,  4.65,  0.35,  5.85,  2.55,  1.25,  1.05,
        5.8 , 14.9 , 23.  , 18.  , 16.  ,  2.75,  3.6 ,  4.5 ,  4.1 ,
       19.99,  6.95, 18.75, 23.5 , 33.  , 19.75,  4.35, 14.25,  3.95,
        1.5 ,  5.25, 14.5 , 14.73, 12.5 ,  3.49, 35.  ,  5.9 ,  3.45,
        3.8 , 11.25,  3.51,  4.  , 20.75, 17.  ,  7.05,  9.65,  1.75,
        1.7 ,  1.65,  1.45,  1.35,  1.2 ,  1.15,  1.11,  1.1 ,  1.  ,
        0.95,  0.9 ,  0.75,  0.8 ,  0.78,  0.72,  0.65,  0.6 ,  0.55,
        0.52,  0.51,  0.5 ,  0.48,  0.45,  0.42,  0.4 ,  0.38,  0.31,
        0.3 ,  0.27,  0.25,  0.2 ,  0.18,  0.17,  0.16,  0.15,  0.12,
        0.1 ,  5.75,  5.15,  7.9 ,  4.85, 11.75,  3.15,  6.45,  3.5 ,
        8.25,  5.11,  2.7 ,  6.15, 11.45,  3.9 ,  9.1 ,  4.8 ,  2.  ,
        5.35,  6.25,  5.95,  5.2 ,  3.75, 12.9 ,  5.  ,  5.4 ,  7.2 ,
       10.25,  8.5 ,  8.4 ,  9.15,  6.6 ,  3.65,  8.35,  6.7 ,  5.3 ,
       10.9 ,  8.65,  9.7 ,  2.1 ,  8.99,  7.4 ,  5.65, 10.11,  6.4 ,
        8.55,  9.5 , 11.5 ])
```

In [20]:
```
car.drop(['Owner'],axis=1,inplace=True)
```

In [21]:
```
car
```

Out[21]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	M
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	M
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	M
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	M
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	Diesel	Dealer	M
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	M
298	city	2009	3.35	11.00	87934	Petrol	Dealer	M
299	city	2017	11.50	12.50	9000	Diesel	Dealer	M
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 8 columns

In [22]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Selling_type   301 non-null    object
 7   Transmission   301 non-null    object
dtypes: float64(2), int64(2), object(4)
memory usage: 18.9+ KB
```

In [23]:
```
car.drop(['Selling_type'],axis=1,inplace=True)
```

In [24]:
```
car
```

Out[24]:

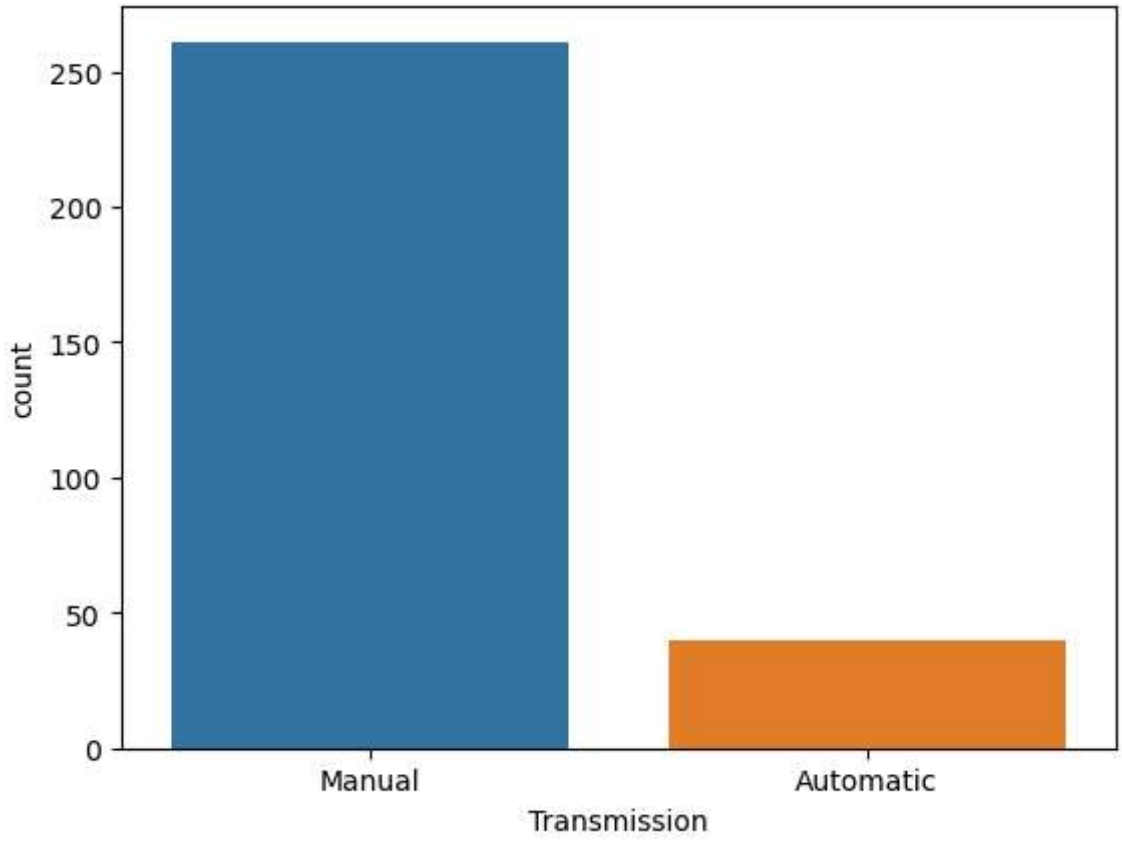
	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Transmission
0	ritz	2014	3.35	5.59	27000	Petrol	Manual
1	sx4	2013	4.75	9.54	43000	Diesel	Manual
2	ciaz	2017	7.25	9.85	6900	Petrol	Manual
3	wagon r	2011	2.85	4.15	5200	Petrol	Manual
4	swift	2014	4.60	6.87	42450	Diesel	Manual
...	...	...	...	...	...	...	...
296	city	2016	9.50	11.60	33988	Diesel	Manual
297	brio	2015	4.00	5.90	60000	Petrol	Manual
298	city	2009	3.35	11.00	87934	Petrol	Manual
299	city	2017	11.50	12.50	9000	Diesel	Manual
300	brio	2016	5.30	5.90	5464	Petrol	Manual

301 rows × 7 columns

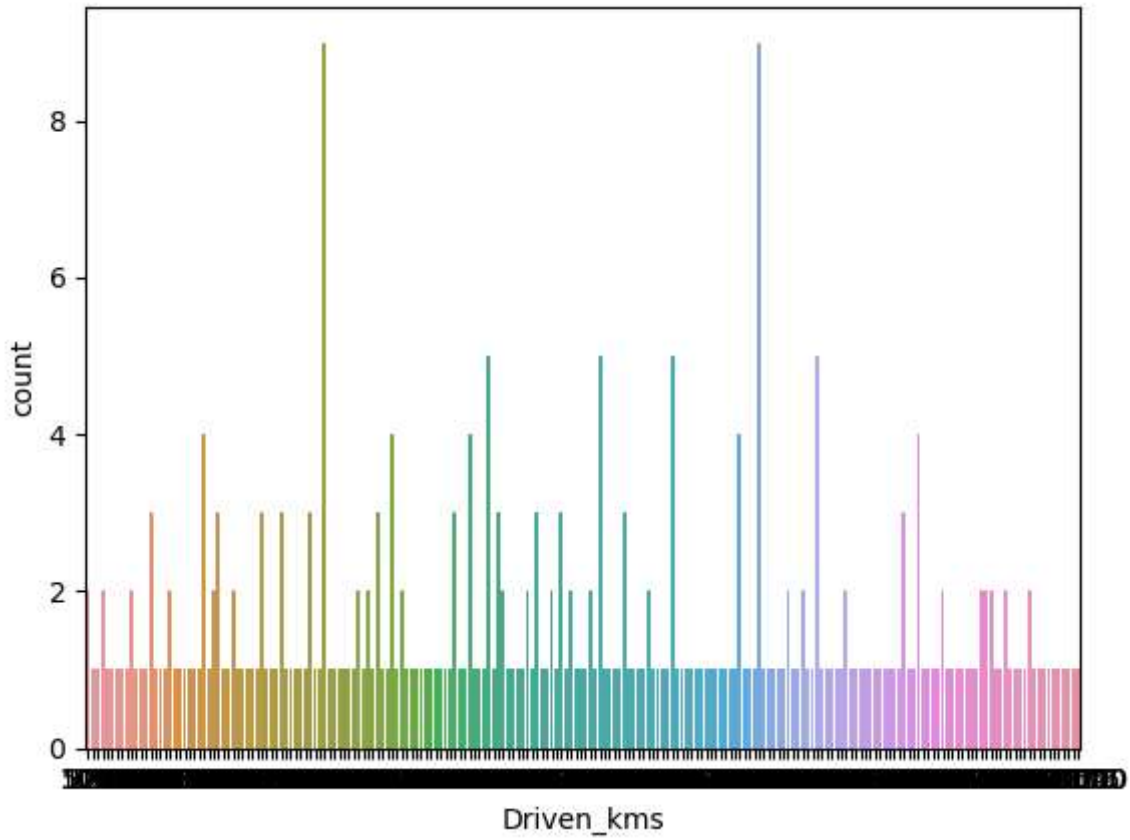
In [25]:
```
car.columns
```

Out[25]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Driven_kms',
       'Fuel_Type', 'Transmission'],
      dtype='object')
```

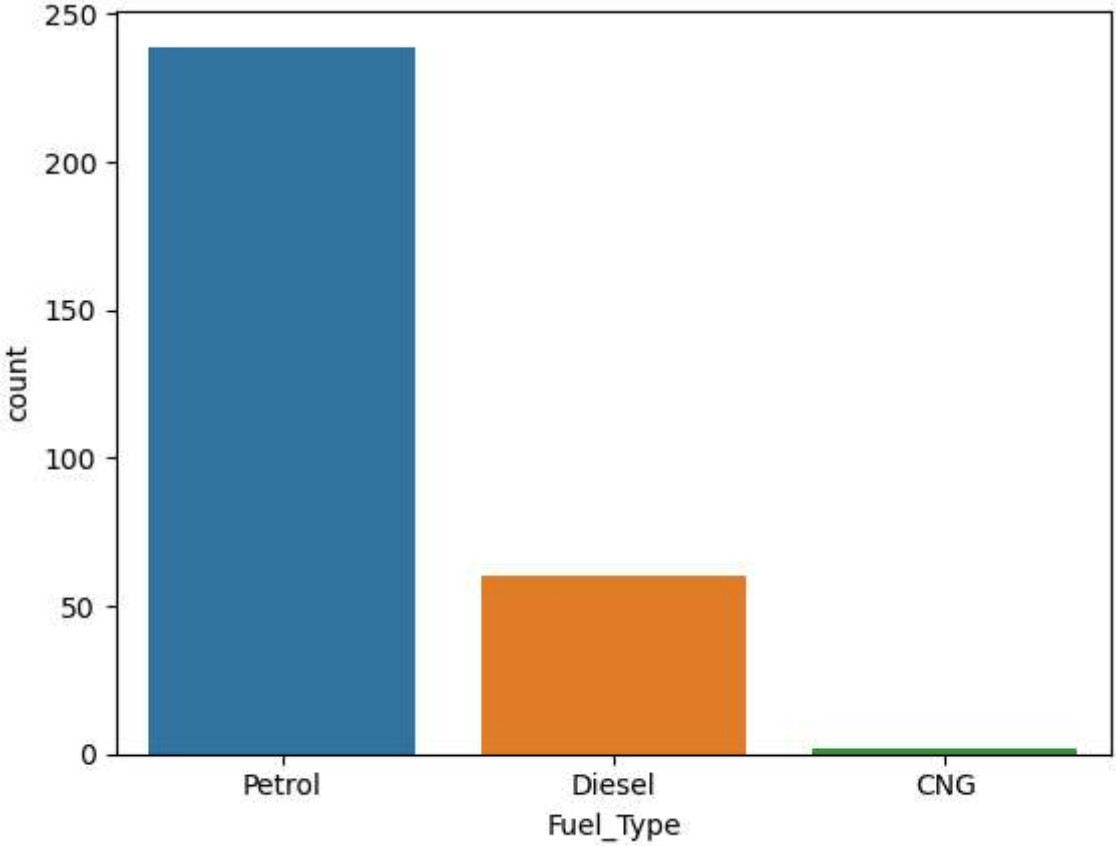
In [26]:
```
ax = sns.countplot(x="Transmission",data=car)
```



In [27]:
```
ax = sns.countplot(x="Driven_kms",data=car)
```

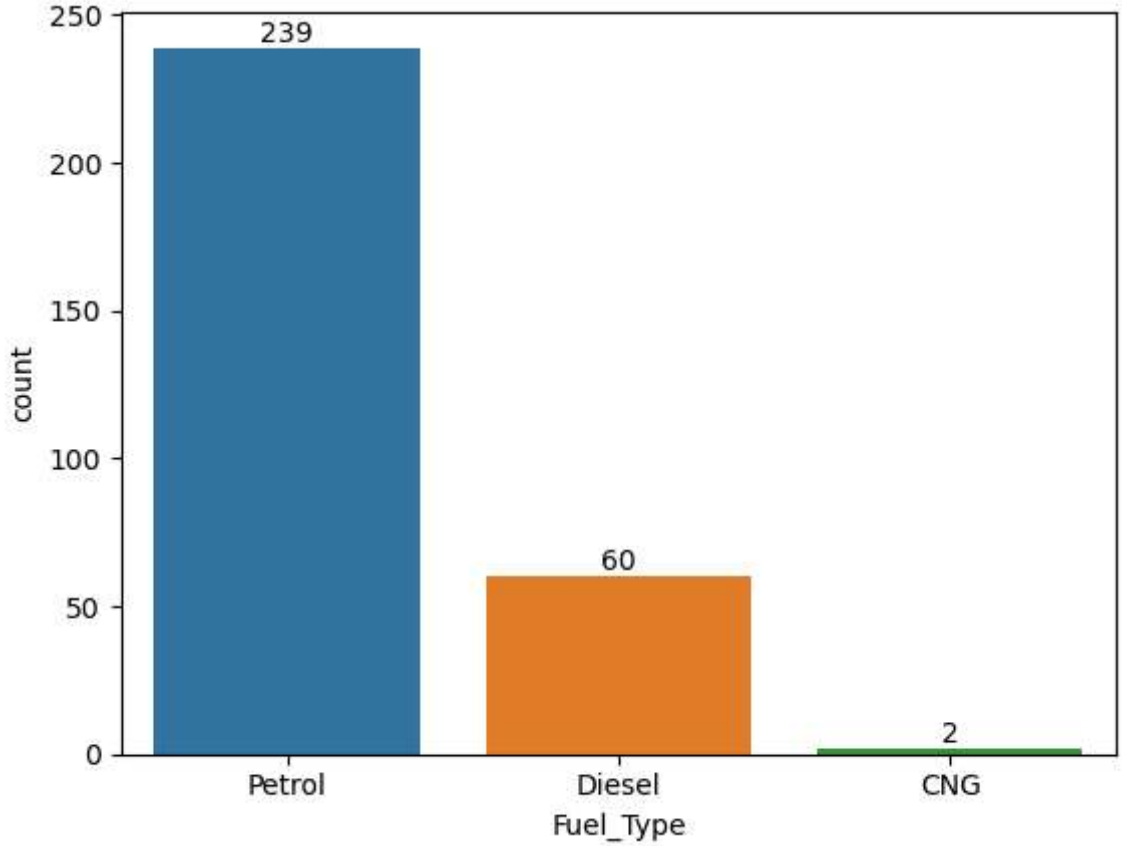


In [28]:
```
ax = sns.countplot(x="Fuel_Type",data=car)
```



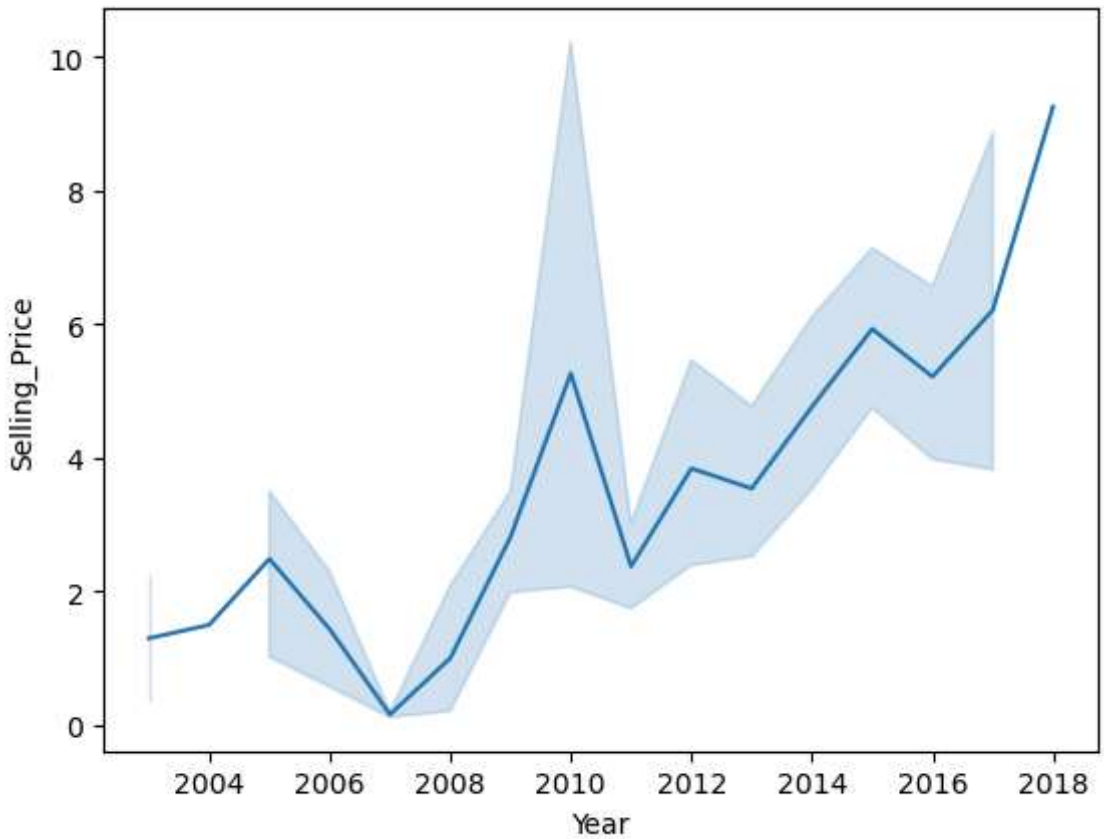
In [29]:
```
ax = sns.countplot(x="Fuel_Type",data=car)

for bars in ax.containers:
    ax.bar_label(bars)
```

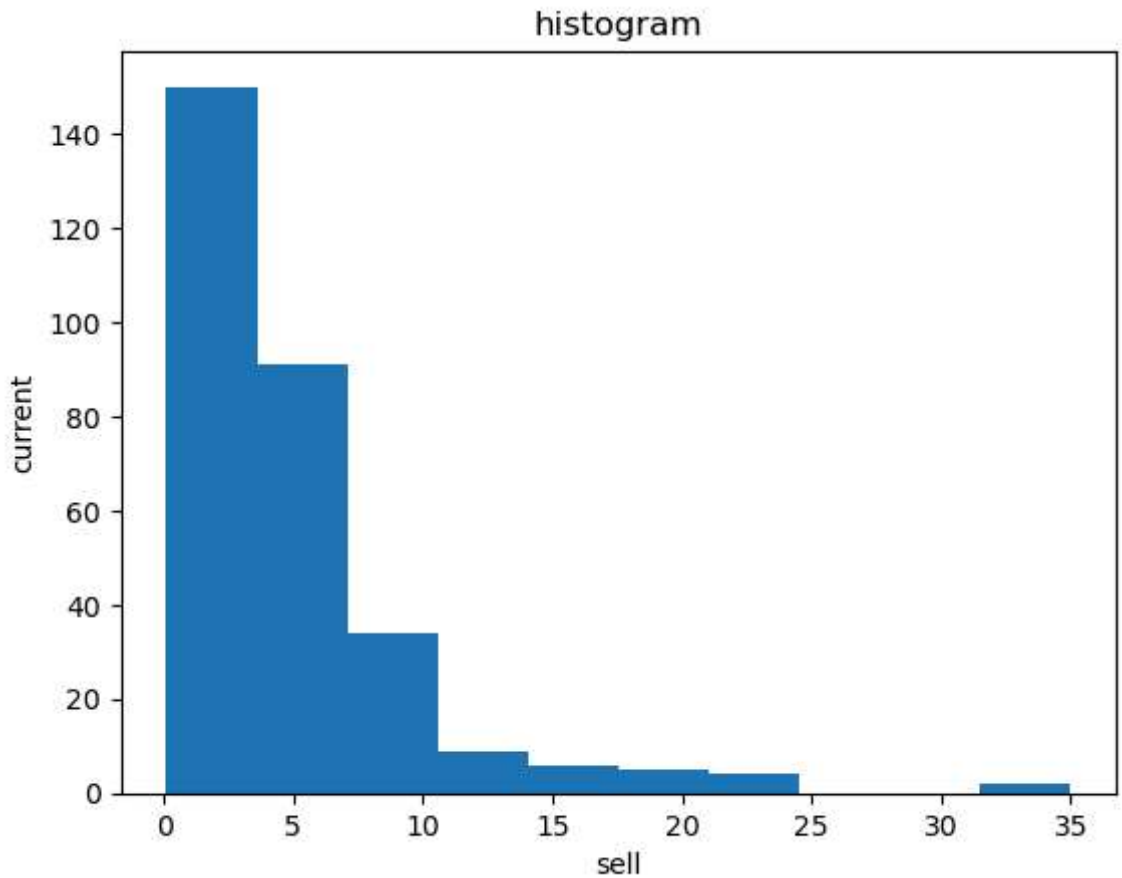


In [30]:
```
sns.lineplot(x="Year",y="Selling_Price",data=car)
```

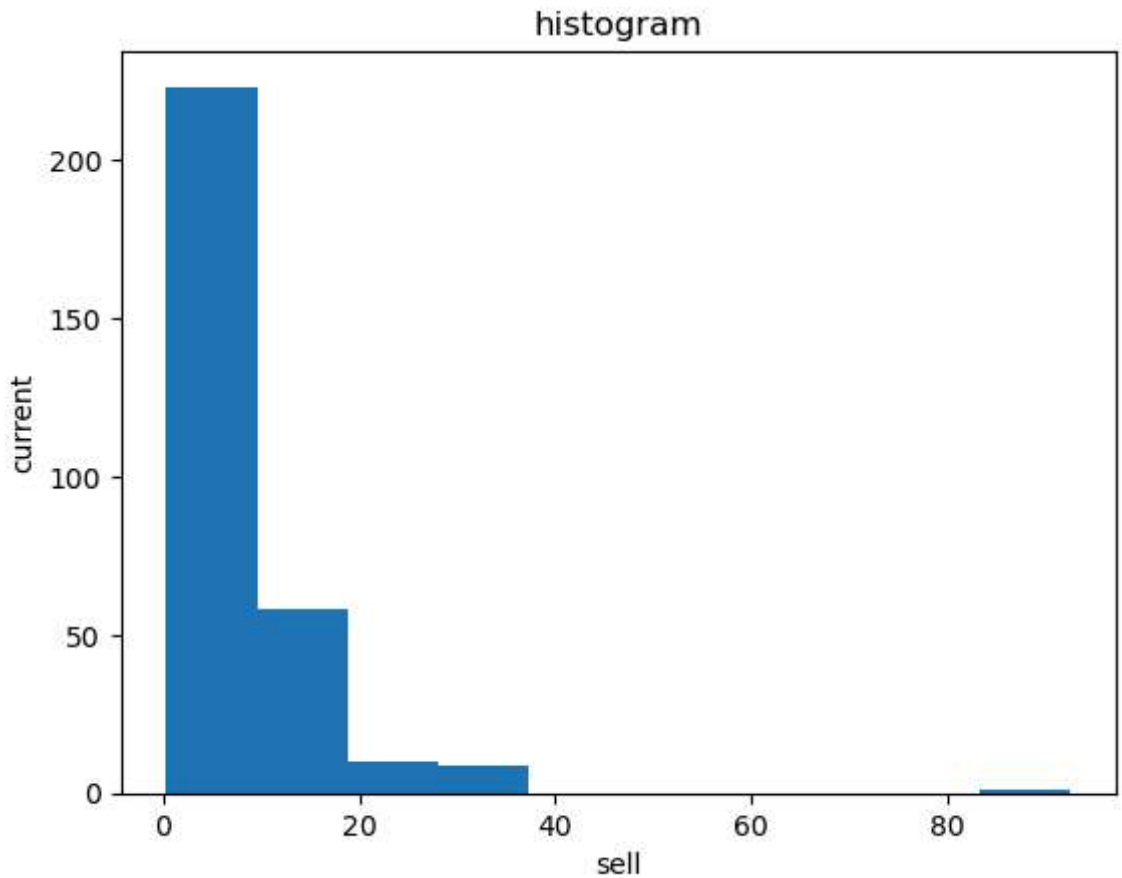
Out[30]: <AxesSubplot:xlabel='Year', ylabel='Selling_Price'>



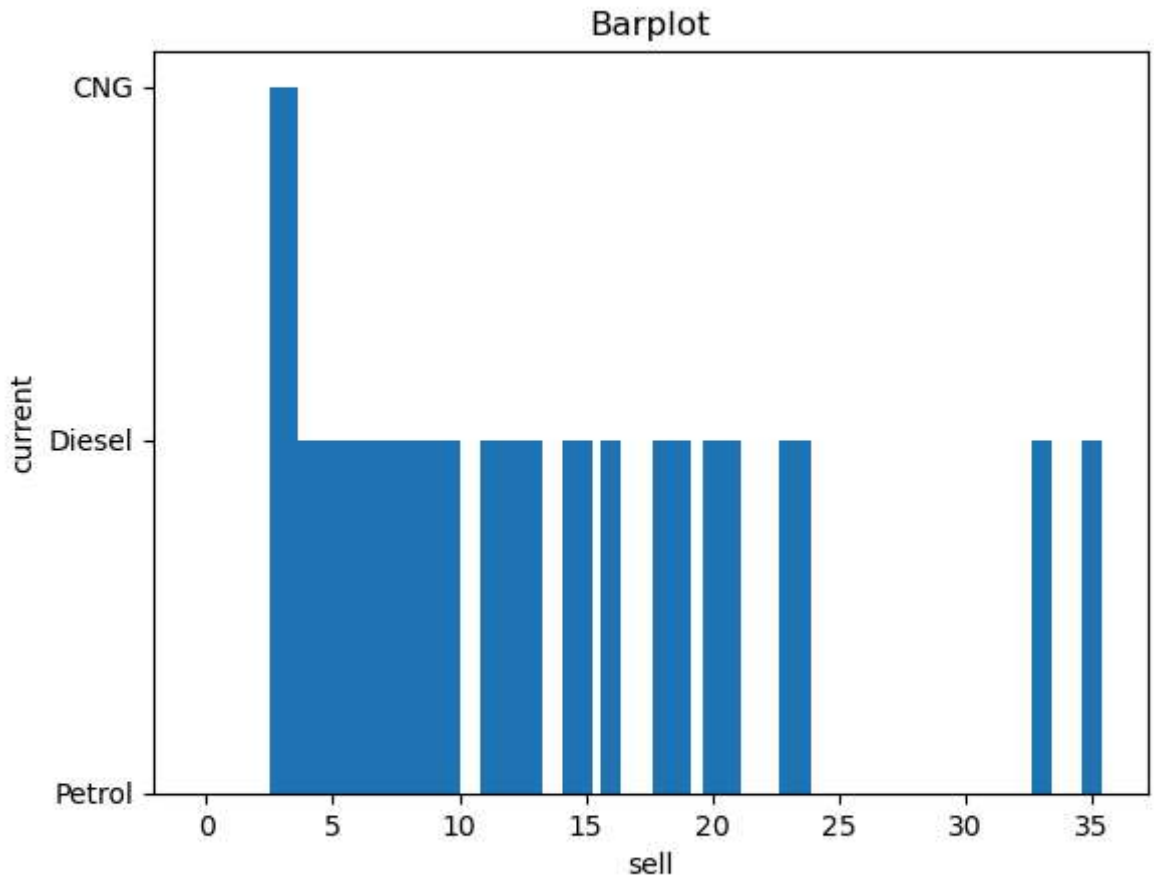
In [31]:
```
x = car["Selling_Price"]
plt.xlabel("sell")
plt.ylabel("current")
plt.title("histogram")
plt.hist(x)
plt.show()
```



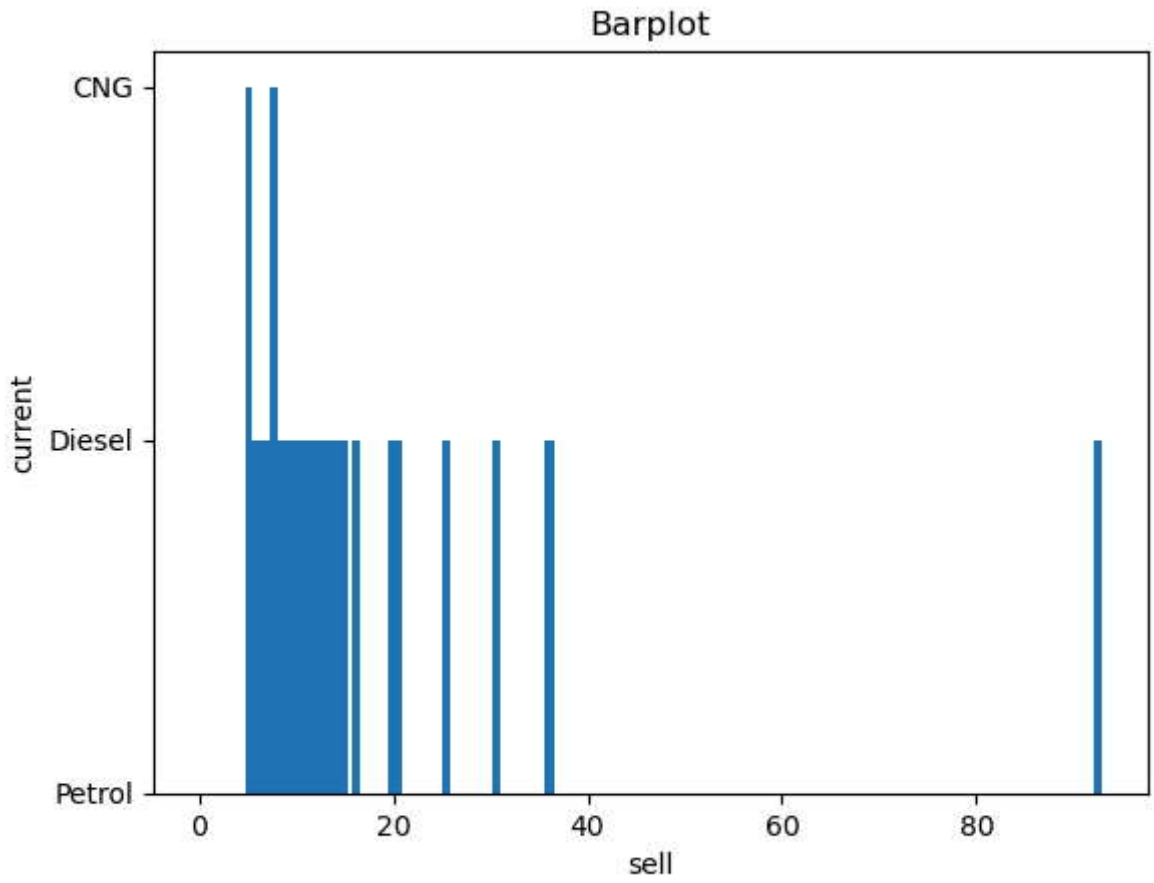
In [32]:
```
x = car["Present_Price"]
plt.xlabel("sell")
plt.ylabel("current")
plt.title("histogram")
plt.hist(x)
plt.show()
```



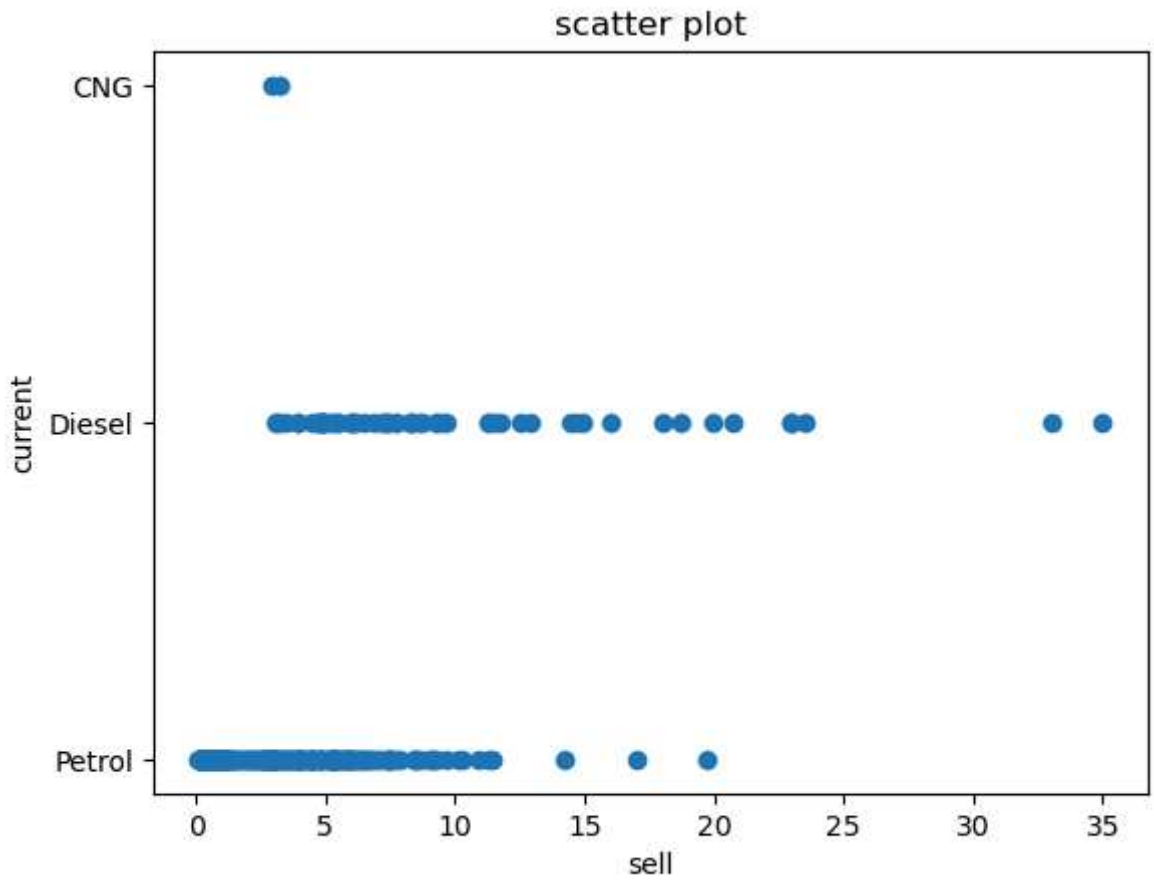
In [33]:
```
x = car["Selling_Price"]
y = car["Fuel_Type"]
plt.xlabel("sell")
plt.ylabel("current")
plt.title("Barplot")
plt.bar(x,y)
plt.show()
```



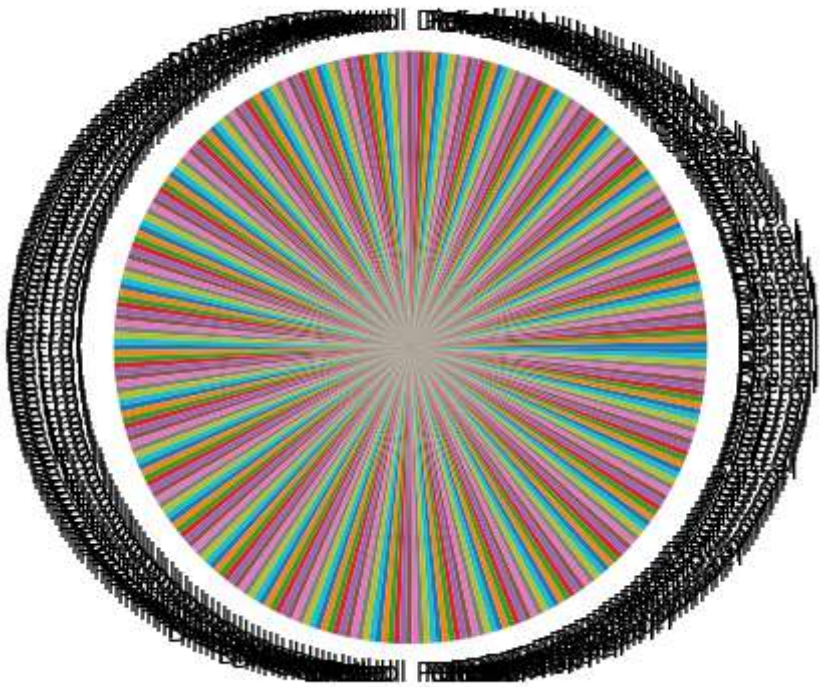
In [34]:
```
x = car["Present_Price"]
y = car["Fuel_Type"]
plt.xlabel("sell")
plt.ylabel("current")
plt.title("Barplot")
plt.bar(x,y)
plt.show()
```



In [35]:
```
x = car["Selling_Price"]
y = car["Fuel_Type"]
plt.xlabel("sell")
plt.ylabel("current")
plt.title("scatter plot")
plt.scatter(x,y)
plt.show()
```



In [36]:
```
x = car["Year"]
y = car["Fuel_Type"]
plt.pie(x,labels=y)
plt.show()
```



In [37]:
```
car.drop(['Car_Name','Year','Driven_kms','Transmission','Fuel_Type'],axis=1,
car
```

Out[37]:

	Selling_Price	Present_Price
0	3.35	5.59
1	4.75	9.54
2	7.25	9.85
3	2.85	4.15
4	4.60	6.87
...	...	...
296	9.50	11.60
297	4.00	5.90
298	3.35	11.00
299	11.50	12.50
300	5.30	5.90

301 rows × 2 columns

In [38]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Selling_Price  301 non-null    float64
 1   Present_Price  301 non-null    float64
dtypes: float64(2)
memory usage: 4.8 KB
```

In [39]:
```
x = car.drop(["Selling_Price"],axis=1)
y = car["Selling_Price"]
```

In [40]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

In [41]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [42]:
```
reg.fit(x_train, y_train)
```

Out[42]:
```
LinearRegression()
```

In [43]:
```
y_pred = reg.predict(x_test)
y_pred
```

Out[43]:
```
array([ 1.13706088,  5.46273972,  8.09724921,  3.6591516 ,  3.65426382,
        5.70712836,  1.60139929,  8.73265966,  1.02464211,  3.03351669,
        4.35321532,  1.0979587 ,  4.19191882,  3.65426382,  3.11660883,
        4.38742973,  9.96437838,  1.22992856,  4.63181836,  8.04348371,
        1.2529011 ,  1.60139929,  1.27880629,  7.51560426,  4.97396245,
       12.02701846,  4.19191882,  5.46273972,  9.96437838,  5.72667945,
       18.44466402,  5.70712836,  2.62783156,  1.32768402,  1.14683643,
        1.12239756,  1.58673597,  6.14214013,  1.72359361,  5.94174145,
        8.14612694,  1.11750979,  1.2592552 ,  3.65426382, 15.82970562,
        3.6591516 ,  3.75201928,  1.11750979,  6.53804972,  4.63670614,
        5.46273972,  1.2529011 ,  1.32768402,  4.18703105,  1.13217311,
        4.58294064,  4.52917514,  3.65426382,  4.17236773,  1.14683643,
        4.20169437,  1.33257179,  1.60139929,  1.44010279,  1.23481634,
        1.13217311,  4.80777818, 18.57663388,  1.27880629,  1.27880629,
        1.26414297,  7.51560426,  3.35610969,  4.70513495,  3.25835423,
        6.75311172,  7.44717544,  1.33257179,  1.27391852,  6.97794926,
        4.58294064,  5.46273972,  1.27880629,  2.89665905,  4.42164414,
        3.11660883,  7.56448198,  6.24478335,  4.72957382,  1.73825692,
        4.35321532])
```

In [44]:
```
car
```

Out[44]:

	Selling_Price	Present_Price
0	3.35	5.59
1	4.75	9.54
2	7.25	9.85
3	2.85	4.15
4	4.60	6.87
...	...	...
296	9.50	11.60
297	4.00	5.90
298	3.35	11.00
299	11.50	12.50
300	5.30	5.90

301 rows × 2 columns

In [45]:
```
car= pd.read_csv(r"C:\Users\786\Downloads\car data.csv")
car
```

Out[45]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transm
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	M
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	M
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	M
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	M
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	Diesel	Dealer	M
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	M
298	city	2009	3.35	11.00	87934	Petrol	Dealer	M
299	city	2017	11.50	12.50	9000	Diesel	Dealer	M
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 9 columns

In [46]:
```
car.head()
```

Out[46]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmiss
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Mar
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Mar
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Mar
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Mar
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Mar

In [47]:
```
car["Year"].describe()
```

Out[47]:
```
count     301.000000
mean     2013.627907
std         2.891554
min      2003.000000
25%      2012.000000
50%      2014.000000
75%      2016.000000
max      2018.000000
Name: Year, dtype: float64
```

In [48]:
```
car.nunique()
```

Out[48]:
```
Car_Name          98
Year              16
Selling_Price    156
Present_Price    148
Driven_kms       206
Fuel_Type          3
Selling_type       2
Transmission       2
Owner              3
dtype: int64
```

In [49]:
```
car.head()
```

Out[49]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmiss
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Mar
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Mar
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Mar
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Mar
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Mar

In [50]:
```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
```

In [51]:
```
car['Car_Name'] = le.fit_transform(car['Car_Name'])
car['Fuel_Type'] = le.fit_transform(car['Fuel_Type'])
car['Selling_type'] = le.fit_transform(car['Selling_type'])
car['Transmission'] = le.fit_transform(car['Transmission'])
car.head()
```

Out[51]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmiss
0	90	2014	3.35	5.59	27000	2	0	
1	93	2013	4.75	9.54	43000	1	0	
2	68	2017	7.25	9.85	6900	2	0	
3	96	2011	2.85	4.15	5200	2	0	
4	92	2014	4.60	6.87	42450	1	0	

In [52]:
```
car.describe()
```

Out[52]:

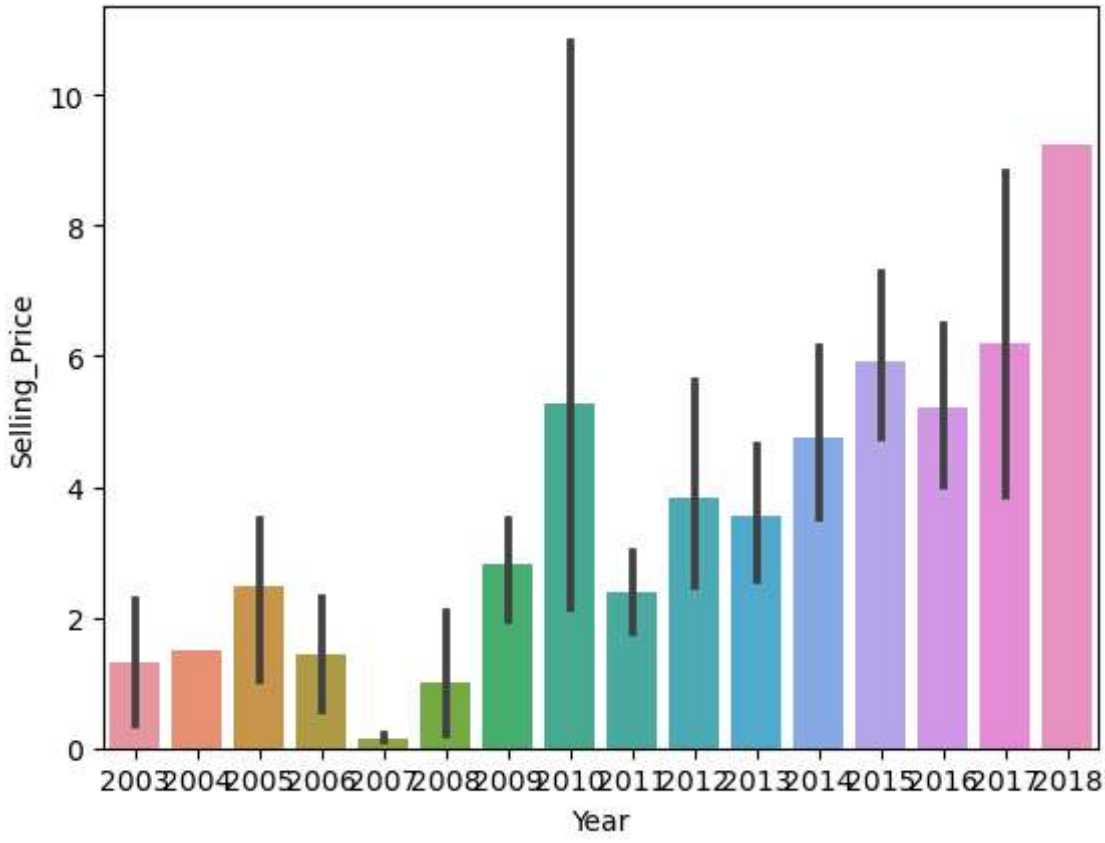
	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Sellin
count	301.000000	301.000000	301.000000	301.000000	301.000000	301.000000	301.0
mean	62.571429	2013.627907	4.661296	7.628472	36947.205980	1.787375	0.3
std	25.573535	2.891554	5.082812	8.642584	38886.883882	0.425801	0.4
min	0.000000	2003.000000	0.100000	0.320000	500.000000	0.000000	0.0
25%	47.000000	2012.000000	0.900000	1.200000	15000.000000	2.000000	0.0
50%	69.000000	2014.000000	3.600000	6.400000	32000.000000	2.000000	0.0
75%	82.000000	2016.000000	6.000000	9.900000	48767.000000	2.000000	1.0
max	97.000000	2018.000000	35.000000	92.600000	500000.000000	2.000000	1.0

In [53]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    int32
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64
 5   Fuel_Type      301 non-null    int32
 6   Selling_type   301 non-null    int32
 7   Transmission   301 non-null    int32
 8   Owner          301 non-null    int64
dtypes: float64(2), int32(4), int64(3)
memory usage: 16.6 KB
```

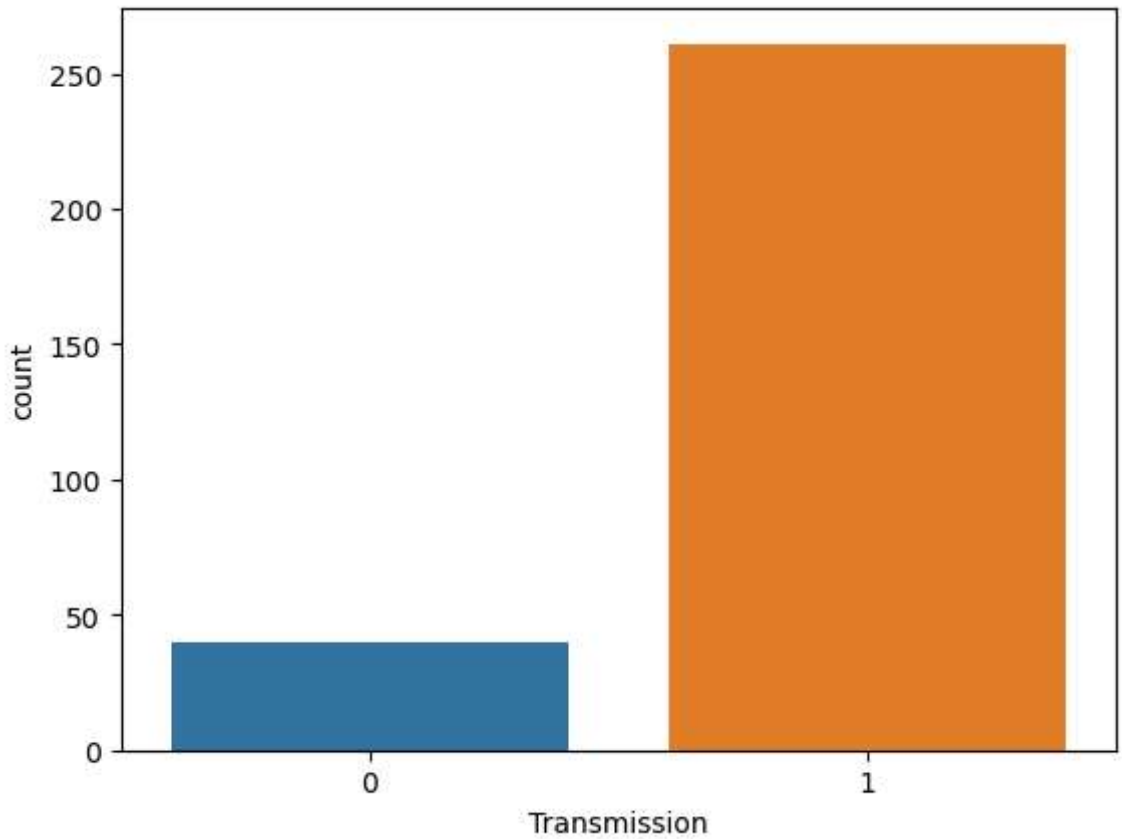
In [54]:
```
sns.barplot(x='Year',y='Selling_Price',data=car)
```

Out[54]:
```
<AxesSubplot:xlabel='Year', ylabel='Selling_Price'>
```



In [55]:
```
sns.countplot(x='Transmission',data=car)
```

Out[55]: <AxesSubplot:xlabel='Transmission', ylabel='count'>



In [56]:
```
x = car.drop(["Transmission"],axis=1)
y = car["Transmission"]
```

In [57]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

In [58]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
```

In [59]:
```
knn.fit(x_train,y_train)
```

Out[59]: KNeighborsClassifier()

In [60]:
```
y_pred = knn.predict(x_test)
y_pred
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classificati
on.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `
kurtosis`), the default behavior of `mode` typically preserves the axis it
acts along. In SciPy 1.11.0, this behavior will change: the default value
of `keepdims` will become False, the `axis` over which the statistic is ta
ken will be eliminated, and the value None will no longer be accepted. Set
`keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[60]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1])
```

In [61]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[61]:
```
array([[ 0, 15],
       [ 1, 75]], dtype=int64)
```

In [62]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        15
           1       0.83      0.99      0.90        76

    accuracy                           0.82        91
   macro avg       0.42      0.49      0.45        91
weighted avg       0.70      0.82      0.75        91
```

In [63]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[63]:
```
0.8241758241758241
```

In [ ]: