In [2]:
```
#reading the data
import pandas as  pd
car_info= pd.read_csv("car%20data.csv")
```

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

Out[3]:

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

In [4]:
```
#check data shape
car_info.shape
```

Out[4]:
```
(301, 9)
```

In [5]:
```
#check null value
car_info.isnull().sum()
```

Out[5]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [6]:
```
car_info.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   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_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 [7]:
```
#mathamatics infomation
car_info.describe()
```

Out[7]:

	Year	Selling_Price	Present_Price	Kms_Driven	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	8.644115	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 [8]:
```
#remove case sensitive (capital/small)
car_info.columns = [cols.lower() for cols in car_info.columns]
```

In [9]:
```
car_info.head()
```

Out[9]:

	car_name	year	selling_price	present_price	kms_driven	fuel_type	seller_type	trai
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	

In [10]:
```
#none mathamatical value (catagical data)
print(car_info["fuel_type"].value_counts())
print(car_info["seller_type"].value_counts())
print(car_info["transmission"].value_counts())
```

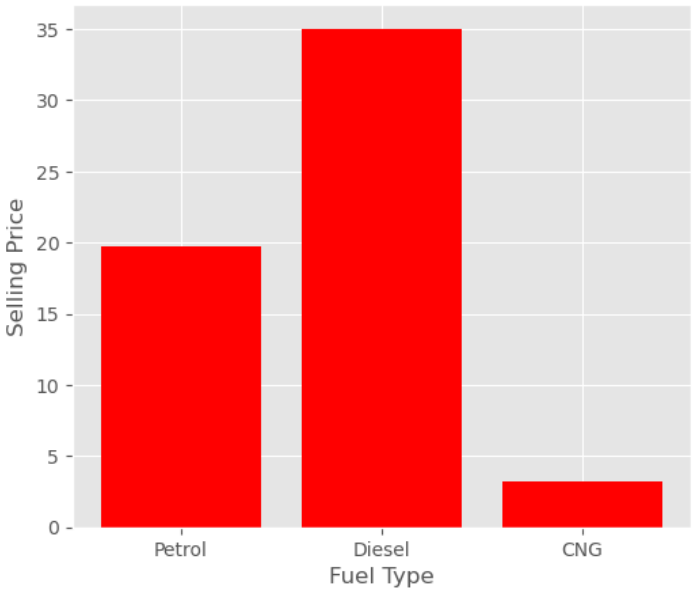
```
fuel_type
Petrol    239
Diesel     60
CNG         2
Name: count, dtype: int64
seller_type
Dealer        195
Individual    106
Name: count, dtype: int64
transmission
Manual       261
Automatic     40
Name: count, dtype: int64
```

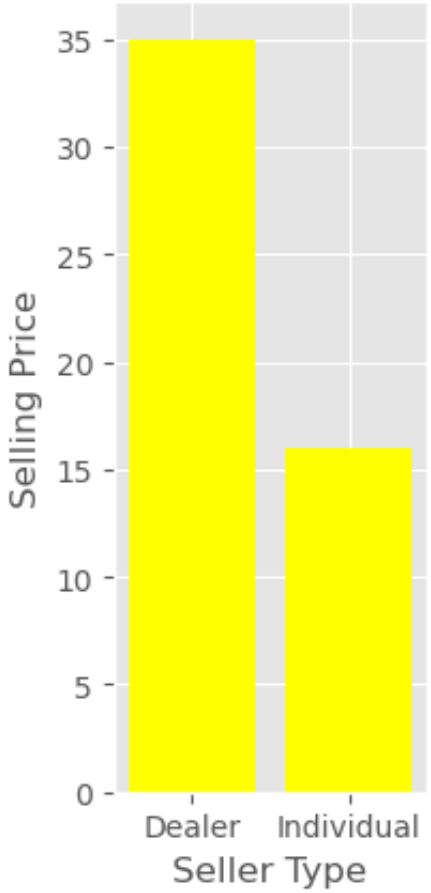
In [11]:
```
#create catagrical data graph
import matplotlib.pyplot as plt
from matplotlib import style
style.available
```

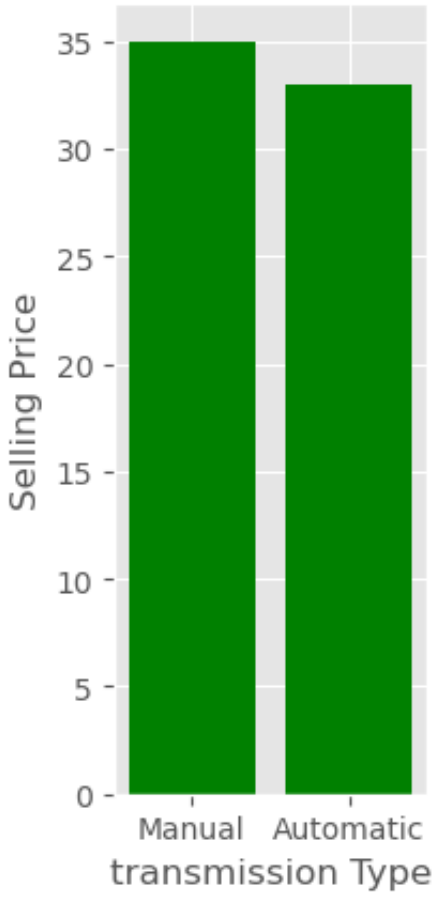
Out[11]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

In [12]:
```
fuel_info = car_info["fuel_type"]
seller_info = car_info["seller_type"]
transmission_info = car_info["transmission"]
selling_price = car_info ["selling_price"]
```

In [13]:
```
style.use("ggplot")
chart = plt.figure(figsize=(20, 5))
chart.suptitle("Categorical chart info")

plt.subplot(1, 3, 1)
plt.bar(fuel_info, selling_price, color="Red")
plt.xlabel("Fuel Type")
plt.ylabel("Selling Price")
plt.show()


plt.subplot(1, 3, 2)
plt.bar(seller_info, selling_price, color="yellow")
plt.xlabel("Seller Type")
plt.ylabel("Selling Price")
plt.show()



plt.subplot(1, 3, 3)
plt.bar(transmission_info, selling_price, color="green")
plt.xlabel("transmission Type")
plt.ylabel("Selling Price")
plt.show()
```




35
30
25
20
15
10
5
0
Selling Price
Dealer
Individual
Seller Type


35
30
25
20
15
10
5
0
Selling Price
Manual
Automatic
transmission Type

In [14]:
```
petrol_data = car_info.groupby("fuel_type").get_group("Petrol")
petrol_data.describe()
```

Out[14]:

	year	selling_price	present_price	kms_driven	owner
count	239.000000	239.000000	239.000000	239.000000	239.000000
mean	2013.539749	3.264184	5.583556	33528.937238	0.050209
std	3.042674	3.135537	5.290685	40308.984886	0.270368
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.600000	0.940000	13850.000000	0.000000
50%	2014.000000	2.650000	4.600000	25870.000000	0.000000
75%	2016.000000	5.200000	7.980000	44271.000000	0.000000
max	2017.000000	19.750000	23.730000	500000.000000	3.000000

In [15]:
```
Diesel_data = car_info.groupby("fuel_type").get_group("Diesel")
Diesel_data.describe()
```

Out[15]:

	year	selling_price	present_price	kms_driven	owner
count	60.000000	60.000000	60.000000	60.000000	60.000000
mean	2014.000000	10.278500	15.814500	50369.916667	0.016667
std	2.201694	7.185159	13.484289	30021.446979	0.129099
min	2005.000000	3.100000	5.700000	2071.000000	0.000000
25%	2013.000000	5.137500	8.912500	38750.000000	0.000000
50%	2014.000000	7.750000	10.585000	45000.000000	0.000000
75%	2015.000000	12.600000	17.010000	59250.000000	0.000000
max	2018.000000	35.000000	92.600000	197176.000000	1.000000

In [16]:
```
cng_data = car_info.groupby("fuel_type").get_group("CNG")
cng_data.describe()
```

Out[16]:

	year	selling_price	present_price	kms_driven	owner
count	2.000000	2.000000	2.000000	2.000000	2.0
mean	2013.000000	3.100000	6.415000	42749.000000	0.0
std	2.828427	0.212132	1.873833	10251.634114	0.0
min	2011.000000	2.950000	5.090000	35500.000000	0.0
25%	2012.000000	3.025000	5.752500	39124.500000	0.0
50%	2013.000000	3.100000	6.415000	42749.000000	0.0
75%	2014.000000	3.175000	7.077500	46373.500000	0.0
max	2015.000000	3.250000	7.740000	49998.000000	0.0

convert letter to numerial value

(assign the numbers eg: Petrol-1,diseal-2,.....)

In [17]:
```
print(car_info["fuel_type"], car_info["seller_type"], car_info["transmiss
```

```
0      Petrol
1      Diesel
2      Petrol
3      Petrol
4      Diesel
        ...
296    Diesel
297    Petrol
298    Petrol
299    Diesel
300    Petrol
Name: fuel_type, Length: 301, dtype: object 0      Dealer
1      Dealer
2      Dealer
3      Dealer
4      Dealer
        ...
296    Dealer
297    Dealer
298    Dealer
299    Dealer
300    Dealer
Name: seller_type, Length: 301, dtype: object 0      Manual
1      Manual
2      Manual
3      Manual
4      Manual
        ...
296    Manual
297    Manual
298    Manual
299    Manual
300    Manual
Name: transmission, Length: 301, dtype: object
```

In [18]:

```
car_info.replace({"fuel_type":{"Petrol":0, "Diesel":1, "CNG":2}}, inplace
car_info.replace({"seller_type":{"Dealer":1, "Individual":2}}, inplace =T
car_info.replace({"transmission":{"Manual":1, "Automatic":2}}, inplace =T

print(car_info["fuel_type"], car_info["seller_type"], car_info["transmiss
```

```
0      0
1      1
2      0
3      0
4      1
      ..
296    1
297    0
298    0
299    1
300    0
Name: fuel_type, Length: 301, dtype: int64 0      1
1      1
2      1
3      1
4      1
      ..
296    1
297    1
298    1
299    1
300    1
Name: seller_type, Length: 301, dtype: int64 0      1
1      1
2      1
3      1
4      1
      ..
296    1
297    1
298    1
299    1
300    1
Name: transmission, Length: 301, dtype: int64
```

In [19]:
```
car_info.describe()
```

Out[19]:

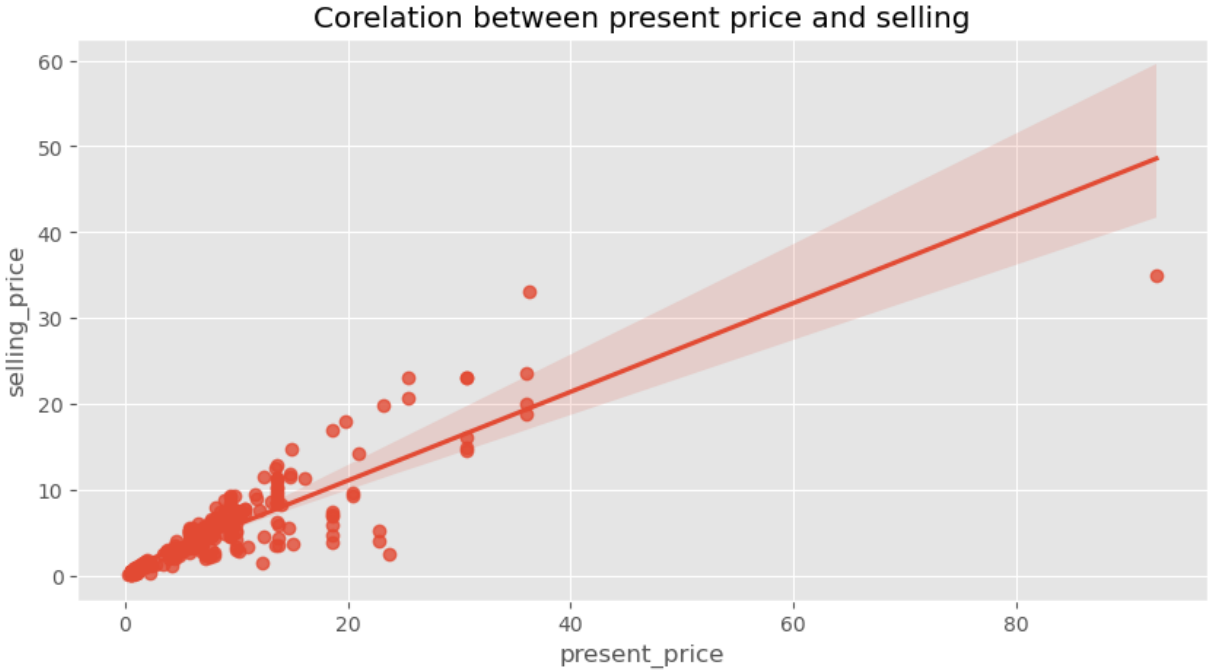
	year	selling_price	present_price	kms_driven	fuel_type	seller_typ
count	301.000000	301.000000	301.000000	301.000000	301.000000	301.00000
mean	2013.627907	4.661296	7.628472	36947.205980	0.212625	1.35215
std	2.891554	5.082812	8.644115	38886.883882	0.425801	0.47843
min	2003.000000	0.100000	0.320000	500.000000	0.000000	1.00000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000	1.00000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000	1.00000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000	2.00000
max	2018.000000	35.000000	92.600000	500000.000000	2.000000	2.00000

In [ ]:
```

```

In [ ]:

In [20]:
```
import seaborn as sea
fig = plt.figure(figsize=(10,5))
plt.title("Corelation between present price and selling")
sea.regplot(x = "present_price" , y = "selling_price" ,data = car_info)
```

Out[20]:
```
<Axes: title={'center': 'Corelation between present price and selling'},
xlabel='present_price', ylabel='selling_price'>
```



In [21]:
```
input_data = car_info.drop(columns = ["car_name" , "selling_price"])
out_data = car_info ["selling_price"]
```

In [22]:
```
print (input_data.shape)
print (out_data.shape)
```

```
(301, 7)
(301,)
```

In [23]:
```
print (input_data)
print (out_data)
```

```
     year  present_price  kms_driven  fuel_type  seller_type  transmissio
n  \
0    2014           5.59       27000          0            1
1
1    2013           9.54       43000          1            1
1
2    2017           9.85        6900          0            1
1
3    2011           4.15        5200          0            1
1
4    2014           6.87       42450          1            1
1
..    ...            ...         ...        ...          ...
...
296  2016          11.60       33988          1            1
1
297  2015           5.90       60000          0            1
1
298  2009          11.00       87934          0            1
1
299  2017          12.50        9000          1            1
1
300  2016           5.90        5464          0            1
1

     owner
0        0
1        0
2        0
3        0
4        0
..     ...
296      0
297      0
298      0
299      0
300      0

[301 rows x 7 columns]
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: selling_price, Length: 301, dtype: float64
```

change number simliar range of number (StandardScaler)

In [24]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
input_data = scaler.fit_transform(input_data)
print (input_data)
```

```
[[ 0.128897   -0.23621461 -0.25622446 ... -0.73728539 -0.39148015
  -0.17450057]
 [-0.21751369  0.22150462  0.1559105  ... -0.73728539 -0.39148015
  -0.17450057]
 [ 1.16812909  0.25742689 -0.77396901 ... -0.73728539 -0.39148015
  -0.17450057]
 ...
 [-1.60315648  0.39068691  1.31334003 ... -0.73728539 -0.39148015
  -0.17450057]
 [ 1.16812909  0.56450434 -0.7198763  ... -0.73728539 -0.39148015
  -0.17450057]
 [ 0.8217184  -0.20029235 -0.81095812 ... -0.73728539 -0.39148015
  -0.17450057]]
```

Model test (allocation datas test,train)

In [25]:
```
from sklearn.model_selection import train_test_split
input_data_train , input_data_test , output_data_train , output_data_test
```

In [26]:
```
print(input_data_train.shape)
print(output_data_train.shape)
print(input_data_test.shape)
print(output_data_test.shape)
```

```
(210, 7)
(210,)
(91, 7)
(91,)
```

In [27]:
```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
model = LinearRegression()
model.fit(input_data_train , output_data_train)
```

Out[27]:

▾ LinearRegression

```
LinearRegression()
```

predict the data

In [28]:
```
predited_sellingprice = model.predict(input_data_test)
```

commpare predit value and assign value

In [29]:
```
from sklearn.metrics import mean_absolute_error , mean_squared_error , r2
```

In [30]:
```
print("Mean Absolute Error:" , (metrics.mean_absolute_error(predited_sell
print("Mean Squared Error:" , (metrics.mean_squared_error(predited_sellin
print("R2 Score:" , (metrics.r2_score(predited_sellingprice, output_data_
```

```
Mean Absolute Error: 1.4492978940813612
Mean Squared Error: 5.635334450464463
R2 Score: 0.8508873413267803
```

In [35]:
```
from sklearn.linear_model import LinearRegression
import joblib
car_model = LinearRegression()
car_model.fit(input_data,out_data)
joblib.dump(car_model , "Car_Selling_Price-Identifier")
```

Out[35]:
```
['Car_Selling_Price-Identifier']
```

In [ ]:

In [ ]: