CAR SALES ANALYSIS IN UKRAINE ¶

Do the EDA and find key metrics

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

import pandas_profiling
```

In [2]:
```
car_s=pd.read_csv(r"D:\Car_Sales.csv")
```

In [3]:
```
car_s
```

Out[3]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9576 rows × 10 columns

In [4]:
```
1 car_s.describe()
```

Out[4]:

	price	mileage	engV	year
count	9576.000000	9576.000000	9142.000000	9576.000000
mean	15633.317316	138.862364	2.646344	2006.605994
std	24106.523436	98.629754	5.927699	7.067924
min	0.000000	0.000000	0.100000	1953.000000
25%	4999.000000	70.000000	1.600000	2004.000000
50%	9200.000000	128.000000	2.000000	2008.000000
75%	16700.000000	194.000000	2.500000	2012.000000
max	547800.000000	999.000000	99.990000	2016.000000

In [5]:
```
1 car_s.shape
```

Out[5]: (9576, 10)

In [6]:
```
1 car_s.info
```

Out[6]:
```
<bound method DataFrame.info of                car    price       body  mileag
e  engV engType registration  \
0              Ford  15500.0  crossover       68   2.5     Gas          yes
1     Mercedes-Benz  20500.0      sedan      173   1.8     Gas          yes
2     Mercedes-Benz  35000.0      other      135   5.5  Petrol          yes
3     Mercedes-Benz  17800.0        van      162   1.8  Diesel          yes
4     Mercedes-Benz  33000.0      vagon       91   NaN   Other          yes
...             ...      ...        ...      ...   ...     ...          ...
9571        Hyundai  14500.0  crossover      140   2.0     Gas          yes
9572     Volkswagen   2200.0      vagon      150   1.6  Petrol          yes
9573  Mercedes-Benz  18500.0  crossover      180   3.5  Petrol          yes
9574          Lexus  16999.0      sedan      150   3.5     Gas          yes
9575           Audi  22500.0      other       71   3.6  Petrol          yes

      year      model  drive
0     2010       Kuga   full
1     2011    E-Class   rear
2     2008     CL 550   rear
3     2012      B 180  front
4     2013    E-Class    NaN
...    ...        ...    ...
9571  2011     Tucson  front
9572  1986  Passat B2  front
9573  2008     ML 350   full
9574  2008     ES 350  front
9575  2007         Q7   full

[9576 rows x 10 columns]>
```

In [7]:
```
1 car_s.head(10)
```

Out[7]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN
5	Nissan	16600.0	crossover	83	2.0	Petrol	yes	2013	X-Trail	full
6	Honda	6500.0	sedan	199	2.0	Petrol	yes	2003	Accord	front
7	Renault	10500.0	vagon	185	1.5	Diesel	yes	2011	Megane	front
8	Mercedes-Benz	21500.0	sedan	146	1.8	Gas	yes	2012	E-Class	rear
9	Mercedes-Benz	22700.0	sedan	125	2.2	Diesel	yes	2010	E-Class	rear

In [8]:
```
1 from pandas_profiling import ProfileReport
```

In [9]:
```
1 car_sales=ProfileReport(car_s,title="Car_Sales")
```

In [10]:
```
1 car_sales
```

Summarize dataset: 100% 36/36 [00:06<00:00, 5.68it/s, Completed]

Generate report structure: 100% 1/1 [00:07<00:00, 7.59s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.31s/it]

Overview

Dataset statistics

Number of variables	10
Number of observations	9576
Missing cells	945
Missing cells (%)	1.0%
Duplicate rows	88
Duplicate rows (%)	0.9%
Total size in memory	748.2 KiB
Average record size in memory	80.0 B

Variable types

Categorical	5
Numeric	4
Boolean	1

Alerts

Dataset has 88 (0.9%) duplicate rows	**Duplicates**
`car` has a high cardinality: 87 distinct values	**High cardinality**

Out[10]:

In [11]:
```
1 car_s.isnull().sum()
```

Out[11]:
```
car              0
price            0
body             0
mileage          0
engV           434
engType          0
registration     0
year             0
model            0
drive          511
dtype: int64
```

In [12]:
```
1 car_s["engV"]
```

Out[12]:
```
0       2.5
1       1.8
2       5.5
3       1.8
4       NaN
       ...
9571    2.0
9572    1.6
9573    3.5
9574    3.5
9575    3.6
Name: engV, Length: 9576, dtype: float64
```

In [13]:
```
1 car_s.sort_values(by=['price'],ascending=False)
```

Out[13]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
7621	Bentley	547800.0	sedan	0	6.75	Petrol	yes	2016	Mulsanne	rear
7914	Bentley	499999.0	crossover	0	6.00	Petrol	yes	2016	Bentayga	full
1611	Bentley	499999.0	crossover	0	6.00	Petrol	yes	2016	Bentayga	full
4134	Bentley	449999.0	crossover	0	6.00	Petrol	yes	2016	Bentayga	full
4325	Mercedes-Benz	300000.0	sedan	68	6.00	Petrol	yes	2011	S 600	NaN
...	...	...	...	...	...	...	...	...	...	...
70	Mercedes-Benz	0.0	crossover	0	3.00	Diesel	yes	2016	GLE-Class	full
158	Land Rover	0.0	crossover	45	3.00	Petrol	yes	2014	Range Rover Sport	full
4786	Mercedes-Benz	0.0	crossover	27	3.00	Diesel	yes	2015	G 350	full
4784	Rolls-Royce	0.0	sedan	22	6.75	Other	yes	2008	Phantom	rear
215	Mercedes-Benz	0.0	sedan	62	3.00	Diesel	yes	2013	S 350	rear

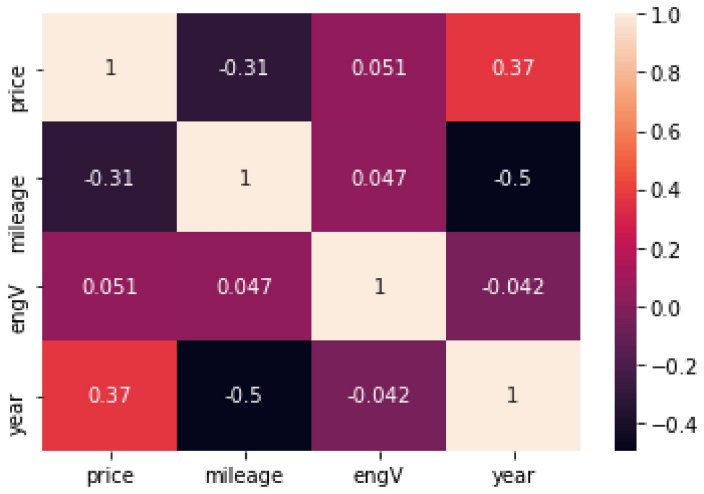
9576 rows × 10 columns

In [14]:
```
1 car_s.value_counts(["car"]).count()
```

Out[14]: 87

In [15]:
```
#Corelation between attributes
sns.heatmap(data=car_s.corr(),annot=True)
```

Out[15]: <AxesSubplot:>



In [16]:
```
car_s
```

Out[16]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9576 rows × 10 columns

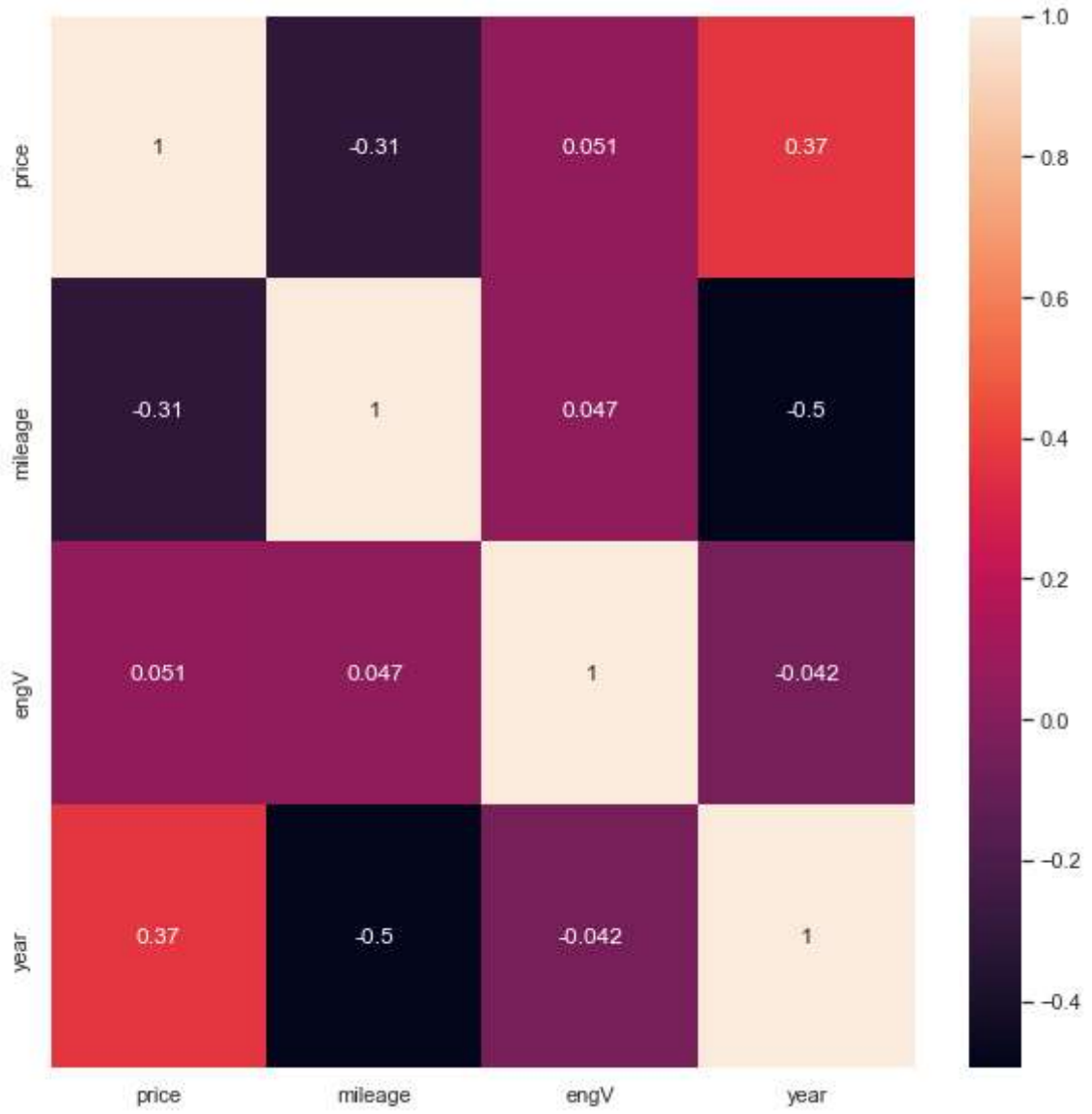
In [17]:
```
# maximum selling car
car_s.groupby("car")["price"].count().sort_values(ascending=False).head()
```

Out[17]:
```
car
Volkswagen       936
Mercedes-Benz    921
BMW              694
Toyota           541
VAZ              489
Name: price, dtype: int64
```

it has been obsserved that the maximum sessling cars is Volkswagen,Mercedez-Benz & BMW

In [18]:
```
# To see the corelation between the parameters.
sns.set()
plt.subplots(figsize=(10,10))
sns.heatmap(car_s.corr(),annot=True)
```

Out[18]: <AxesSubplot:>



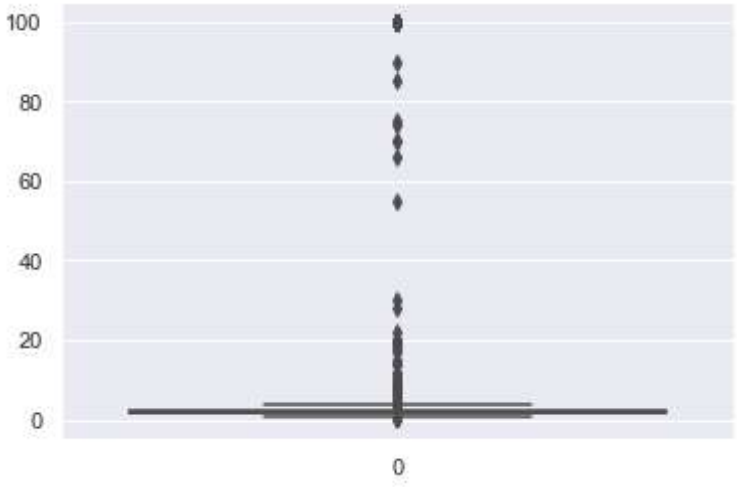
In [19]:
```
# To find the Null Values

car_s.isnull().sum()
```

Out[19]:
```
car              0
price            0
body             0
mileage          0
engV           434
engType          0
registration     0
year             0
model            0
drive          511
dtype: int64
```

In [20]:
```
# We can see the eng V & drive has null values. To fill the values of null c
#To use mean or median, we want to see if the distribution has outlier or no
# To check the outlier we can use the box plot
```

In [21]:
```
#Find the outlier of engV
sns.boxplot(data=car_s.engV)
```

Out[21]: <AxesSubplot:>



In [22]:
```
car_s.value_counts("registration")
```

Out[22]:
```
registration
yes    9015
no      561
dtype: int64
```

In [23]:
```
#To find the duplicate rows
car_s.loc[car_s.duplicated(),:]
```

Out[23]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
18	Nissan	16600.0	crossover	83	2.0	Petrol	yes	2013	X-Trail	full
42	Mercedes-Benz	20400.0	sedan	190	1.8	Gas	yes	2011	E-Class	rear
70	Mercedes-Benz	0.0	crossover	0	3.0	Diesel	yes	2016	GLE-Class	full
86	Toyota	103999.0	crossover	0	4.5	Diesel	yes	2016	Land Cruiser 200	full
98	Mercedes-Benz	20400.0	sedan	190	1.8	Gas	yes	2011	E-Class	rear
...	...	...	...	...	...	...	...	...	...	...
9156	Volkswagen	15700.0	sedan	110	1.8	Petrol	yes	2011	Passat B7	front
9163	Mercedes-Benz	20500.0	sedan	222	5.5	Petrol	yes	2006	S 500	rear
9164	VAZ	3900.0	hatch	121	1.4	Petrol	yes	2008	1119	front
9169	Hyundai	12900.0	crossover	49	2.7	Petrol	yes	2008	Tucson	full
9477	BMW	77777.0	sedan	8	4.4	Petrol	yes	2014	750	full

113 rows × 10 columns

In [24]:
```
car_s.duplicated().sum()
```

Out[24]: 113

In [25]:
```
#To remove the null value "drop_duplicates"
```

In [26]:
```
car_s.drop_duplicates(inplace=True)
```

In [27]:
```
1 car_s.value_counts("engV")
```

Out[27]:
```
engV
2.00    1525
1.60    1224
1.50     689
3.00     672
1.80     575
        ...
4.67       1
1.12       1
5.40       1
0.65       1
0.10       1
Length: 117, dtype: int64
```

In [28]:
```
1 car_s.loc[car_s.duplicated(),:]
```

Out[28]:

car	price	body	mileage	engV	engType	registration	year	model	drive

There is no duplicates in the data

In [29]:
```
1 car_s
```

Out[29]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9463 rows × 10 columns

In [30]:
```
1 car_s_copy=car_s.copy()
```

In [31]:
```
1 car_s_copy.head()
```

Out[31]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN

In [32]:
```
1 car_s_copy.loc[car_s_copy.duplicated()]
```

Out[32]:

car	price	body	mileage	engV	engType	registration	year	model	drive

In [33]:
```
1 from pandas_profiling import ProfileReport
```

In [34]:

```
1 ProfileReport(car_s_copy,title="car_s_copy")
```

Summarize dataset: 100% 36/36 [00:06<00:00, 5.47it/s, Completed]

Generate report structure: 100% 1/1 [02:58<00:00, 178.52s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.07s/it]

Overview

Dataset statistics

Number of variables	10
Number of observations	9463
Missing cells	944
Missing cells (%)	1.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.0 MiB
Average record size in memory	115.9 B

Variable types

Categorical	5
Numeric	4
Boolean	1

Alerts

`car` has a high cardinality: 87 distinct values	High cardinality
`model` has a high cardinality: 863 distinct values	High cardinality

Out[34]:

In [35]:
```
1 car_s_copy['drive'].mode()
```

Out[35]:
```
0    front
Name: drive, dtype: object
```

In [36]:
```
car_s_copy['drive']=car_s_copy['drive'].fillna('front')
car_s_copy['drive'].isnull().sum()
```

Out[36]:
```
0
```

In [37]:
```
car_s_copy['price'].isnull().sum()
```

Out[37]:
```
0
```

In [38]:
```
car_s_copy['engV']=car_s_copy.groupby(['car','body'])['engV'].transform(lamb
car_s_copy['engV'].isnull().sum()
```

Out[38]:
```
10
```

In [39]:
```
car_s_copy.isnull().sum()
```

Out[39]:
```
car              0
price            0
body             0
mileage          0
engV            10
engType          0
registration     0
year             0
model            0
drive            0
dtype: int64
```

In [40]:
```
car_s_copy[car_s_copy.engV.isnull()]
```

Out[40]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
319	Tesla	58000.0	hatch	52	NaN	Other	yes	2013	Model S	front
1437	Tesla	178500.0	crossover	0	NaN	Other	yes	2016	Model X	full
2486	Tesla	185000.0	crossover	1	NaN	Other	yes	2016	Model X	full
5084	GAZ	0.0	crossover	1	NaN	Petrol	yes	1963	69	full
6773	UAZ	3000.0	other	1	NaN	Other	yes	1985	3303	full
8569	Tesla	176900.0	crossover	0	NaN	Other	yes	2016	Model X	full
8824	Fisker	0.0	other	100	NaN	Other	yes	2001	Karma	front
8905	Changan	6028.0	crossover	101	NaN	Other	yes	2005	Ideal	front
9360	Barkas	5500.0	van	80	NaN	Petrol	yes	2015	B1000	front
9566	UAZ	850.0	van	255	NaN	Other	yes	1981	3962	front

In [41]:
```
car_s_copy['engV']
```

Out[41]:
```
0       2.5
1       1.8
2       5.5
3       1.8
4       2.3
       ...
9571    2.0
9572    1.6
9573    3.5
9574    3.5
9575    3.6
Name: engV, Length: 9463, dtype: float64
```

In [43]:
```
#Drop Null Values
car_s_copy.dropna(subset=['engV'],inplace=True)
```

In [44]:
```
car_s_copy.isnull().sum()
```

Out[44]:
```
car             0
price           0
body            0
mileage         0
engV            0
engType         0
registration    0
year            0
model           0
drive           0
dtype: int64
```

In [45]:
```
b=car_s_copy.mileage.value_counts()
```

In [46]:
```
car_s_copy['mileage'].replace(0,'b')
```

Out[46]:
```
0        68
1       173
2       135
3       162
4        91
       ...
9571    140
9572    150
9573    180
9574    150
9575     71
Name: mileage, Length: 9453, dtype: object
```

In [47]:
```
1 car_s_copy['mileage'].value_counts()
```

Out[47]:
```
0      308
1      295
200    170
150    130
250    128
      ...
362      1
547      1
413      1
332      1
427      1
Name: mileage, Length: 442, dtype: int64
```

In [48]:
```
1 car_s_copy['mileage']=car_s['mileage']
```

In [49]:
```
1 car_s_copy['mileage']
```

Out[49]:
```
0        68
1       173
2       135
3       162
4        91
       ...
9571    140
9572    150
9573    180
9574    150
9575     71
Name: mileage, Length: 9453, dtype: int64
```

In [50]:
```
1 car_s_copy['mileage']=car_s_copy['mileage'].replace(0,'b')
```

In [53]:
```
1 car_s_copy['mileage']
```

Out[53]:
```
0        68
1       173
2       135
3       162
4        91
       ...
9571    140
9572    150
9573    180
9574    150
9575     71
Name: mileage, Length: 9453, dtype: object
```

In [54]:
```
1 profile_cleaned=ProfileReport(car_s_copy,title="Cleaned Profile")
```

In [55]:
```
1 profile_cleaned
```

Summarize dataset: 100% 28/28 [00:03<00:00, 7.30it/s, Completed]

Generate report structure: 100% 1/1 [00:02<00:00, 2.23s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.10it/s]

Overview

Dataset statistics

Number of variables	10
Number of observations	9453
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	183
Duplicate rows (%)	1.9%
Total size in memory	1.0 MiB
Average record size in memory	116.0 B

Variable types

Categorical	5
Numeric	3
Unsupported	1
Boolean	1

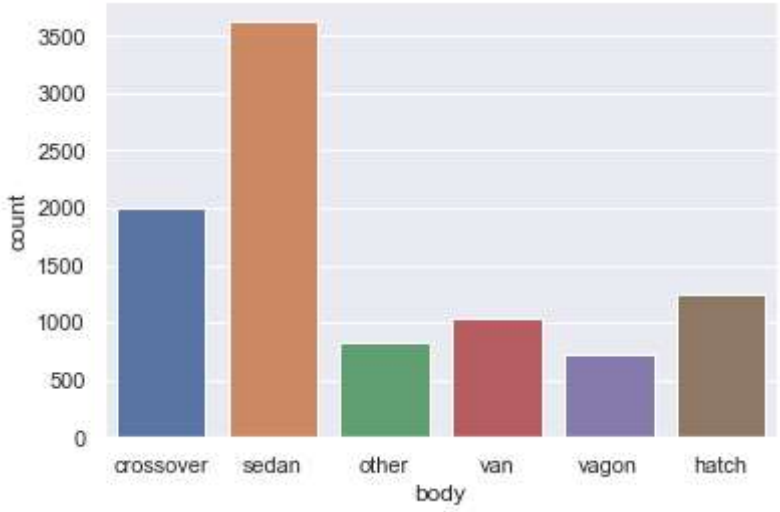
Alerts

Dataset has 183 (1.9%) duplicate rows	**Duplicates**

Out[55]:

In [56]:
```
#The maximum sold cars body
sns.countplot(data=car_s_copy,x='body')
```

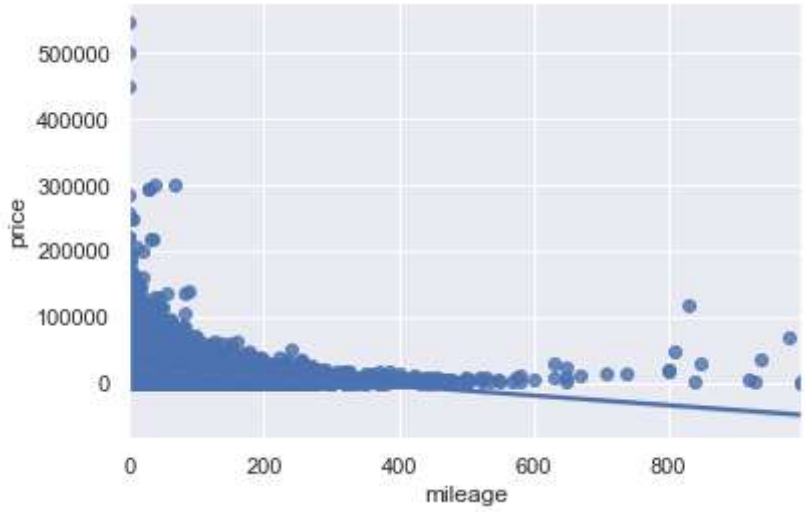
Out[56]: <AxesSubplot:xlabel='body', ylabel='count'>



From the graph. it is understood that sedan cars are sold maximum

In [57]:
```
#Corelation between price & milage
sns.regplot(x='mileage',y='price',data=car_s)
```

Out[57]: <AxesSubplot:xlabel='mileage', ylabel='price'>



From the above graph, it seems majority of thr car price falls under 150000 and the milage in between 0 to 400

In [58]:
```
#Total number of cars registered
car_s_copy
```

Out[58]:

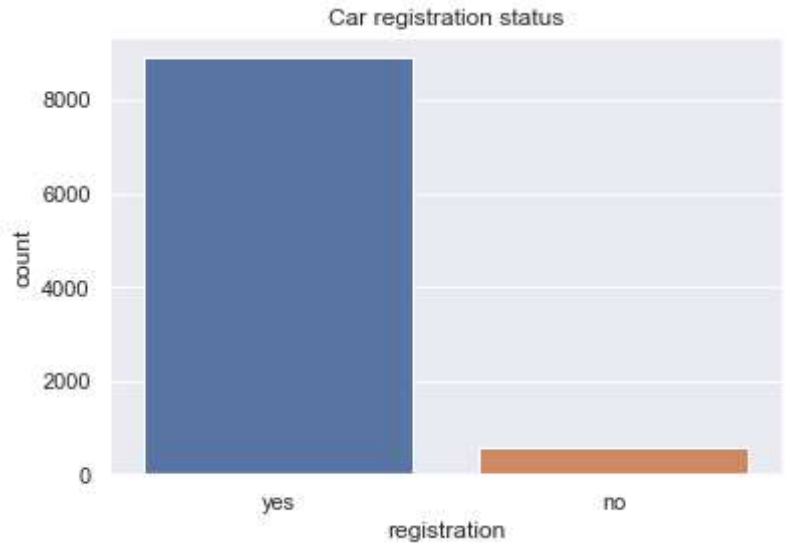
	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	2.3	Other	yes	2013	E-Class	front
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9453 rows × 10 columns

In [59]:
```
1 sns.countplot('registration',data=car_s_copy).set_title('Car registration st
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[59]: Text(0.5, 1.0, 'Car registration status')



From the above graph, it can observe that 8000+ cars are registered and less than 500 cars are not registered

In [60]:
```
#Price distribution between  registered and non registered cars
car_s_copy.groupby(['registration','body'])['price'].mean()
```

Out[60]:
```
registration  body
no            crossover     7816.542373
              hatch         2563.750000
              other         3817.393939
              sedan         3906.605691
              vagon         3055.507353
              van           3488.457143
yes           crossover    29400.477100
              hatch         8606.659842
              other        19714.860379
              sedan        12613.659852
              vagon         9944.810356
              van          10531.747228
Name: price, dtype: float64
```

In [61]:
```
car_s_copy
```

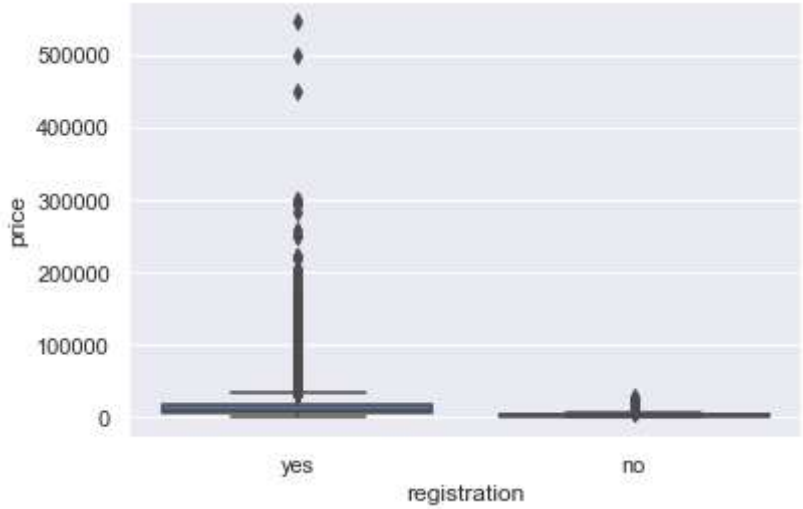
Out[61]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	2.3	Other	yes	2013	E-Class	front
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9453 rows × 10 columns

In [62]:
```
#To see the price distribution
sns.boxplot(x='registration',y='price',data=car_s_copy)
```

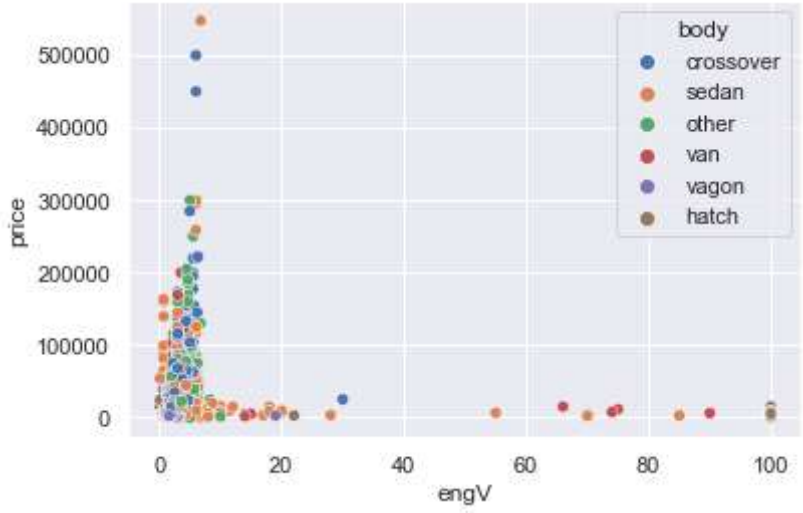
Out[62]: <AxesSubplot:xlabel='registration', ylabel='price'>



Majority of the cars are registered and the price is less than 30000. Non registered cars are cheaper in cost

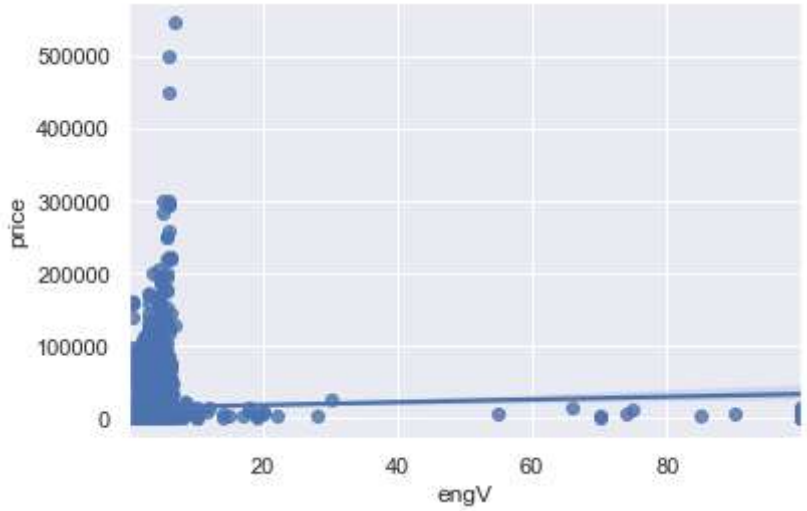
In [63]:
```
#car price distribution based out of engine value
sns.scatterplot(x='engV',y='price',data=car_s_copy,hue='body')
```

Out[63]: <AxesSubplot:xlabel='engV', ylabel='price'>



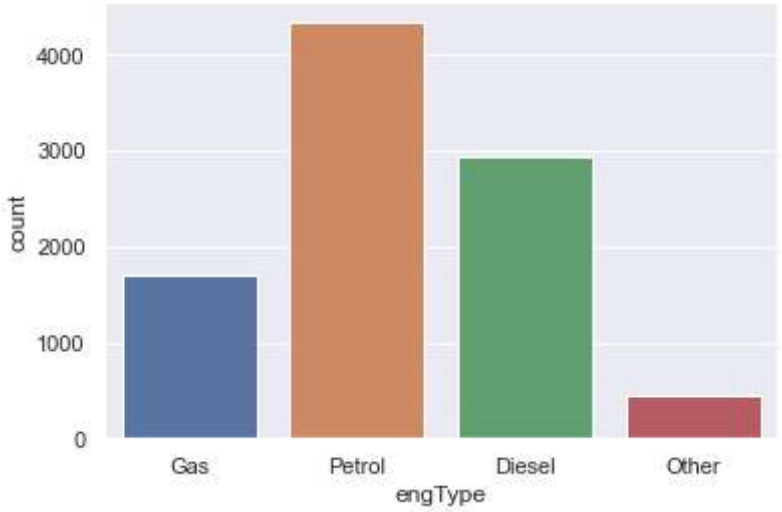
In [64]:
```
##car price distribution based out of engine value
sns.regplot(x='engV',y='price',data=car_s_copy)
```

Out[64]: <AxesSubplot:xlabel='engV', ylabel='price'>



In [65]:
```
#Type of engine cars users prefers maximum
sns.countplot(data=car_s_copy,x='engType')
```

Out[65]: <AxesSubplot:xlabel='engType', ylabel='count'>



Petrol cars are more prefered by users followed by diesel, Gas and other

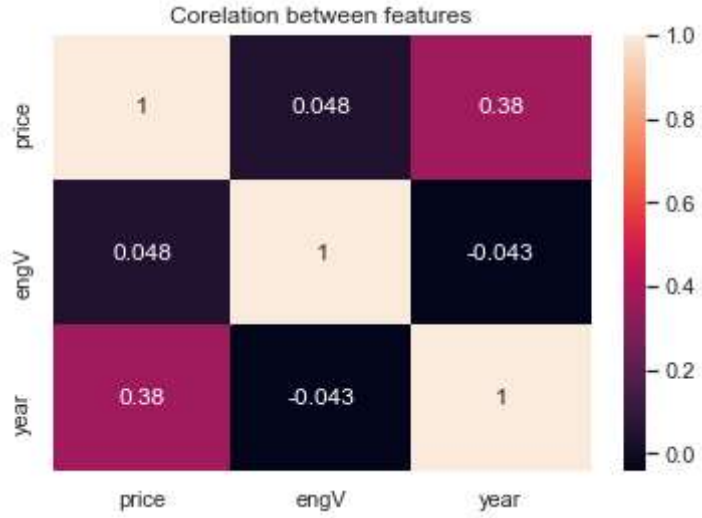
In [66]:
```
car_s_copy.groupby('engType')['price'].count().sort_values(ascending=False)
```

Out[66]:
```
engType
Petrol    4339
Diesel    2950
Gas       1710
Other      454
Name: price, dtype: int64
```

In [67]:
```
#Corelation between the data
car_cor=car_s_copy.corr()
```

In [68]:
```
sns.heatmap(data=car_cor,annot=True,linecolor='black',)
plt.title("Corelation between features")
```

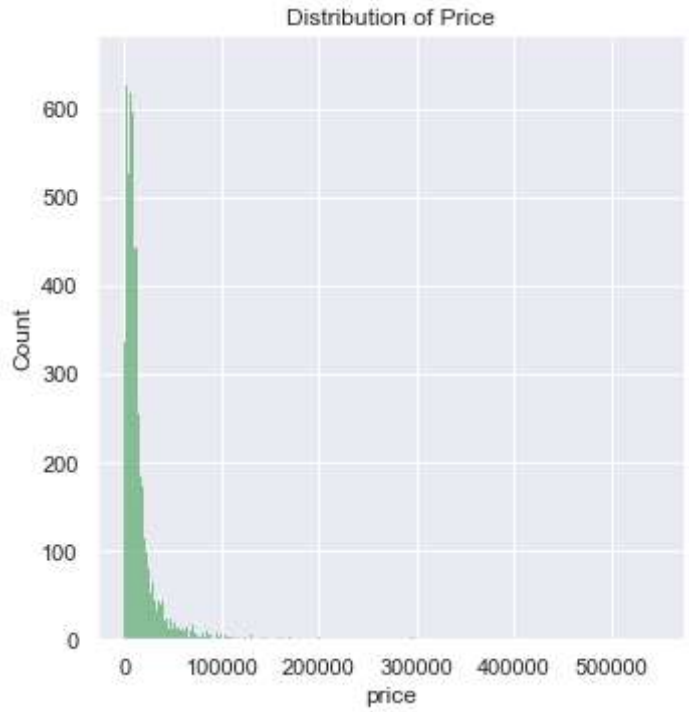
Out[68]: Text(0.5, 1.0, 'Corelation between features')



-Mileage is negatively corelated with year. -Price is also negatively corelated with year -Engine Value is positively corelated with Mileage and price

In [69]:
```
#Distribution of price
sns.displot(car_s_copy['price'],color='g')
plt.title("Distribution of Price")
```

Out[69]: Text(0.5, 1.0, 'Distribution of Price')



From the above distribution plot, we can see most of the car sales price lies between 0 to 80000