EDA with Data Collection, Data Cleaning and Pre-processing with

Vehicle dataset

Data cleaning and Pre-procrossing

To import library

In [1]:

```
import numpy as np
import pandas as pd
```

To import dataset

In [2]:

```
d=pd.read_csv(r"c:\Users\user\Downloads\ve.csv")
d
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495
...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1549 rows × 11 columns

To get top 10 record

In [3]:

```
d.head(10)
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	l
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.417
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.68227
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.611559
7	8.0	lounge	51.0	1521.0	49076.0	1.0	41.903221	12.49565
8	9.0	sport	73.0	4049.0	76000.0	1.0	45.548000	11.54946
9	10.0	sport	51.0	3653.0	89000.0	1.0	45.438301	10.99170

To get last 10

In [4]:

```
d.tail(10)
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	
1539	NaN	NaN	NaN	NaN	NaN	NaN	NaN	avg	8576.00
1540	NaN	NaN	NaN	NaN	NaN	NaN	NaN	count	
1541	NaN	NaN	NaN	NaN	NaN	NaN	NaN	countif	
1542	NaN	NaN	NaN	NaN	NaN	NaN	NaN	sumif	401
1543	NaN	NaN	NaN	NaN	NaN	NaN	NaN	counta (not empty)	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lon
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	

To describe statistics Analysis

In [5]:

```
d.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	la
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.00000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.54136
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.13351
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.85583
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.80299
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.39409
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.46796
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.79561

To get rows and columns

In [6]:

```
np.shape(d)
```

Out[6]:

```
(1549, 11)
```

To get number of elements

In [7]:

```
np.size(d)
```

Out[7]:

```
17039
```

To get the missing value

In [8]:

```
d.isna()
```

Out[8]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
1544	True	True	True	True	True	True	True	False	False
1545	True	True	True	True	True	True	True	False	False
1546	True	True	True	True	True	True	True	False	False
1547	True	True	True	True	True	True	True	False	False
1548	True	True	True	True	True	True	True	False	False

1549 rows × 11 columns

To drop the missing elements

In [9]:

```
d.dropna(axis=1,how='any')
```

Out[9]:

	lon	price
0	8.611559868	8900
1	12.24188995	8800
2	11.41784	4200
3	17.63460922	6000
4	12.49565029	5700
...	...	...
1544	length	5
1545	concat	lonprice
1546	Null values	NO
1547	find	1
1548	search	1

1549 rows × 2 columns

In [10]:

```
d["engine_power"]
```

Out[10]:

```
0       51.0
1       51.0
2       74.0
3       51.0
4       73.0
        ...
1544     NaN
1545     NaN
1546     NaN
1547     NaN
1548     NaN
Name: engine_power, Length: 1549, dtype: float64
```

Visualization

In [11]:

```
data=pd.DataFrame(d[['engine_power','km']][0:500])
data
```

Out[11]:

	engine_power	km
0	51.0	25000.0
1	51.0	32500.0
2	74.0	142228.0
3	51.0	160000.0
4	73.0	106880.0
...	...	...
495	51.0	15003.0
496	51.0	38718.0
497	51.0	17488.0
498	51.0	24281.0
499	51.0	25076.0

500 rows × 2 columns

In [12]:

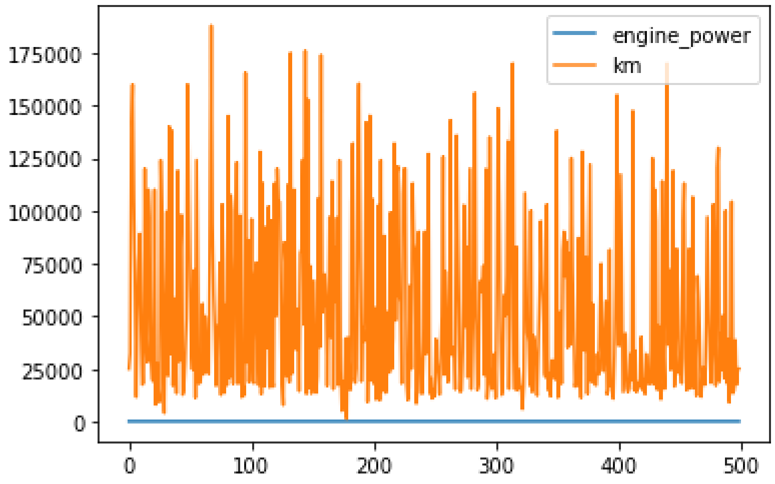
```
import matplotlib.pyplot as pp
```

In [13]:

```
data.plot.line()
```

Out[13]:

```
<AxesSubplot:>
```

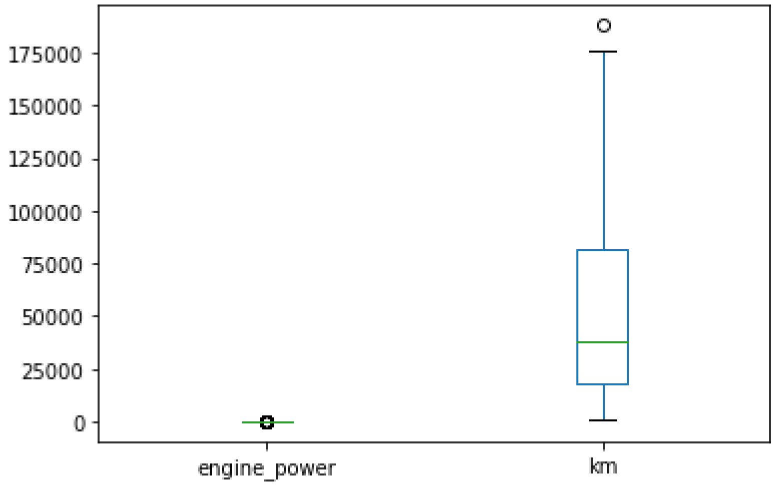


In [14]:

```
data.plot.box()
```

Out[14]:

```
<AxesSubplot:>
```

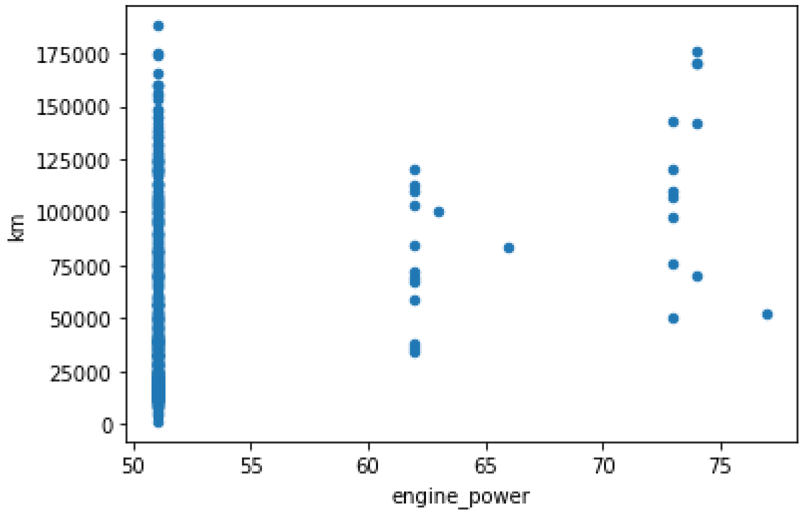


In [15]:

```
data.plot.scatter(x="engine_power",y="km")
```

Out[15]:

```
<AxesSubplot:xlabel='engine_power', ylabel='km'>
```

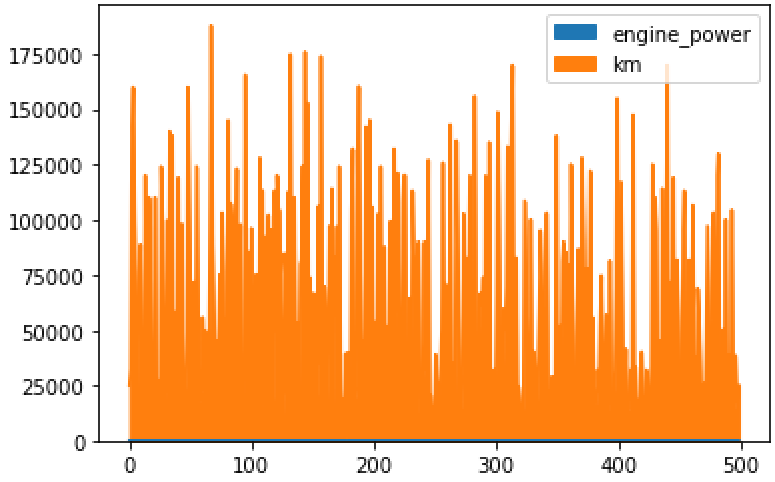


In [16]:

```
data.plot.area()
```

Out[16]:

```
<AxesSubplot:>
```

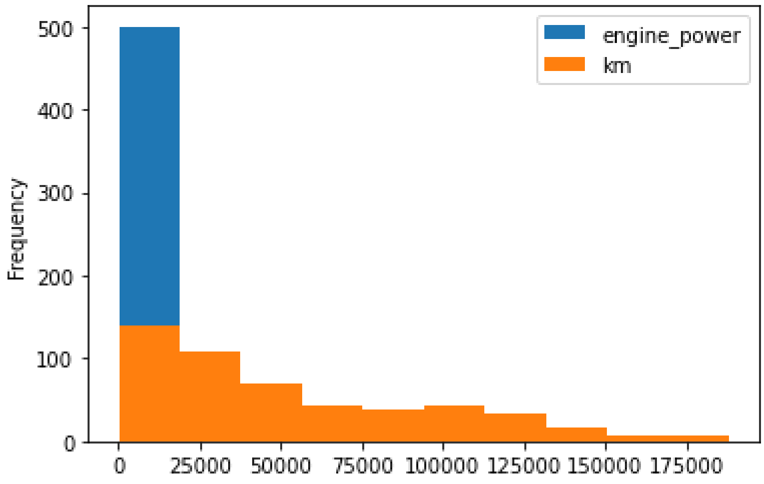


In [17]:

```
data.plot.hist()
```

Out[17]:

```
<AxesSubplot:ylabel='Frequency'>
```

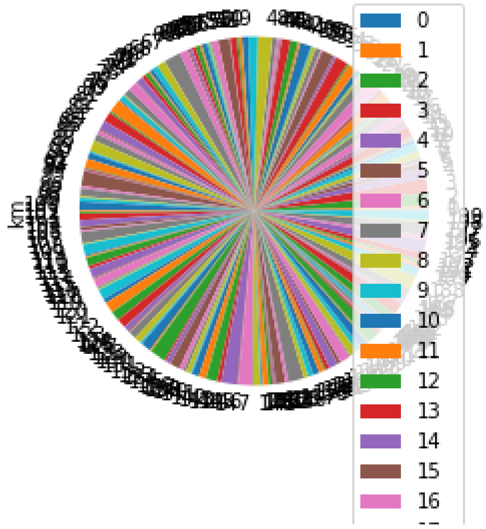


In [18]:

```
data=pd.DataFrame(d[['engine_power','km']][0:200])
data.plot.pie(y="km")
```

Out[18]:

```
<AxesSubplot:ylabel='km'>
```



Statistics

Mean,median,mode,describe

In [19]:

```
data=pd.DataFrame(d[['engine_power','km']][0:500])
data
```

Out[19]:

	engine_power	km
0	51.0	25000.0
1	51.0	32500.0
2	74.0	142228.0
3	51.0	160000.0
4	73.0	106880.0
...	...	...
495	51.0	15003.0
496	51.0	38718.0
497	51.0	17488.0
498	51.0	24281.0
499	51.0	25076.0

500 rows × 2 columns

In [20]:

```
print(data.mean())
```

```
engine_power       51.908
km              53279.784
dtype: float64
```

In [21]:

```
print(data.median())
```

```
engine_power       51.0
km              38000.0
dtype: float64
```

In [22]:

```
print(data.mode())
```

```
   engine_power       km
0          51.0  32057.0
```

In [24]:

```
data.fillna(value=1)
```

Out[24]:

	engine_power	km
0	51.0	25000.0
1	51.0	32500.0
2	74.0	142228.0
3	51.0	160000.0
4	73.0	106880.0
...	...	...
495	51.0	15003.0
496	51.0	38718.0
497	51.0	17488.0
498	51.0	24281.0
499	51.0	25076.0

500 rows × 2 columns

In [26]:

```
print(data.describe())
```

```
       engine_power             km
count     500.00000     500.000000
mean       51.90800   53279.784000
std         4.03337   41893.569817
min        51.00000    1232.000000
25%        51.00000   18199.500000
50%        51.00000   38000.000000
75%        51.00000   81900.000000
max        77.00000  188000.000000
```

Sum,cumsum,count,min,max

In [27]:

```
print(data.sum())
```

```
engine_power       25954.0
km              26639892.0
dtype: float64
```

In [28]:

```
print(data.cumsum())
```

```
     engine_power          km
0            51.0     25000.0
1           102.0     57500.0
2           176.0    199728.0
3           227.0    359728.0
4           300.0    466608.0
..            ...         ...
495       25750.0  26534329.0
496       25801.0  26573047.0
497       25852.0  26590535.0
498       25903.0  26614816.0
499       25954.0  26639892.0

[500 rows x 2 columns]
```

In [29]:

```
print(data.count())
```

```
engine_power    500
km              500
dtype: int64
```

In [30]:

```
print(data.min())
```

```
engine_power      51.0
km              1232.0
dtype: float64
```

In [31]:

```
print(data.max())
```

```
engine_power        77.0
km              188000.0
dtype: float64
```

covariance and correlation (spearman and pearsons)

In [32]:

```
data1=data['engine_power'][0:10]
data1
```

Out[32]:

```
0    51.0
1    51.0
2    74.0
3    51.0
4    73.0
5    74.0
6    51.0
7    51.0
8    73.0
9    51.0
Name: engine_power, dtype: float64
```

In [33]:

```
data2=data['km'][0:10]
data2
```

Out[33]:

```
0     25000.0
1     32500.0
2    142228.0
3    160000.0
4    106880.0
5     70225.0
6     11600.0
7     49076.0
8     76000.0
9     89000.0
Name: km, dtype: float64
```

In [34]:

```
from numpy import cov
print(cov(data1,data2))
```

```
[[1.35111111e+02 2.27466444e+05]
 [2.27466444e+05 2.44032836e+09]]
```

In [35]:

```
from scipy.stats import pearsonr
print(pearsonr(data1,data2))
```

```
(0.39613906530125964, 0.25710544510156774)
```

In [36]:

```
from scipy.stats import spearmanr
print(spearmanr(data1,data2))
```

```
SpearmanrResult(correlation=0.4128614119223852, pvalue=0.2357037774356011
1)
```

In [ ]: