Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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

In [2]:
```
df=pd.read_csv("fiat")
```

To display top 10 rows

In [3]:
```
df.head(10)
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Unnamed: 9	Unnamed: 10	Unnam
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611560	8900.0	NaN	NaN	CONCATE
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241890	8800.0	NaN	NaN	Length
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.417840	4200.0	NaN	NaN	No. of w days btw 23 to 17
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634609	6000.0	NaN	NaN	
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495650	5700.0	NaN	NaN	Ave
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.682270	7900.0	NaN	NaN	c
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.611560	10750.0	NaN	NaN	
7	8.0	lounge	51.0	1521.0	49076.0	1.0	41.903221	12.495650	9190.0	NaN	NaN	
8	9.0	sport	73.0	4049.0	76000.0	1.0	45.548000	11.549470	5600.0	NaN	NaN	
9	10.0	sport	51.0	3653.0	89000.0	1.0	45.438301	10.991700	6000.0	NaN	NaN	

Data Cleaning And Pre-Processing

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1541 entries, 0 to 1540
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   float64
 1   model            1538 non-null   object 
 2   engine_power     1539 non-null   float64
 3   age_in_days      1539 non-null   float64
 4   km               1539 non-null   float64
 5   previous_owners  1538 non-null   float64
 6   lat              1539 non-null   float64
 7   lon              1539 non-null   float64
 8   price            1541 non-null   float64
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      float64
 11  Unnamed: 11      6 non-null      object 
 12  Unnamed: 12      10 non-null     object 
 13  Unnamed: 13      0 non-null      float64
 14  Unnamed: 14      3 non-null      object 
dtypes: float64(11), object(4)
memory usage: 180.7+ KB
```

In [5]:
```
# Display the statistical summary
df.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price	Unnamed: 9
count	1538.000000	1539.000000	1.539000e+03	1539.000000	1538.000000	1539.000000	1539.000000	1.541000e+03	0.0
mean	769.500000	52.870045	3.001763e+03	53396.011704	1.123537	87.026139	23.111829	2.568905e+04	NaN
std	444.126671	38.090731	5.300702e+04	40033.809481	0.416423	1705.913084	453.050800	4.747172e+05	NaN
min	1.000000	51.000000	3.660000e+02	1232.000000	1.000000	36.855839	7.245400	2.500000e+03	NaN
25%	385.250000	51.000000	6.700000e+02	20012.500000	1.000000	41.802990	9.505090	7.190000e+03	NaN
50%	769.500000	51.000000	1.035000e+03	39038.000000	1.000000	44.399971	11.869260	9.000000e+03	NaN
75%	1153.750000	51.000000	2.616000e+03	79535.500000	1.000000	45.467960	12.774520	1.000000e+04	NaN
max	1538.000000	1538.000000	2.080506e+06	235000.000000	4.000000	66966.613720	17784.552790	1.318989e+07	NaN

In [6]:
```
# To display the col headings
df.columns
```

Out[6]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11',
       'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14'],
      dtype='object')
```

In [7]:
```
cols=df.dropna(axis=1)
cols
```

Out[7]:

	price
0	8.900000e+03
1	8.800000e+03
2	4.200000e+03
3	6.000000e+03
4	5.700000e+03
...	...
1536	5.990000e+03
1537	7.900000e+03
1538	1.318989e+07
1539	1.714086e+04
1540	1.318989e+07

1541 rows × 1 columns

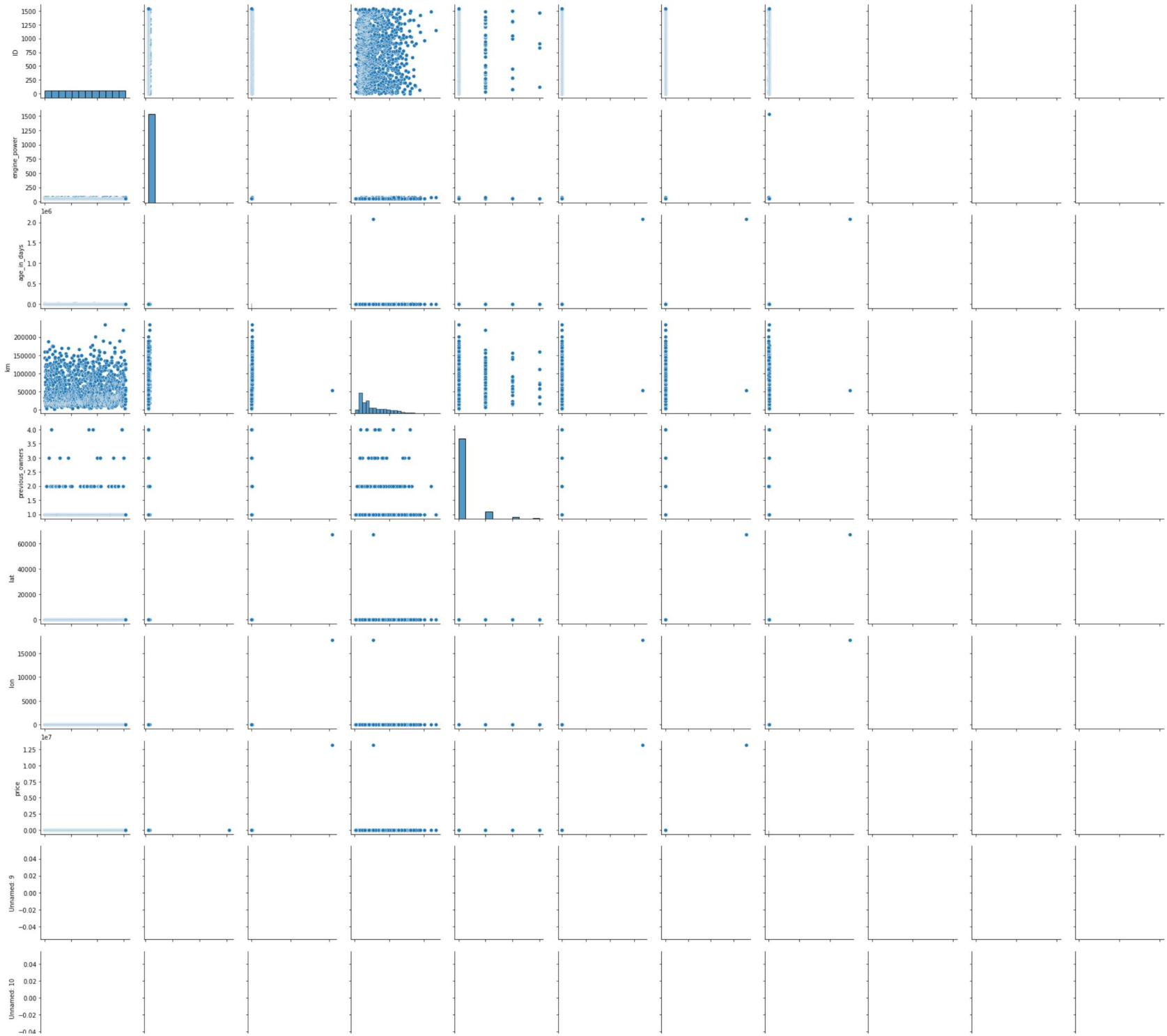
In [8]:
```
cols.columns
```

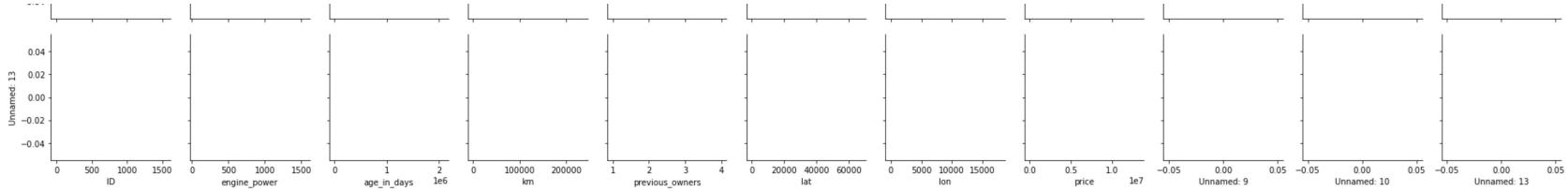
Out[8]:
```
Index(['price'], dtype='object')
```

EDA and Visualization

In [9]:
```
sns.pairplot(df)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x2211632e580>
```

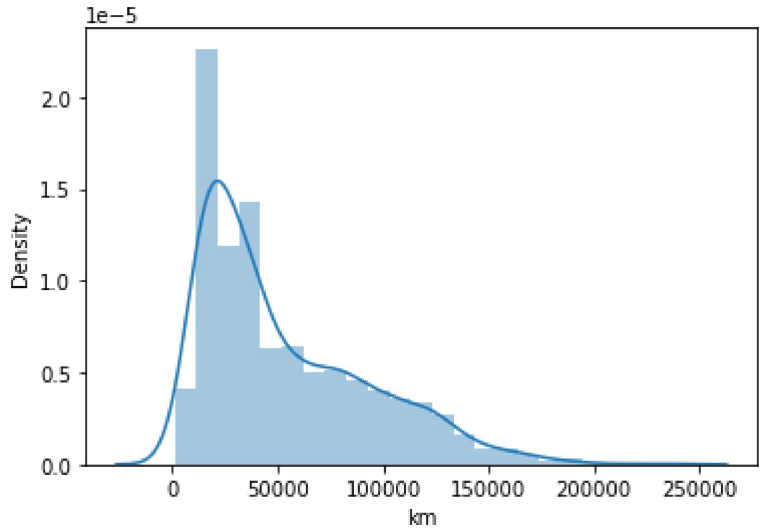

ID
engine_power
age_in_days
km
previous_owners
lat
lon
price
Unnamed: 9
Unnamed: 10



In [11]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['km'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]: <AxesSubplot:xlabel='km', ylabel='Density'>



In [13]:
```
df1=df[['engine_power', 'age_in_days', 'km', 'previous_owners']]
df1
```

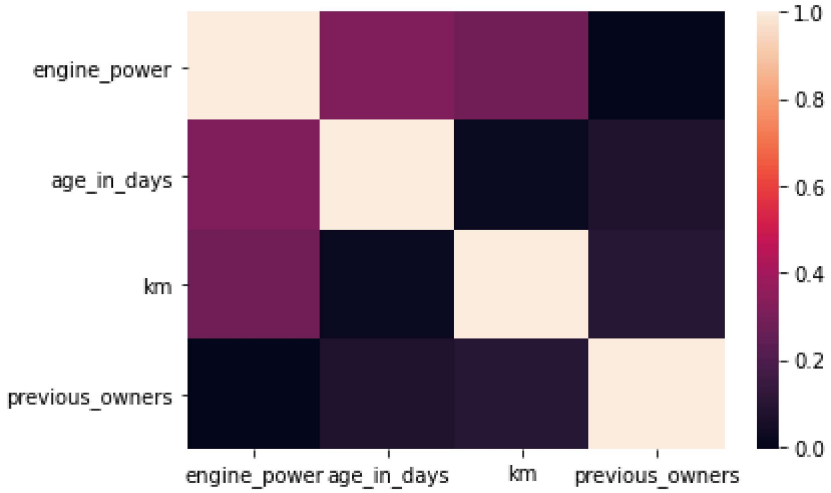
Out[13]:

	engine_power	age_in_days	km	previous_owners
0	51.0	882.0	25000.0000	1.0
1	51.0	1186.0	32500.0000	1.0
2	74.0	4658.0	142228.0000	1.0
3	51.0	2739.0	160000.0000	1.0
4	73.0	3074.0	106880.0000	1.0
...	...	...	...	...
1536	51.0	2557.0	80750.0000	1.0
1537	51.0	1766.0	54276.0000	1.0
1538	NaN	2080506.0	53396.0117	NaN
1539	1538.0	NaN	NaN	NaN
1540	NaN	NaN	NaN	NaN

1541 rows × 4 columns

In [14]:
```
sns.heatmap(df1.corr())
```

Out[14]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [22]:
```
x=df[['price']]
y=df[['price']]
```

To split the dataset into test data

In [23]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [24]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [25]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
print(lr.intercept_)
```

```
[7.27595761e-12]
```

In [27]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

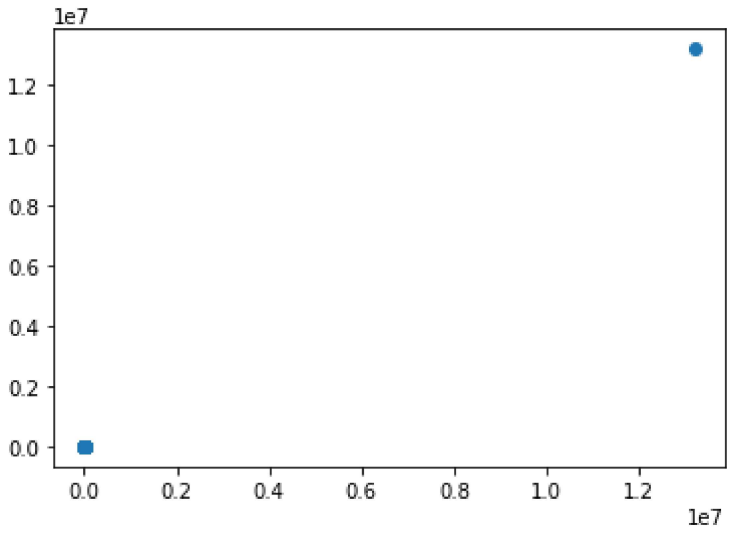
In [28]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[28]:

	0
0	1.0

In [29]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x221b2521250>
```



In [ ]: