In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```

```

In [3]:

```
df = pd.read_csv('cars24.csv')
df.shape
```

Out[3]:

```
(19980, 11)
```

In [4]:

```
df.head()
```

Out[4]:

	full_name	selling_price	year	seller_type	km_driven	fuel_type	transmission_type	mileage
0	Maruti Alto Std	1.20	2012.0	Individual	120000	Petrol	Manual	19.70
1	Hyundai Grand i10 Asta	5.50	2016.0	Individual	20000	Petrol	Manual	18.90
2	Hyundai i20 Asta	2.15	2010.0	Individual	60000	Petrol	Manual	17.00
3	Maruti Alto K10 2010-2014 VXI	2.26	2012.0	Individual	37000	Petrol	Manual	20.92
4	Ford Ecosport 2015-2021 1.5 TDCi Titanium BSIV	5.70	2015.0	Dealer	30000	Diesel	Manual	22.77

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19980 entries, 0 to 19979
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   full_name          19980 non-null  object 
 1   selling_price      19980 non-null  float64
 2   year               19980 non-null  float64
 3   seller_type        19980 non-null  object 
 4   km_driven          19980 non-null  int64  
 5   fuel_type          19980 non-null  object 
 6   transmission_type  19980 non-null  object 
 7   mileage            19980 non-null  float64
 8   engine             19980 non-null  float64
 9   max_power          19980 non-null  float64
 10  seats              19980 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.7+ MB
```

In [6]:

```
df.describe()
```

Out[6]:

	selling_price	year	km_driven	mileage	engine	max_power	
count	19980.000000	19980.000000	1.998000e+04	19980.000000	19980.000000	19980.000000	19
mean	7.392066	2014.525125	5.824488e+04	19.347219	1476.327401	97.854443	
std	9.103088	3.249185	5.172509e+04	4.620053	520.449398	45.080670	
min	0.250000	1991.000000	1.000000e+02	0.000000	0.000000	5.000000	
25%	3.400000	2013.000000	3.116425e+04	16.800000	1197.000000	73.900000	
50%	5.200000	2015.000000	5.200000e+04	19.160000	1248.000000	86.700000	
75%	7.850000	2017.000000	7.400000e+04	22.320000	1582.000000	112.000000	
max	395.000000	2021.000000	3.800000e+06	120.000000	6752.000000	626.000000	

In [9]:

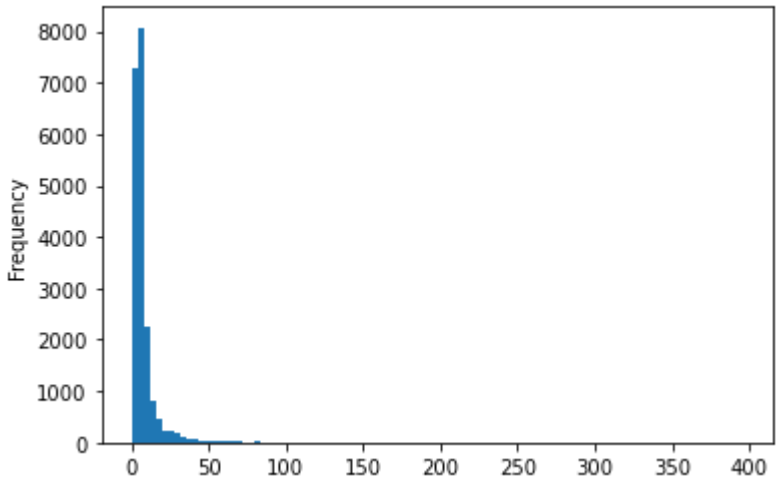
```
# plt.hist(df['selling_price'])
```

In [12]:

```
df['selling_price'].plot(kind='hist', bins=100)
```

Out[12]:

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



In [17]:

```
df[df['selling_price']>100].shape[0]/19000
```

Out[17]:

```
0.0005789473684210527
```

In [ ]:

```

```

In [18]:

```
df[df['selling_price']>100]
```

Out[18]:

	full_name	selling_price	year	seller_type	km_driven	fuel_type	transmission_type	m
475	Bentley Mulsanne 6.8	235.0	2012.0	Dealer	15000	Petrol	Automatic	
1536	Bentley Continental GT Speed Convertible BSIV	145.0	2012.0	Dealer	9000	Petrol	Automatic	
2478	Mercedes-Benz S-Class S 500	110.0	2016.0	Dealer	41000	Petrol	Automatic	
3980	Lamborghini Gallardo Spyder	150.0	2013.0	Dealer	4000	Petrol	Automatic	
3983	Jaguar XJ 50 Special Edition	110.0	2019.0	Individual	5400	Diesel	Automatic	
4967	Ferrari GTC4Lusso T	395.0	2019.0	Dealer	3800	Petrol	Automatic	
12299	Porsche Cayenne GTS	111.0	2017.0	Dealer	24000	Petrol	Automatic	
12741	Mercedes-Benz S-Class Maybach S500	130.0	2018.0	Dealer	4000	Petrol	Automatic	
13863	Porsche Cayenne Base	132.0	2019.0	Individual	5000	Petrol	Automatic	
14261	Rolls-Royce Ghost Series II Extended Wheelbase	242.0	2017.0	Individual	5000	Petrol	Automatic	
14283	Mercedes-Benz S-Class S 450	110.0	2019.0	Dealer	18000	Petrol	Automatic	

In [20]:

```
np.percentile(df['selling_price'], 99.9)
```

Out[20]:

```
82.01050000000032
```

In [ ]:

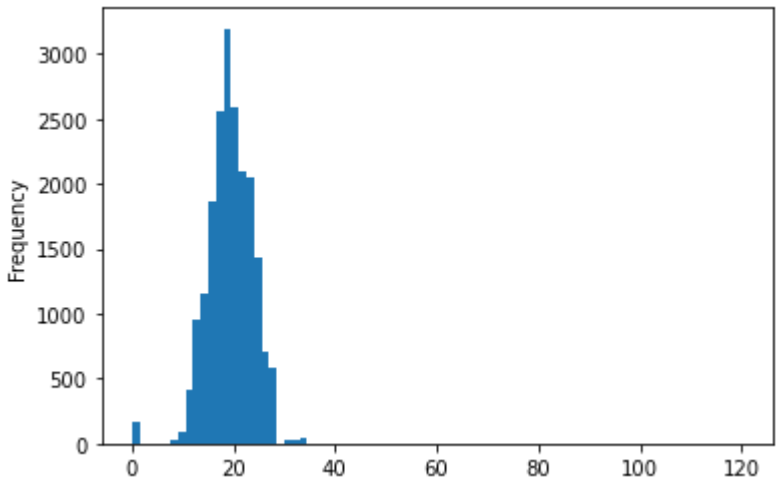
```

```

In [27]:

```
df['mileage'].plot(kind='hist', bins=80)
```

Out[27]:

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



In [31]:

```
df[(df['mileage']==0)]
```

Out[31]:

	full_name	selling_price	year	seller_type	km_driven	fuel_type	transmission_type	mil
146	Hyundai Santro LP	1.60	2007.0	Individual	54000	Petrol	Manual	
270	Hyundai Santro AT	0.71	2004.0	Individual	110000	Petrol	Automatic	
304	Ford Figo Aspire Facelift	5.50	2017.0	Individual	15000	Diesel	Manual	
351	Hyundai Santro Xing XL	1.15	2005.0	Individual	120000	Petrol	Manual	
420	Toyota Fortuner 4X2 AT	12.35	2012.0	Dealer	85000	Petrol	Automatic	
...	...	...	...	...	...	...	...	
19772	Hyundai Kona Premium	21.75	2019.0	Dealer	6000	Electric	Automatic	
19824	Mahindra Jeep MM 550 DP	3.50	2006.0	Individual	120000	Diesel	Manual	
19880	Toyota Fortuner 4X2 AT	14.50	2014.0	Dealer	127000	Petrol	Automatic	
19968	Hyundai Santro Xing GL	1.15	2008.0	Individual	80000	Petrol	Manual	
19970	Mercedes-Benz M-Class ML 350 4Matic	32.00	2014.0	Dealer	32000	Diesel	Automatic	

160 rows × 11 columns

In [ ]:

```
```

In [34]:

```
df[(df['mileage']> 40)]
```

Out[34]:

	full_name	selling_price	year	seller_type	km_driven	fuel_type	transmission_type
11966	Mahindra e2oPlus P6	4.00	2018.0	Dealer	26000	Electric	Automatic
15684	Mahindra e2o T2	3.60	2015.0	Dealer	42862	Electric	Automatic
17782	Mahindra e2o T2	4.50	2015.0	Dealer	40000	Electric	Automatic
18401	Mahindra e2o Premium	3.25	2013.0	Individual	50000	Electric	Automatic

In [ ]:

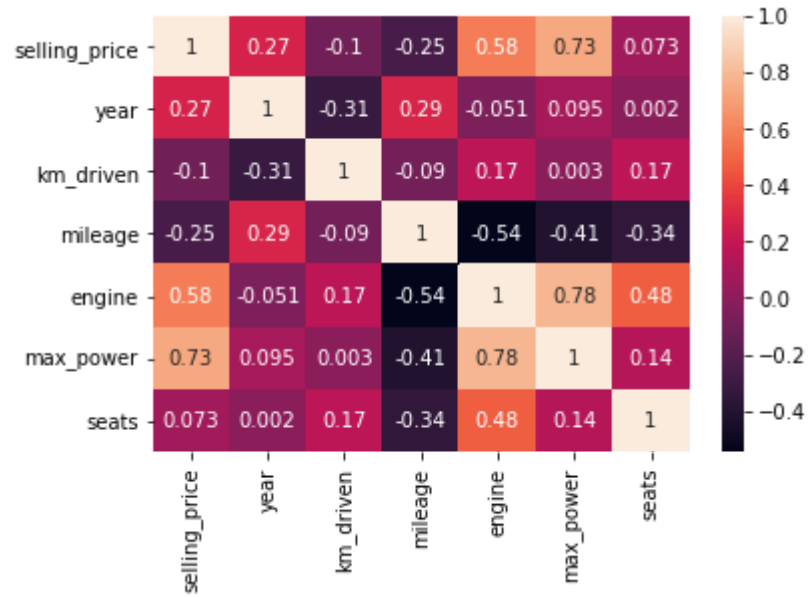
```
```

In [36]:

```
sns.heatmap(df.corr(), annot=True)
```

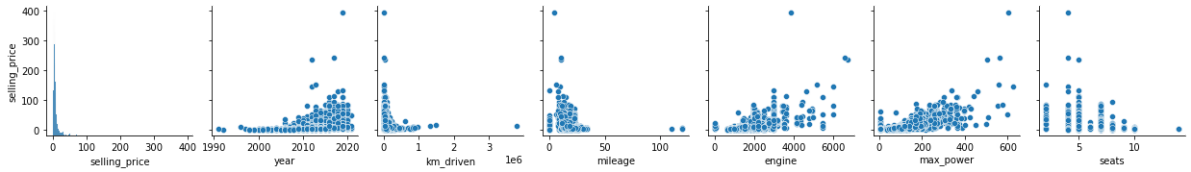
Out[36]:

```
<AxesSubplot:>
```



In [37]:

```
sns.pairplot(df, y_vars=['selling_price'])
plt.show()
```



In [ ]:

```

```

In [41]:

```
df['seats'].value_counts(normalize=True)
```

Out[41]:

```
5.0     0.834885
7.0     0.118619
8.0     0.022022
4.0     0.011261
6.0     0.007157
9.0     0.003604
10.0    0.001301
2.0     0.001051
14.0    0.000100
Name: seats, dtype: float64
```

In [42]:

```
df
```

Out[42]:

	full_name	selling_price	year	seller_type	km_driven	fuel_type	transmission_type	mile
0	Maruti Alto Std	1.20	2012.0	Individual	120000	Petrol	Manual	1
1	Hyundai Grand i10 Asta	5.50	2016.0	Individual	20000	Petrol	Manual	1
2	Hyundai i20 Asta	2.15	2010.0	Individual	60000	Petrol	Manual	1
3	Maruti Alto K10 2010-2014 VXI	2.26	2012.0	Individual	37000	Petrol	Manual	2
4	Ford Ecosport 2015-2021 1.5 TDCi Titanium BSIV	5.70	2015.0	Dealer	30000	Diesel	Manual	2
...	...	...	...	...	...	...	...	
19975	Toyota Platinum Etios 1.4 GXD	6.50	2017.0	Dealer	69480	Diesel	Manual	2
19976	Maruti Ertiga BSIV VXI	9.25	2019.0	Dealer	18000	Petrol	Manual	1
19977	Skoda Rapid 1.5 TDI Elegance	4.25	2015.0	Dealer	67000	Diesel	Manual	2
19978	Mahindra XUV500 W6 2WD	12.25	2016.0	Dealer	3800000	Diesel	Manual	1
19979	Honda City i-VTEC CVT VX	12.00	2019.0	Dealer	13000	Petrol	Automatic	1

19980 rows × 11 columns

In [46]:

```
df['make'] = df['full_name'].apply(lambda x: x.split()[0])
```

In [50]:

```
df['model'] = df['full_name'].apply(lambda x: " ".join(x.split()[1:]))
```

In [52]:

```
df.drop(columns=['full_name'], inplace = True)
```

In [53]:

```
df.head()
```

Out[53]:

	selling_price	year	seller_type	km_driven	fuel_type	transmission_type	mileage	engine	m
0	1.20	2012.0	Individual	120000	Petrol	Manual	19.70	796.0	
1	5.50	2016.0	Individual	20000	Petrol	Manual	18.90	1197.0	
2	2.15	2010.0	Individual	60000	Petrol	Manual	17.00	1197.0	
3	2.26	2012.0	Individual	37000	Petrol	Manual	20.92	998.0	
4	5.70	2015.0	Dealer	30000	Diesel	Manual	22.77	1498.0	

In [ ]:

```
```

In [55]:

```
df.describe(include='object')
```

Out[55]:

	seller_type	fuel_type	transmission_type	make	model
count	19980	19980	19980	19980	19980
unique	3	5	2	42	3321
top	Dealer	Diesel	Manual	Maruti	Swift Dzire VDI
freq	11973	9817	16029	5650	210

In [56]:

```
df["fuel_type"].value_counts()
```

Out[56]:

```
Diesel      9817
Petrol      9767
CNG          316
LPG           66
Electric      14
Name: fuel_type, dtype: int64
```

In [57]:

```
df["transmission_type"].value_counts()
```

Out[57]:

```
Manual       16029
Automatic     3951
Name: transmission_type, dtype: int64
```

In [58]:

```
df["seller_type"].value_counts()
```

Out[58]:

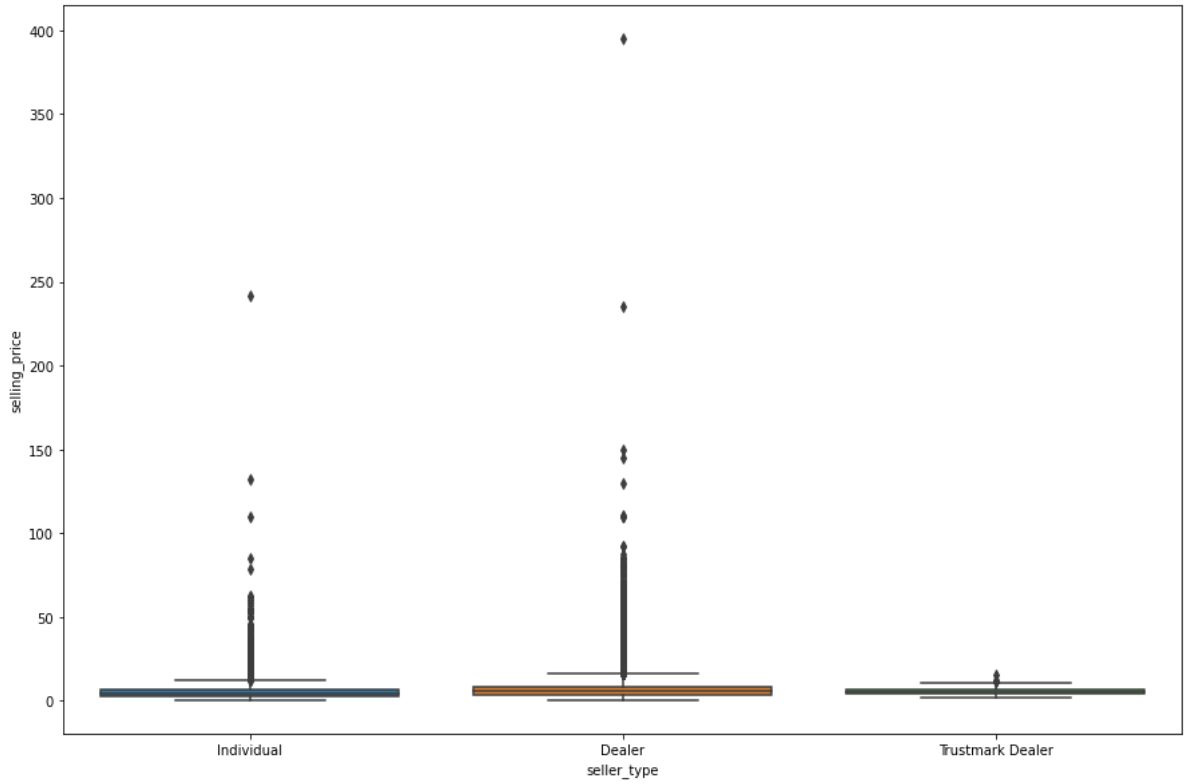
```
Dealer              11973
Individual           7817
Trustmark Dealer      190
Name: seller_type, dtype: int64
```

In [60]:

```
plt.figure(figsize=(15,10))
sns.boxplot(data=df, x='seller_type', y='selling_price')
```

Out[60]:

```
<AxesSubplot:xlabel='seller_type', ylabel='selling_price'>
```

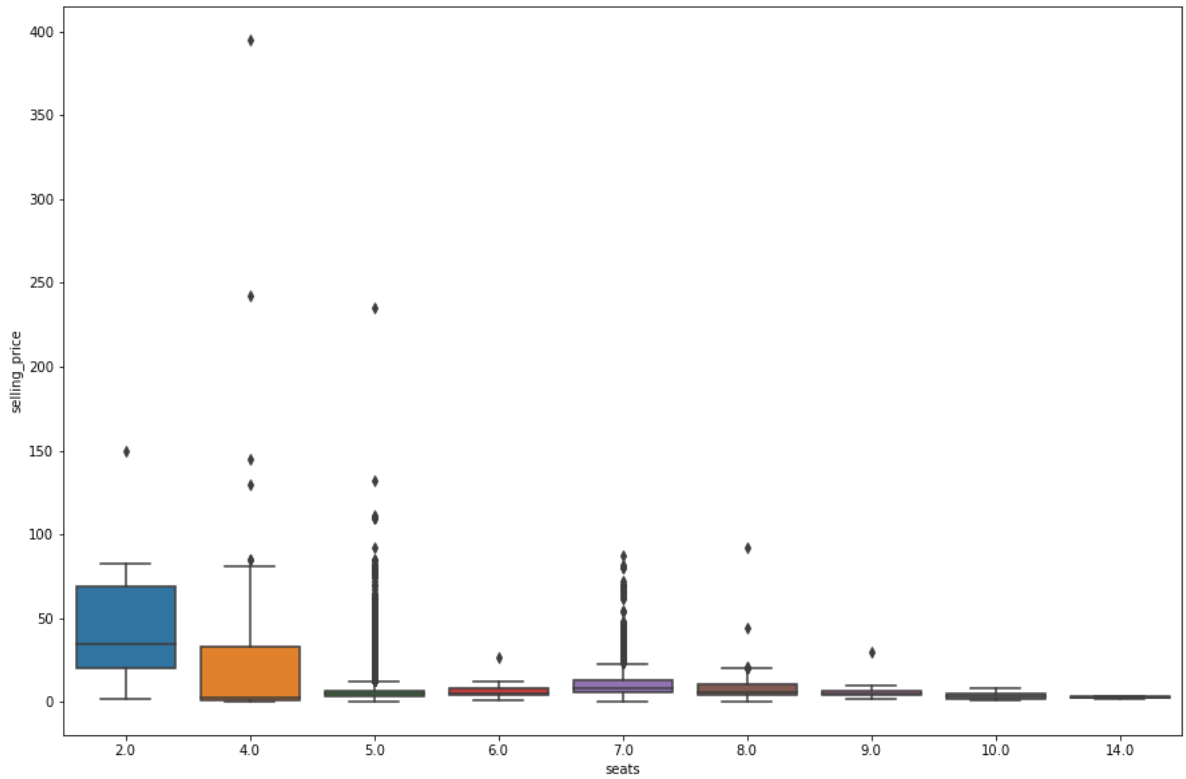


In [61]:

```
plt.figure(figsize=(15,10))
sns.boxplot(data=df, x='seats', y='selling_price')
```

Out[61]:

```
<AxesSubplot:xlabel='seats', ylabel='selling_price'>
```



In [ ]:

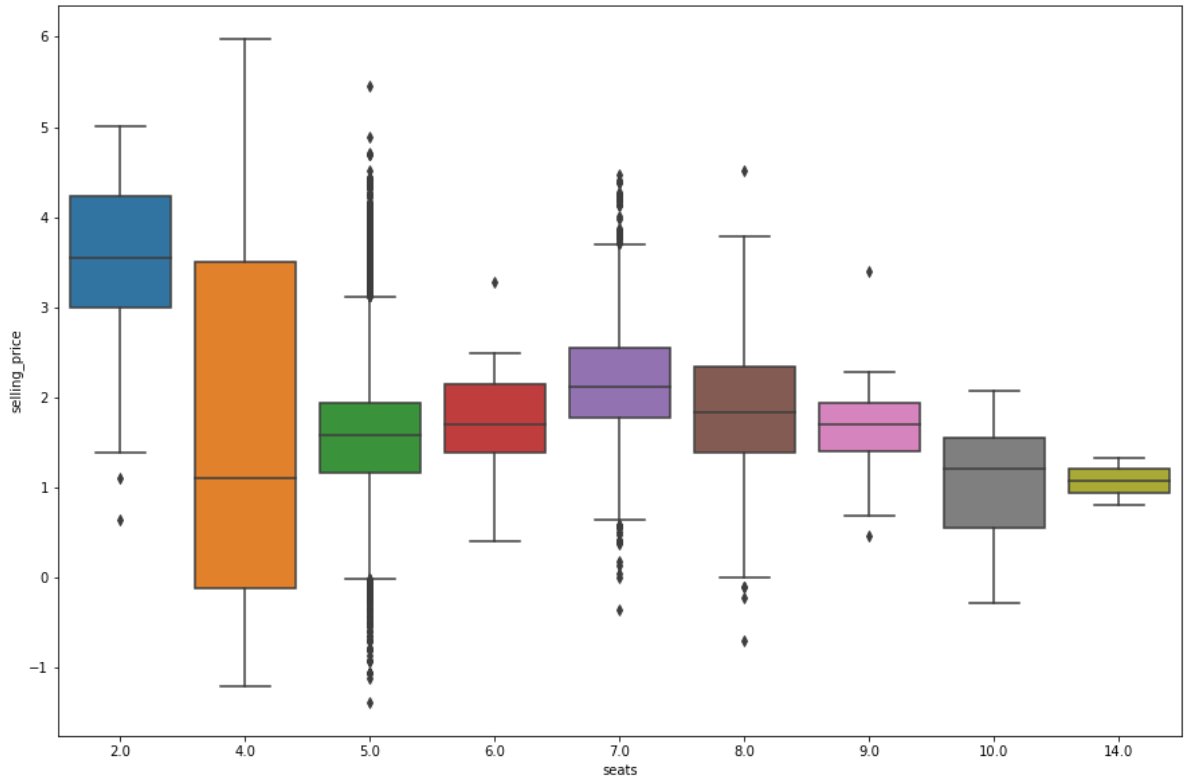
```
```

In [62]:

```
# Using log selling price.

plt.figure(figsize=(15,10))
sns.boxplot(data=df, x='seats', y=np.log(df['selling_price']))
```

Out[62]:

```
<AxesSubplot:xlabel='seats', ylabel='selling_price'>
```



In [64]:

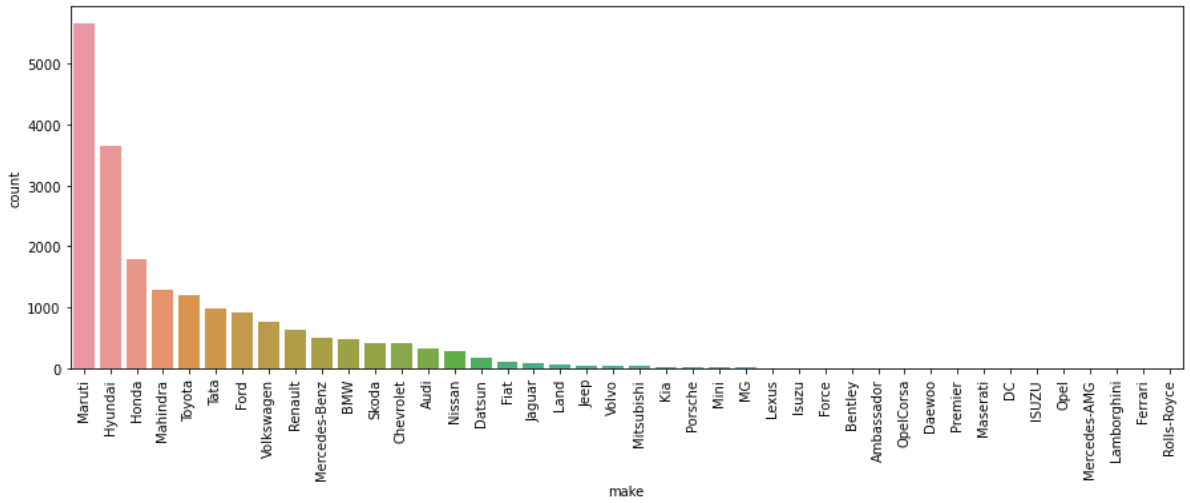
```
df['selling_price_log'] = np.log(df['selling_price'])
```

In [ ]:

```

```

In [69]:

```
plt.figure(figsize=(15,5))
sns.countplot(x='make', data=df, order = df['make'].value_counts().index)
plt.xticks(rotation=90)
plt.show()
```



In [ ]:

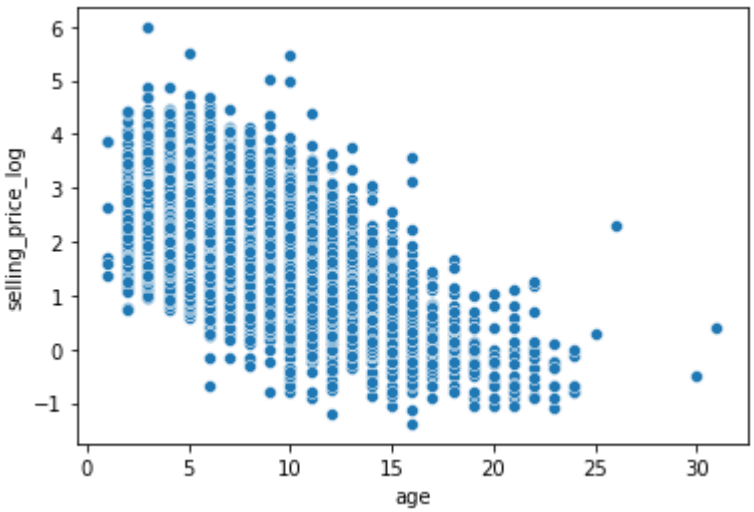
```
```

In [73]:

```
curr_year = 2022
df['age'] = curr_year - df['year']

sns.scatterplot(data = df, x = 'age', y='selling_price_log')
```

Out[73]:

```
<AxesSubplot:xlabel='age', ylabel='selling_price_log'>
```



In [74]:

```
df.head()
```

Out[74]:

	selling_price	year	seller_type	km_driven	fuel_type	transmission_type	mileage	engine	m
0	1.20	2012.0	Individual	120000	Petrol	Manual	19.70	796.0	
1	5.50	2016.0	Individual	20000	Petrol	Manual	18.90	1197.0	
2	2.15	2010.0	Individual	60000	Petrol	Manual	17.00	1197.0	
3	2.26	2012.0	Individual	37000	Petrol	Manual	20.92	998.0	
4	5.70	2015.0	Dealer	30000	Diesel	Manual	22.77	1498.0	

In [ ]:

In [77]:

```
df['transmission_type'] = df['transmission_type'].astype('category').cat.codes
```

In [81]:

```
# you can drop a manual column.
fuel_dummy = pd.get_dummies(df['fuel_type'], drop_first=True)
fuel_dummy
```

Out[81]:

	Diesel	Electric	LPG	Petrol
0	0	0	0	1
1	0	0	0	1
2	0	0	0	1
3	0	0	0	1
4	1	0	0	0
...	...	...	...	...
19975	1	0	0	0
19976	0	0	0	1
19977	1	0	0	0
19978	1	0	0	0
19979	0	0	0	1

19980 rows × 4 columns

In [83]:

```
df = pd.concat([df, fuel_dummy], axis=1)
```

In [84]:

```
df.drop(columns=['fuel_type'], inplace=True)
```

In [85]:

```
df.head()
```

Out[85]:

	selling_price	year	seller_type	km_driven	transmission_type	mileage	engine	max_power
0	1.20	2012.0	Individual	120000	1	19.70	796.0	46.30
1	5.50	2016.0	Individual	20000	1	18.90	1197.0	82.00
2	2.15	2010.0	Individual	60000	1	17.00	1197.0	80.00
3	2.26	2012.0	Individual	37000	1	20.92	998.0	67.10
4	5.70	2015.0	Dealer	30000	1	22.77	1498.0	98.59

In [88]:

```
df['make'].nunique()
```

Out[88]:

```
42
```

In [92]:

```
df[df['make'] == 'Maruti']['selling_price'].mean()
```

Out[92]:

```
4.684721460177021
```

In [99]:

```
encode_make = df.groupby(by='make').mean()['selling_price']
```

In [113]:

```
df['make'].head()
```

Out[113]:

```
0      Maruti
1     Hyundai
2     Hyundai
3      Maruti
4        Ford
Name: make, dtype: object
```

In [116]:

```
make_df = encode_make.reset_index()
make_df.columns = ['make', 'encoded_make']
make_df.head()
```

Out[116]:

	make	encoded_make
0	Ambassador	1.452500
1	Audi	20.795525
2	BMW	26.547723
3	Bentley	128.250000
4	Chevrolet	2.723621

In [118]:

```
df = pd.merge(df, make_df, how='left', on='make')
```

In [93]:

```
# from category_encoders import TargetEncoder
```

In [ ]:

```

```

In [120]:

```
df.drop(columns=['seller_type', 'make', 'model'], inplace = True)
```

In [122]:

```
df.head()
```

Out[122]:

	selling_price	year	km_driven	transmission_type	mileage	engine	max_power	seats
0	1.20	2012.0	120000	1	19.70	796.0	46.30	5.0
1	5.50	2016.0	20000	1	18.90	1197.0	82.00	5.0
2	2.15	2010.0	60000	1	17.00	1197.0	80.00	5.0
3	2.26	2012.0	37000	1	20.92	998.0	67.10	5.0
4	5.70	2015.0	30000	1	22.77	1498.0	98.59	5.0

In [ ]:

```

```

In [123]:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [124]:

```
mu = df.mean()
sig = df.std()
```

In [125]:

```
df = (df-mu)/sig
```

In [126]:

```
df
```

Out[126]:

	selling_price	year	km_driven	transmission_type	mileage	engine	max_power
0	-0.680216	-0.777156	1.193910	0.496466	0.076359	-1.307192	-1.143604
1	-0.207849	0.453921	-0.739387	0.496466	-0.096799	-0.536704	-0.351690
2	-0.575856	-1.392695	0.033932	0.496466	-0.508050	-0.536704	-0.396055
3	-0.563772	-0.777156	-0.410727	0.496466	0.340425	-0.919066	-0.682209
4	-0.185878	0.146152	-0.546058	0.496466	0.740853	0.041642	0.016316
...	...	...	...	...	...	...	...
19975	-0.097996	0.761691	0.217208	0.496466	0.918340	-0.215828	-0.683318
19976	0.204099	1.377230	-0.778053	0.496466	-0.399826	-0.198535	-0.149830
19977	-0.345165	0.146152	0.169263	0.496466	0.388043	0.041642	0.125676
19978	0.533658	0.453921	72.339267	0.496466	-0.724498	1.350127	0.934892
19979	0.506195	1.377230	-0.874718	-2.014136	-0.291602	0.039721	0.438005

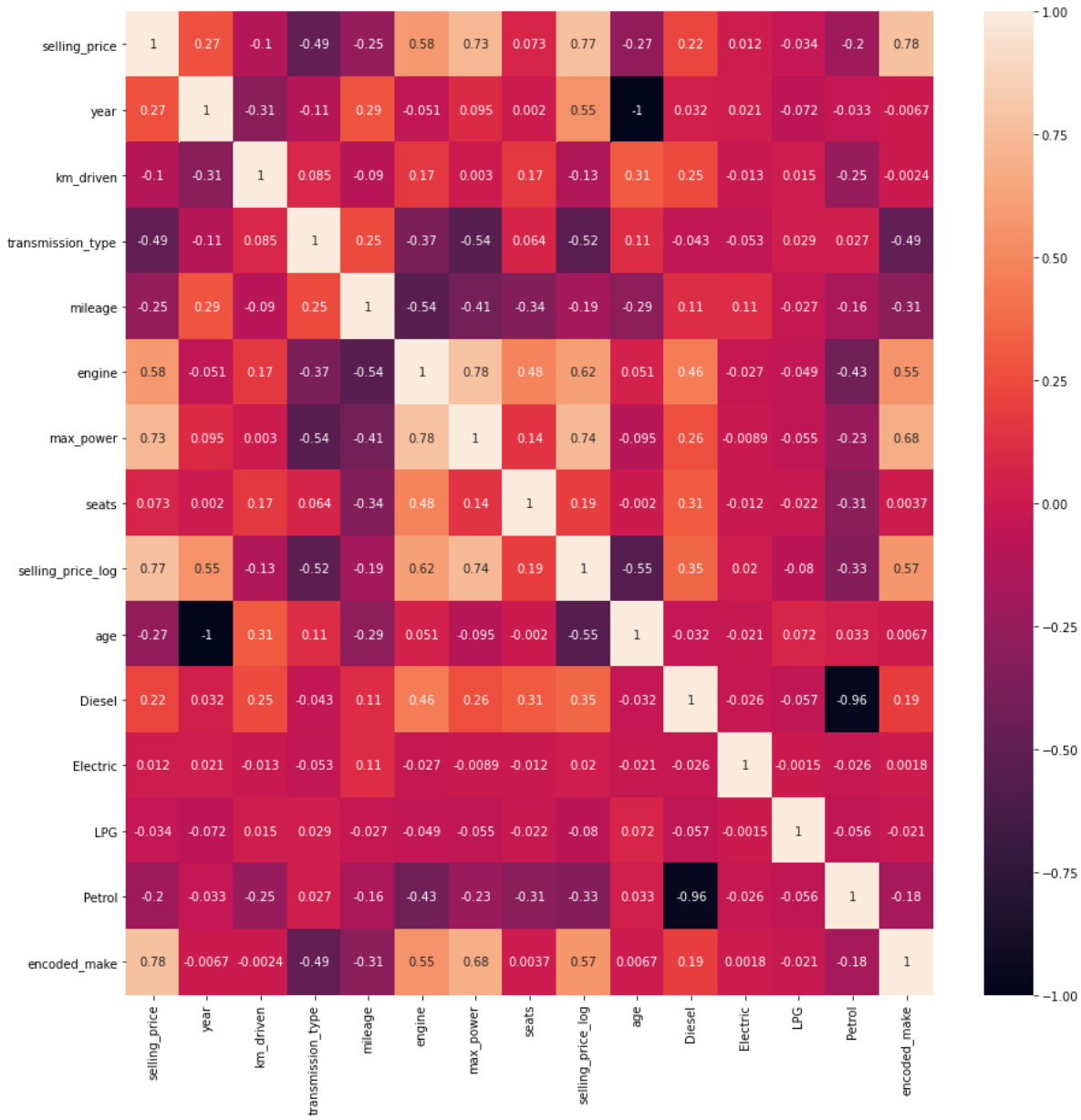
19980 rows × 15 columns

In [128]:

```
plt.figure(figsize=(15, 15))
sns.heatmap(df.corr(), annot=True)
```

Out[128]:

<AxesSubplot:>



In [ ]:

```
```

In [131]:

```
def merge_seats(x):
    if 2 <= x <= 4:
        return '2-4'
    elif x > 5:
        return '>5'
    else:
        return '5'

def preprocess(df):
    df = df.loc[df.mileage != 0].copy()
    outlier_theshold = df.selling_price.quantile(0.95)
    df.loc[df.selling_price > outlier_theshold, 'selling_price'] = outlier_theshold
    df['age'] = pd.to_datetime('now', utc=True).date().year - df['year']
    df['full_name'] = df['full_name'].str.upper()
    df['seats'] = df['seats'].apply(merge_seats)
    df.drop(columns=['year'])
    return df

def feature_engineering(df):
    df['make'] = df.full_name.apply(lambda x : x.split()[0])
    df['model'] = df.full_name.apply(lambda x : " ".join(x.split()[1:]))
    one_hot_encode_cols = ['seller_type', 'fuel_type', 'transmission_type', 'seats']
    for column in one_hot_encode_cols:
        df = pd.concat([df, pd.get_dummies(df[column]).iloc[: , 1:]], axis=1)
    df.drop(columns=one_hot_encode_cols + ['full_name'], inplace=True)
    df['make'] = df.groupby('make')['selling_price'].transform('median')
    df['model'] = df.groupby('model')['selling_price'].transform('median')
    scaler = MinMaxScaler()
    df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
    return df
```

In [ ]:

```
```

In [132]:

```
df = pd.read_csv('cars24.csv')
df = preprocess(df)
df = feature_engineering(df)
df.head()
```

Out[132]:

ear	km_driven	mileage	engine	max_power	age	make	model	Individual	Trustma Dea
655	0.031553	0.135345	0.117891	0.066506	0.310345	0.184371	0.042522	1.0	
586	0.005237	0.128448	0.177281	0.123994	0.172414	0.207231	0.225975	1.0	
690	0.015764	0.112069	0.177281	0.120773	0.379310	0.207231	0.120277	1.0	
655	0.009711	0.145862	0.147808	0.100000	0.310345	0.184371	0.093549	1.0	
103	0.007869	0.161810	0.221860	0.150709	0.206897	0.224624	0.300085	0.0	

In [ ]:

```

```

In [ ]:

```

```