In [1]:
```python
import pandas as pd
import numpy as np
```

In [3]:
```python
data=pd.read_csv(r"D:\ML\real_estate_price_size_year_view  3rd march new.csv")
data
```

Out[3]:

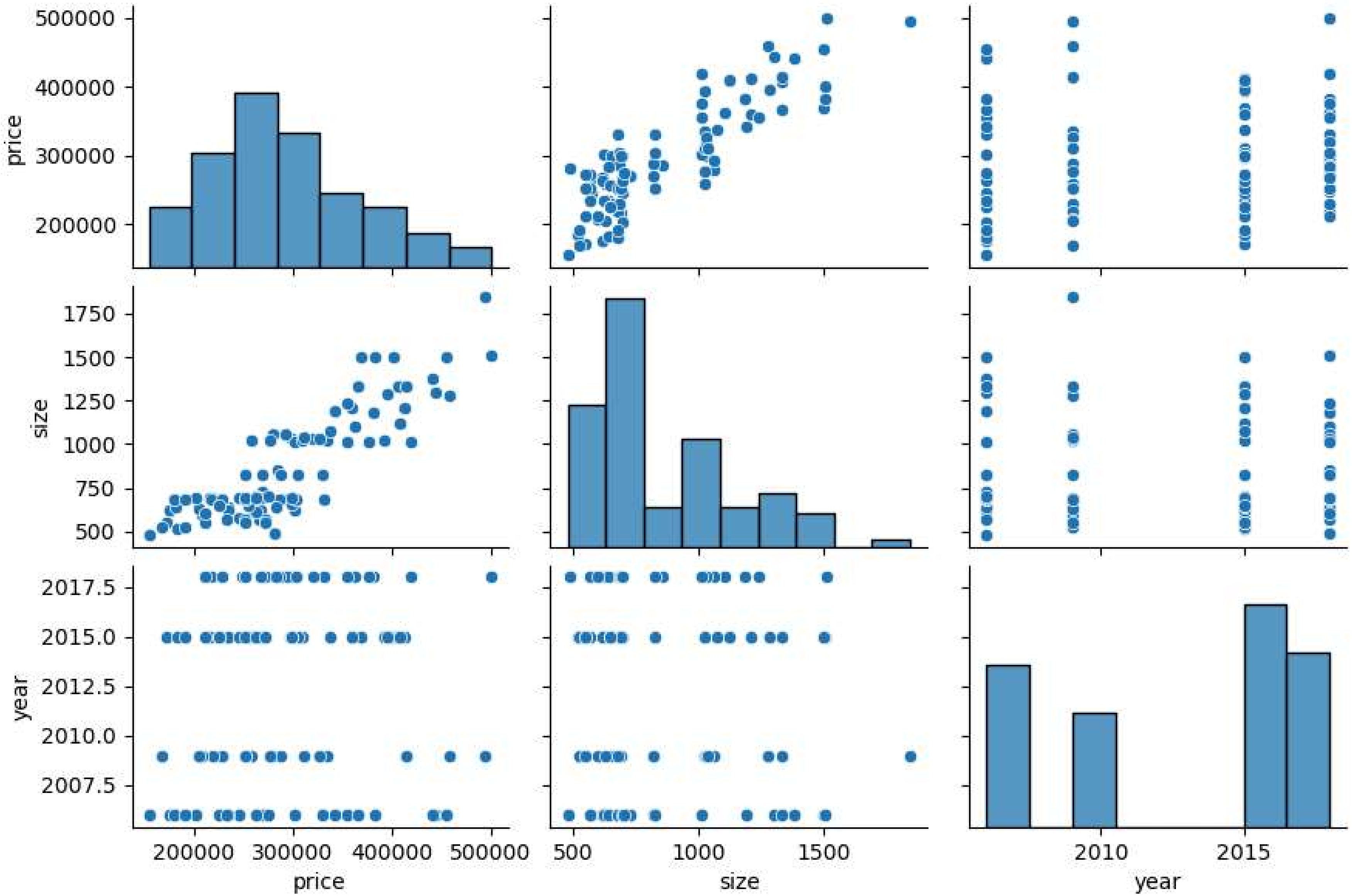
	price	size	year	view
0	234314.144	643.09	2015	No sea view
1	228581.528	656.22	2009	No sea view
2	281626.336	487.29	2018	Sea view
3	401255.608	1504.75	2015	No sea view
4	458674.256	1275.46	2009	Sea view
...	...	...	...	...
95	252460.400	549.80	2009	Sea view
96	310522.592	1037.44	2009	No sea view
97	383635.568	1504.75	2006	No sea view
98	225145.248	648.29	2015	No sea view
99	274922.856	705.29	2006	Sea view

100 rows × 4 columns

In [4]:
```python
import seaborn as sns
sns.pairplot(data,height=2,aspect=1.5)
```

```
D:\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[4]:
```
<seaborn.axisgrid.PairGrid at 0x25dd80ab050>
```



In [7]:
```
data.describe()
```

Out[7]:

	price	size	year
count	100.000000	100.000000	100.000000
mean	292289.470160	853.024200	2012.600000
std	77051.727525	297.941951	4.729021
min	154282.128000	479.750000	2006.000000
25%	234280.148000	643.330000	2009.000000
50%	280590.716000	696.405000	2015.000000
75%	335723.696000	1029.322500	2018.000000
max	500681.128000	1842.510000	2018.000000

In [8]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   price   100 non-null    float64
 1   size    100 non-null    float64
 2   year    100 non-null    int64  
 3   view    100 non-null    object 
dtypes: float64(2), int64(1), object(1)
memory usage: 3.3+ KB
```

In [15]:

```
data1=pd.get_dummies(data)
data1
```

Out[15]:

	price	size	year	view_No sea view	view_Sea view
0	234314.144	643.09	2015	True	False
1	228581.528	656.22	2009	True	False
2	281626.336	487.29	2018	False	True
3	401255.608	1504.75	2015	True	False
4	458674.256	1275.46	2009	False	True
...	...	...	...	...	...
95	252460.400	549.80	2009	False	True
96	310522.592	1037.44	2009	True	False
97	383635.568	1504.75	2006	True	False
98	225145.248	648.29	2015	True	False
99	274922.856	705.29	2006	False	True

100 rows × 5 columns

In [18]:

```
x=data1.drop('size' ,axis='columns')
y=data1.price
print(x)
print(y)
```

```
         price  year  view_No sea view  view_Sea view
0   234314.144  2015              True          False
1   228581.528  2009              True          False
2   281626.336  2018             False           True
3   401255.608  2015              True          False
4   458674.256  2009             False           True
..         ...   ...               ...            ...
95  252460.400  2009             False           True
96  310522.592  2009              True          False
97  383635.568  2006              True          False
98  225145.248  2015              True          False
99  274922.856  2006             False           True

[100 rows x 4 columns]
0     234314.144
1     228581.528
2     281626.336
3     401255.608
4     458674.256
         ...
95    252460.400
96    310522.592
97    383635.568
98    225145.248
99    274922.856
Name: price, Length: 100, dtype: float64
```

In [19]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.4, random_state=2)
```

In [20]:

```
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

	price	year	view_No sea view	view_Sea view
12	215472.104	2015	True	False
53	269523.056	2006	False	True
87	327252.112	2009	False	True
54	255629.160	2015	False	True
95	252460.400	2009	False	True
32	207742.248	2009	True	False
19	299416.976	2018	True	False
26	271793.312	2018	False	True
60	251188.824	2018	True	False
55	500681.128	2018	False	True
9	218630.608	2009	True	False
96	310522.592	2009	True	False
17	234178.160	2006	False	True
59	251332.592	2015	False	True
57	395242.096	2015	False	True
41	217468.224	2018	True	False
64	302393.384	2015	False	True
45	300061.480	2015	False	True
97	383635.568	2006	True	False
8	331101.344	2018	False	True
71	181587.576	2006	True	False
94	262477.856	2006	False	True
90	251140.656	2018	True	False
98	225145.248	2015	True	False
86	154282.128	2006	True	False
80	180307.216	2006	True	False
50	225656.120	2015	True	False
52	258637.008	2009	True	False
66	355251.200	2006	False	True
88	211904.536	2018	True	False
70	276875.632	2009	True	False
46	204302.976	2009	True	False
68	294582.944	2018	False	True
69	454512.760	2006	False	True
81	408637.816	2015	False	True
58	330677.128	2006	False	True
33	191486.896	2015	True	False
38	292965.216	2018	False	True
51	393069.760	2015	False	True
42	287350.000	2009	False	True
4	458674.256	2009	False	True
67	271726.752	2015	False	True
39	245747.200	2015	True	False

37	233493.208	2006	False	True
20	268125.080	2015	False	True
31	225452.320	2006	True	False
63	334938.872	2009	False	True
47	201778.048	2006	True	False
85	376253.808	2018	False	True
93	266684.248	2018	True	False
49	262423.504	2015	False	True
34	285223.176	2018	True	False
7	175716.480	2006	True	False
75	286161.600	2018	False	True
82	190909.056	2006	True	False
43	414682.648	2009	True	False
22	412569.472	2015	False	True
72	298926.496	2015	False	True
15	440201.616	2006	False	True
40	310045.712	2015	True	False
	price	year	view_No sea view	view_Sea view
83	282683.544	2018	False	True
30	301635.728	2006	True	False
56	320345.520	2018	True	False
24	168047.264	2009	True	False
16	248337.600	2018	True	False
23	183459.488	2015	True	False
2	281626.336	2018	False	True
27	406852.304	2015	False	True
28	297760.440	2015	False	True
13	418753.008	2018	True	False
99	274922.856	2006	False	True
92	298170.880	2015	False	True
76	382120.152	2018	False	True
14	444192.008	2006	False	True
0	234314.144	2015	True	False
21	171795.240	2015	True	False
3	401255.608	2015	True	False
29	368988.432	2015	True	False
61	263311.696	2015	True	False
79	342988.456	2006	False	True
35	302000.920	2018	False	True
11	494778.992	2009	False	True
84	303597.216	2018	False	True
44	293044.496	2018	True	False
73	211724.096	2015	True	False
5	245050.280	2006	False	True

25	362519.720	2018	False	True
77	365863.936	2006	True	False
74	228313.024	2018	True	False
62	359674.440	2015	True	False
65	304587.272	2015	True	False
1	228581.528	2009	True	False
18	225451.984	2006	True	False
48	257828.416	2015	False	True
36	269225.920	2006	True	False
78	251560.040	2009	False	True
6	265129.064	2015	False	True
89	354512.112	2018	True	False
91	338078.168	2015	True	False
10	279555.096	2009	True	False

12	215472.104
53	269523.056
87	327252.112
54	255629.160
95	252460.400
32	207742.248
19	299416.976
26	271793.312
60	251188.824
55	500681.128
9	218630.608
96	310522.592
17	234178.160
59	251332.592
57	395242.096
41	217468.224
64	302393.384
45	300061.480
97	383635.568
8	331101.344
71	181587.576
94	262477.856
90	251140.656
98	225145.248
86	154282.128
80	180307.216
50	225656.120
52	258637.008
66	355251.200
88	211904.536

```
70    276875.632
46    204302.976
68    294582.944
69    454512.760
81    408637.816
58    330677.128
33    191486.896
38    292965.216
51    393069.760
42    287350.000
4     458674.256
67    271726.752
39    245747.200
37    233493.208
20    268125.080
31    225452.320
63    334938.872
47    201778.048
85    376253.808
93    266684.248
49    262423.504
34    285223.176
7     175716.480
75    286161.600
82    190909.056
43    414682.648
22    412569.472
72    298926.496
15    440201.616
40    310045.712
Name: price, dtype: float64
83    282683.544
30    301635.728
56    320345.520
24    168047.264
16    248337.600
23    183459.488
2     281626.336
27    406852.304
28    297760.440
13    418753.008
99    274922.856
92    298170.880
76    382120.152
```

```
14    444192.008
0     234314.144
21    171795.240
3     401255.608
29    368988.432
61    263311.696
79    342988.456
35    302000.920
11    494778.992
84    303597.216
44    293044.496
73    211724.096
5     245050.280
25    362519.720
77    365863.936
74    228313.024
62    359674.440
65    304587.272
1     228581.528
18    225451.984
48    257828.416
36    269225.920
78    251560.040
6     265129.064
89    354512.112
91    338078.168
10    279555.096
Name: price, dtype: float64
```

In [21]:
```
from sklearn.linear_model import LinearRegression
```

In [22]:
```
equation=LinearRegression()
```

In [23]:
```
equation.fit(x_train, y_train)
```

Out[23]:
```
▾ LinearRegression
LinearRegression()
```

In [24]:
```
equation.intercept_
```

Out[24]: -1.3969838619232178e-09

In [25]:
```
equation.coef_
```

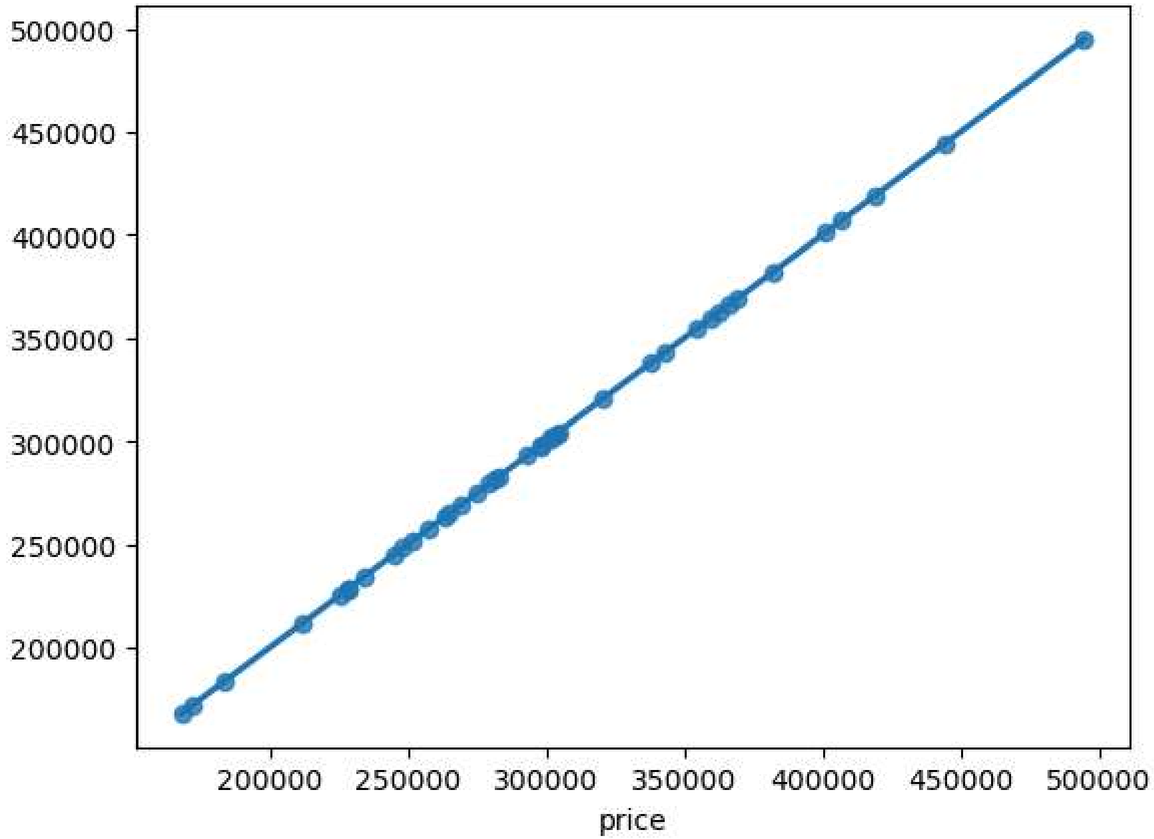
Out[25]: array([1.00000000e+00, 6.51492980e-13, 2.48453213e-12, -2.48453213e-12])

In [26]:
```
y_test_predicted=equation.predict(x_test)
y_test_predicted
```

Out[26]:
```
array([282683.544, 301635.728, 320345.52 , 168047.264, 248337.6  ,
       183459.488, 281626.336, 406852.304, 297760.44 , 418753.008,
       274922.856, 298170.88 , 382120.152, 444192.008, 234314.144,
       171795.24 , 401255.608, 368988.432, 263311.696, 342988.456,
       302000.92 , 494778.992, 303597.216, 293044.496, 211724.096,
       245050.28 , 362519.72 , 365863.936, 228313.024, 359674.44 ,
       304587.272, 228581.528, 225451.984, 257828.416, 269225.92 ,
       251560.04 , 265129.064, 354512.112, 338078.168, 279555.096])
```

In [27]:
```
sns.regplot(x=y_test, y=y_test_predicted)
```

Out[27]: <Axes: xlabel='price'>


500000
450000
400000
350000
300000
250000
200000
200000
250000
300000
350000
400000
450000
500000
price

```
In [ ]:
```