House sales in King County, Seattle

00. Introduction

1) 팀원

- 홍준표
- 윤강열
- 윤현선

2) 목표: test data의 sales price(집값) 예측

3) 데이터

- 위치: 미국 Washington주의 King County
- 총 data : 21,613개의 집값 데이터
  - 출처 (https://www.kaggle.com/harlfoxem/housesalesprediction)
- Train Datasets: 70%
- Test Datasets: 30%

4) 평가기준

- R-squared (R^2)

5) 진행순서

- **데이터 탐색(EDA)**
  - load datasets
  - target data
  - independent variables
    - pairplot
    - heatmaps
    - histograms
    - target vs. X scatter plots
  - 다중공선성
  - 이상값제거
  - category변환

- **Modeling**
  - train data 전처리
  - final model
    - performance
    - 부분회귀

- **Prediction**
  - test data 전처리
  - test R-squared

0. Import Packages and Load Data

In [1]:

```
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
from scipy import stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from scipy.stats import norm, skew
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import warnings
warnings.simplefilter('ignore')

fs = (13, 7)
```

In [2]:

```
with open("data.pkl", "rb") as f:
    df = pickle.load(f)
```

In [3]:

```
# load csv

# df = pd.read_csv("data/kc_house_data.csv")
```

In [4]:

```
df.head().T
```

Out[4]:

	0	1	2	3	4
bathrooms	1	2.25	1	3	2
bedrooms	3	3	2	4	3
condition	3	3	3	5	3
date	2014-10-13 00:00:00	2014-12-09 00:00:00	2015-02-25 00:00:00	2014-12-09 00:00:00	2015-02-18 00:00:00
date_ord	735519	735576	735654	735576	735647
floors	1	2	1	1	1
grade	7	7	6	7	8
lat	47.5112	47.721	47.7379	47.5208	47.6168
long	-122.257	-122.319	-122.233	-122.393	-122.045
sqft_above	1180	2170	770	1050	1680
sqft_basement	0	400	0	910	0
sqft_living	1180	2570	770	1960	1680
sqft_living15	1340	1690	2720	1360	1800
sqft_lot	5650	7242	10000	5000	8080
sqft_lot15	5650	7639	8062	5000	7503
view	0	0	0	0	0
waterfront	0	0	0	0	0
yr_built	1955	1951	1933	1965	1987
yr_renovated	0	1991	0	0	0
zipcode	98178	98125	98028	98136	98074
price	221900	538000	180000	604000	510000

In [5]:

```
df.dtypes
```

Out[5]:

```
bathrooms               float64
bedrooms                  int64
condition                 int64
date             datetime64[ns]
date_ord                  int64
floors                  float64
grade                     int64
lat                     float64
long                    float64
sqft_above                int64
sqft_basement             int64
sqft_living               int64
sqft_living15             int64
sqft_lot                  int64
sqft_lot15                int64
view                      int64
waterfront                int64
yr_built                  int64
yr_renovated              int64
zipcode                   int64
price                   float64
dtype: object
```

1. Preprocessing

Columns

- id: a notation for a house
- date: Date house was sold
- price: Price is prediction target
- bedrooms: Number of Bedrooms/House
- bathrooms: Number of bathrooms/House
- sqft_living: square footage of the home
- sqft_lot: square footage of the lot
- floors: Total floors (levels) in house
- waterfront: House which has a view to a waterfront
- view: Has been viewed
- condition: How good the condition is (Overall)
- grade: overall grade given to the housing unit, based on King County grading system
- sqft_above: square footage of house apart from basement
- sqft_basement: square footage of the basement
- yr_built: Built Year
- yr_renovated: Year when house was renovated
- zipcode: zip
- lat: Latitude coordinate
- long: Longitude coordinate
- sqft_living15: Living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area
- sqft_lot15: lotSize area in 2015(implies-- some renovations)

Y	Dependent variables	Type	Description	Minimun, Mean, Max
1	price	numerical	Target for prediction	75000, 540088, 7700000

X	Independent variables	Type	Description	Minimun, Mean, Max
-	Id	-	a notation for a house	
1	date	-	date house was sold(May 2014 - May 2015)	735355, 735535, 735745(ordinal)
2	bedrooms	numerical	number of bedrooms	0, 3.37, 33
3	bathrooms	numerical	number of bathrooms	0, 2.12, 8
4	floors	categorical	total floors (levels) in house	1, 1.5, 3.5
5	waterfront	categorical	house which has a view to a waterfront	0, 0.08, 1
6	sqft_living	numerical	square footage of the home	290, 2079.9, 13540
7	sqft_living15	numerical	living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area	399, 1986.55, 6210
8	sqft_above	numerical	square footage of house apart from basement	0, 1.06, 28
9	sqft_basement	numerical	square footage of the basement	0, 117.32, 2490
10	sqft_lot	numerical	square footage of the lot	520, 15107, 1651359
11	sqft_lot15	numerical	lotSize area in 2015(implies-- some renovations)	651, 12768.46, 871200
12	view	categorical	has been viewed	0, 0.23, 4
13	condition	categorical	how good the condition is (Overall)	1, 3.41, 5

X	Independent variables	Type	Description	Minimun, Mean, Max
14	grade	categorical	overall grade given to the housing unit, based on King County grading system	1, 7.66, 13
15	yr_built	numerical	built Year	1900, 1971, 2015
16	yr_renovated	numerical	Year when house was renovated	0, 84.4, 2015
17	zipcode	categorical	zipcode	98001, 98077.94, 98199(70 types)
18	lat	-	latitude coordinate	47.16, 47.56, 47.78
19	long	-	longitude coordinate	-122.52, -122.21, -121.32

In [6]:

```
df.describe()
```

Out[6]:

	bathrooms	bedrooms	condition	date_ord	floors	grade	2
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	2
mean	2.114757	3.370842	3.409430	735535.193078	1.494309	7.656873	
std	0.770163	0.930062	0.650743	113.048011	0.539989	1.175459	
min	0.000000	0.000000	1.000000	735355.000000	1.000000	1.000000	
25%	1.750000	3.000000	3.000000	735436.000000	1.000000	7.000000	
50%	2.250000	3.000000	3.000000	735522.000000	1.500000	7.000000	
75%	2.500000	4.000000	4.000000	735646.000000	2.000000	8.000000	
max	8.000000	33.000000	5.000000	735745.000000	3.500000	13.000000	

- Comments
  - Count값 동일, is null 확인: missig data 없음
  - Mean vs. Median(50%) 몇몇 데이터의 치우침이 있음 ex) sqft_living

Convert date to datetime64 and add column date_ord (ordinal)

In [7]:

```
df.date = df.date.astype("datetime64")
df["date_ord"] = df.date.apply(lambda x: x.toordinal())
```

In [8]:

```
# sort columns and move 'price' (y) to the last position

# cols = list(df.columns)
# cols.sort()
# cols.remove("price")
# cols.append("price")
# df = df[cols]
```

In [9]:

```
# check for any null values

# before = len(df)
# df = df.dropna()
# after = len(df)
# before, after, before == after
```

In [10]:

```
# drop column id

# df = df.drop("id", axis=1)
```

In [11]:

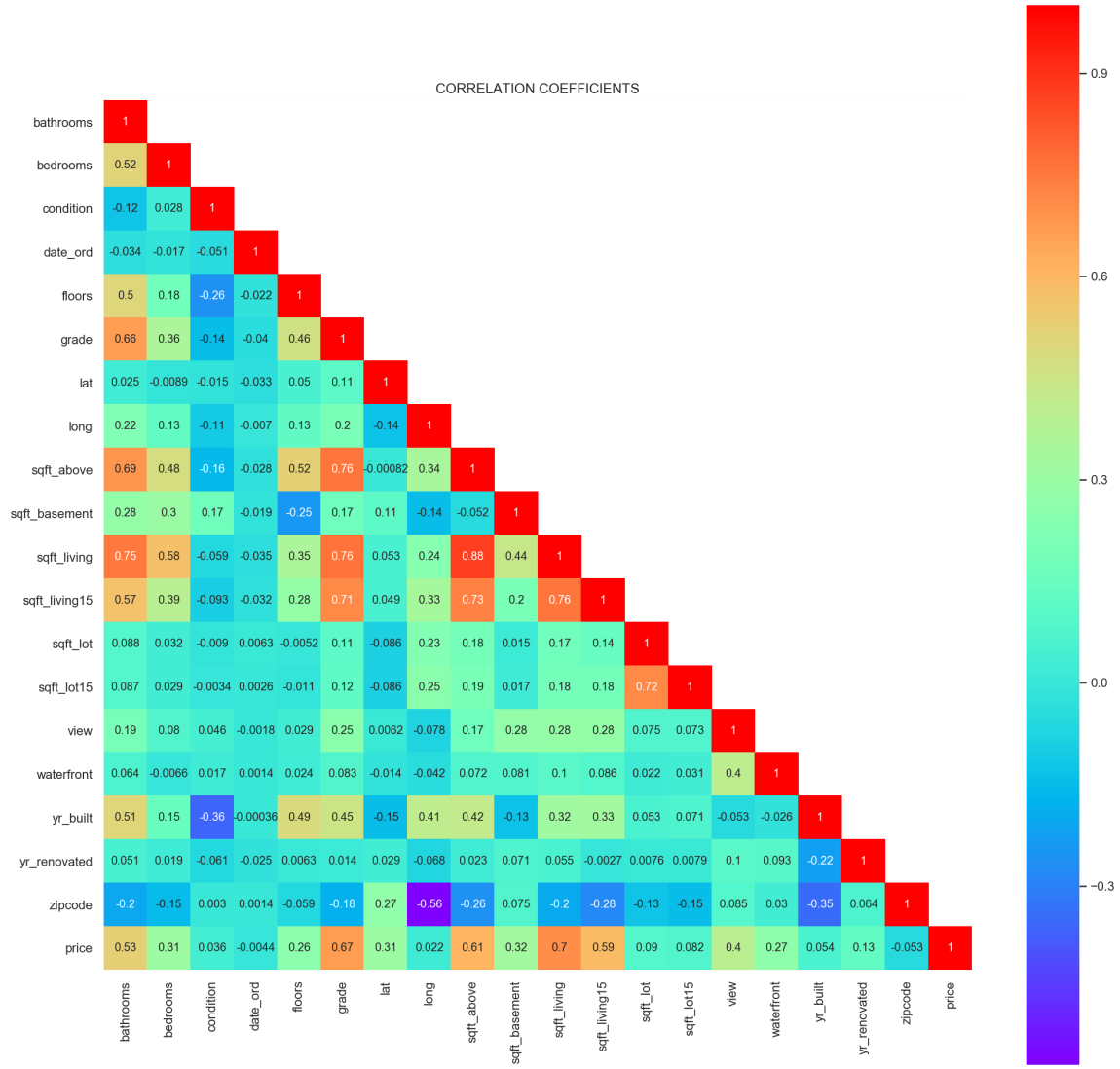
```
# save as binary file
# with open("data/data.pkl", "wb") as f:
#     pickle.dump(df, f)
```

2. EDA

2.1 Heatmap

In [12]:

```
r = df.corr()
r_sq = r.apply(lambda x: x ** 2)
mask = np.zeros_like(r, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
mask[np.diag_indices_from(mask)] = False

# heatmap of R between columns
plt.figure(figsize=(17, 17))
sns.heatmap(r, annot=True, cmap="rainbow", square=True, mask=mask).set_title("CORREI
plt.savefig("figures/heatmap_r.png")
plt.show()
```



2.1 Pairplot

In [13]:

```
# pairplot
sns.pairplot(df)
plt.savefig("figures/pairplot.png")
plt.show()
```

```
---------------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
 last)
<ipython-input-13-51cd85c34d10> in <module>
      2 sns.pairplot(df)
      3 plt.savefig("figures/pairplot.png")
----> 4 plt.show()

/anaconda3/lib/python3.7/site-packages/matplotlib/pyplot.py in show(*a
rgs, **kw)
    261     """
    262     global _show
--> 263     return _show(*args, **kw)
    264
    265

/anaconda3/lib/python3.7/site-packages/ipykernel/pylab/backend_inline.
py in show(close, block)
```

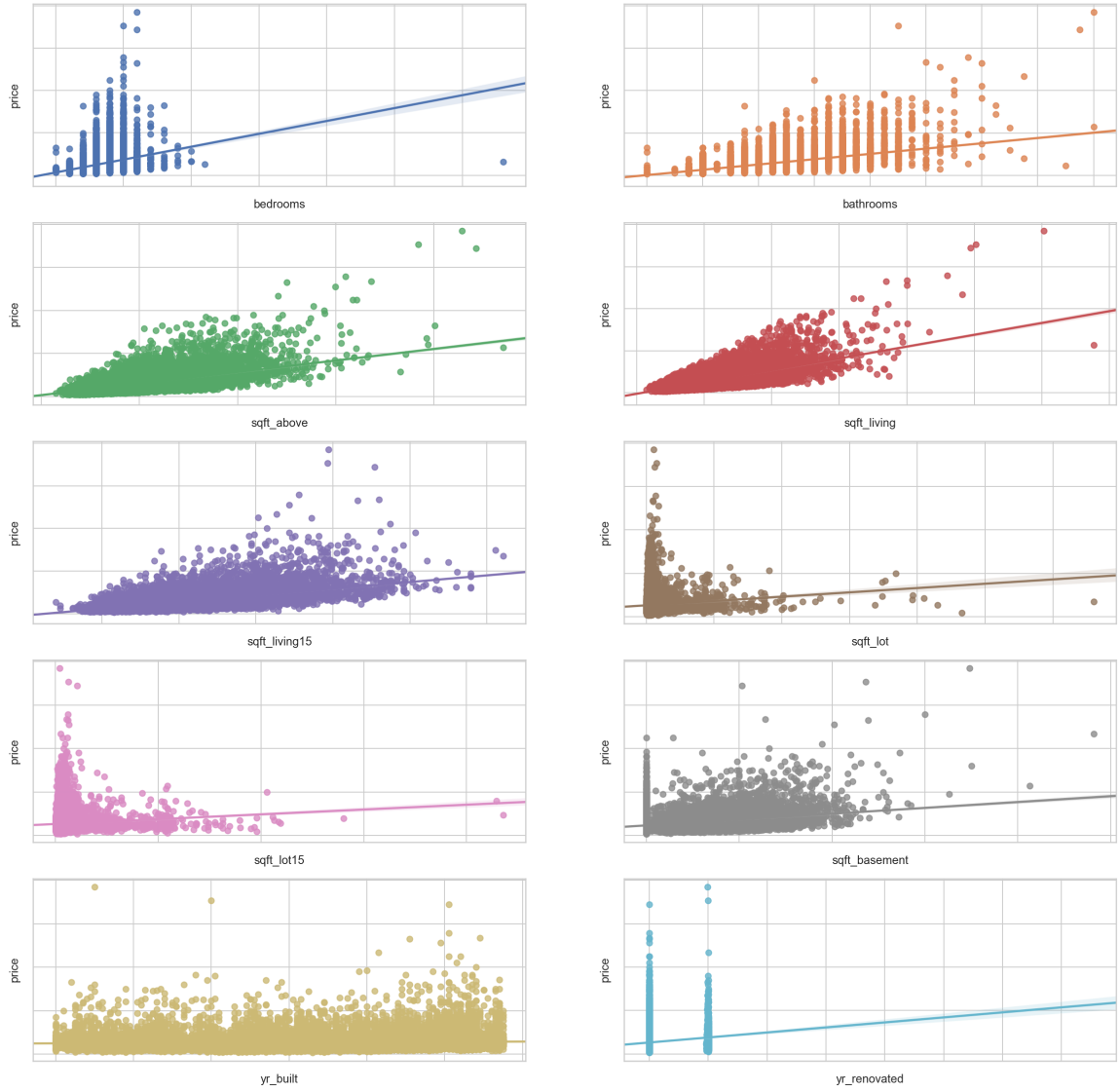
In [13]:

```
col_cat = [
    "condition",
    "floors",
    "waterfront",
    "zipcode",
    "grade"
]

col_num = [
    "bedrooms",
    "bathrooms",
    "sqft_above",
    "sqft_living",
    "sqft_living15",
    "sqft_lot",
    "sqft_lot15",
    "sqft_basement",
    "yr_built",
    "yr_renovated",
]
```

(1) Numerical 변수와 Price과의 상관그래프

In [14]:

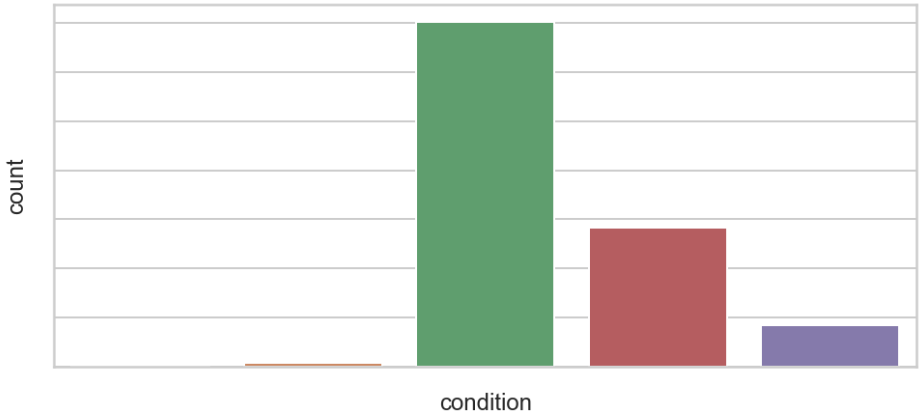
```
fig, ax = plt.subplots(5,2, figsize = (20,20))

for idx, n in enumerate(col_num):
    if n == 'price':
        continue
    sns.regplot(x=n, y='price', data=df, ax = ax[idx//2,idx%2])
    ax[idx//2, idx % 2].set(yticklabels=[])
    ax[idx//2, idx % 2].set(xticklabels=[])

    continue
```

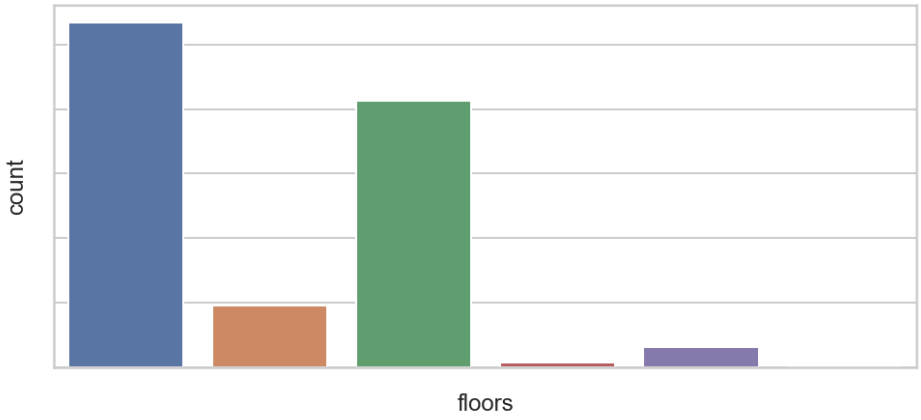


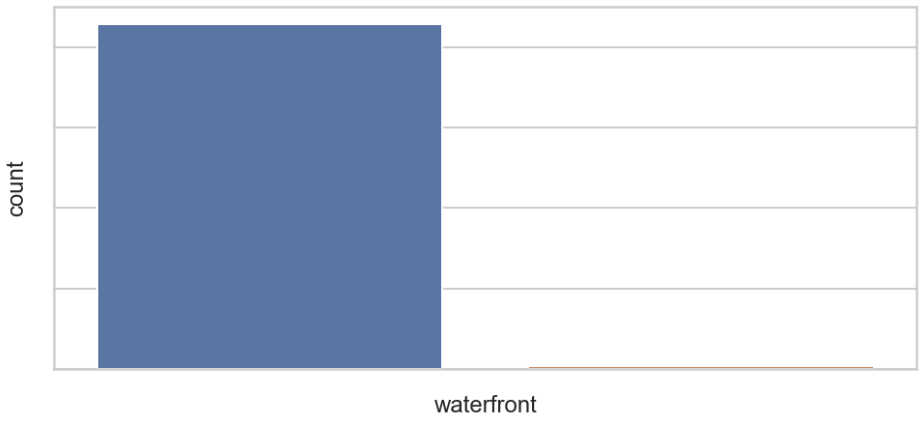
(2) Categorical 변수의 그래프

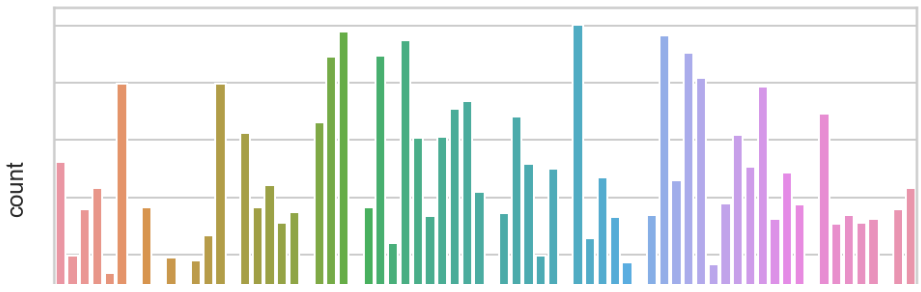
In [15]:

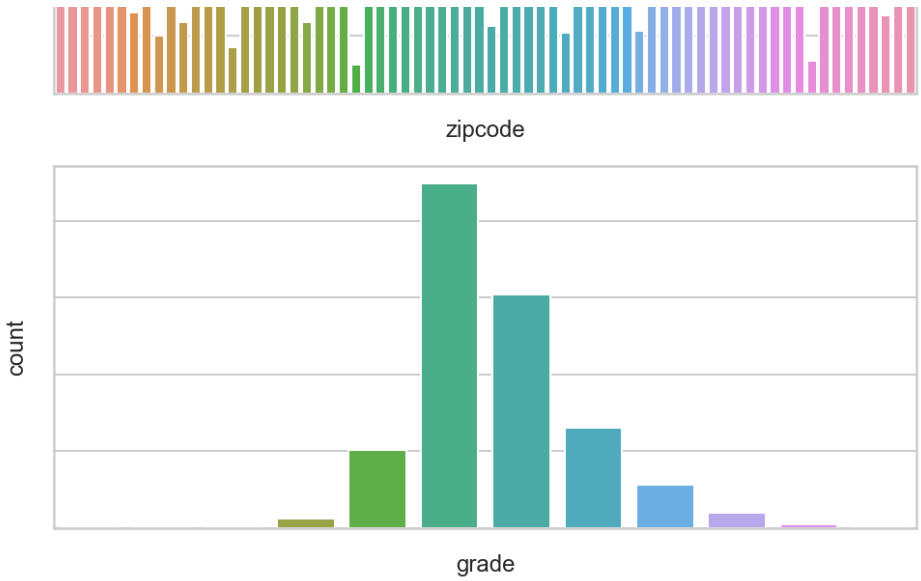
```
fig, ax = plt.subplots(5, figsize = (8,20))

for idx, n in enumerate(col_cat):
    if n == 'price':
        continue
    sns.countplot(x=n, data=df, ax = ax[idx])
    ax[idx].set(yticklabels=[])
    ax[idx].set(xticklabels=[])

    continue
```







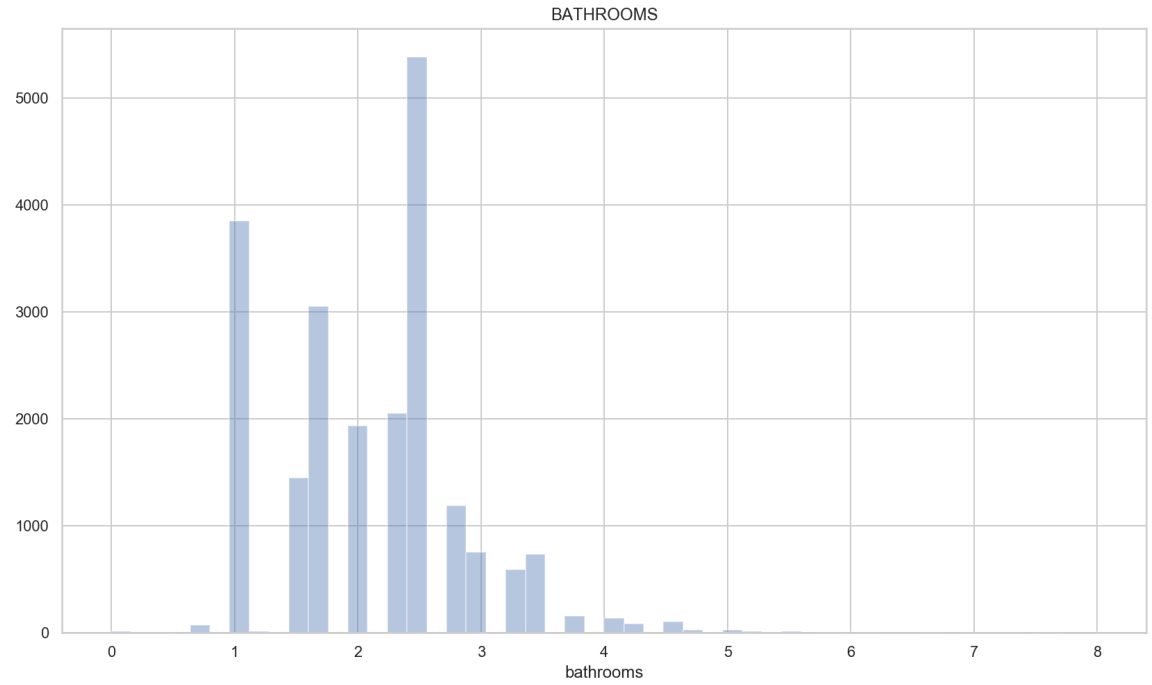




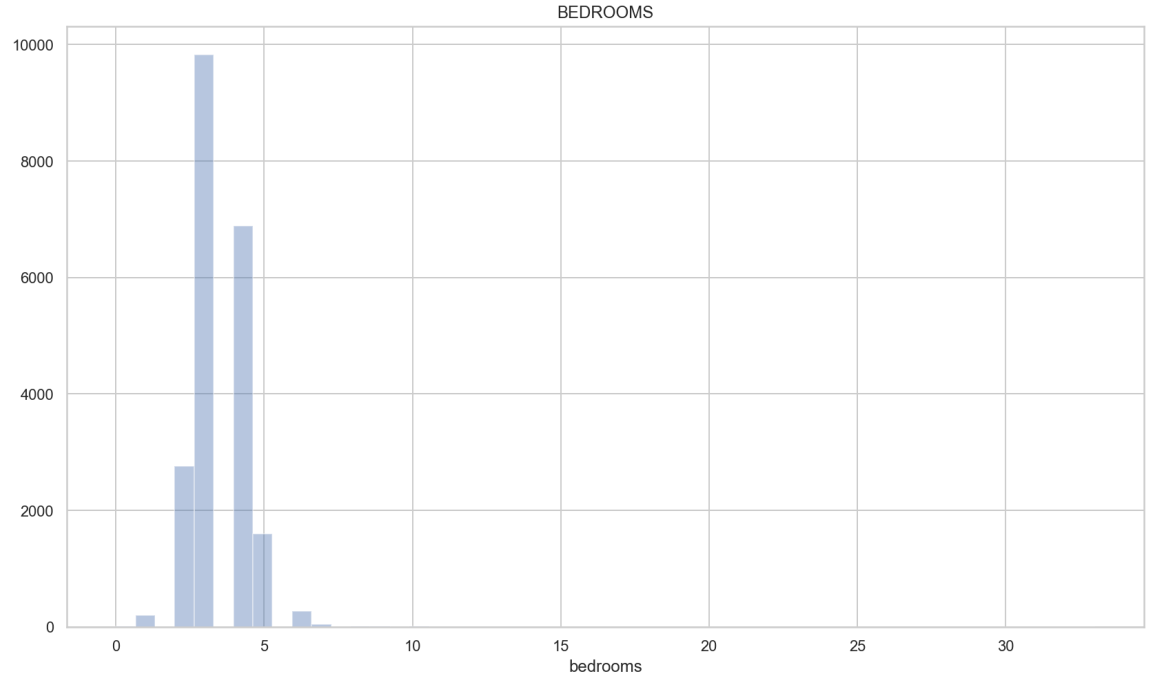
2.3 Histograms

In [20]:

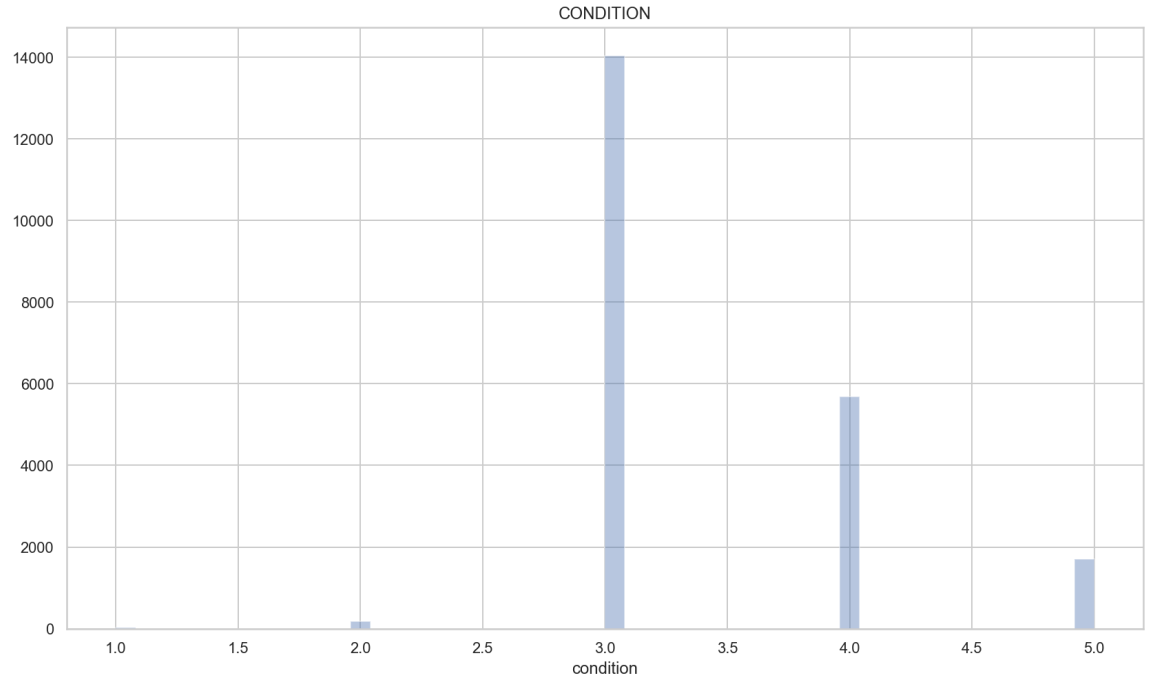
```
cols = list(df.columns)
cols.remove("date")

for col in cols:
    plt.figure(figsize=fs)
    sns.distplot(df[col], kde=False).set_title("{}".format(col.upper()))
    plt.savefig("figures/histograms/histogram_{}.png".format(col))
    plt.show()
    print(df[col].describe())
```



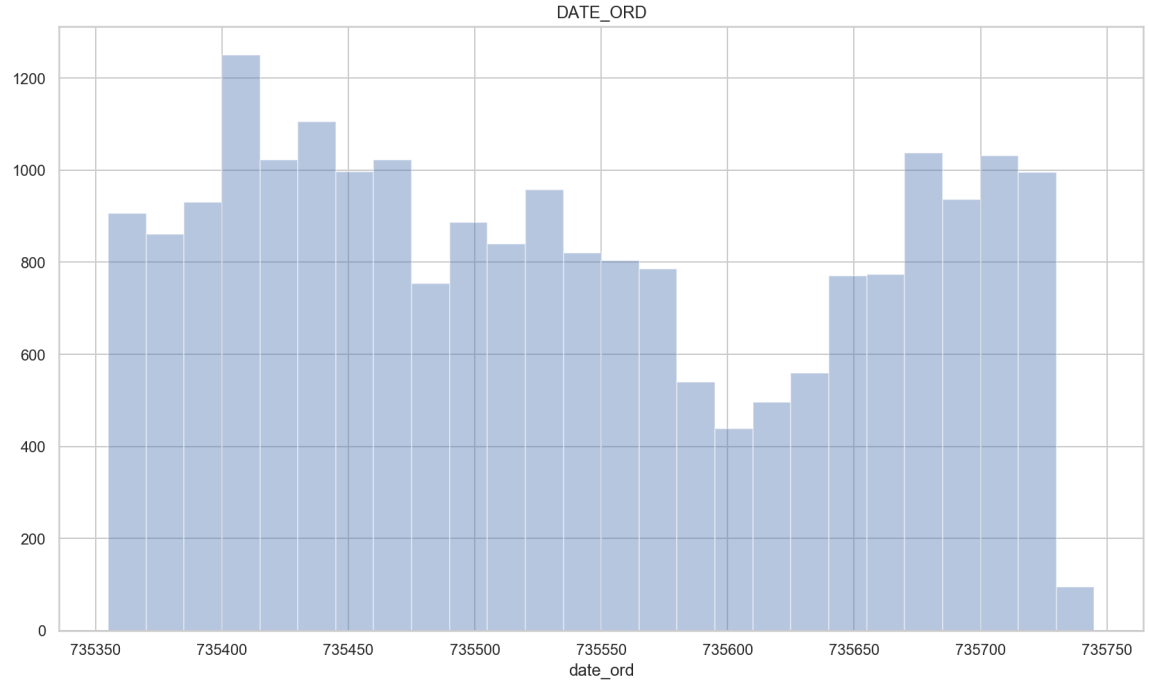

```
count    21613.000000
mean         2.114757
std          0.770163
min          0.000000
25%          1.750000
50%          2.250000
75%          2.500000
max          8.000000
Name: bathrooms, dtype: float64
```



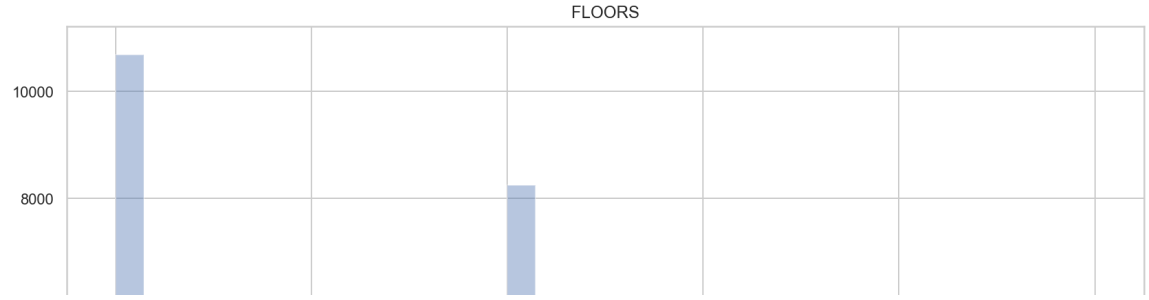

```
count    21613.000000
mean         3.370842
std          0.930062
min          0.000000
25%          3.000000
50%          3.000000
75%          4.000000
max         33.000000
Name: bedrooms, dtype: float64
```



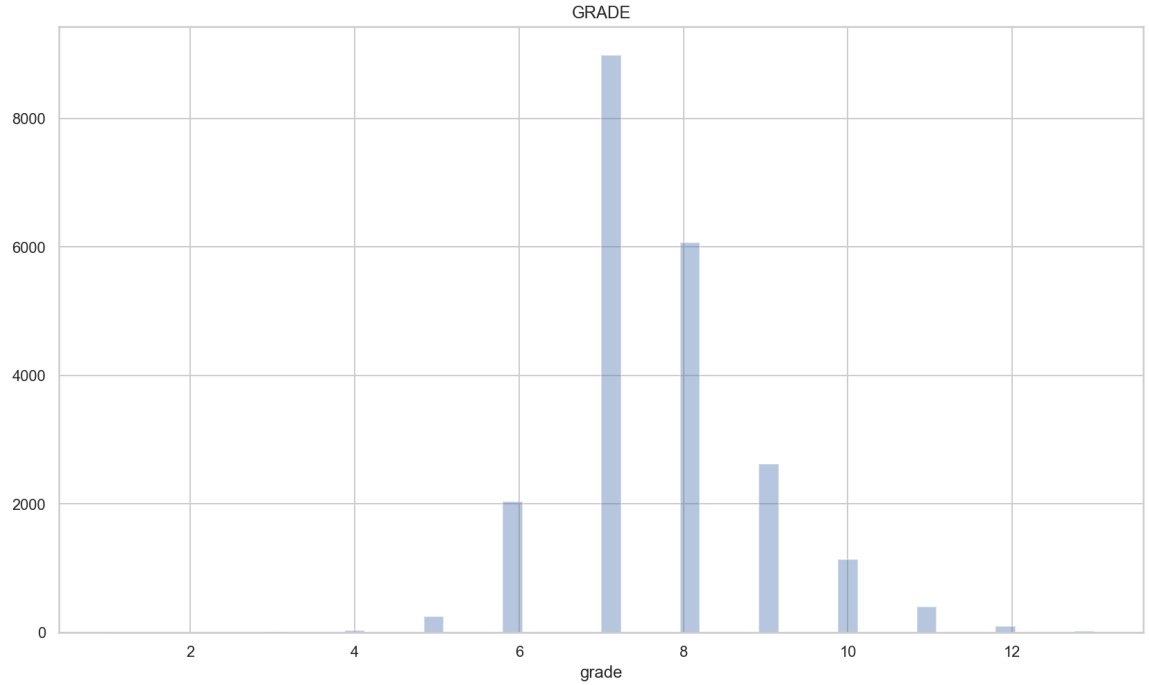
```
count    21613.000000
mean         3.409430
std          0.650743
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          5.000000
Name: condition, dtype: float64
```



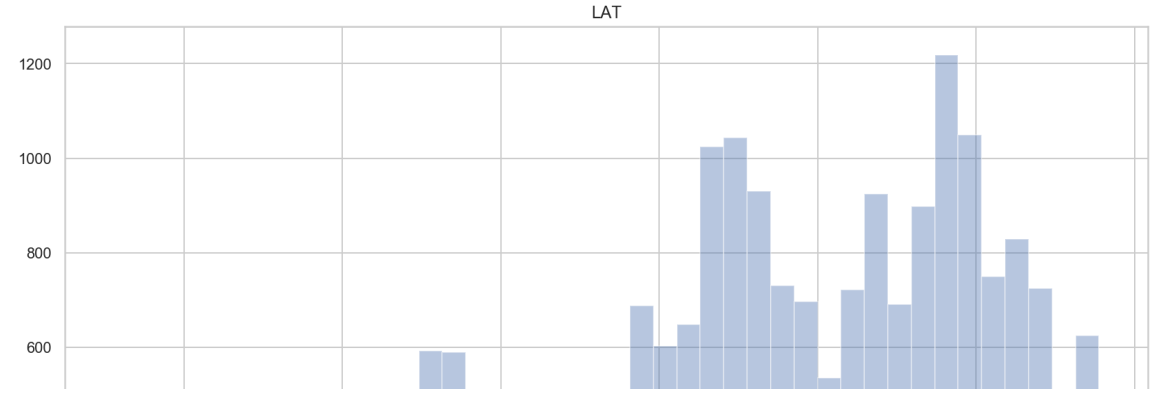

```
count     21613.000000
mean     735535.193078
std         113.048011
min      735355.000000
25%      735436.000000
50%      735522.000000
75%      735646.000000
max      735745.000000
Name: date_ord, dtype: float64
```



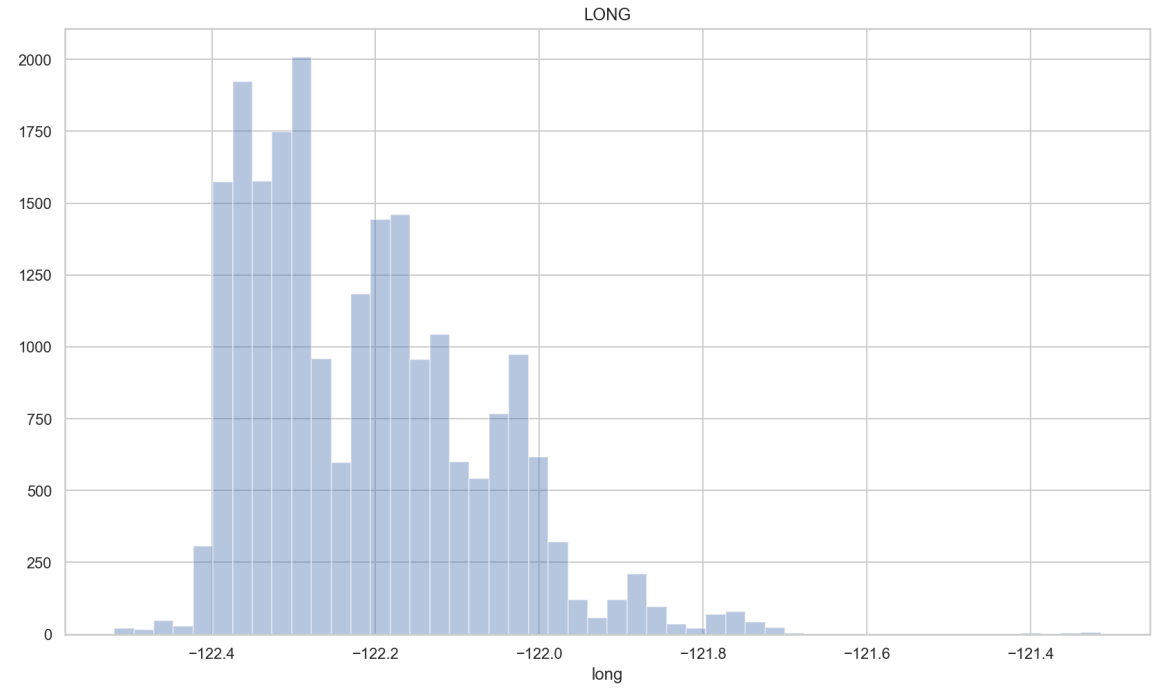

```
count    21613.000000
mean         1.494309
std          0.539989
min          1.000000
25%          1.000000
50%          1.500000
75%          2.000000
max          3.500000
Name: floors, dtype: float64
```



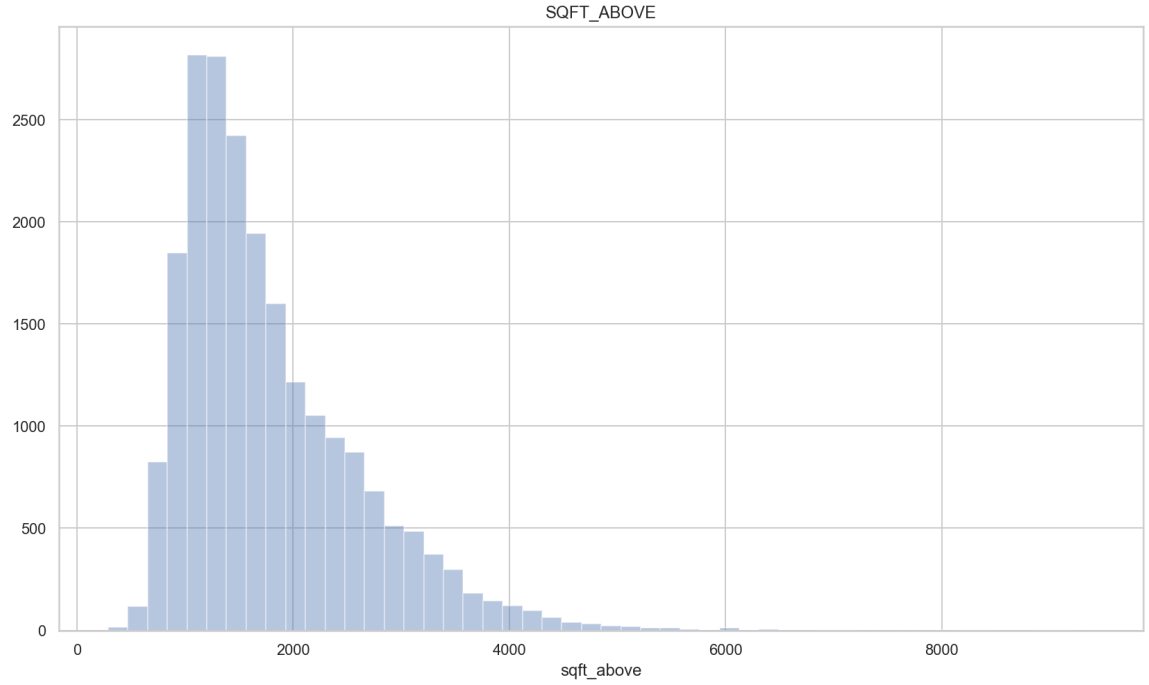

```
count    21613.000000
mean         7.656873
std          1.175459
min          1.000000
25%          7.000000
50%          7.000000
75%          8.000000
max         13.000000
Name: grade, dtype: float64
```



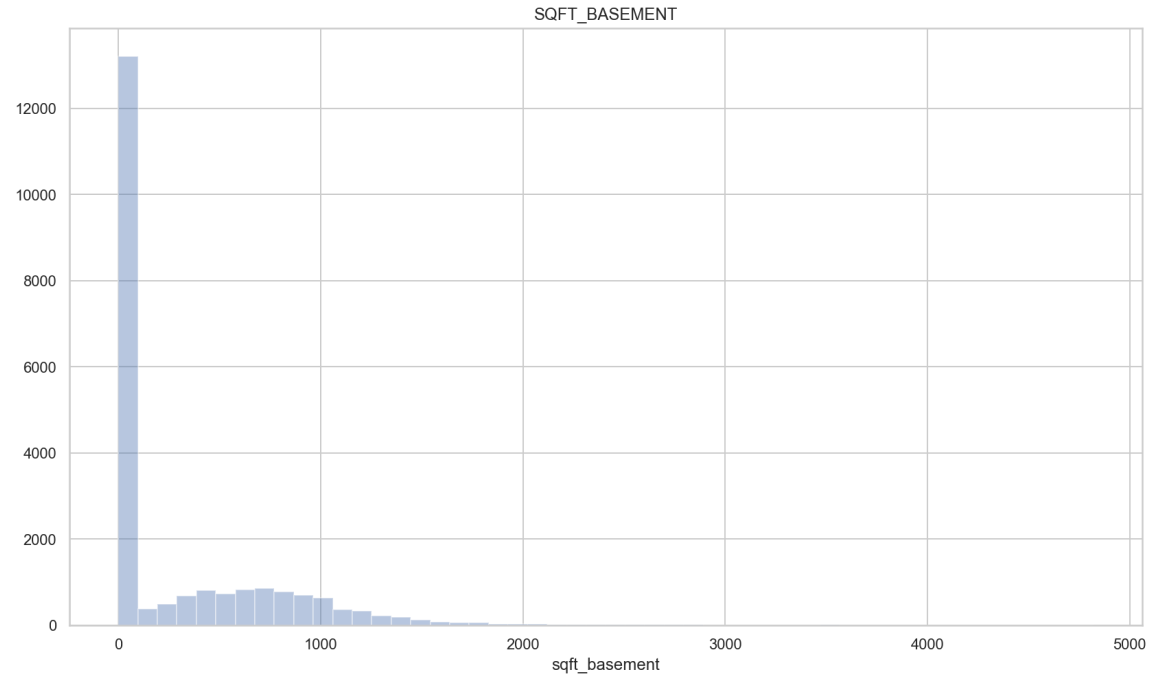

```
count    21613.000000
mean        47.560053
std          0.138564
min         47.155900
25%         47.471000
50%         47.571800
75%         47.678000
max         47.777600
Name: lat, dtype: float64
```



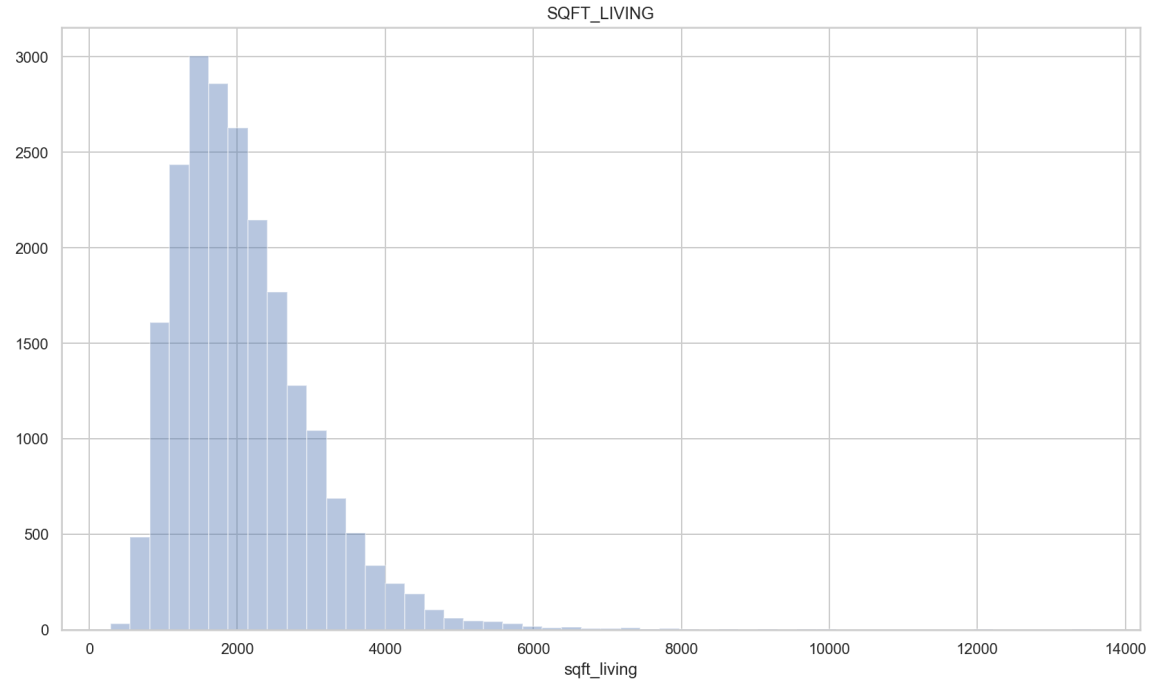
```
count    21613.000000
mean      -122.213896
std          0.140828
min       -122.519000
25%       -122.328000
50%       -122.230000
75%       -122.125000
max       -121.315000
Name: long, dtype: float64
```



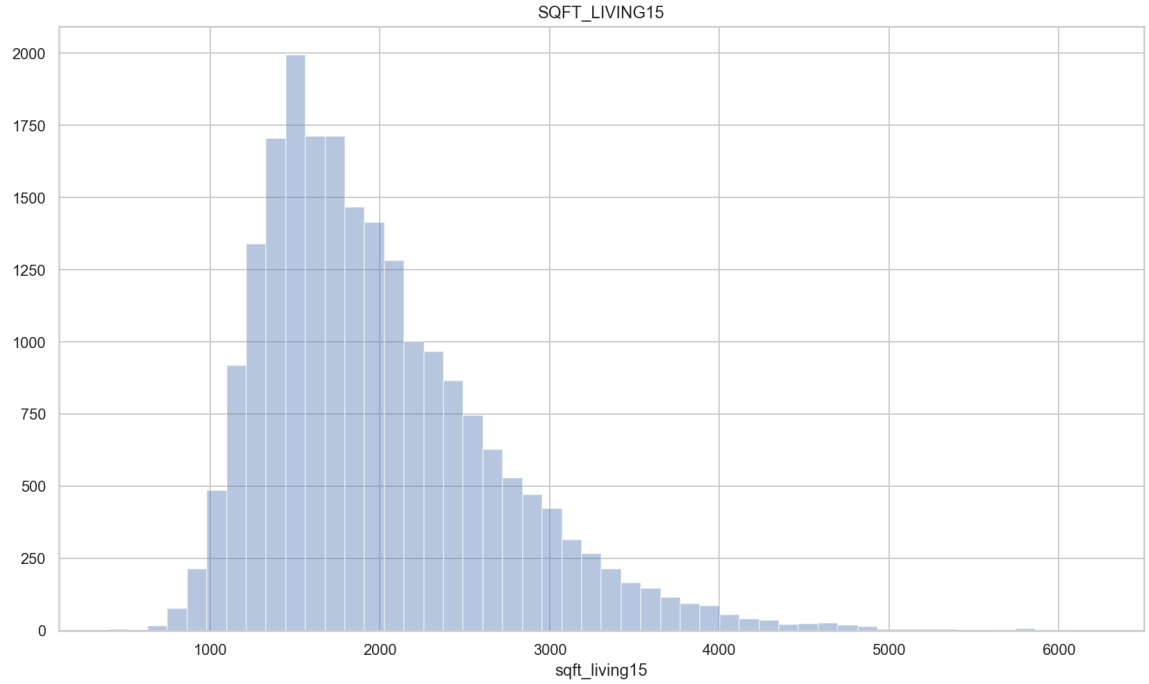

```
count    21613.000000
mean      1788.390691
std        828.090978
min        290.000000
25%       1190.000000
50%       1560.000000
75%       2210.000000
max       9410.000000
Name: sqft_above, dtype: float64
```



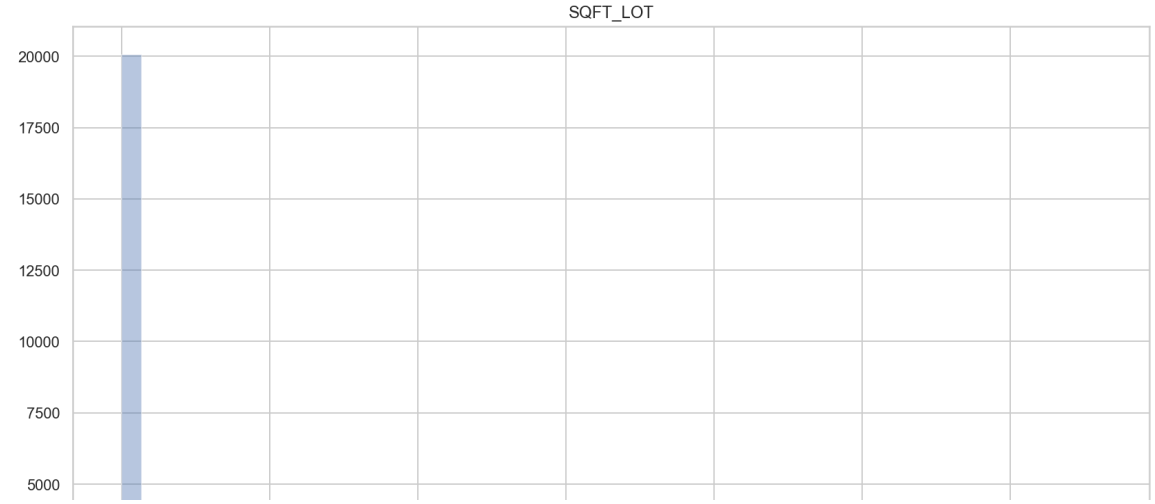

```
count    21613.000000
mean       291.509045
std        442.575043
min          0.000000
25%          0.000000
50%          0.000000
75%        560.000000
max       4820.000000
Name: sqft_basement, dtype: float64
```



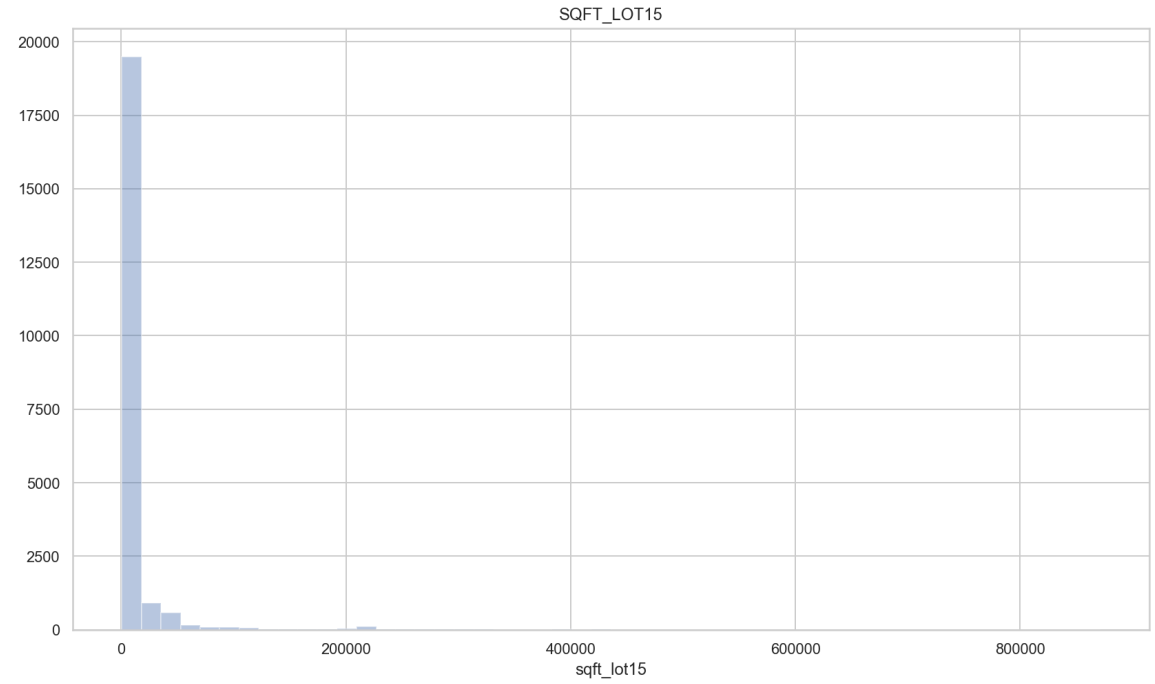
```
count    21613.000000
mean      2079.899736
std        918.440897
min        290.000000
25%       1427.000000
50%       1910.000000
75%       2550.000000
max      13540.000000
Name: sqft_living, dtype: float64
```




```
count    21613.000000
mean      1986.552492
std        685.391304
min        399.000000
25%       1490.000000
50%       1840.000000
75%       2360.000000
max       6210.000000
Name: sqft_living15, dtype: float64
```

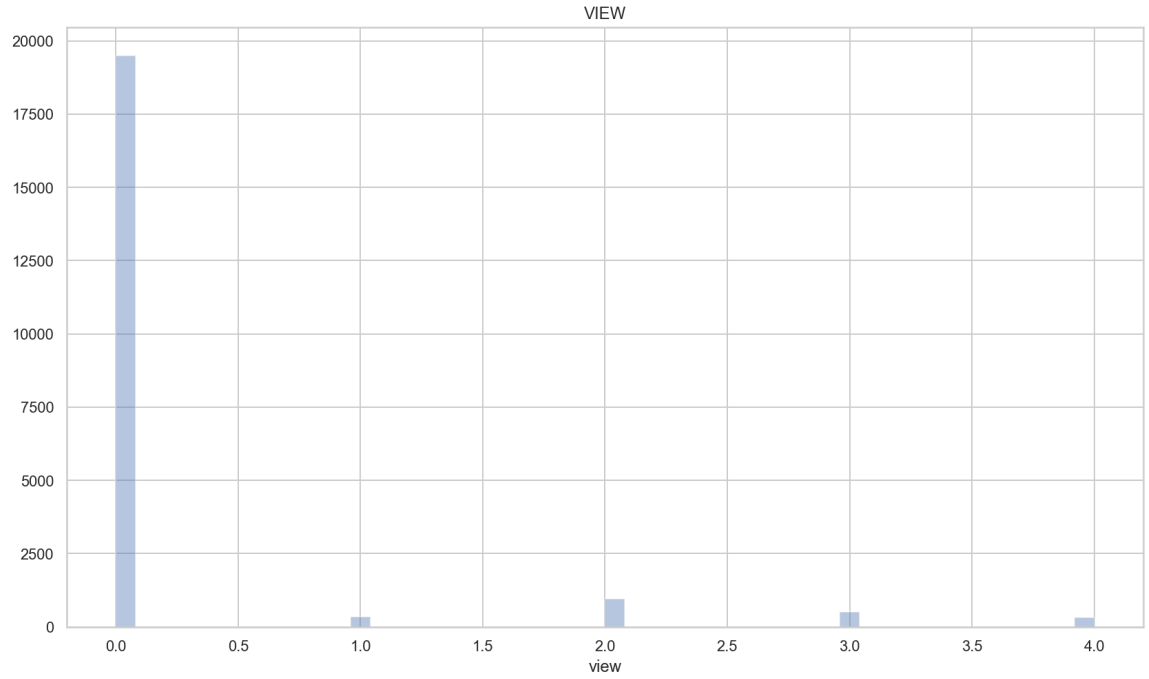



```
count    2.161300e+04
mean     1.510697e+04
std      4.142051e+04
min      5.200000e+02
25%      5.040000e+03
50%      7.618000e+03
75%      1.068800e+04
max      1.651359e+06
Name: sqft_lot, dtype: float64
```

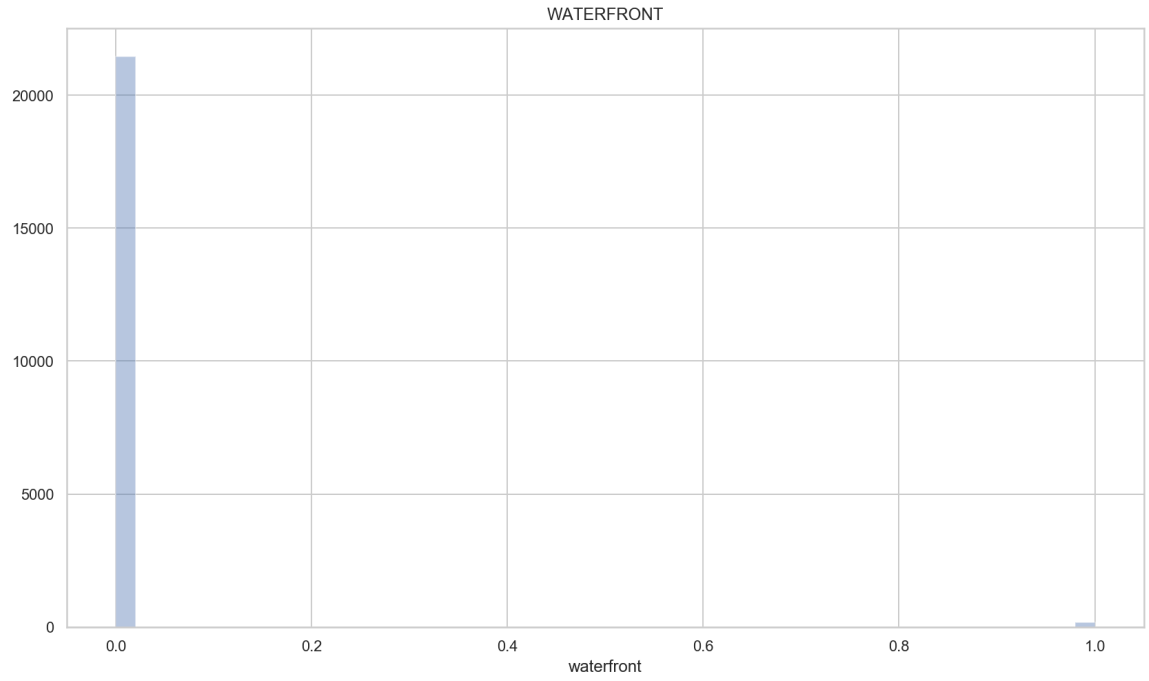



```
count     21613.000000
mean      12768.455652
std       27304.179631
min         651.000000
```

```
25%        5100.000000
50%        7620.000000
75%       10083.000000
max      871200.000000
Name: sqft_lot15, dtype: float64
```

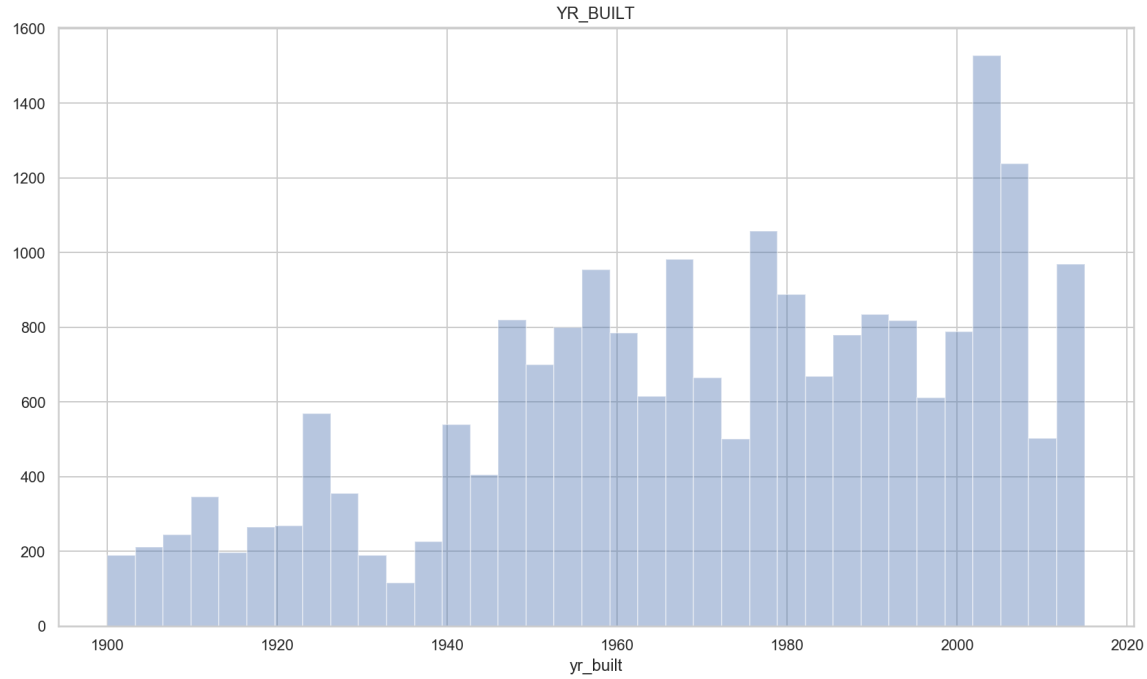



```
count    21613.000000
mean         0.234303
std          0.766318
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          4.000000
Name: view, dtype: float64
```

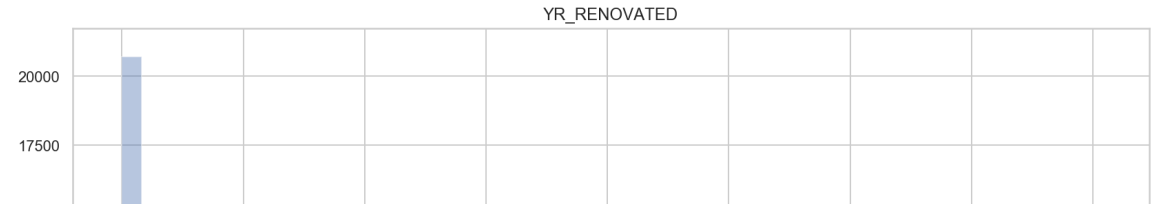



```
count    21613.000000
mean         0.007542
std          0.086517
```

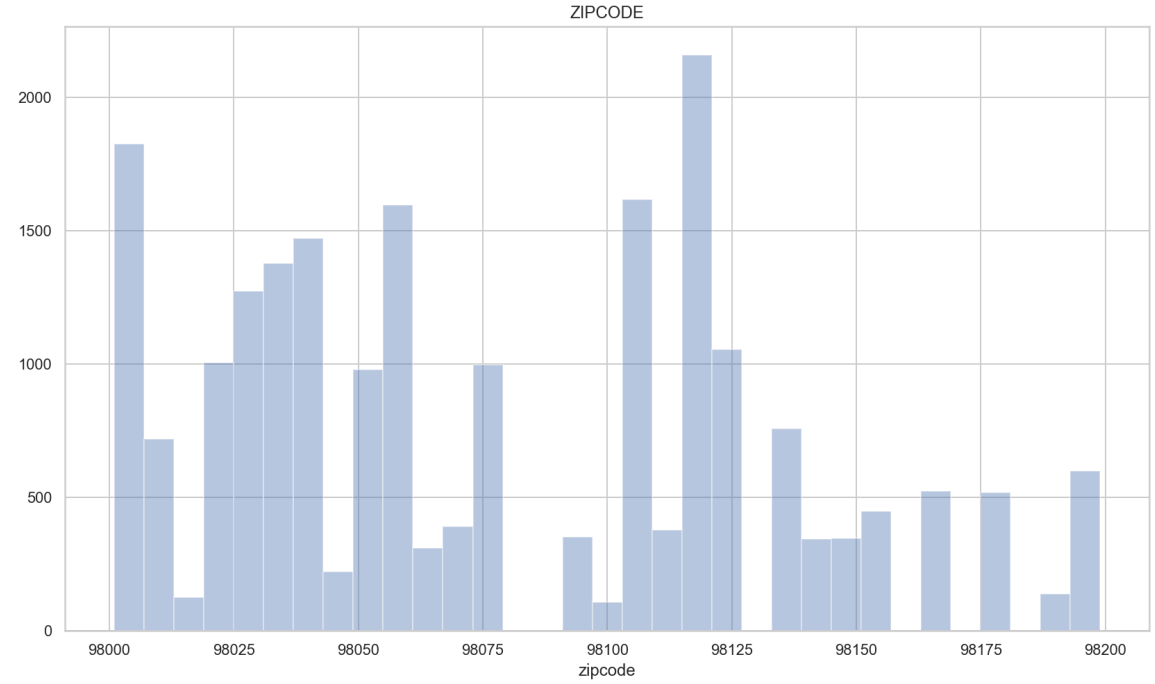
```
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max          1.000000
Name: waterfront, dtype: float64
```



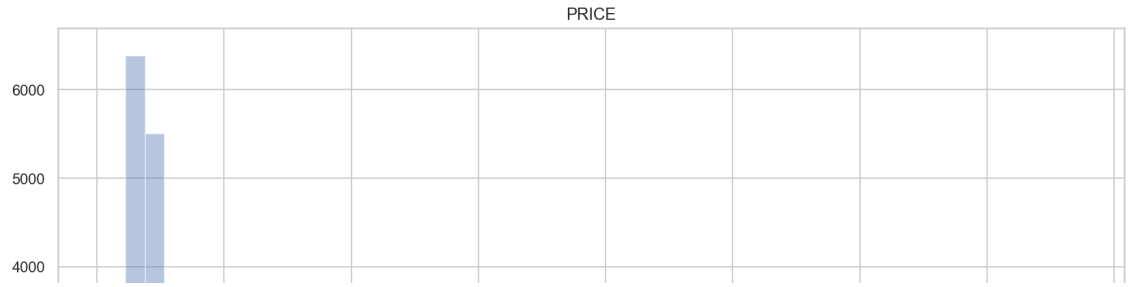

```
count    21613.000000
mean      1971.005136
std         29.373411
min       1900.000000
25%       1951.000000
50%       1975.000000
75%       1997.000000
max       2015.000000
Name: yr_built, dtype: float64
```




```
count    21613.000000
mean        84.402258
std        401.679240
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max       2015.000000
Name: yr_renovated, dtype: float64
```




```
count    21613.000000
mean     98077.939805
std         53.505026
min      98001.000000
25%      98033.000000
50%      98065.000000
75%      98118.000000
max      98199.000000
Name: zipcode, dtype: float64
```




```
count    2.161300e+04
mean     5.400881e+05
std      3.671272e+05
min      7.500000e+04
25%      3.219500e+05
50%      4.500000e+05
75%      6.450000e+05
max      7.700000e+06
Name: price, dtype: float64
```

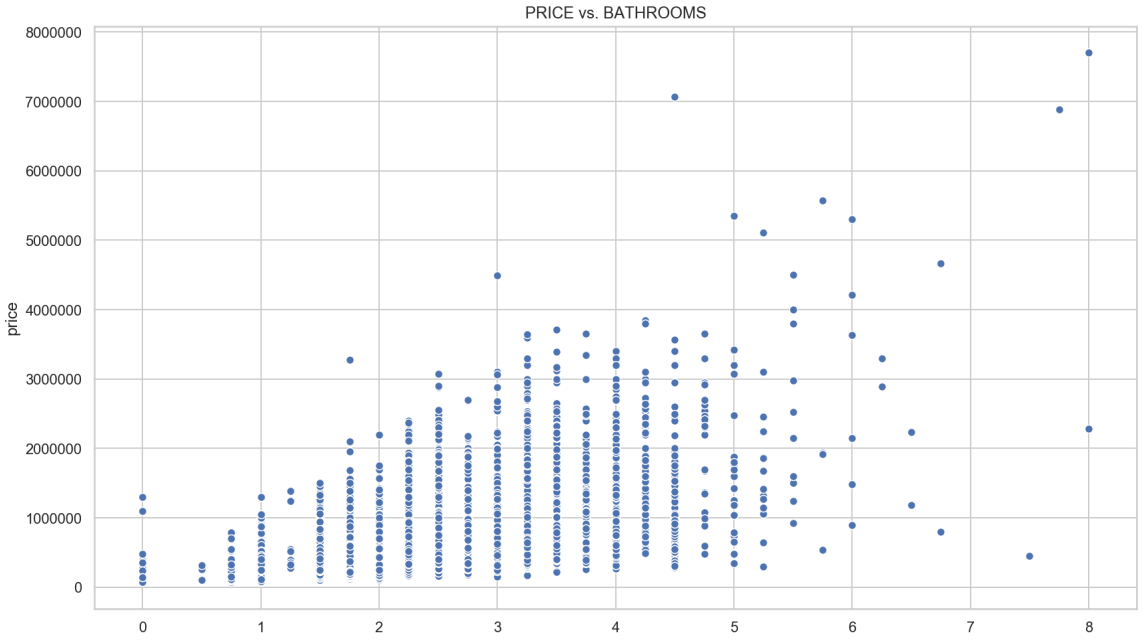
2.4 Y vs X Scatter Plots

In [16]:

```
y = "price"
x_cols = list(df.columns)
x_cols.remove(y)
x_cols.remove("date")
```

In [18]:

```
for x in x_cols:
    plt.figure(figsize=fs)
    sns.scatterplot(df[x], df[y]).set_title("{} vs. {}".format(y.upper(), x.upper())
    plt.savefig("figures/scatterplots/scatter_{}_{}.png".format(y, x))
    plt.show()
```



- Comments
  - 각 feature와 price간의 상관관계를 자세히 확인

2.5 Preprocessing

변수간 다중공선성

- 다중공선성(multicollinearity)란 독립 변수의 일부가 다른 독립 변수의 조합으로 표현될 수 있는 경우를 말하며, 독립변수들이 서로 독립이 아니라 상호상관관계가 강한 경우 발생함
- 다중 공선성이 있으면 독립변수의 공분산 행렬의 조건수(conditional number)가 증가
- 상호의존적인 독립변수를 선택하는 방법으로는 VIF(Variance Inflation Factor)를 사용하며 숫자가 높을수록 다른 변수에 의존적임을 알 수 있음

$$\mathrm{VIF}_i = \frac{\sigma^2}{(n-1)\mathrm{Var}[X_i]} \cdot \frac{1}{1-R_i^2}$$

In [17]:

```
col_cat = [
    "condition",
    "floors",
    "waterfront",
    "zipcode",
    "grade",
    "view"
]

col_num = [
    "bedrooms",
    "bathrooms",
    "sqft_above",
    "sqft_basement",
    "sqft_living",
    "sqft_living15",
    "sqft_lot",
    "sqft_lot15",
    "yr_built",
    "yr_renovated",
]
```

In [18]:

```
x0 = df.drop("price", axis=1)
y = df.price
```

In [19]:

```
from patsy import dmatrix

f_num = ["scale(" + col + ")" for col in col_num]
formula = "+".join(col_cat + f_num)

dfX = dmatrix(formula, x0, return_type="dataframe")
dfy = pd.DataFrame(y, columns=["price"])
```

In [20]:

```
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(
    dfX.values, i) for i in range(dfX.shape[1])]
vif["features"] = dfX.columns
vif = vif.sort_values("VIF Factor").reset_index(drop=True)
vif
```

Out[20]:

	VIF Factor	features
0	1.147351e+00	scale(yr_renovated)
1	1.203243e+00	waterfront
2	1.246040e+00	condition
3	1.306180e+00	zipcode
4	1.419601e+00	view
5	1.649619e+00	scale(bedrooms)
6	1.985198e+00	floors
7	2.091007e+00	scale(sqft_lot)
8	2.120834e+00	scale(sqft_lot15)
9	2.222699e+00	scale(yr_built)
10	2.899714e+00	scale(sqft_living15)
11	3.250774e+00	grade
12	3.347965e+00	scale(bathrooms)
13	4.386414e+06	Intercept
14	inf	scale(sqft_basement)
15	inf	scale(sqft_above)
16	inf	scale(sqft_living)

- commnet
  - vif값이 높은 feature들을 모델에서 제거
    - sqft_above, sqft_basement, sqft_living

이상값 제거(anomaly)

In [21]:

```
idx1 = df[df.bathrooms == 0].index
idx2 = df[df.bedrooms == 0].index
df = df.drop(idx1, errors="ignore")
df = df.drop(idx2, errors="ignore")
idx = np.where(df.bedrooms == 33)
df.bedrooms.iloc[idx] = 3
```

In [23]:

```
# idx1 = list(np.where(df.sqft_living15==1330)[0])
# idx2 = list(np.where(df.bedrooms==33)[0])
# idx = list(set(idx1) - set(idx2))
# np.mean(df.bedrooms.iloc[idx])
```

Out[23]:

```
2.8688524590163933
```

- comment
  - bathroom과 bedroom이 0인 데이터 drop
  - bedroom값이 33개인 데이터 3으로 변경
    - 면적이 같은 집들의 평균 bedroom 값으로 대체

카테고리 변수

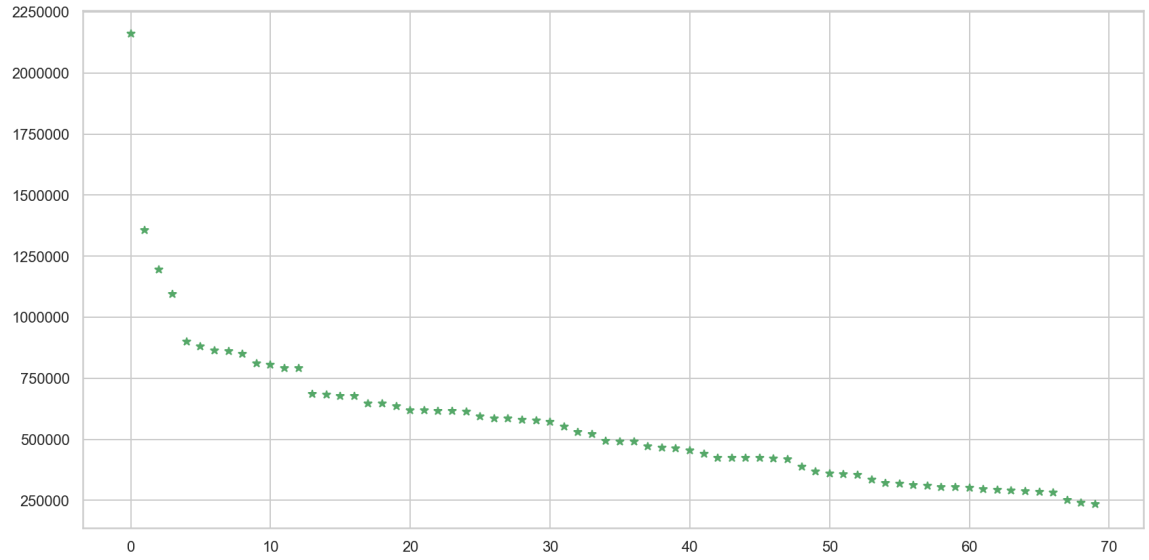
- Zipcode
  - Groupby, 70개의 type
- Latitude, Longitude 대체,

In [24]:

```
temp = df[["price", "zipcode"]]
temp = temp.groupby(by=["zipcode"], sort=False).mean()
temp["zipcode"] = temp.index
temp = temp.sort_values(by="price", ascending=False)
temp = temp.reset_index(drop=True)
```

In [25]:

```
plt.figure(figsize=(14, 7))
plt.plot(temp.index, temp.price, "g*")
plt.show()
```



- comments

- zipcode VS. 평균 집값
- 지역에 따른 집값 차이가 있음을 확인

3. Model Testing

Model_1

In [26]:

```
x0 = df.drop("price", axis=1)
y = df.price
```

In [27]:

```
col_cat = [
    "waterfront",
    "condition",
    "zipcode",
    "view",
    "grade"
]

col_num = [
    "bedrooms",
    "bathrooms",
    "sqft_living15",
    "sqft_lot",
    "sqft_lot15",
    "yr_built",
    "yr_renovated",
]
```

In [28]:

```
f_cat = ["C(" + col +")" for col in col_cat]
f_num = ["scale(" + col + ")" for col in col_num]
formula = "price ~ " + " + ".join(f_cat + f_num) + " + 0"
formula
```

Out[28]:

```
'price ~ C(waterfront) + C(condition) + C(zipcode) + C(view) + C(grad
e) + scale(bedrooms) + scale(bathrooms) + scale(sqft_living15) + scale
(sqft_lot) + scale(sqft_lot15) + scale(yr_built) + scale(yr_renovated)
+ 0'
```

In [29]:

```
x_train, x_test, y_train, y_test = train_test_split(x0, y, test_size=0.3, random_sta
```

In [30]:

```
model = sm.OLS.from_formula(formula, data=pd.concat([x_train, y_train], axis=1))
result = model.fit()
```

In [31]:

```
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                   price   R-squared:
0.807
Model:                             OLS   Adj. R-squared:
0.805
Method:                  Least Squares   F-statistic:
659.5
Date:                 Wed, 10 Jul 2019   Prob (F-statistic):
0.00
Time:                         16:02:29   Log-Likelihood:            -2.
0295e+05
No. Observations:               15117   AIC:
4.061e+05
Df Residuals:                   15021   BIC:
4.068e+05
Df Model:                          95
Covariance Type:            nonrobust
==============================================================================
==================
                          coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
------------------
C(waterfront)[0]      2.828e+05   1.69e+05      1.669      0.095   -4.
93e+04    6.15e+05
C(waterfront)[1]      9.058e+05    1.7e+05      5.313      0.000    5.
72e+05    1.24e+06
C(condition)[T.2]     8.799e+04   4.02e+04      2.190      0.029    92
47.772    1.67e+05
C(condition)[T.3]     8.343e+04   3.74e+04      2.231      0.026    1.
01e+04    1.57e+05
C(condition)[T.4]     1.136e+05   3.74e+04      3.035      0.002    4.
02e+04    1.87e+05
C(condition)[T.5]     1.531e+05   3.76e+04      4.070      0.000    7.
94e+04    2.27e+05
C(zipcode)[T.98002]   9177.2051   1.77e+04      0.518      0.604   -2.
55e+04    4.39e+04
C(zipcode)[T.98003]  -1.806e+04   1.57e+04     -1.150      0.250   -4.
89e+04    1.27e+04
C(zipcode)[T.98004]    7.49e+05   1.52e+04     49.168      0.000    7.
19e+05    7.79e+05
C(zipcode)[T.98005]   2.852e+05   1.82e+04     15.643      0.000    2.
49e+05    3.21e+05
C(zipcode)[T.98006]    2.15e+05    1.4e+04     15.357      0.000    1.
88e+05    2.42e+05
C(zipcode)[T.98007]   2.291e+05      2e+04     11.441      0.000
1.9e+05    2.68e+05
C(zipcode)[T.98008]   2.409e+05   1.59e+04     15.162      0.000
2.1e+05    2.72e+05
C(zipcode)[T.98010]   6.618e+04   2.33e+04      2.841      0.005    2.
05e+04    1.12e+05
C(zipcode)[T.98011]    1.29e+05   1.73e+04      7.478      0.000    9.
52e+04    1.63e+05
C(zipcode)[T.98014]   1.048e+05   2.03e+04      5.168      0.000
6.5e+04    1.44e+05
```

```
C(zipcode)[T.98019]   9.096e+04   1.75e+04      5.195      0.000    5.
66e+04    1.25e+05
C(zipcode)[T.98022]  -1.632e+04   1.65e+04     -0.990      0.322   -4.
86e+04     1.6e+04
C(zipcode)[T.98023]   -3.76e+04   1.34e+04     -2.796      0.005    -
6.4e+04   -1.12e+04
C(zipcode)[T.98024]   1.525e+05   2.48e+04      6.140      0.000    1.
04e+05    2.01e+05
C(zipcode)[T.98027]   1.498e+05   1.43e+04     10.458      0.000    1.
22e+05    1.78e+05
C(zipcode)[T.98028]   1.213e+05   1.55e+04      7.849      0.000
9.1e+04    1.52e+05
C(zipcode)[T.98029]   1.892e+05   1.53e+04     12.331      0.000    1.
59e+05    2.19e+05
C(zipcode)[T.98030]  -1570.6776   1.59e+04     -0.099      0.921   -3.
27e+04    2.96e+04
C(zipcode)[T.98031]  -6470.3284   1.58e+04     -0.410      0.682   -3.
74e+04    2.45e+04
C(zipcode)[T.98032]  -1495.0284   2.01e+04     -0.075      0.941   -4.
08e+04    3.78e+04
C(zipcode)[T.98033]   3.518e+05   1.43e+04     24.575      0.000    3.
24e+05     3.8e+05
C(zipcode)[T.98034]   1.814e+05   1.33e+04     13.631      0.000    1.
55e+05    2.08e+05
C(zipcode)[T.98038]   2.703e+04   1.33e+04      2.032      0.042     9
60.596    5.31e+04
C(zipcode)[T.98039]   1.224e+06   2.94e+04     41.652      0.000    1.
17e+06    1.28e+06
C(zipcode)[T.98040]   4.872e+05   1.62e+04     30.101      0.000    4.
55e+05    5.19e+05
C(zipcode)[T.98042]   1557.4350   1.36e+04      0.115      0.909   -2.
51e+04    2.82e+04
C(zipcode)[T.98045]   1.111e+05   1.71e+04      6.487      0.000    7.
76e+04    1.45e+05
C(zipcode)[T.98052]   2.238e+05   1.33e+04     16.834      0.000    1.
98e+05     2.5e+05
C(zipcode)[T.98053]     2.3e+05   1.46e+04     15.744      0.000    2.
01e+05    2.59e+05
C(zipcode)[T.98055]   3.115e+04   1.56e+04      1.994      0.046     5
23.977    6.18e+04
C(zipcode)[T.98056]   8.866e+04   1.43e+04      6.185      0.000    6.
06e+04    1.17e+05
C(zipcode)[T.98058]   1.584e+04   1.39e+04      1.137      0.256   -1.
15e+04    4.32e+04
C(zipcode)[T.98059]   7.916e+04   1.38e+04      5.748      0.000    5.
22e+04    1.06e+05
C(zipcode)[T.98065]   1.013e+05   1.53e+04      6.644      0.000    7.
14e+04    1.31e+05
C(zipcode)[T.98070]  -6.072e+04   2.25e+04     -2.704      0.007   -1.
05e+05   -1.67e+04
C(zipcode)[T.98072]   1.518e+05   1.59e+04      9.559      0.000    1.
21e+05    1.83e+05
C(zipcode)[T.98074]   1.575e+05   1.41e+04     11.140      0.000
1.3e+05    1.85e+05
C(zipcode)[T.98075]   1.586e+05   1.51e+04     10.533      0.000    1.
29e+05    1.88e+05
C(zipcode)[T.98077]   1.133e+05   1.72e+04      6.593      0.000    7.
96e+04    1.47e+05
C(zipcode)[T.98092]  -2.872e+04   1.46e+04     -1.961      0.050   -5.
74e+04     -13.065
C(zipcode)[T.98102]   4.232e+05   2.14e+04     19.791      0.000    3.
```

```
81e+05    4.65e+05
C(zipcode)[T.98103]    3.05e+05   1.33e+04     22.954      0.000    2.
79e+05    3.31e+05
C(zipcode)[T.98105]   4.297e+05   1.71e+04     25.193      0.000    3.
96e+05    4.63e+05
C(zipcode)[T.98106]    1.03e+05    1.5e+04      6.863      0.000    7.
36e+04    1.32e+05
C(zipcode)[T.98107]   2.978e+05    1.6e+04     18.604      0.000    2.
66e+05    3.29e+05
C(zipcode)[T.98108]    9.37e+04   1.76e+04      5.327      0.000    5.
92e+04    1.28e+05
C(zipcode)[T.98109]   4.423e+05   2.15e+04     20.549      0.000
4e+05    4.84e+05
C(zipcode)[T.98112]   6.189e+05   1.62e+04     38.117      0.000    5.
87e+05    6.51e+05
C(zipcode)[T.98115]   3.166e+05   1.33e+04     23.802      0.000    2.
91e+05    3.43e+05
C(zipcode)[T.98116]   2.682e+05   1.53e+04     17.562      0.000    2.
38e+05    2.98e+05
C(zipcode)[T.98117]   2.903e+05   1.36e+04     21.369      0.000    2.
64e+05    3.17e+05
C(zipcode)[T.98118]   1.339e+05   1.38e+04      9.674      0.000    1.
07e+05    1.61e+05
C(zipcode)[T.98119]   4.423e+05   1.81e+04     24.500      0.000    4.
07e+05    4.78e+05
C(zipcode)[T.98122]   2.972e+05   1.57e+04     18.921      0.000    2.
66e+05    3.28e+05
C(zipcode)[T.98125]   1.888e+05   1.43e+04     13.198      0.000    1.
61e+05    2.17e+05
C(zipcode)[T.98126]   1.757e+05    1.5e+04     11.731      0.000    1.
46e+05    2.05e+05
C(zipcode)[T.98133]   1.507e+05   1.37e+04     11.007      0.000    1.
24e+05    1.78e+05
C(zipcode)[T.98136]   2.193e+05    1.6e+04     13.677      0.000    1.
88e+05    2.51e+05
C(zipcode)[T.98144]   2.502e+05    1.5e+04     16.725      0.000    2.
21e+05     2.8e+05
C(zipcode)[T.98146]   9.211e+04   1.57e+04      5.870      0.000    6.
14e+04    1.23e+05
C(zipcode)[T.98148]    6.75e+04   2.84e+04      2.379      0.017    1.
19e+04    1.23e+05
C(zipcode)[T.98155]   1.434e+05   1.39e+04     10.308      0.000    1.
16e+05    1.71e+05
C(zipcode)[T.98166]   5.139e+04   1.58e+04      3.255      0.001    2.
04e+04    8.23e+04
C(zipcode)[T.98168]   5.126e+04    1.6e+04      3.207      0.001    1.
99e+04    8.26e+04
C(zipcode)[T.98177]   2.054e+05   1.63e+04     12.615      0.000    1.
74e+05    2.37e+05
C(zipcode)[T.98178]   2.287e+04    1.6e+04      1.427      0.154   -85
48.890    5.43e+04
C(zipcode)[T.98188]   2.637e+04   1.98e+04      1.331      0.183   -1.
25e+04    6.52e+04
C(zipcode)[T.98198]  -7629.0346   1.55e+04     -0.491      0.624   -3.
81e+04    2.28e+04
C(zipcode)[T.98199]   3.625e+05   1.56e+04     23.223      0.000    3.
32e+05    3.93e+05
C(view)[T.1]          9.121e+04   1.07e+04      8.489      0.000    7.
02e+04    1.12e+05
C(view)[T.2]          6.808e+04   6740.810     10.100      0.000    5.
49e+04    8.13e+04
```

```
C(view)[T.3]          1.401e+05   9187.520     15.253      0.000    1.
22e+05    1.58e+05
C(view)[T.4]          2.866e+05   1.39e+04     20.567      0.000    2.
59e+05    3.14e+05
C(grade)[T.4]        -1.209e+05   1.69e+05     -0.715      0.474   -4.
52e+05     2.1e+05
C(grade)[T.5]        -1.406e+05   1.65e+05     -0.850      0.395   -4.
65e+05    1.84e+05
C(grade)[T.6]        -1.135e+05   1.65e+05     -0.688      0.491   -4.
37e+05     2.1e+05
C(grade)[T.7]        -9.484e+04   1.65e+05     -0.575      0.565   -4.
18e+05    2.29e+05
C(grade)[T.8]        -4.647e+04   1.65e+05     -0.282      0.778    -
3.7e+05    2.77e+05
C(grade)[T.9]         7.987e+04   1.65e+05      0.484      0.629   -2.
44e+05    4.04e+05
C(grade)[T.10]        2.437e+05   1.65e+05      1.475      0.140   -8.
02e+04    5.68e+05
C(grade)[T.11]        5.151e+05   1.66e+05      3.112      0.002    1.
91e+05    8.39e+05
C(grade)[T.12]        1.074e+06   1.67e+05      6.440      0.000    7.
47e+05     1.4e+06
C(grade)[T.13]        2.436e+06   1.72e+05     14.142      0.000
2.1e+06    2.77e+06
scale(bedrooms)       1.704e+04   1690.779     10.078      0.000    1.
37e+04    2.04e+04
scale(bathrooms)      6.133e+04   2227.898     27.527      0.000
5.7e+04    6.57e+04
scale(sqft_living15)  4.906e+04   2298.945     21.342      0.000    4.
46e+04    5.36e+04
scale(sqft_lot)       1.534e+04   1885.183      8.137      0.000    1.
16e+04     1.9e+04
scale(sqft_lot15)      798.1381   1952.172      0.409      0.683   -30
28.356    4624.632
scale(yr_built)      -2.647e+04   2251.603    -11.755      0.000   -3.
09e+04   -2.21e+04
scale(yr_renovated)   1.136e+04   1459.504      7.786      0.000    85
02.657    1.42e+04
==============================================================================
========
Omnibus:                    13955.213   Durbin-Watson:
2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          302
9559.348
Skew:                           3.825   Prob(JB):
0.00
Kurtosis:                      71.929   Cond. No.
672.
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

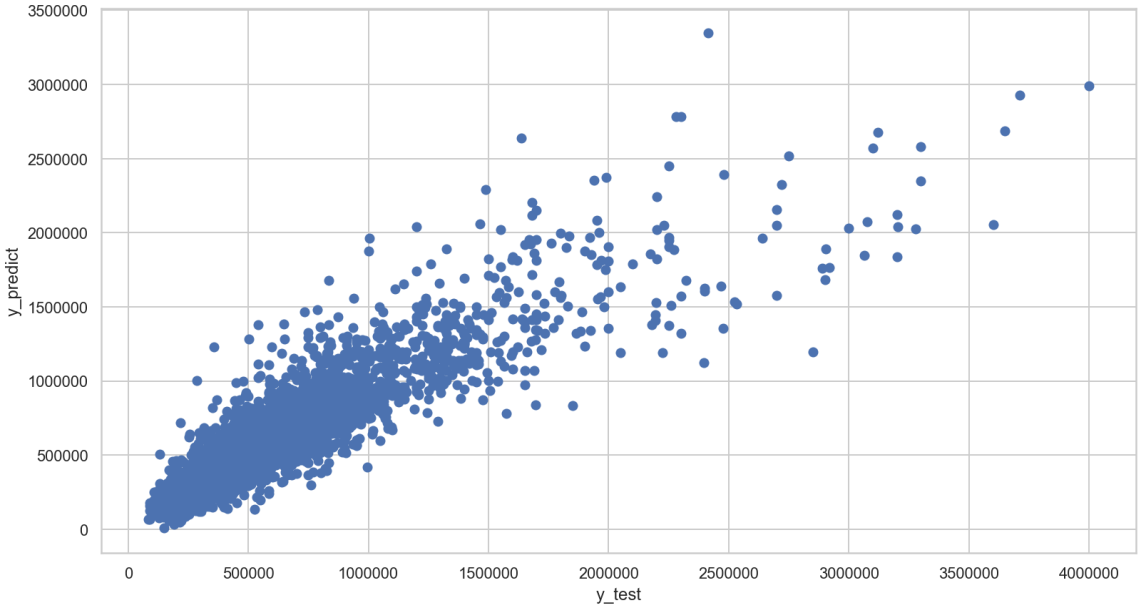
- comments
  - Numerical변수의 p-value값이 0.05보다 높은 변수 제거

In [32]:

```
y_predict = result.predict(x_test)
```

In [33]:

```
plt.figure(figsize=fs)
plt.xlabel("y_test")
plt.ylabel("y_predict")
plt.scatter(y_test, y_predict)
plt.show()
```



In [34]:

```
r2_score(y_test, y_predict)
```

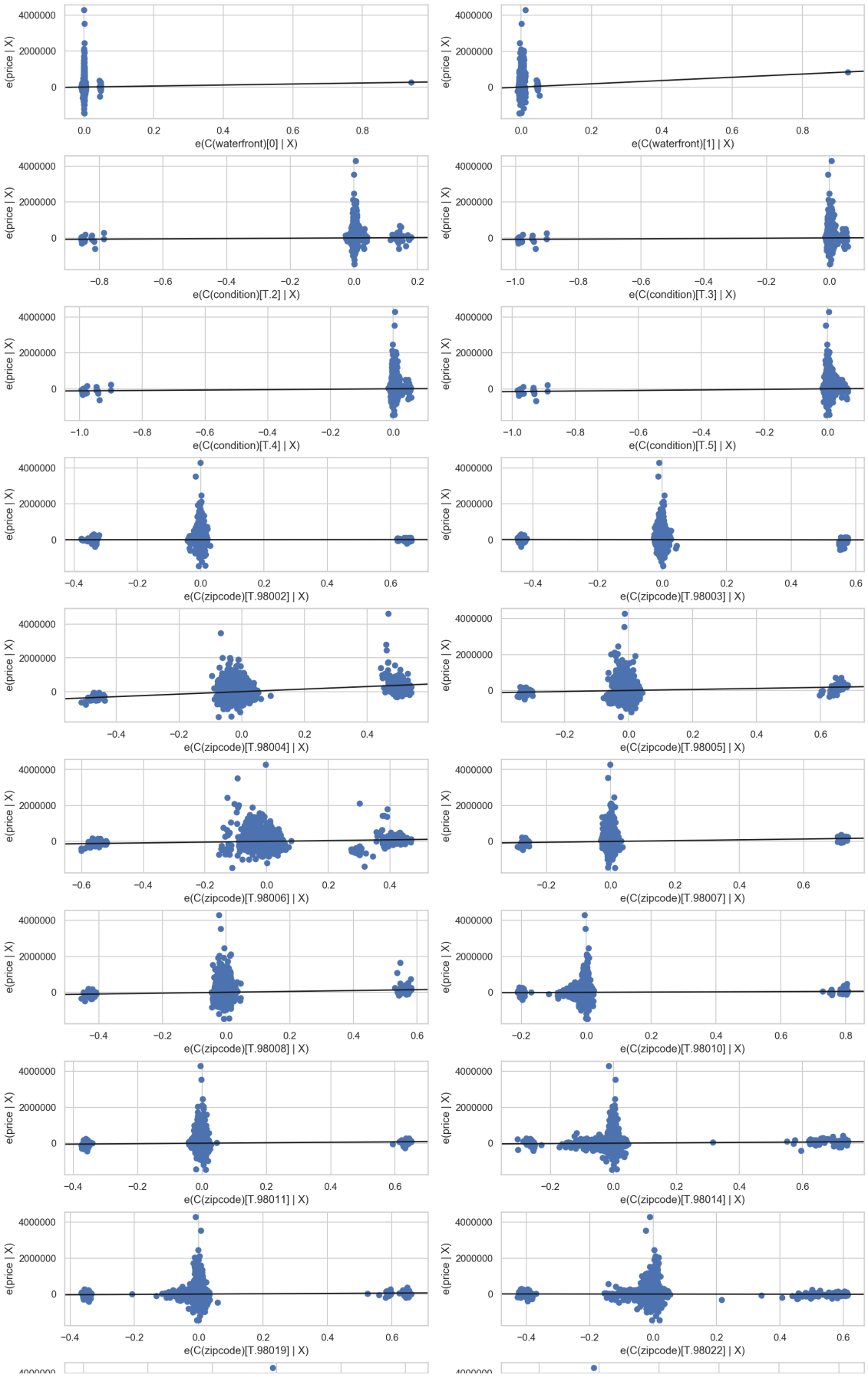
Out[34]:

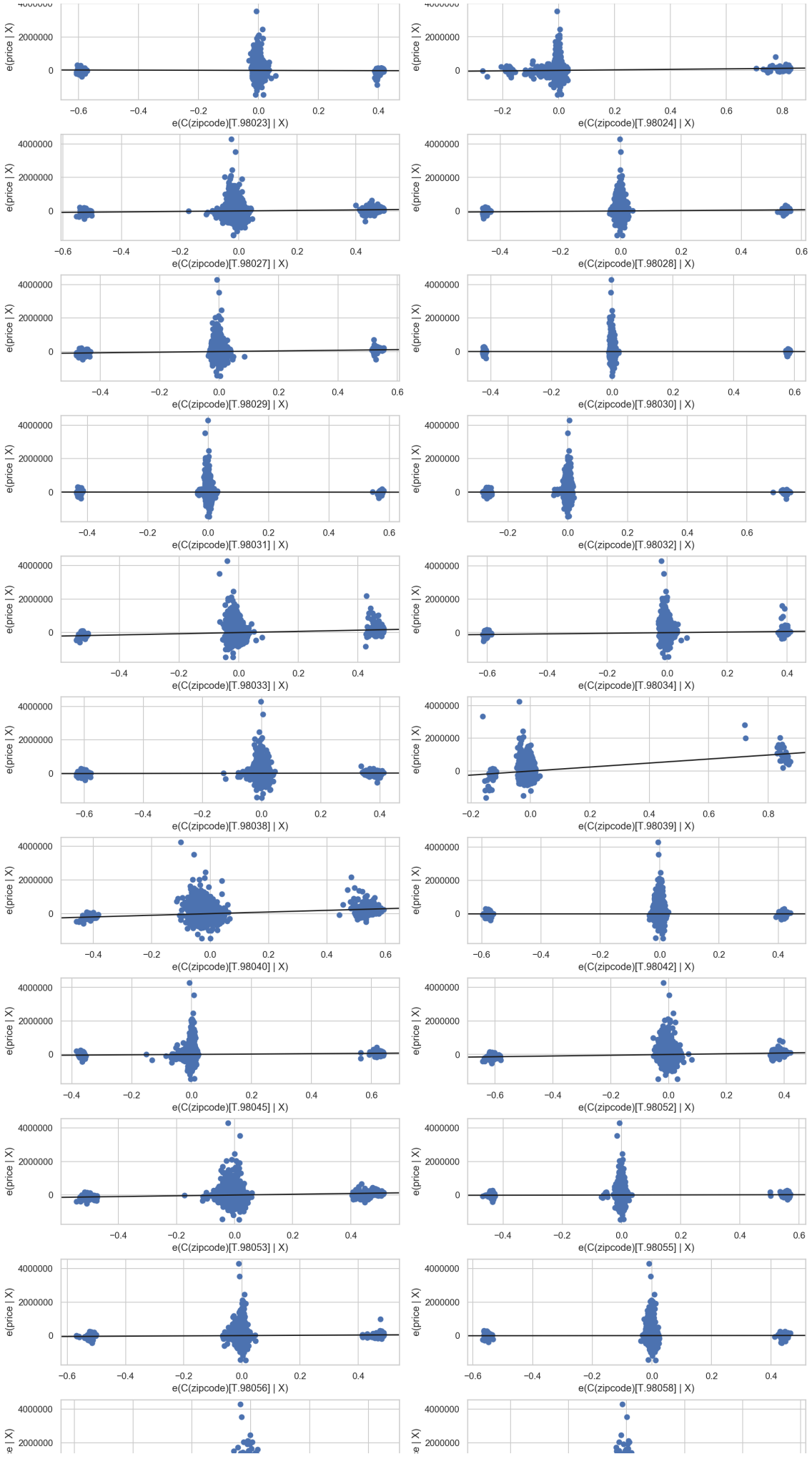
```
0.8161989418392456
```

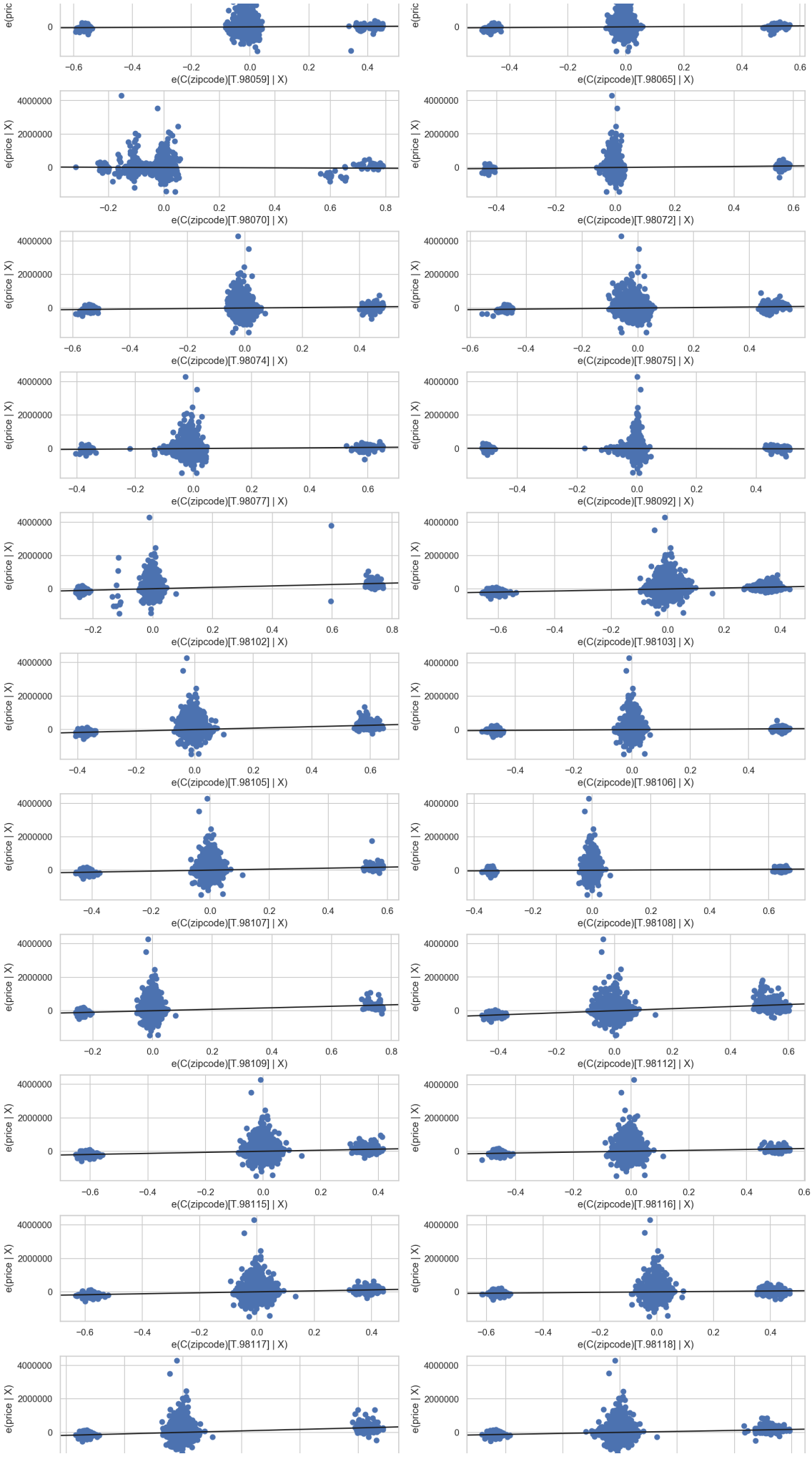
- comments
  - test결과 확인

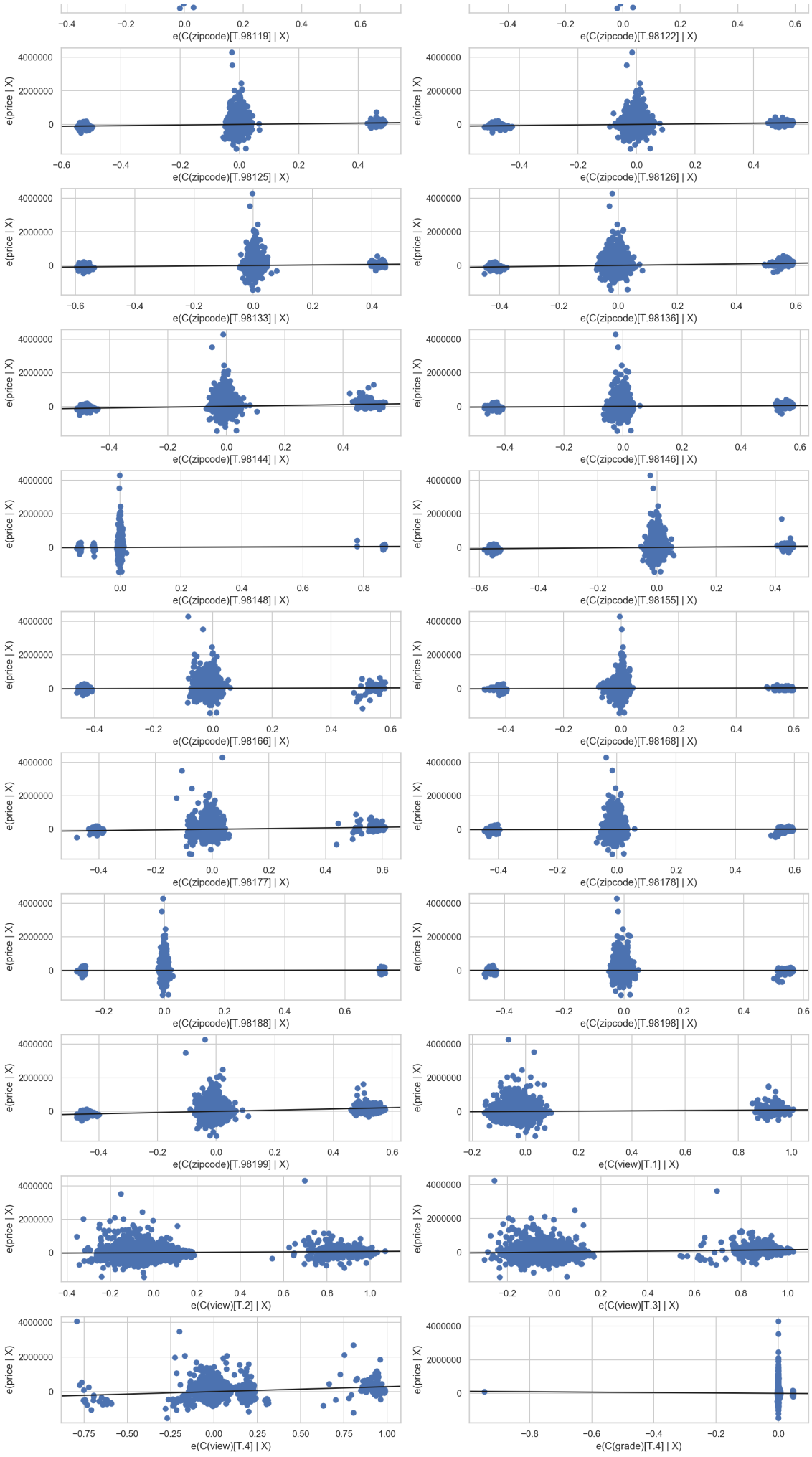
In [35]:

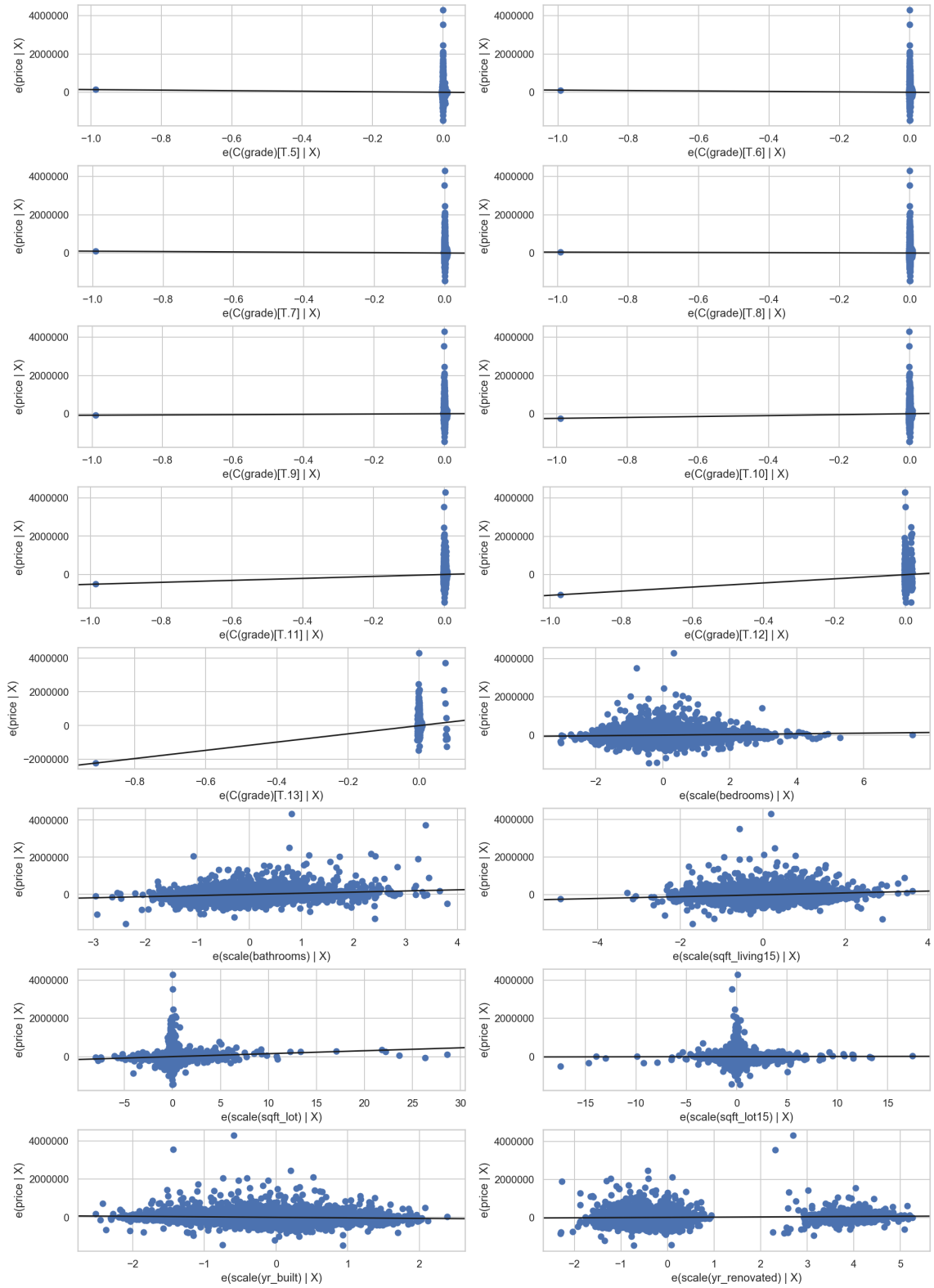
```
fig = plt.figure(figsize=(14, 120))
sm.graphics.plot_partregress_grid(result, fig=fig)
fig.suptitle("")
plt.show()
```




e(price | X)
e(C(zipcode)[T.98023] | X)
e(C(zipcode)[T.98024] | X)
e(C(zipcode)[T.98027] | X)
e(C(zipcode)[T.98028] | X)
e(C(zipcode)[T.98029] | X)
e(C(zipcode)[T.98030] | X)
e(C(zipcode)[T.98031] | X)
e(C(zipcode)[T.98032] | X)
e(C(zipcode)[T.98033] | X)
e(C(zipcode)[T.98034] | X)
e(C(zipcode)[T.98038] | X)
e(C(zipcode)[T.98039] | X)
e(C(zipcode)[T.98040] | X)
e(C(zipcode)[T.98042] | X)
e(C(zipcode)[T.98045] | X)
e(C(zipcode)[T.98052] | X)
e(C(zipcode)[T.98053] | X)
e(C(zipcode)[T.98055] | X)
e(C(zipcode)[T.98056] | X)
e(C(zipcode)[T.98058] | X)


e(price | X)
e(C(zipcode)[T.98059] | X)
e(C(zipcode)[T.98065] | X)
e(C(zipcode)[T.98070] | X)
e(C(zipcode)[T.98072] | X)
e(C(zipcode)[T.98074] | X)
e(C(zipcode)[T.98075] | X)
e(C(zipcode)[T.98077] | X)
e(C(zipcode)[T.98092] | X)
e(C(zipcode)[T.98102] | X)
e(C(zipcode)[T.98103] | X)
e(C(zipcode)[T.98105] | X)
e(C(zipcode)[T.98106] | X)
e(C(zipcode)[T.98107] | X)
e(C(zipcode)[T.98108] | X)
e(C(zipcode)[T.98109] | X)
e(C(zipcode)[T.98112] | X)
e(C(zipcode)[T.98115] | X)
e(C(zipcode)[T.98116] | X)
e(C(zipcode)[T.98117] | X)
e(C(zipcode)[T.98118] | X)


e(C(zipcode)[T.98119] | X)
e(C(zipcode)[T.98122] | X)
e(C(zipcode)[T.98125] | X)
e(C(zipcode)[T.98126] | X)
e(C(zipcode)[T.98133] | X)
e(C(zipcode)[T.98136] | X)
e(C(zipcode)[T.98144] | X)
e(C(zipcode)[T.98146] | X)
e(C(zipcode)[T.98148] | X)
e(C(zipcode)[T.98155] | X)
e(C(zipcode)[T.98166] | X)
e(C(zipcode)[T.98168] | X)
e(C(zipcode)[T.98177] | X)
e(C(zipcode)[T.98178] | X)
e(C(zipcode)[T.98188] | X)
e(C(zipcode)[T.98198] | X)
e(C(zipcode)[T.98199] | X)
e(C(view)[T.1] | X)
e(C(view)[T.2] | X)
e(C(view)[T.3] | X)
e(C(view)[T.4] | X)
e(C(grade)[T.4] | X)
e(price | X)



- comments
  - 부분회귀결과 상관관계가 없는 변수 제거
    - sqft_lot15, yr_built, yr_renovated

Model_2

In [36]:

```
formula = "np.log(price) ~ scale(np.log(bathrooms)) + scale(np.log(bedrooms)) + \
scale(np.log(sqft_living15)) + scale(np.log(sqft_lot)) + C(condition) + \
C(grade) + C(zipcode) + C(view) + 0"
```

In [37]:

```
x0 = df.drop("price", axis=1)
y = df.price
```

In [38]:

```
x_train, x_test, y_train, y_test = train_test_split(x0, y, test_size=0.3, random_sta
```

In [39]:

```
model2 = sm.OLS.from_formula(formula, data=pd.concat([x_train, y_train], axis=1))
result2 = model2.fit()
```

In [40]:

```
print(result2.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:          np.log(price)   R-squared:
0.859
Model:                            OLS   Adj. R-squared:
0.858
Method:                 Least Squares   F-statistic:
1006.
Date:                Wed, 10 Jul 2019   Prob (F-statistic):
0.00
Time:                        16:06:12   Log-Likelihood:
3056.4
No. Observations:               15117   AIC:
-5929.
Df Residuals:                   15025   BIC:
-5227.
Df Model:                          91
Covariance Type:            nonrobust
==============================================================================
==========================
                                 coef    std err          t      P>|
t|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
C(condition)[1]                12.4491      0.205     60.874      0.0
00      12.048      12.850
C(condition)[2]                12.5686      0.200     62.735      0.0
00      12.176      12.961
C(condition)[3]                12.6881      0.199     63.603      0.0
00      12.297      13.079
C(condition)[4]                12.7316      0.200     63.815      0.0
00      12.341      13.123
C(condition)[5]                12.8012      0.199     64.199      0.0
00      12.410      13.192
C(grade)[T.4]                  -0.5349      0.204     -2.623      0.0
09      -0.935      -0.135
C(grade)[T.5]                  -0.4758      0.200     -2.384      0.0
17      -0.867      -0.085
C(grade)[T.6]                  -0.3619      0.199     -1.818      0.0
69      -0.752       0.028
C(grade)[T.7]                  -0.2610      0.199     -1.311      0.1
90      -0.651       0.129
C(grade)[T.8]                  -0.1371      0.199     -0.688      0.4
91      -0.527       0.253
C(grade)[T.9]                   0.0362      0.199      0.182      0.8
56      -0.354       0.427
C(grade)[T.10]                  0.1648      0.199      0.827      0.4
08      -0.226       0.556
C(grade)[T.11]                  0.3347      0.200      1.676      0.0
94      -0.057       0.726
C(grade)[T.12]                  0.5103      0.201      2.537      0.0
11       0.116       0.905
C(grade)[T.13]                  0.8420      0.208      4.054      0.0
00       0.435       1.249
C(zipcode)[T.98002]             0.0256      0.021      1.195      0.2
32      -0.016       0.068
```

```
C(zipcode)[T.98003]              0.0201      0.019      1.059      0.2
89      -0.017       0.057
C(zipcode)[T.98004]              1.1060      0.018     60.322      0.0
00       1.070       1.142
C(zipcode)[T.98005]              0.7189      0.022     32.752      0.0
00       0.676       0.762
C(zipcode)[T.98006]              0.6096      0.017     36.127      0.0
00       0.576       0.643
C(zipcode)[T.98007]              0.6461      0.024     26.750      0.0
00       0.599       0.693
C(zipcode)[T.98008]              0.6484      0.019     33.854      0.0
00       0.611       0.686
C(zipcode)[T.98010]              0.2312      0.028      8.246      0.0
00       0.176       0.286
C(zipcode)[T.98011]              0.4458      0.021     21.415      0.0
00       0.405       0.487
C(zipcode)[T.98014]              0.2871      0.024     11.834      0.0
00       0.240       0.335
C(zipcode)[T.98019]              0.3177      0.021     15.087      0.0
00       0.276       0.359
C(zipcode)[T.98022]              0.0392      0.020      1.985      0.0
47       0.000       0.078
C(zipcode)[T.98023]             -0.0341      0.016     -2.104      0.0
35      -0.066      -0.002
C(zipcode)[T.98024]              0.4022      0.030     13.496      0.0
00       0.344       0.461
C(zipcode)[T.98027]              0.5038      0.017     29.206      0.0
00       0.470       0.538
C(zipcode)[T.98028]              0.4134      0.019     22.174      0.0
00       0.377       0.450
C(zipcode)[T.98029]              0.5956      0.019     32.060      0.0
00       0.559       0.632
C(zipcode)[T.98030]              0.0498      0.019      2.597      0.0
09       0.012       0.087
C(zipcode)[T.98031]              0.0589      0.019      3.090      0.0
02       0.022       0.096
C(zipcode)[T.98032]             -0.0093      0.024     -0.385      0.7
01      -0.057       0.038
C(zipcode)[T.98033]              0.7932      0.017     45.909      0.0
00       0.759       0.827
C(zipcode)[T.98034]              0.5343      0.016     33.273      0.0
00       0.503       0.566
C(zipcode)[T.98038]              0.1636      0.016     10.216      0.0
00       0.132       0.195
C(zipcode)[T.98039]              1.2907      0.035     36.507      0.0
00       1.221       1.360
C(zipcode)[T.98040]              0.8647      0.019     44.434      0.0
00       0.827       0.903
C(zipcode)[T.98042]              0.0599      0.016      3.661      0.0
00       0.028       0.092
C(zipcode)[T.98045]              0.3152      0.021     15.230      0.0
00       0.275       0.356
C(zipcode)[T.98052]              0.6302      0.016     39.270      0.0
00       0.599       0.662
C(zipcode)[T.98053]              0.5985      0.018     34.031      0.0
00       0.564       0.633
C(zipcode)[T.98055]              0.1635      0.019      8.677      0.0
00       0.127       0.200
C(zipcode)[T.98056]              0.3358      0.017     19.392      0.0
00       0.302       0.370
C(zipcode)[T.98058]              0.1515      0.017      9.011      0.0
```

```
00        0.119        0.184
C(zipcode)[T.98059]              0.3517      0.017     21.171       0.0
00        0.319        0.384
C(zipcode)[T.98065]              0.4396      0.018     23.894       0.0
00        0.404        0.476
C(zipcode)[T.98070]              0.3499      0.027     13.189       0.0
00        0.298        0.402
C(zipcode)[T.98072]              0.4517      0.019     23.558       0.0
00        0.414        0.489
C(zipcode)[T.98074]              0.5364      0.017     31.451       0.0
00        0.503        0.570
C(zipcode)[T.98075]              0.5602      0.018     30.881       0.0
00        0.525        0.596
C(zipcode)[T.98077]              0.3949      0.021     19.033       0.0
00        0.354        0.436
C(zipcode)[T.98092]              0.0106      0.018      0.600       0.5
49       -0.024        0.045
C(zipcode)[T.98102]              0.9770      0.026     37.886       0.0
00        0.926        1.028
C(zipcode)[T.98103]              0.8845      0.016     55.144       0.0
00        0.853        0.916
C(zipcode)[T.98105]              0.9900      0.020     48.661       0.0
00        0.950        1.030
C(zipcode)[T.98106]              0.3877      0.018     21.325       0.0
00        0.352        0.423
C(zipcode)[T.98107]              0.8920      0.019     46.067       0.0
00        0.854        0.930
C(zipcode)[T.98108]              0.4021      0.021     18.983       0.0
00        0.361        0.444
C(zipcode)[T.98109]              1.0560      0.026     40.936       0.0
00        1.005        1.107
C(zipcode)[T.98112]              1.1290      0.019     58.232       0.0
00        1.091        1.167
C(zipcode)[T.98115]              0.8636      0.016     54.228       0.0
00        0.832        0.895
C(zipcode)[T.98116]              0.8110      0.018     44.143       0.0
00        0.775        0.847
C(zipcode)[T.98117]              0.8647      0.016     53.072       0.0
00        0.833        0.897
C(zipcode)[T.98118]              0.5052      0.017     30.390       0.0
00        0.473        0.538
C(zipcode)[T.98119]              1.0282      0.022     47.473       0.0
00        0.986        1.071
C(zipcode)[T.98122]              0.8617      0.019     45.623       0.0
00        0.825        0.899
C(zipcode)[T.98125]              0.6035      0.017     35.073       0.0
00        0.570        0.637
C(zipcode)[T.98126]              0.6082      0.018     33.682       0.0
00        0.573        0.644
C(zipcode)[T.98133]              0.4946      0.017     29.930       0.0
00        0.462        0.527
C(zipcode)[T.98136]              0.7175      0.019     37.179       0.0
00        0.680        0.755
C(zipcode)[T.98144]              0.7371      0.018     40.707       0.0
00        0.702        0.773
C(zipcode)[T.98146]              0.3140      0.019     16.619       0.0
00        0.277        0.351
C(zipcode)[T.98148]              0.1961      0.034      5.734       0.0
00        0.129        0.263
C(zipcode)[T.98155]              0.4539      0.017     27.114       0.0
00        0.421        0.487
```

```
C(zipcode)[T.98166]              0.3397      0.019     17.904      0.0
00       0.302       0.377
C(zipcode)[T.98168]              0.0847      0.019      4.404      0.0
00       0.047       0.122
C(zipcode)[T.98177]              0.5909      0.020     30.221      0.0
00       0.553       0.629
C(zipcode)[T.98178]              0.1738      0.019      9.006      0.0
00       0.136       0.212
C(zipcode)[T.98188]              0.1132      0.024      4.738      0.0
00       0.066       0.160
C(zipcode)[T.98198]              0.0835      0.019      4.455      0.0
00       0.047       0.120
C(zipcode)[T.98199]              0.8921      0.019     47.495      0.0
00       0.855       0.929
C(view)[T.1]                     0.1090      0.013      8.408      0.0
00       0.084       0.134
C(view)[T.2]                     0.1112      0.008     13.715      0.0
00       0.095       0.127
C(view)[T.3]                     0.2005      0.011     18.247      0.0
00       0.179       0.222
C(view)[T.4]                     0.4739      0.014     33.816      0.0
00       0.446       0.501
scale(np.log(bathrooms))         0.0737      0.002     29.537      0.0
00       0.069       0.079
scale(np.log(bedrooms))          0.0315      0.002     15.487      0.0
00       0.028       0.036
scale(np.log(sqft_living15))     0.0892      0.003     32.858      0.0
00       0.084       0.095
scale(np.log(sqft_lot))          0.0770      0.002     34.721      0.0
00       0.073       0.081
==============================================================================
========
Omnibus:                      839.394   Durbin-Watson:
1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
3310.172
Skew:                          -0.107   Prob(JB):
0.00
Kurtosis:                       5.282   Cond. No.
713.
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

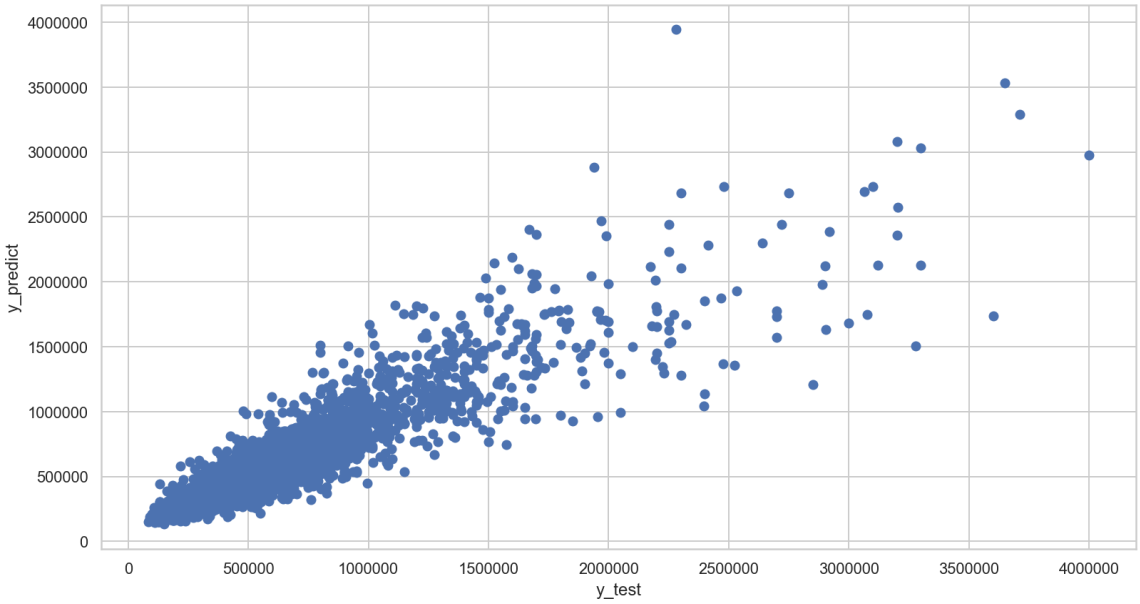
- comments
  - 부분회귀결과를 반영후 성능향상
  - numerical 변수는 p-value가 0이므로 모두 유의미한 변수임

In [41]:

```
y_predict2 = result2.predict(x_test).apply(np.exp)
```

In [42]:

```
plt.figure(figsize=fs)
plt.xlabel("y_test")
plt.ylabel("y_predict")
plt.scatter(y_test, y_predict2)
plt.show()
```



In [43]:

```
r2_score(y_test, y_predict2)
```

Out[43]:

```
0.8379459892098651
```

- comments
  - test결과 확인

In [ ]:

```

```

In [ ]:

```

```