Simple Model 1

In [514]:

```
# %load_ext autoreload
%autoreload 2
%reload_ext autoreload
import glob
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import folium
import geopandas
import geopy
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
import statsmodels.api as sm
import statsmodels.stats.api as sms
from statsmodels.formula.api import ols
from statsmodels.stats.diagnostic import linear_rainbow, het_breuschpagan
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.metrics import mean_squared_error, make_scorer
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter
import folium.plugins as plugins
import math
from math import sin, cos, sqrt, atan2, radians
from haversine import haversine
from itertools import combinations
import json
%matplotlib inline
pd.set_option("display.max_columns", 200)
pd.set_option("display.max_rows", 200)
import sys
import os
path_to_src = os.path.join('..\..', 'src')
sys.path.insert(0, path_to_src)
from useful_functions import *
```

In [515]:

```
df = pd.read_csv('..\..\data\processed\cleaned_data.csv')
len(df)
```

Out[515]:

```
15851
```

In [516]:

```
# create a simple linear model with just one independent variable
# As single feature need to reshape X_train into column vector
df1 = df.copy()
X_train = np.array(df1['SqFtTotLiving']).reshape(-1,1)
y_train = df1['SalePrice']
```

In [517]:

```
X_int = sm.add_constant(X_train)
model = sm.OLS(y_train, X_int).fit()
model.summary()
```

Out[517]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.389
Model:	OLS	**Adj. R-squared:**	0.389
Method:	Least Squares	**F-statistic:**	1.010e+04
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:20:56	**Log-Likelihood:**	-2.2843e+05
No. Observations:	15851	**AIC:**	4.569e+05
Df Residuals:	15849	**BIC:**	4.569e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-9291.9955	8504.112	-1.093	0.275	-2.6e+04	7377.030
x1	374.8940	3.730	100.509	0.000	367.583	382.205

Omnibus:	19945.027	**Durbin-Watson:**	0.685
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9825842.224
Skew:	6.470	**Prob(JB):**	0.00
Kurtosis:	124.284	**Cond. No.**	5.56e+03

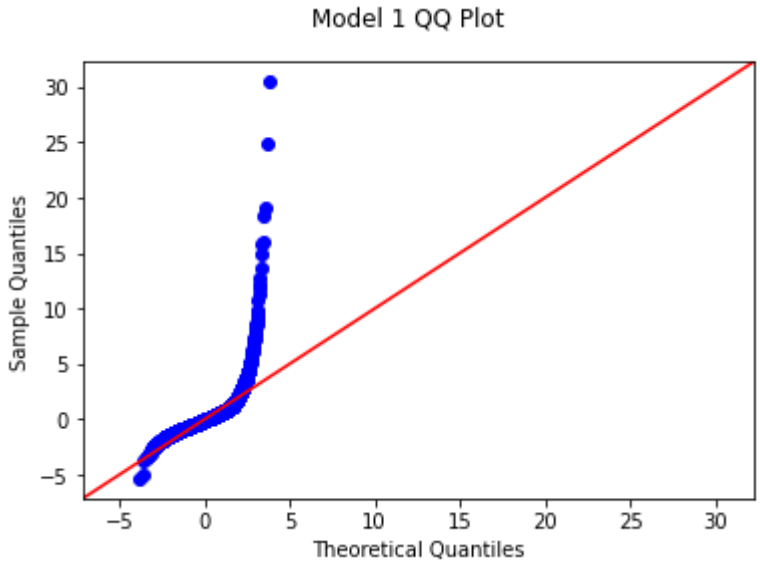
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.56e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Ok so there are issues with this model of course, only a single feature has been used. It still manages to explain 39% of the variance in SalePrice, which I'm actually surprised at - this suggests the total square footage of living space is an important factor in house prices - perhaps this is less surprising!

In [518]:

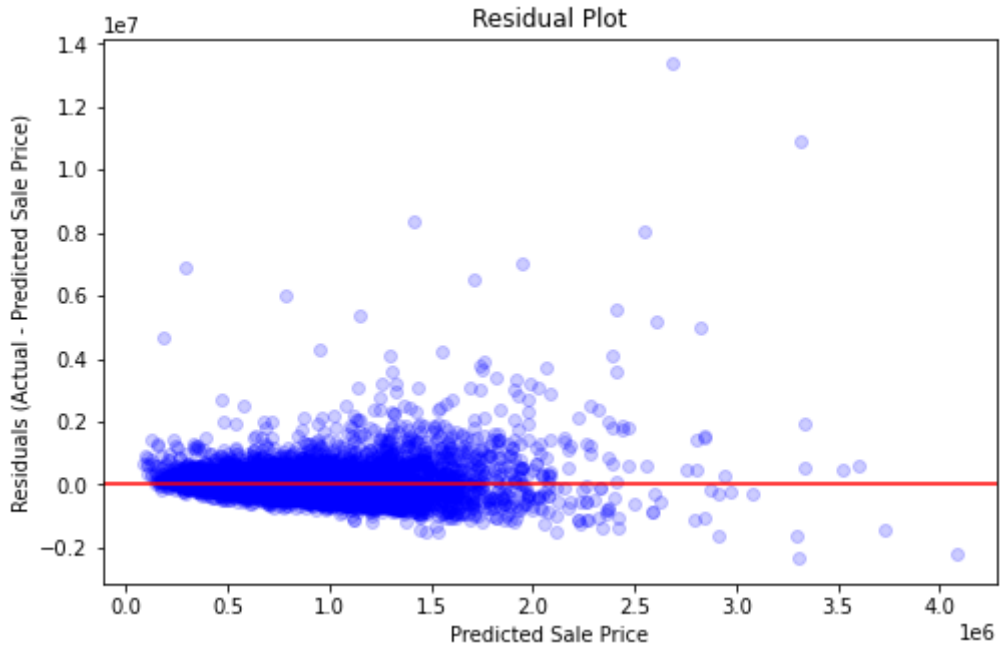
```
get_qq(model, 'Model 1')
```

```
..\..\src\useful_functions.py:70: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cann
ot show the figure.
  fig.show()
```



In [519]:

```
get_resid(df1, model)
```



In [520]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg.fit(X_train, y_train)

scores = cross_val_score(
            linreg,
            X_train,
            y_train,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores = np.sqrt(-scores)

display(rmse_scores.mean())
display(rmse_scores.std())
```

```
398611.71066358173

342506.0943804986
```

In [521]:

```
# Define table as a dataframe with specific columns. Each column's values will be a list th
summary = pd.DataFrame({'Model': [], 'Description':[], ' No. Features' : [], 'R^2':[],
                                'Adj R^2':[], 'RMSE': [],'RMSE sd':[], 'JB':[]})

# Add data for simple linear regression
summary.loc[0] = ['Simple Model - one independent variable', 'Square ft Total living', mode
                  , round(model.rsquared_adj,3), int(rmse_scores.mean()), int(rmse_scores.s

summary
```

Out[521]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0

It is pretty clear that this model violates the assumptions that residuals are normally distributed. This is also reflected with the extremely high JB number. The mean squared error is also very high, 398,611USD meaning that on average the house price predicted by the model could be +/- 400k USD away from the actual value - not the best.

I will remove outliers from this model to try and improve these metrics.

Simple Model 2 (outliers removed)

In [522]:

```
df2 = df.copy()
df2 = drop_outliers(df2, 'SalePrice', 3)
df2 = drop_outliers(df2, 'SqFtTotLiving', 3)
```

In [523]:

```
X_train2 = np.array(df2['SqFtTotLiving']).reshape(-1,1)
y_train2 = df2['SalePrice']
```

In [524]:

```
X_int2 = sm.add_constant(X_train2)
model2 = sm.OLS(y_train2, X_int2).fit()
model2.summary()
```

Out[524]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.354
Model:	OLS	**Adj. R-squared:**	0.354
Method:	Least Squares	**F-statistic:**	8444.
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:21:03	**Log-Likelihood:**	-2.1568e+05
No. Observations:	15411	**AIC:**	4.314e+05
Df Residuals:	15409	**BIC:**	4.314e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.671e+05	6371.126	26.230	0.000	1.55e+05	1.8e+05
x1	272.4155	2.965	91.893	0.000	266.605	278.226

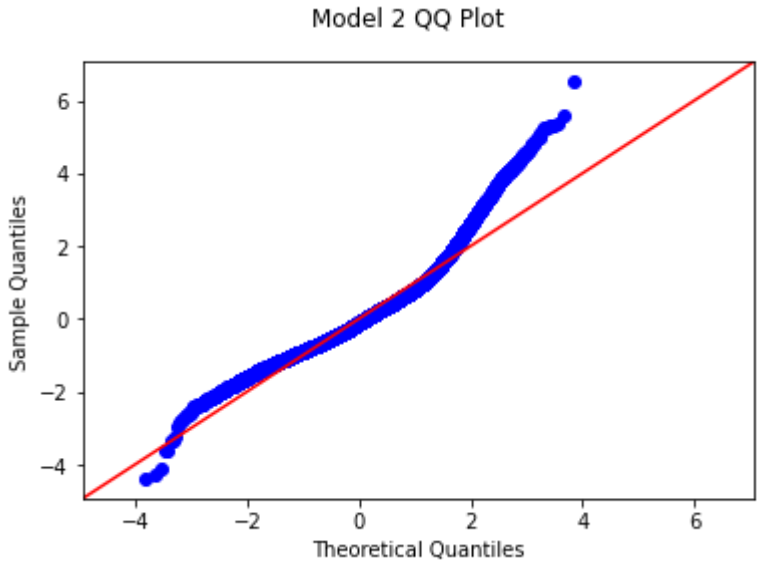
Omnibus:	2916.013	**Durbin-Watson:**	0.678
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	7132.340
Skew:	1.055	**Prob(JB):**	0.00
Kurtosis:	5.579	**Cond. No.**	5.87e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.87e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [525]:

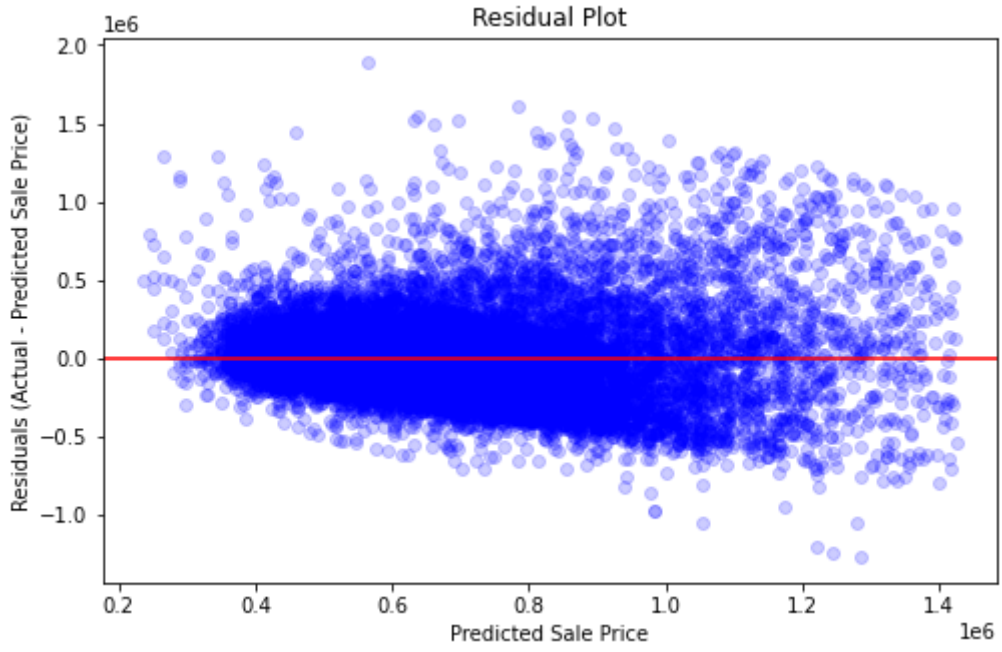
```
get_qq(model2, 'Model 2')
```

```
..\..\src\useful_functions.py:70: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cann
ot show the figure.
  fig.show()
```



In [526]:

```
get_resid(df2, model2)
```



In [527]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg2 = linreg.fit(X_train2, y_train2)

scores2 = cross_val_score(
            linreg2,
            X_train2,
            y_train2,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores2 = np.sqrt(-scores2)

display(rmse_scores2.mean())
display(rmse_scores2.std())
```

```
294579.90938941936

168991.91647047078
```

In [528]:

```
summary.loc[1] = ['Simple Model - outliers removed', 'Square ft Total living', model2.df_mo
                  , round(model2.rsquared_adj,3), int(rmse_scores2.mean()), int(rmse_scores

summary
```

Out[528]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0

Interesting that the R-squared value reduced, but the JB number has reduced significantly, and checking the qq plot it appears to be somewhat honouring residuals normality assumption to some extent, certainly better than previously. It is not honouring the homoscedasticity assumption.

Model 3 - Adding Features

Before adding new features, its important to check that they are have a linear relationship with SalePrice. This can be checked visually by plotting each potential feature vs SalePrice

In [529]:

```
# work from the dataframe that had outliers in price and sqfttotliving removed
df3 = df2.copy()
df3.drop(columns=['Unnamed: 0'], inplace=True)
```

In [530]:

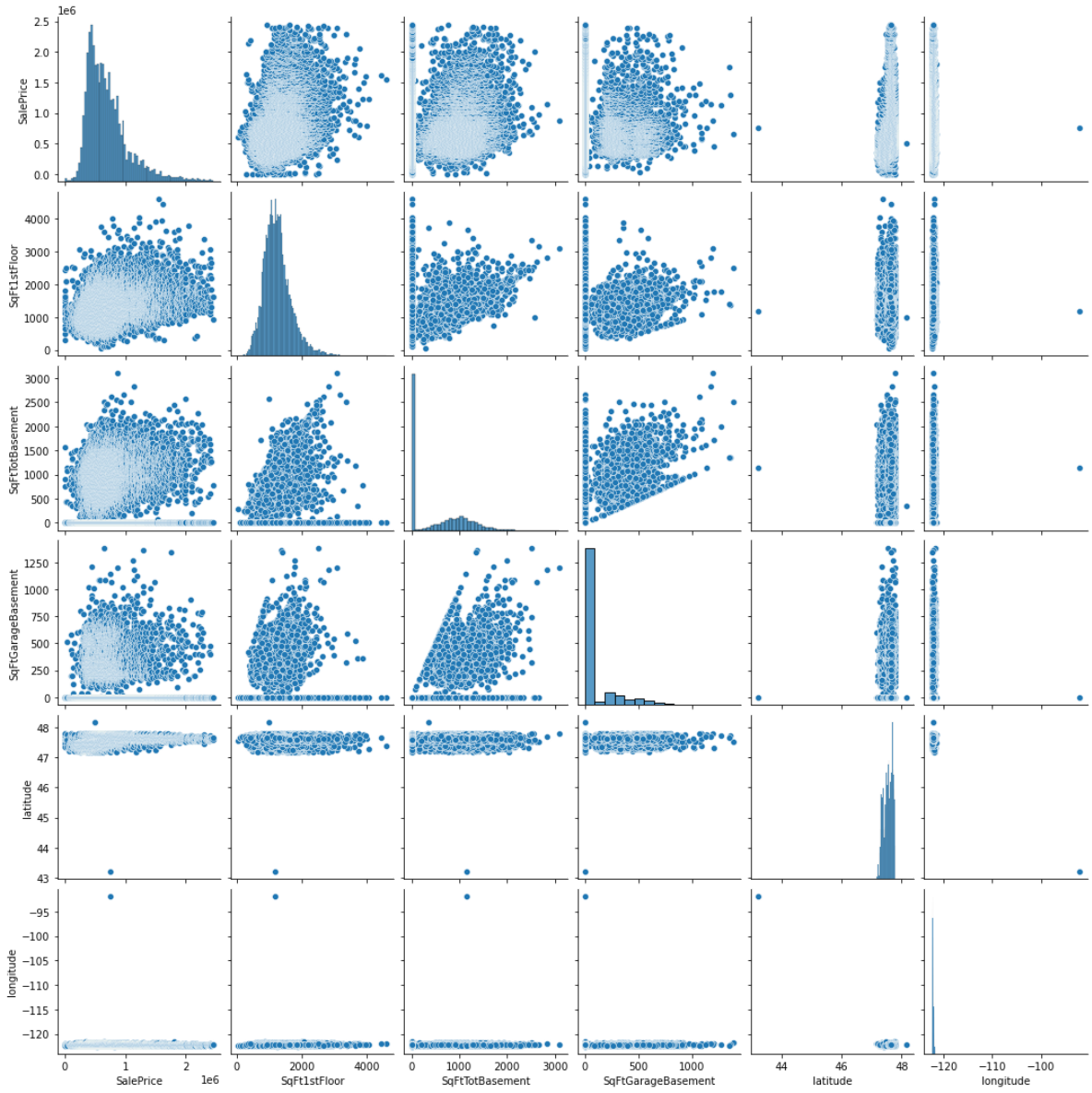
```
df3.columns
```

Out[530]:

```
Index(['Address', 'BuildingNumber', 'DirectionPrefix', 'StreetName',
       'StreetType', 'DirectionSuffix', 'ZipCode', 'Stories', 'BldgGrade',
       'SqFt1stFloor', 'SqFtHalfFloor', 'SqFt2ndFloor', 'SqFtUpperFloor',
       'SqFtTotLiving', 'SqFtTotBasement', 'SqFtFinBasement',
       'FinBasementGrade', 'SqFtGarageBasement', 'SqFtGarageAttached',
       'DaylightBasement', 'SqFtOpenPorch', 'SqFtEnclosedPorch', 'SqFtDeck',
       'HeatSystem', 'HeatSource', 'BrickStone', 'ViewUtilization', 'Bedroom
s',
       'BathHalfCount', 'Bath3qtrCount', 'BathFullCount', 'YrBuilt',
       'YrRenovated', 'Condition', 'id', 'Township', 'Section',
       'QuarterSection', 'Area', 'DistrictName', 'SqFtLot', 'Access',
       'Topography', 'InadequateParking', 'MtRainier', 'Olympics', 'Cascade
s',
       'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington',
       'LakeSammamish', 'SmallLakeRiverCreek', 'OtherView', 'WfntLocation',
       'TrafficNoise', 'PowerLines', 'OtherNuisances', 'AdjacentGreenbelt',
       'Easements', 'DocumentDate', 'SalePrice', 'SaleWarning', 'address',
       'excellent_view', 'latitude', 'longitude'],
      dtype='object')
```

In [531]:

```
cont_pairplot = ['SalePrice','SqFt1stFloor','SqFtTotBasement','SqFtGarageBasement', 'latitu

sns.pairplot(df3[cont_pairplot]);
```



These plots not very useful in their current state, however they have highlighted the need to remove further outliers..

In [532]:

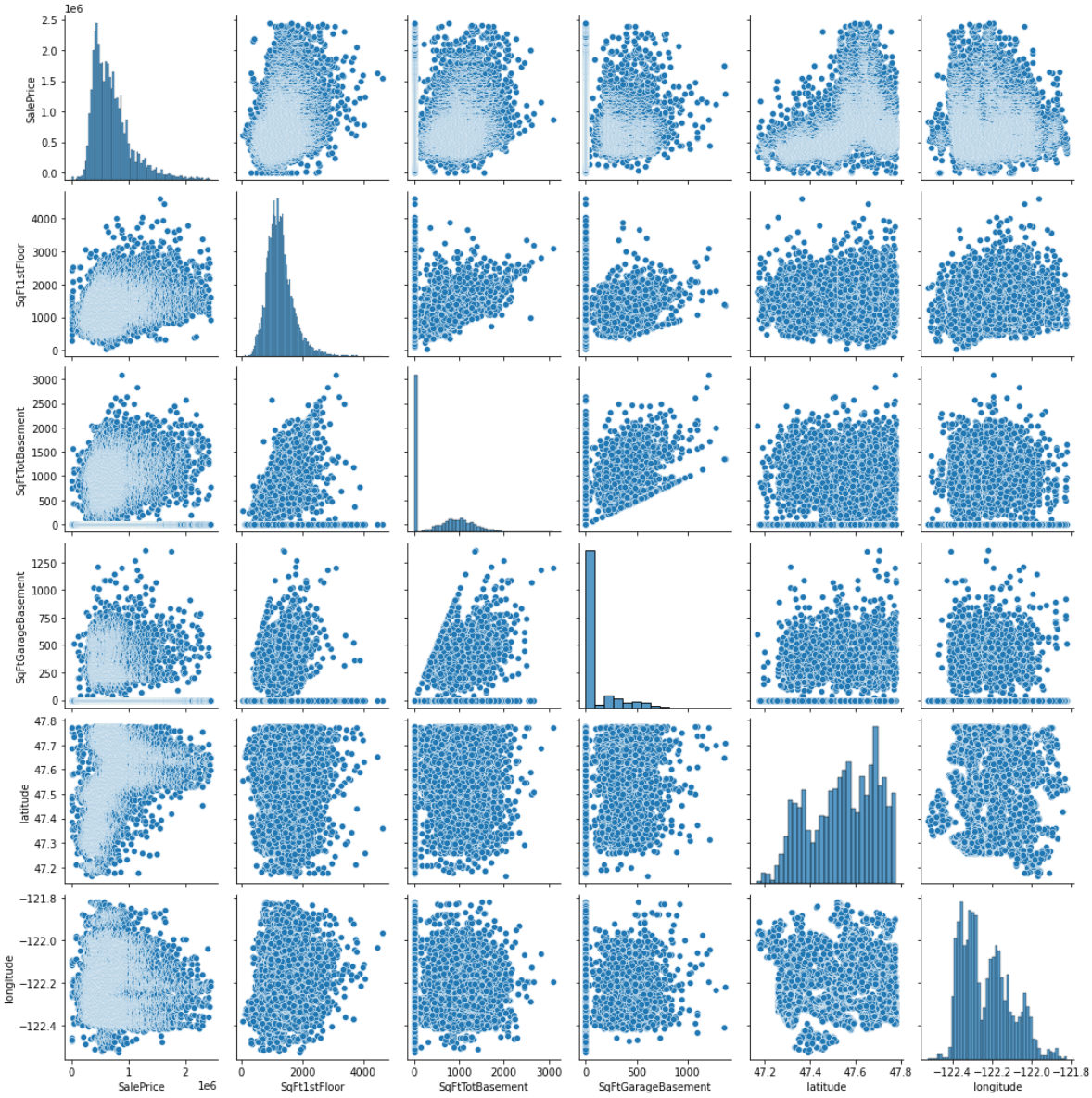
```
df3 = drop_outliers(df3, 'latitude', 3)
df3 = drop_outliers(df3, 'longitude', 3)
```

In [533]:

```
sns.pairplot(df3[cont_pairplot])
```

Out[533]:

```
<seaborn.axisgrid.PairGrid at 0x2210c1abdc0>
```



Messy data which is difficult to spot useful correlations but I think there are some trends particularly with SqfT1stFloor and latitude/longitude. It may be better to change some of these features into binary columns due to the high number of zeroes in each. For example SqFtGarageBasement could become (has_garagebasement).Aside from that, looking at the histograms, none of the features are normally distributed and could probably benefit from log transformation.

In [534]:

```
# drop outliers from 'SqFt1stFloor'
df3 = drop_outliers(df3, 'SqFt1stFloor', 3)
```

In [535]:

```
features_3 = ['SqFtTotLiving','SqFt1stFloor', 'latitude', 'longitude']
X_train3 = df3[features_3]
y_train3 = df3['SalePrice']
X_int3 = sm.add_constant(X_train3)
model3 = sm.OLS(y_train3, X_int3).fit()
model3.summary()
```

Out[535]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.535
Model:	OLS	**Adj. R-squared:**	0.535
Method:	Least Squares	**F-statistic:**	4327.
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:21:45	**Log-Likelihood:**	-2.0765e+05
No. Observations:	15030	**AIC:**	4.153e+05
Df Residuals:	15025	**BIC:**	4.154e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-7.291e+07	2.05e+06	-35.589	0.000	-7.69e+07	-6.89e+07
SqFtTotLiving	275.8215	3.236	85.238	0.000	269.479	282.164
SqFt1stFloor	-27.4989	6.371	-4.316	0.000	-39.988	-15.010
latitude	1.007e+06	1.37e+04	73.621	0.000	9.8e+05	1.03e+06
longitude	-2.064e+05	1.66e+04	-12.414	0.000	-2.39e+05	-1.74e+05

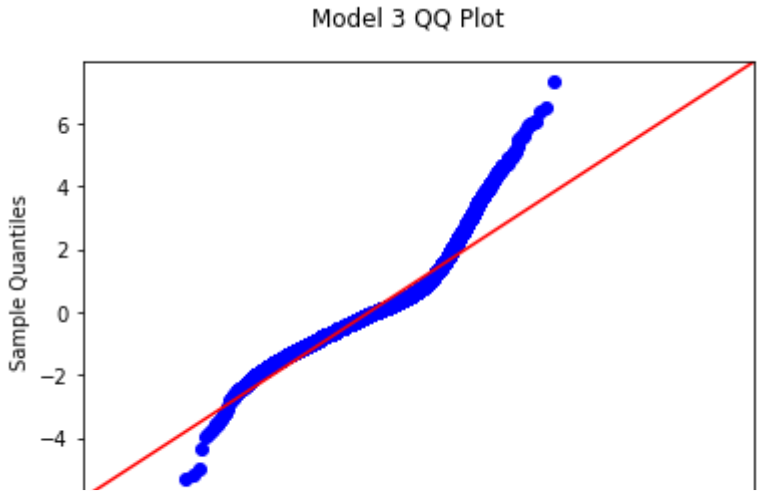
Omnibus:	4182.863	**Durbin-Watson:**	0.842
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18348.360
Skew:	1.302	**Prob(JB):**	0.00
Kurtosis:	7.746	**Cond. No.**	2.56e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.56e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [536]:

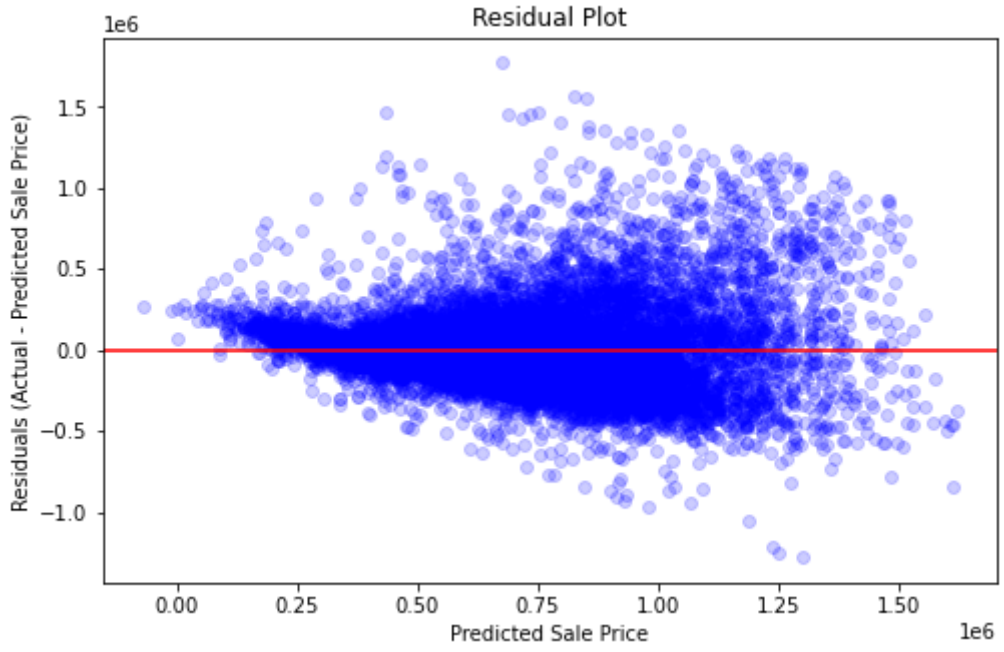
```
get_qq(model3, 'Model 3')
```

```
..\..\src\useful_functions.py:70: UserWarning: Matplotlib is currently usi
ng module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so
cannot show the figure.
  fig.show()
```



In [537]:

```
get_resid(df3, model3)
```



In [538]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg3 = linreg.fit(X_train3, y_train3)

scores3 = cross_val_score(
            linreg3,
            X_train3,
            y_train3,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores3 = np.sqrt(-scores3)

display(rmse_scores3.mean())
display(rmse_scores3.std())
```

```
238663.29701269456

158954.98852596313
```

In [539]:

```
summary.loc[2] = ['Multiple Linear Model 1', 'Continuous features', model3.df_model, round(
                  , round(model3.rsquared_adj,3), int(rmse_scores3.mean()), int(rmse_scores
                  round((sms.jarque_bera(model3.resid)[0]),0)]

summary
```

Out[539]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0

Model 4 (Distance to Expensive Areas and Age Features Added)

1. Most expensive house in mercer island is in 5330 Butterworth Rd, Mercer Island, Washington
2. Most expensive house in medina is in medina 7887 Overlake Drive West
3. NE Laurelcrest Lane has the most expensive house in Seattle.
4. Another populous area is Kent. 148th Ave SE is the most expensive neighbourhood in Kent and quite populous, I will add these four locations using Geocode

In [540]:

```
from geopy.geocoders import Nominatim
expensive = ['5330 Butterworth Rd, Mercer Island, Washington', '7887 Overlake Drive West, M
             'NE Laurelcrest Lane, Seattle, WA', '148th Ave SE, Kent, WA']
locator = Nominatim(user_agent='myGeocoder')
for loc in expensive:
    location = locator.geocode(loc)
    print(loc)
    print('Latitude = {}, Longitude = {}'.format(location.latitude, location.longitude))
```

```
5330 Butterworth Rd, Mercer Island, Washington
Latitude = 47.55621241561222, Longitude = -122.2129871659772
7887 Overlake Drive West, Medina, WA
Latitude = 47.6160689485921, Longitude = -122.23362543624816
NE Laurelcrest Lane, Seattle, WA
Latitude = 47.6560723, Longitude = -122.2725167
148th Ave SE, Kent, WA
Latitude = 47.3844953, Longitude = -122.1426915
```

In [541]:

```
laurelcrest = [47.6560723, -122.2725167]
medina = [47.6160689485921,-122.23362543624816]
mercer = [47.55621241561222, -122.2129871659772]
kent = [47.3844953, -122.1426915]
```

In [542]:

```
df4 = df3.copy()
```

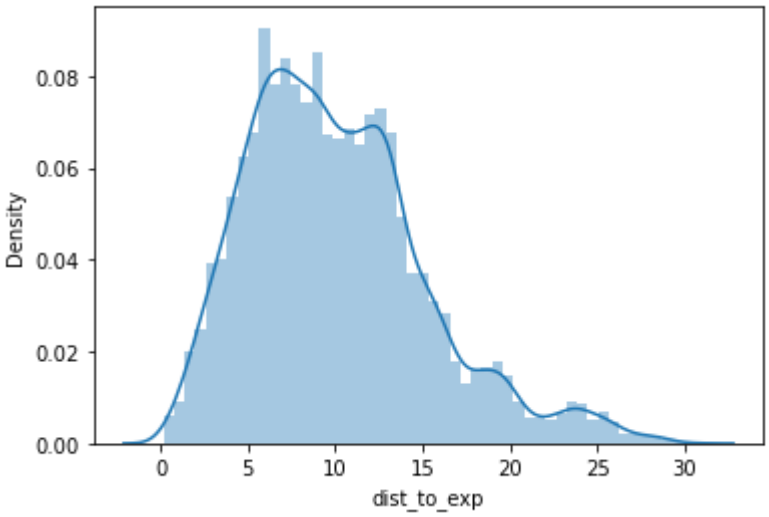
In [543]:

```
# create column in df4 with tuple containing lat long info
df4['lat_long'] = tuple(zip(df4.latitude, df4.longitude))
# calculate distance to each of the four expensive areas highlighted and create a column fo
df4['laurelcrest'] = calc_distances(df4['lat_long'], laurelcrest, df4)
df4['medina'] = calc_distances(df4['lat_long'], medina, df4)
df4['mercer'] = calc_distances(df4['lat_long'], mercer, df4)
df4['kent'] = calc_distances(df4['lat_long'], kent, df4)

# new column with the minimum value across these four columns
df4['dist_to_exp'] = df4[['laurelcrest', 'medina', 'mercer', 'kent']].min(axis=1, numeric_o

#drop the rest original 4 and work with minimum distance to an expensive area
df4.drop(['laurelcrest', 'medina', 'mercer', 'kent'], axis=1, inplace=True)
```

In [544]:

```
sns.distplot(df4.dist_to_exp);
```

```
C:\Users\Andrew\anaconda3\envs\geo-env\lib\site-packages\seaborn\distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and will be r
emoved in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-le
vel function for histograms).
  warnings.warn(msg, FutureWarning)
```



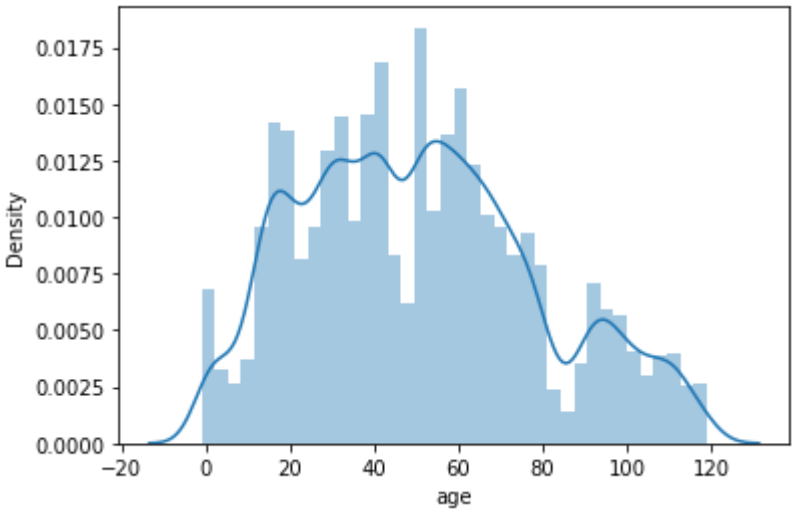
In [545]:

```
# create a new column for age feature
df4['age'] = 2019 - df4['YrBuilt']
sns.distplot(df4.age)
```

```
C:\Users\Andrew\anaconda3\envs\geo-env\lib\site-packages\seaborn\distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and will be r
emoved in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-le
vel function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[545]:

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



In [546]:

```
features_4 = ['SqFtTotLiving','SqFt1stFloor', 'latitude', 'longitude', 'dist_to_exp', 'age'
X_train4 = df4[features_4]
y_train4 = df4['SalePrice']
X_int4 = sm.add_constant(X_train4)
model4 = sm.OLS(y_train4, X_int4).fit()
model4.summary()
```

Out[546]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.567
Model:	OLS	**Adj. R-squared:**	0.566
Method:	Least Squares	**F-statistic:**	3274.
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:21:58	**Log-Likelihood:**	-2.0713e+05
No. Observations:	15030	**AIC:**	4.143e+05
Df Residuals:	15023	**BIC:**	4.143e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.927e+07	2.18e+06	-22.633	0.000	-5.35e+07	-4.5e+07
SqFtTotLiving	288.1086	3.320	86.790	0.000	281.602	294.615
SqFt1stFloor	-31.7000	6.287	-5.042	0.000	-44.024	-19.376
latitude	9.297e+05	1.35e+04	69.111	0.000	9.03e+05	9.56e+05
longitude	-4.327e+04	1.73e+04	-2.498	0.013	-7.72e+04	-9309.938
dist_to_exp	-1.057e+04	382.960	-27.596	0.000	-1.13e+04	-9817.377
age	1149.7452	79.533	14.456	0.000	993.850	1305.640

Omnibus:	3693.396	**Durbin-Watson:**	0.925
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	16023.068
Skew:	1.145	**Prob(JB):**	0.00
Kurtosis:	7.510	**Cond. No.**	2.82e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.82e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [547]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg4 = linreg.fit(X_train4, y_train4)

scores4 = cross_val_score(
            linreg4,
            X_train4,
            y_train4,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores4 = np.sqrt(-scores4)

display(rmse_scores4.mean())
display(rmse_scores4.std())
```

```
236962.5891214435

153473.79543835213
```

In [548]:

```
summary.loc[3] = ['Multiple Linear Model 2', 'Continuous features + age/dist_to_exp', model
                  , round(model4.rsquared_adj,3), int(rmse_scores4.mean()), int(rmse_scores
                  round((sms.jarque_bera(model4.resid)[0]),0)]

summary
```

Out[548]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.0	153473.0	16023.0

Model 5 - Add Binary Features

The next model I will test is one with the binary columns added to see what impact this will have.

In [549]:

```
df5 = df4.copy()
df5.columns
```

Out[549]:

```
Index(['Address', 'BuildingNumber', 'DirectionPrefix', 'StreetName',
       'StreetType', 'DirectionSuffix', 'ZipCode', 'Stories', 'BldgGrade',
       'SqFt1stFloor', 'SqFtHalfFloor', 'SqFt2ndFloor', 'SqFtUpperFloor',
       'SqFtTotLiving', 'SqFtTotBasement', 'SqFtFinBasement',
       'FinBasementGrade', 'SqFtGarageBasement', 'SqFtGarageAttached',
       'DaylightBasement', 'SqFtOpenPorch', 'SqFtEnclosedPorch', 'SqFtDeck',
       'HeatSystem', 'HeatSource', 'BrickStone', 'ViewUtilization', 'Bedroom
s',
       'BathHalfCount', 'Bath3qtrCount', 'BathFullCount', 'YrBuilt',
       'YrRenovated', 'Condition', 'id', 'Township', 'Section',
       'QuarterSection', 'Area', 'DistrictName', 'SqFtLot', 'Access',
       'Topography', 'InadequateParking', 'MtRainier', 'Olympics', 'Cascade
s',
       'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington',
       'LakeSammamish', 'SmallLakeRiverCreek', 'OtherView', 'WfntLocation',
       'TrafficNoise', 'PowerLines', 'OtherNuisances', 'AdjacentGreenbelt',
       'Easements', 'DocumentDate', 'SalePrice', 'SaleWarning', 'address',
       'excellent_view', 'latitude', 'longitude', 'lat_long', 'dist_to_exp',
       'age'],
      dtype='object')
```

In [550]:

```
# engineer column for traffic noise
df5.loc[df5['TrafficNoise']!= 0, 'TrafficNoise'] = 1

# engineer a column for has_basement
df5['has_basement'] = 0
df5.loc[(df5['SqFtGarageBasement']!= 0)|(df5['SqFtTotBasement']!= 0)|\
        (df5['DaylightBasement']!= 0), 'has_basement'] = 1

# engineer a column for has_deck
df5['has_deck'] = 0
df5.loc[(df5['SqFtDeck']!= 0),'has_deck'] = 1

# engineer a column for has_porch
df5['has_porch'] = 0
df5.loc[(df5['SqFtOpenPorch']!= 0)|(df5['SqFtEnclosedPorch']!= 0), 'has_porch'] = 1

# engineer a column for has_renovation
df5['has_renovation'] = 0
df5.loc[df5['YrRenovated'] != 0, 'has_renovation'] = 1

# engineer a column for has_problem
df5['has_problem'] = 0
df5.loc[(df5['PowerLines'] != 0)|(df5['OtherNuisances'] != 0), 'has_problem'] = 1
```

In [551]:

```
binary_cols5 = ['AdjacentGreenbelt','excellent_view','Topography', 'InadequateParking', 'Mt
                'Olympics', 'Cascades','Territorial', 'SeattleSkyline', 'PugetSound', 'LakeW
                'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'SaleWarnin
                'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_problem'

cont_5 = ['SqFtTotLiving','SqFt1stFloor', 'latitude', 'longitude', 'dist_to_exp', 'age']
```

In [552]:

```
features_5 = binary_cols5 + cont_5
X_train5 = df5[features_5]
y_train5 = df5['SalePrice']
X_int5 = sm.add_constant(X_train5)
model5 = sm.OLS(y_train5, X_int5).fit()
model5.summary()
```

Out[552]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.622
Model:	OLS	**Adj. R-squared:**	0.622
Method:	Least Squares	**F-statistic:**	852.7
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:22:04	**Log-Likelihood:**	-2.0609e+05
No. Observations:	15030	**AIC:**	4.122e+05
Df Residuals:	15000	**BIC:**	4.125e+05
Df Model:	29		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.473e+07	2.19e+06	-20.387	0.000	-4.9e+07	-4.04e+07
AdjacentGreenbelt	1.538e+04	1.18e+04	1.308	0.191	-7662.162	3.84e+04
excellent_view	1.13e+05	1.93e+04	5.853	0.000	7.52e+04	1.51e+05
Topography	1.856e+04	6930.618	2.677	0.007	4971.100	3.21e+04
InadequateParking	3.499e+04	3690.286	9.482	0.000	2.78e+04	4.22e+04
MtRainier	7.174e+04	2.15e+04	3.342	0.001	2.97e+04	1.14e+05
Olympics	7.625e+04	1.6e+04	4.775	0.000	4.49e+04	1.08e+05
Cascades	-6411.7770	1.32e+04	-0.485	0.627	-3.23e+04	1.95e+04
Territorial	5629.9652	9303.120	0.605	0.545	-1.26e+04	2.39e+04
SeattleSkyline	1.806e+05	2.13e+04	8.482	0.000	1.39e+05	2.22e+05
PugetSound	9.376e+04	1.44e+04	6.503	0.000	6.55e+04	1.22e+05
LakeWashington	1.588e+05	1.4e+04	11.338	0.000	1.31e+05	1.86e+05
LakeSammamish	3.203e+05	2.85e+04	11.239	0.000	2.64e+05	3.76e+05
SmallLakeRiverCreek	-3.239e+04	2.64e+04	-1.229	0.219	-8.41e+04	1.93e+04
OtherView	1.02e+05	3.15e+04	3.234	0.001	4.02e+04	1.64e+05
WfntLocation	2.695e+05	2.38e+04	11.321	0.000	2.23e+05	3.16e+05
Easements	2.628e+04	1.32e+04	1.985	0.047	335.462	5.22e+04
SaleWarning	-1.038e+05	7446.043	-13.945	0.000	-1.18e+05	-8.92e+04
ViewUtilization	1.977e+05	1.39e+04	14.211	0.000	1.7e+05	2.25e+05
has_porch	4.345e+04	3772.634	11.516	0.000	3.61e+04	5.08e+04
has_renovation	5.696e+04	8279.590	6.880	0.000	4.07e+04	7.32e+04

has_basement	-1.747e+04	4306.812	-4.057	0.000	-2.59e+04	-9030.786
TrafficNoise	-3.003e+04	5205.521	-5.769	0.000	-4.02e+04	-1.98e+04
has_problem	-2.974e+04	9259.612	-3.211	0.001	-4.79e+04	-1.16e+04
SqFtTotLiving	251.4487	3.557	70.700	0.000	244.477	258.420
SqFt1stFloor	-16.5864	6.160	-2.693	0.007	-28.661	-4.512
latitude	9.458e+05	1.28e+04	73.632	0.000	9.21e+05	9.71e+05
longitude	-103.5302	1.75e+04	-0.006	0.995	-3.43e+04	3.41e+04
dist_to_exp	-1.147e+04	371.311	-30.882	0.000	-1.22e+04	-1.07e+04
age	922.0280	81.847	11.265	0.000	761.598	1082.458

Omnibus:	3390.296	**Durbin-Watson:**	1.049
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	16415.648
Skew:	1.013	**Prob(JB):**	0.00
Kurtosis:	7.702	**Cond. No.**	3.04e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.04e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [553]:

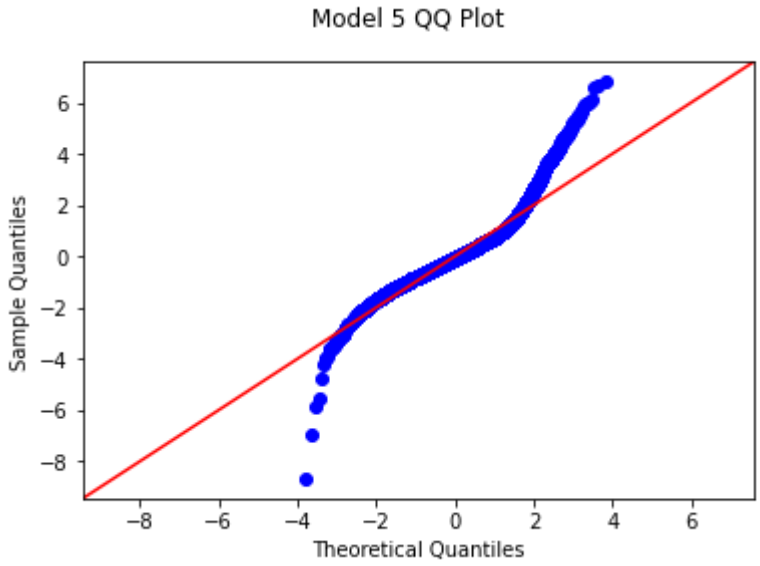
```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg5 = linreg.fit(X_train5, y_train5)

scores5 = cross_val_score(
            linreg5,
            X_train5,
            y_train5,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores5 = np.sqrt(-scores5)

display(rmse_scores5.mean())
display(rmse_scores5.std())
```

```
227275.879266802

146517.14238741784
```

In [554]:

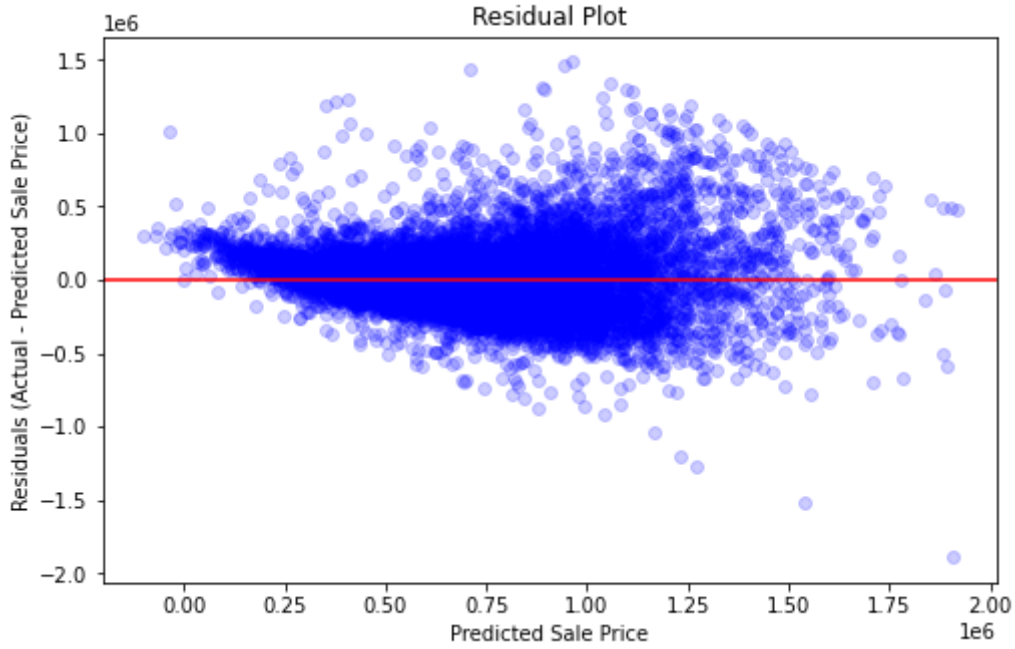
```
get_qq(model5, 'Model 5')
```

```
..\..\src\useful_functions.py:70: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cann
ot show the figure.
  fig.show()
```



In [555]:

```
get_resid(df5, model5)
```



In [556]:

```
summary.loc[4] = ['Multiple Linear Model 3', 'continuous + binary features', model5.df_mode
                  , round(model5.rsquared_adj,3), int(rmse_scores5.mean()), int(rmse_scores
                  round((sms.jarque_bera(model5.resid)[0]),0)]

summary
```

Out[556]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.0	153473.0	16023.0
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.0	146517.0	16416.0

Model 6 - Adding Categorical Features

ok now it is time to add further categorical features, however, we will need to onehotencode them first.

In [557]:

```
df5.columns
```

Out[557]:

```
Index(['Address', 'BuildingNumber', 'DirectionPrefix', 'StreetName',
       'StreetType', 'DirectionSuffix', 'ZipCode', 'Stories', 'BldgGrade',
       'SqFt1stFloor', 'SqFtHalfFloor', 'SqFt2ndFloor', 'SqFtUpperFloor',
       'SqFtTotLiving', 'SqFtTotBasement', 'SqFtFinBasement',
       'FinBasementGrade', 'SqFtGarageBasement', 'SqFtGarageAttached',
       'DaylightBasement', 'SqFtOpenPorch', 'SqFtEnclosedPorch', 'SqFtDeck',
       'HeatSystem', 'HeatSource', 'BrickStone', 'ViewUtilization', 'Bedroom
s',
       'BathHalfCount', 'Bath3qtrCount', 'BathFullCount', 'YrBuilt',
       'YrRenovated', 'Condition', 'id', 'Township', 'Section',
       'QuarterSection', 'Area', 'DistrictName', 'SqFtLot', 'Access',
       'Topography', 'InadequateParking', 'MtRainier', 'Olympics', 'Cascade
s',
       'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington',
       'LakeSammamish', 'SmallLakeRiverCreek', 'OtherView', 'WfntLocation',
       'TrafficNoise', 'PowerLines', 'OtherNuisances', 'AdjacentGreenbelt',
       'Easements', 'DocumentDate', 'SalePrice', 'SaleWarning', 'address',
       'excellent_view', 'latitude', 'longitude', 'lat_long', 'dist_to_exp',
       'age', 'has_basement', 'has_deck', 'has_porch', 'has_renovation',
       'has_problem'],
      dtype='object')
```

In [558]:

```
features_6 = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'Mt
              'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
              'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
             'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving', 'SqFt1stFloor
              'dist_to_exp', 'age', 'SalePrice']
```

Prior to adding in ZipCodes into the analysis, I will have a quick QC of the values. I don't think including zips with few entries will be valuable

In [559]:

```
df6 = df5.copy()
df6.ZipCode.value_counts()
```

Out[559]:

```
98042    521
98023    499
98058    443
98115    414
98038    403
98034    392
98117    383
98001    342
98118    337
98103    335
98052    331
98133    322
98155    304
98059    296
98006    294
98033    275
98056    273
98003    273
```

In [560]:

```
zips_to_keep = ['98042', '98023', '98058', '98117', '98115', '98034', '98038', '98133',
       '98001', '98118', '98103', '98155', '98052', '98003', '98031', '98059',
       '98125', '98056', '98168', '98106', '98126', '98092', '98146', '98198',
       '98006', '98178', '98116', '98002', '98033', '98030', '98074', '98022',
       '98177', '98008', '98028', '98055', '98122', '98144', '98199', '98136',
       '98045', '98107', '98072', '98108', '98166', '98011', '98029', '98027',
       '98053', '98105', '98188', '98112', '98032', '98065', '98075', '98119',
       '98019', '98077', '98007', '98070', '98014', '98040', '98148', '98005',
       '98109', '98004', '98102', '98024', '98010', '98047', '98051']
```

In [561]:

```
df6 = df6[df6['ZipCode'].isin(zips_to_keep)]
```

In [562]:

```
categoricals6 = ['HeatSystem', 'HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms',

cat6_ohe = df6[categoricals6]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder.fit(cat6_ohe)
cat6_ohe = encoder.transform(cat6_ohe)
cat6_ohe = pd.DataFrame(cat6_ohe, columns = encoder.get_feature_names())
cont6 = df6[features_6]
cat6_ohe.reset_index(drop=True, inplace=True)
cont6.reset_index(drop=True, inplace=True)
preprocessed6 = pd.concat([cont6, cat6_ohe], axis=1)
X_train6 = preprocessed6.drop('SalePrice', axis=1)
y_train6 = preprocessed6['SalePrice']
```

In [563]:

```
X_int6 = sm.add_constant(X_train6)
model6 = sm.OLS(y_train6, X_int6).fit()
model6.summary()
```

Out[563]:

OLS Regression Results

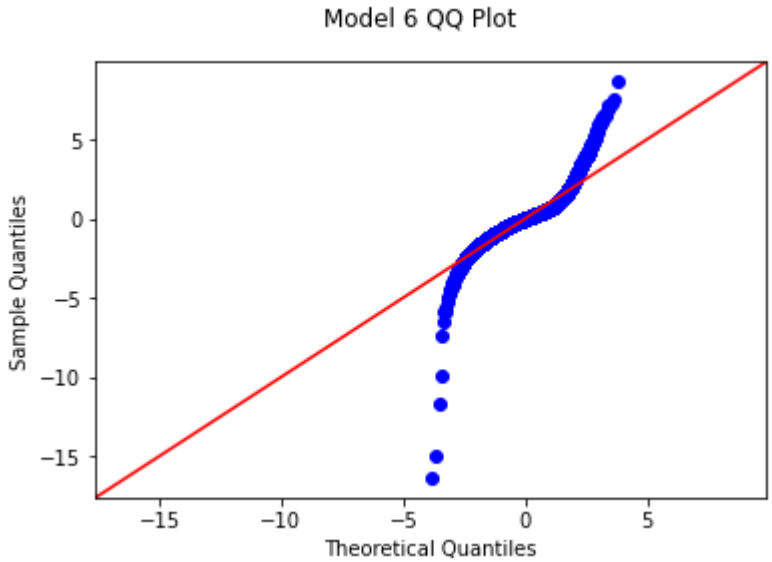
Dep. Variable:	SalePrice	**R-squared:**	0.816
Model:	OLS	**Adj. R-squared:**	0.814
Method:	Least Squares	**F-statistic:**	474.0
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:22:17	**Log-Likelihood:**	-1.9993e+05
No. Observations:	14980	**AIC:**	4.001e+05
Df Residuals:	14840	**BIC:**	4.012e+05
Df Model:	139		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**

In [564]:

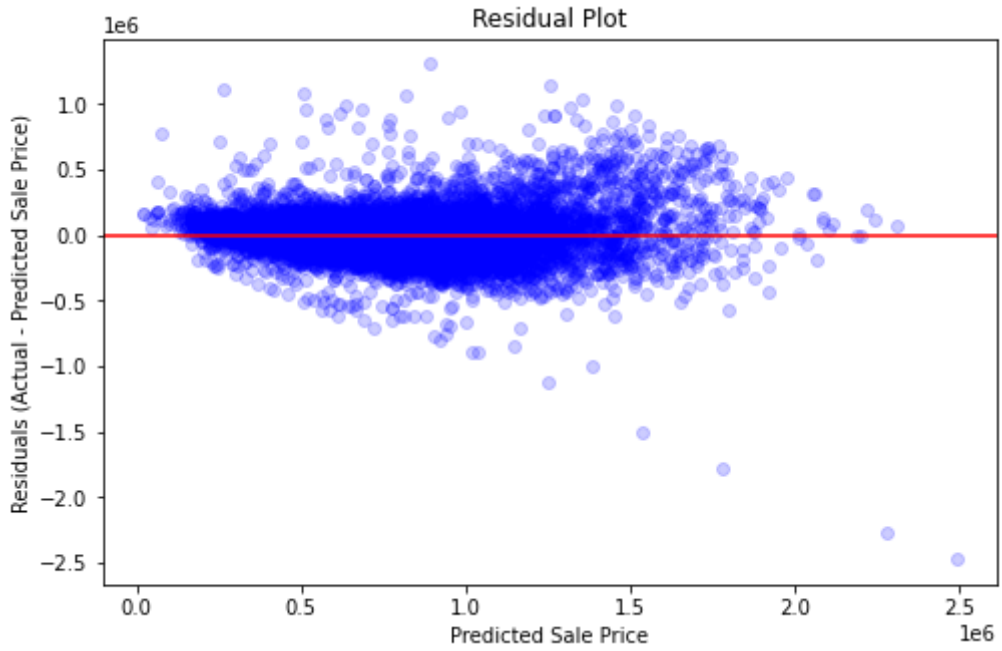
```
get_qq(model6, 'Model 6')
```

```
..\..\src\useful_functions.py:70: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cann
ot show the figure.
  fig.show()
```



In [565]:

```
get_resid(preprocessed6, model6)
```



In [566]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg6 = linreg.fit(X_train6, y_train6)

scores6 = cross_val_score(
            linreg6,
            X_train6,
            y_train6,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores6 = np.sqrt(-scores6)

display(rmse_scores6.mean())
display(rmse_scores6.std())
```

```
166951.55691694707
```

```
118312.22299791012
```

In [567]:

```
summary.loc[5] = ['Multiple Linear Model 4', 'continuous + binary features + cat', model6.d
                  , round(model6.rsquared_adj,3), int(rmse_scores6.mean()), int(rmse_scores
                  round((sms.jarque_bera(model6.resid)[0]),0)]

summary
```

Out[567]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.0	153473.0	16023.0
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.0	146517.0	16416.0
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.0	118312.0	165256.0

The JB metric on this model has increased by an order of magnitude and its clear from the qq plot that the normality of residuals assumption is not being met, there could also be strong multicollinearity at play. I will work to try and improve this model both in terms of R^2 but more importantly the JB number to provide confidence in the results.

In [568]:

```
# check for highly correlated features
get_multicol(preprocessed6)
```

Out[568]:

pairs	cc
(x2_PUBLIC, x2_PRIVATE)	0.985493
(x4_4, x4_3)	0.748812

Ok so two dummy variables are highly correlated, this would lead me to suspect the variable being dropped when onehotencoding has a relatively low count. It may be worse removing these rows from the analysis for the next model.

In [569]:

```
get_multicol(df6)
```

Out[569]:

	cc
pairs	
(age, YrBuilt)	1.000000
(has_renovation, YrRenovated)	0.999959
(latitude, Township)	0.984267
(has_basement, SqFtTotBasement)	0.882418
(OtherNuisances, has_problem)	0.846419
(FinBasementGrade, SqFtFinBasement)	0.845789
(SqFtFinBasement, SqFtTotBasement)	0.837436
(Stories, SqFt2ndFloor)	0.824724
(has_basement, FinBasementGrade)	0.822696
(SqFtTotBasement, FinBasementGrade)	0.811965
(SqFtTotLiving, BldgGrade)	0.715628
(SqFtFinBasement, has_basement)	0.702382

Model 7 - Cleaning features, adding SqFt2ndFloor

In [570]:

```
df7 = df6.copy()
```

In [571]:

```
df7.Condition.value_counts()
```

Out[571]:

```
3    8487
4    4497
5    1862
2     119
1      15
Name: Condition, dtype: int64
```

In [572]:

```
df7.Access.value_counts()
```

Out[572]:

```
PUBLIC               14096
PRIVATE                860
WALK IN                 11
RESTRICTED              11
LEGAL/UNDEVELOPED        2
Name: Access, dtype: int64
```

I will drop rows containing 1 and 2 in 'Condition' and keep only PUBLIC and PRIVATE in Access to try and reduce multicollinearity

In [573]:

```
df7 = df7[(df7['Condition']==3)|(df7['Condition']==4)|(df7['Condition']==5)]
df7 = df7[(df7['Access']=='PUBLIC')|(df7['Access']=='PRIVATE')]
```

In [574]:

```
df7.Condition.value_counts()
```

Out[574]:

```
3    8475
4    4489
5    1858
Name: Condition, dtype: int64
```

In [575]:

```
df7.Access.value_counts()
```

Out[575]:

```
PUBLIC     13970
PRIVATE      852
Name: Access, dtype: int64
```

In [576]:

```
# check other categorical features for such imbalances
df7.Bedrooms.value_counts()
```

Out[576]:

```
3     6784
4     4834
2     1801
5     1091
6      142
1      137
0       14
7       13
8        4
9        1
10       1
Name: Bedrooms, dtype: int64
```

In [577]:

```
# remove high and low values
df7 = df7[(df7['Bedrooms']>1)&(df7['Bedrooms']<7)]
```

In [578]:

```
df7.Bedrooms.value_counts()
```

Out[578]:

```
3    6784
4    4834
2    1801
5    1091
6     142
Name: Bedrooms, dtype: int64
```

In [579]:

```
df7.HeatSource.value_counts()
```

Out[579]:

```
Gas                  10348
Electricity           2578
Oil                   1676
Gas/Solar               30
Other                    9
Electricity/Solar        8
Oil/Solar                3
Name: HeatSource, dtype: int64
```

In [580]:

```
# remove low number entries
df7 = df7[(df7['HeatSource']=='Gas')|(df7['HeatSource']=='Electricity')|(df7['HeatSource']=
```

In [581]:

```
df7.HeatSource.value_counts()
```

Out[581]:

```
Gas            10348
Electricity     2578
Oil             1676
Name: HeatSource, dtype: int64
```

In [582]:

```
# add SqFt2ndFloor
features_7 = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'Mt
              'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
              'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
             'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving', 'SqFt1stFloor
              'dist_to_exp', 'age', 'SalePrice', 'SqFt2ndFloor']
categoricals7 = ['HeatSystem', 'HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms',
```

In [583]:

```
cat7_ohe = df7[categoricals7]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder7 = encoder.fit(cat7_ohe)
cat7_ohe = encoder7.transform(cat7_ohe)
cat7_ohe = pd.DataFrame(cat7_ohe, columns = encoder7.get_feature_names())
cont7 = df7[features_7]
cat7_ohe.reset_index(drop=True, inplace=True)
cont7.reset_index(drop=True, inplace=True)
preprocessed7 = pd.concat([cont7, cat7_ohe], axis=1)
X_train7 = preprocessed7.drop('SalePrice', axis=1)
y_train7 = preprocessed7['SalePrice']
```

In [584]:

```
X_int7 = sm.add_constant(X_train7)
model7 = sm.OLS(y_train7, X_int7).fit()
model7.summary()
```

Out[584]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.823
Model:	OLS	**Adj. R-squared:**	0.821
Method:	Least Squares	**F-statistic:**	547.0
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:22:45	**Log-Likelihood:**	-1.9462e+05
No. Observations:	14602	**AIC:**	3.895e+05
Df Residuals:	14478	**BIC:**	3.904e+05
Df Model:	123		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**

In [585]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg7 = linreg.fit(X_train7, y_train7)

scores7 = cross_val_score(
            linreg7,
            X_train7,
            y_train7,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores7 = np.sqrt(-scores7)

display(rmse_scores7.mean())
display(rmse_scores7.std())
```

```
164932.23909308625

118703.0456134486
```

In [586]:

```
summary.loc[6] = ['Multiple Linear Model 5', 'MLP 4 + SqFt2nd + Clean Cat', model7.df_model
                  , round(model7.rsquared_adj,3), int(rmse_scores7.mean()), int(rmse_scores
                  round((sms.jarque_bera(model7.resid)[0]),0)]

summary
```

Out[586]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.0	153473.0	16023.0
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.0	146517.0	16416.0
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.0	118312.0	165256.0
6	Multiple Linear Model 5	MLP 4 + SqFt2nd + Clean Cat	123.0	0.823	0.821	164932.0	118703.0	206712.0

By reducing the number of features, we actually managed to improve the R-squared, which is perhaps and indication that the features that were dropped were causing problems. However, the JB number is still high. Another pass of outliers may be required.

Longitude has been a persistent bad actor, again this is worth investigating further

In [587]:

```
df7.HeatSystem.value_counts()
```

Out[587]:

```
Forced Air    11710
Elec BB         986
Heat Pump       974
Floor-Wall      420
Hot Water       320
Radiant         152
Gravity          36
Other             4
Name: HeatSystem, dtype: int64
```

It was noted that the majority of the dummy variables pertaining to the Heat System column had high P-values. It might be worth making this a binary column, either forced air or not. I will create a new column for this rather

than lose the data here.

In [588]:

```
df7.SalePrice.describe()
```

Out[588]:

```
count    1.460200e+04
mean     7.077948e+05
std      3.533408e+05
min      1.000000e+01
25%      4.500000e+05
50%      6.300000e+05
75%      8.520000e+05
max      2.450000e+06
Name: SalePrice, dtype: float64
```

A very large spread on this, but interestingly some low values that I am keen to investigate more.

In [589]:

```
df7[df7['SalePrice']< 100000].sort_values('SalePrice')
```

Out[589]:

	Address	BuildingNumber	DirectionPrefix	StreetName	StreetType	DirectionSuffix	ZipCode	Stor
0	17701 185TH AVE NE 98072	17701		185TH	AVE	NE	98072	
1	9508 167TH AVE NE 98052	9508		167TH	AVE	NE	98052	
2	19361 61ST AVE NE 98028	19361		61ST	AVE	NE	98028	
3	15915 VASHON HWY	15915		VASHON	HWY	SW	98070	

There are several properties in sought after areas (Mercer Island, Bellevue) that have clearly had the wrong price entered. I will remove these for the next model.

In [590]:

```
df7.describe()
```

Out[590]:

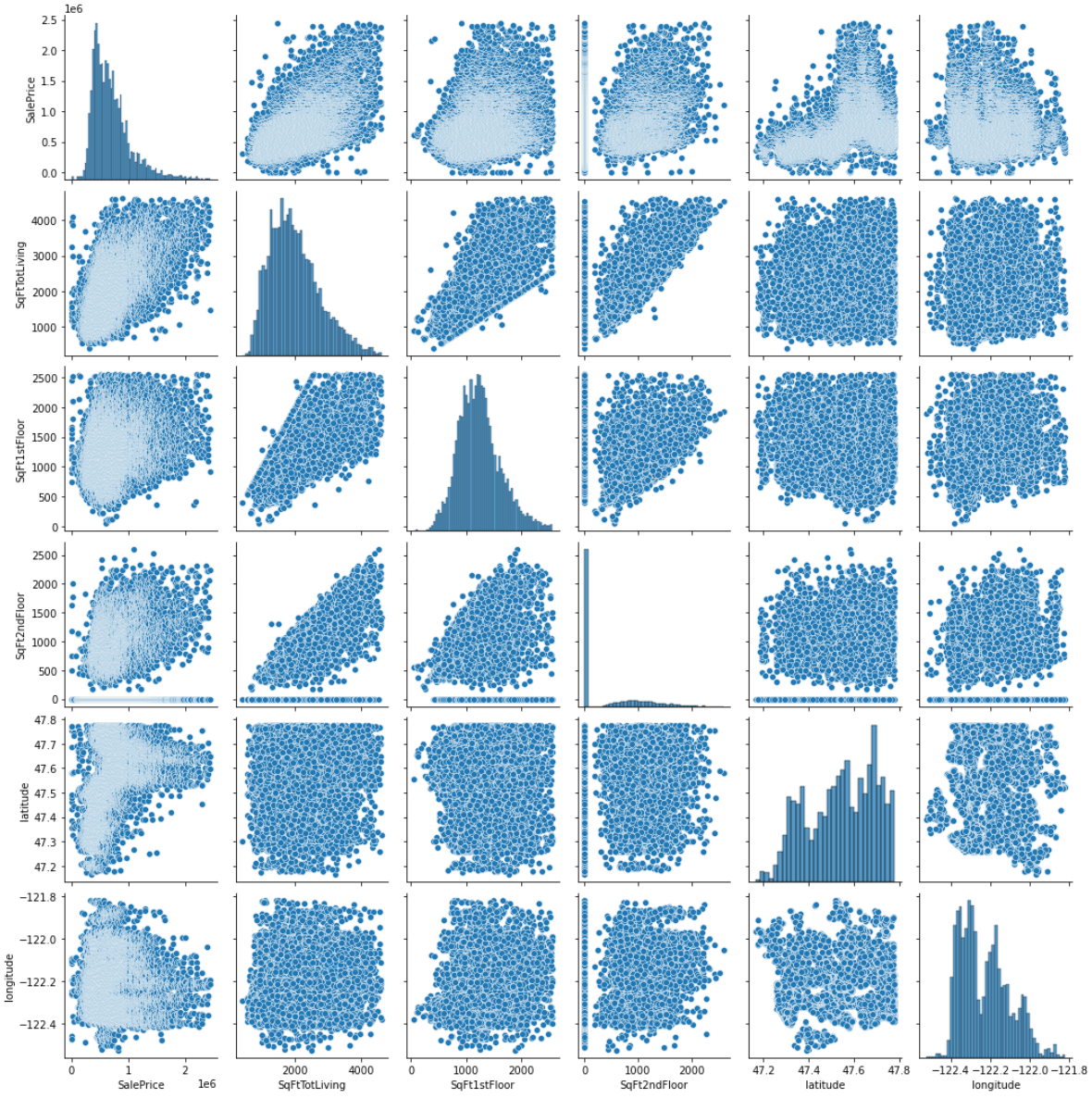
	BuildingNumber	Stories	BldgGrade	SqFt1stFloor	SqFtHalfFloor	SqFt2ndFloor
count	14602.000000	14602.000000	14602.000000	14602.000000	14602.000000	14602.000000
mean	10789.922202	1.424702	7.517532	1246.618956	49.453910	388.716683
std	9420.323006	0.501587	1.027597	389.750327	169.907707	570.707007
min	6.000000	1.000000	5.000000	50.000000	0.000000	0.000000
25%	2818.000000	1.000000	7.000000	980.000000	0.000000	0.000000
50%	8410.000000	1.000000	7.000000	1210.000000	0.000000	0.000000
75%	16519.000000	2.000000	8.000000	1460.000000	0.000000	830.000000
max	48617.000000	3.000000	12.000000	2560.000000	1930.000000	2600.000000

In [591]:

```
cont_pairplot7 = ['SalePrice','SqFtTotLiving','SqFt1stFloor','SqFt2ndFloor', 'latitude', 'l
sns.pairplot(df7[cont_pairplot7])
```

Out[591]:

```
<seaborn.axisgrid.PairGrid at 0x2211f8942b0>
```



Looking at this I am not convinced by longitude, it is not linear. I will remove it from the next model.

Model 8 - More outlier removal, drop longitude

In [592]:

```
df8 = df7.copy()
```

In [593]:

```
# get rid of low and probably erroneous home values - I could replace with average home val
df8 = df8[df8['SalePrice']> 75000]

# creating category for heat system type
df8['heating'] = 0
df8.loc[(df8['HeatSystem'] == 'Forced Air'),'heating'] = 1
```

In [594]:

```
df8[df8.SqFt1stFloor<300]
```

Out[594]:

	Address	BuildingNumber	DirectionPrefix	StreetName	StreetType	DirectionSuffix	Zip
1595	3863 21ST AVE SW 98106	3863		21ST	AVE	SW	9
4894	819 B NW 97TH ST 98117	819	NW	97TH	ST		9
5681	1116 A 13TH AVE 98122	1116		13TH	AVE		9
5935	1116 B 13TH AVE 98122	1116		13TH	AVE		9
7145	4262 A WINSLOW PL N 98103	4262		WINSLOW	PL	N	9
7150	3905 C SW HUDSON ST 98116	3905	SW	HUDSON	ST		9
7771	6342 5TH AVE NE 98115	6342		5TH	AVE	NE	9
8013	3636 C WHITMAN AVE N 98103	3636		WHITMAN	AVE	N	9
8613	4414 A MERIDIAN AVE N 98103	4414		MERIDIAN	AVE	N	9
9227	921 C N 35TH ST 98103	921	N	35TH	ST		9
10375	7407 4TH AVE NE 98115	7407		4TH	AVE	NE	9

In [595]:

```
# get rid of the outliers in the 1st floor category on the low side, they maybe represent s
# setting these first floor values to their second floor equivalents
df8.loc[df8.SqFt1stFloor<300, 'SqFt1stFloor'] = df8['SqFt2ndFloor']
```

In [596]:

```
# remove longitude, add new heating feature, remove heatsystem from categoricals
features_8 = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'Mt
              'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
              'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
             'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving', 'SqFt1stFloor
              'dist_to_exp', 'age', 'SalePrice', 'SqFt2ndFloor', 'heating']
categoricals8 = ['HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms', 'BldgGrade']
```

In [597]:

```
cat8_ohe = df8[categoricals8]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder8 = encoder.fit(cat8_ohe)
cat8_ohe = encoder8.transform(cat8_ohe)
cat8_ohe = pd.DataFrame(cat8_ohe, columns = encoder8.get_feature_names())
cont8 = df8[features_8]
cat8_ohe.reset_index(drop=True, inplace=True)
cont8.reset_index(drop=True, inplace=True)
preprocessed8 = pd.concat([cont8, cat8_ohe], axis=1)
X_train8 = preprocessed8.drop('SalePrice', axis=1)
y_train8 = preprocessed8['SalePrice']
```

In [598]:

```
X_int8 = sm.add_constant(X_train8)
model8 = sm.OLS(y_train8, X_int8).fit()
model8.summary()
```

Out[598]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.834
Model:	OLS	**Adj. R-squared:**	0.833
Method:	Least Squares	**F-statistic:**	626.6
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:23:10	**Log-Likelihood:**	-1.9383e+05
No. Observations:	14580	**AIC:**	3.879e+05
Df Residuals:	14463	**BIC:**	3.888e+05
Df Model:	116		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**

In [599]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg8 = linreg.fit(X_train8, y_train8)

scores8 = cross_val_score(
            linreg8,
            X_train8,
            y_train8,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores8 = np.sqrt(-scores8)

display(rmse_scores8.mean())
display(rmse_scores8.std())
```

```
158260.95861193296

111643.75578564298
```

In [600]:

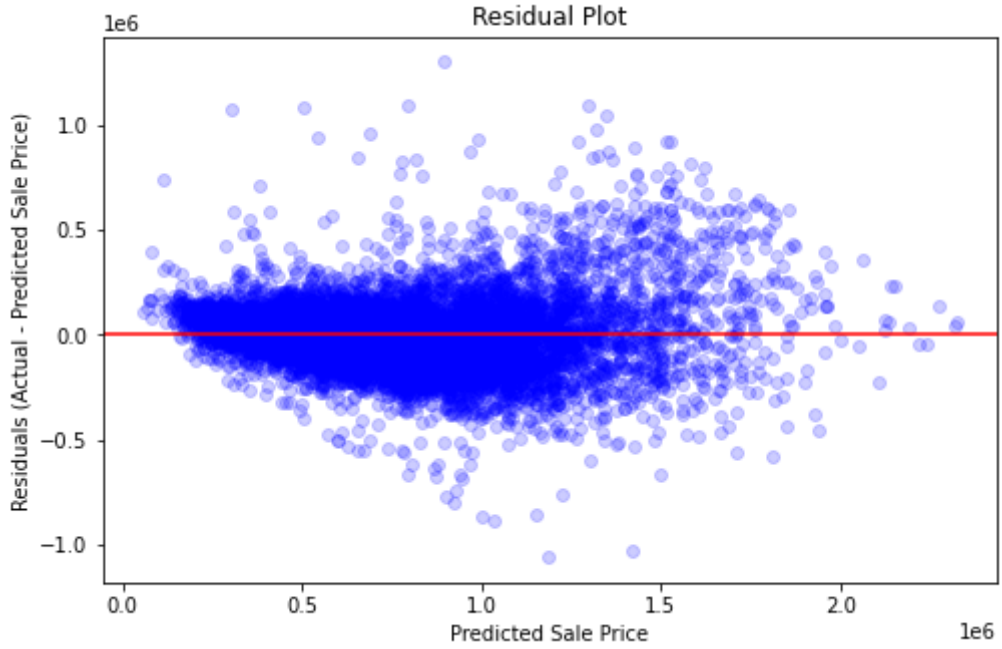
```
summary.loc[7] = ['Multiple Linear Model 6', 'MLP 5 outliers longitude removed', model8.df_
                  , round(model8.rsquared_adj,3), int(rmse_scores8.mean()), int(rmse_scores
                  round((sms.jarque_bera(model8.resid)[0]),0)]

summary
```

Out[600]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.0	153473.0	16023.0
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.0	146517.0	16416.0
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.0	118312.0	165256.0
6	Multiple Linear Model 5	MLP 4 + SqFt2nd + Clean Cat	123.0	0.823	0.821	164932.0	118703.0	206712.0
7	Multiple Linear Model 6	MLP 5 outliers longitude removed	116.0	0.834	0.833	158260.0	111643.0	32901.0

In [601]:

```
get_resid(preprocessed8, model8)
```



Model 9 - YES I FORGOT BATHROOMS!!

In [602]:

```
df9 = df8.copy()
```

In [603]:

```
df9.columns
```

Out[603]:

```
Index(['Address', 'BuildingNumber', 'DirectionPrefix', 'StreetName',
       'StreetType', 'DirectionSuffix', 'ZipCode', 'Stories', 'BldgGrade',
       'SqFt1stFloor', 'SqFtHalfFloor', 'SqFt2ndFloor', 'SqFtUpperFloor',
       'SqFtTotLiving', 'SqFtTotBasement', 'SqFtFinBasement',
       'FinBasementGrade', 'SqFtGarageBasement', 'SqFtGarageAttached',
       'DaylightBasement', 'SqFtOpenPorch', 'SqFtEnclosedPorch', 'SqFtDeck',
       'HeatSystem', 'HeatSource', 'BrickStone', 'ViewUtilization', 'Bedroom
s',
       'BathHalfCount', 'Bath3qtrCount', 'BathFullCount', 'YrBuilt',
       'YrRenovated', 'Condition', 'id', 'Township', 'Section',
       'QuarterSection', 'Area', 'DistrictName', 'SqFtLot', 'Access',
       'Topography', 'InadequateParking', 'MtRainier', 'Olympics', 'Cascade
s',
       'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington',
       'LakeSammamish', 'SmallLakeRiverCreek', 'OtherView', 'WfntLocation',
       'TrafficNoise', 'PowerLines', 'OtherNuisances', 'AdjacentGreenbelt',
       'Easements', 'DocumentDate', 'SalePrice', 'SaleWarning', 'address',
       'excellent_view', 'latitude', 'longitude', 'lat_long', 'dist_to_exp',
       'age', 'has_basement', 'has_deck', 'has_porch', 'has_renovation',
       'has_problem', 'heating'],
      dtype='object')
```

In [604]:

```
df9['bathrooms'] = df9['BathHalfCount']*0.5 + df9['Bath3qtrCount']*0.75 + df9['BathFullCoun
```

In [605]:

```
df9.bathrooms.value_counts()
```

Out[605]:

```
2.50    3404
1.75    2444
1.00    2407
2.25    1495
2.00    1409
1.50    1006
2.75     930
3.00     492
3.50     386
3.25     376
3.75      68
4.00      54
4.25      38
0.75      35
4.50      16
1.25      12
4.75       4
5.25       2
6.25       1
5.50       1
Name: bathrooms, dtype: int64
```

In [606]:

```
df9['bedbath'] = df9.Bedrooms - df9.bathrooms
```

In [607]:

```
df9.bedbath.value_counts()
```

Out[607]:

```
 1.50    2373
 1.00    2076
 1.25    2072
 0.50    1798
 2.00    1680
 0.75    1025
 2.25     778
 1.75     636
 0.25     492
 2.50     432
 3.00     305
 0.00     262
-0.25     157
-0.50     154
 3.25     107
 2.75      93
 3.50      49
 4.00      32
-0.75      19
 4.25      13
-1.25       8
-1.00       7
 4.50       5
 3.75       5
-1.50       2
Name: bedbath, dtype: int64
```

In [608]:

```
df9.bathbins = 0
df9.loc[(df9['bedbath'] < 1),'bathbins'] = 'less than 1'
df9.loc[(df9['bedbath'] > 0.99) & (df9['bedbath'] < 2), 'bathbins'] = '1 to 2'
df9.loc[(df9['bedbath'] > 1.99), 'bathbins'] = 'greater than 2'
```

In [609]:

```
df9.head()
```

Out[609]:

	Address	BuildingNumber	DirectionPrefix	StreetName	StreetType	DirectionSuffix	ZipCode
27	5131 S 324TH ST 98001	5131	S	324TH	ST		98001
28	26636 188TH AVE SE 98042	26636		188TH	AVE	SE	98042
29	21058 98TH AVE S 98031	21058		98TH	AVE	S	98031
32	8709 14TH AVE NW 98117	8709		14TH	AVE	NW	98117
33	7045 S 125TH ST 98178	7045	S	125TH	ST		98178

In [610]:

```
df9['bathbins'].value_counts()
```

Out[610]:

```
1 to 2            7157
less than 1       3924
greater than 2    3499
Name: bathbins, dtype: int64
```

I will experiment with these bathroom columns and see what happens to the model r-squared.

In [611]:

```
#add bedbath and also the bathbins
features_9 = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'Mt
              'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
              'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
             'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving', 'SqFt1stFloor
              'dist_to_exp', 'age', 'SalePrice', 'SqFt2ndFloor', 'heating', 'bedbath']
categoricals9 = ['HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms', 'BldgGrade', '
```

In [612]:

```
cat9_ohe = df9[categoricals9]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder9 = encoder.fit(cat9_ohe)
cat9_ohe = encoder9.transform(cat9_ohe)
cat9_ohe = pd.DataFrame(cat9_ohe, columns = encoder9.get_feature_names())
cont9 = df9[features_9]
cat9_ohe.reset_index(drop=True, inplace=True)
cont9.reset_index(drop=True, inplace=True)
preprocessed9 = pd.concat([cont9, cat9_ohe], axis=1)
X_train9 = preprocessed9.drop('SalePrice', axis=1)
y_train9 = preprocessed9['SalePrice']
```

In [613]:

```
X_int9 = sm.add_constant(X_train9)
model9 = sm.OLS(y_train9, X_int9).fit()
model9.summary()
```

x2_98040	6.042e+05	2.21e+04	27.350	0.000	5.61e+05	6.48e+05
x2_98042	-7.894e+04	1.25e+04	-6.301	0.000	-1.03e+05	-5.44e+04
x2_98045	1.224e+05	1.03e+05	1.192	0.233	-7.89e+04	3.24e+05
x2_98047	5.432e+04	2.81e+04	1.934	0.053	-744.773	1.09e+05
x2_98051	1.234e+05	2.79e+04	4.432	0.000	6.88e+04	1.78e+05
x2_98052	3.446e+05	2.59e+04	13.279	0.000	2.94e+05	3.95e+05
x2_98053	3.536e+05	2.77e+04	12.778	0.000	2.99e+05	4.08e+05
x2_98055	3182.4957	1.66e+04	0.191	0.848	-2.94e+04	3.58e+04
x2_98056	4.038e+04	1.79e+04	2.257	0.024	5304.138	7.55e+04
x2_98058	-1.954e+04	1.4e+04	-1.391	0.164	-4.71e+04	7992.117
x2_98059	7.174e+04	1.65e+04	4.358	0.000	3.95e+04	1.04e+05
x2_98065	2.736e+05	2.42e+04	11.321	0.000	2.26e+05	3.21e+05
x2_98070	2.195e+05	2.14e+04	10.263	0.000	1.78e+05	2.61e+05

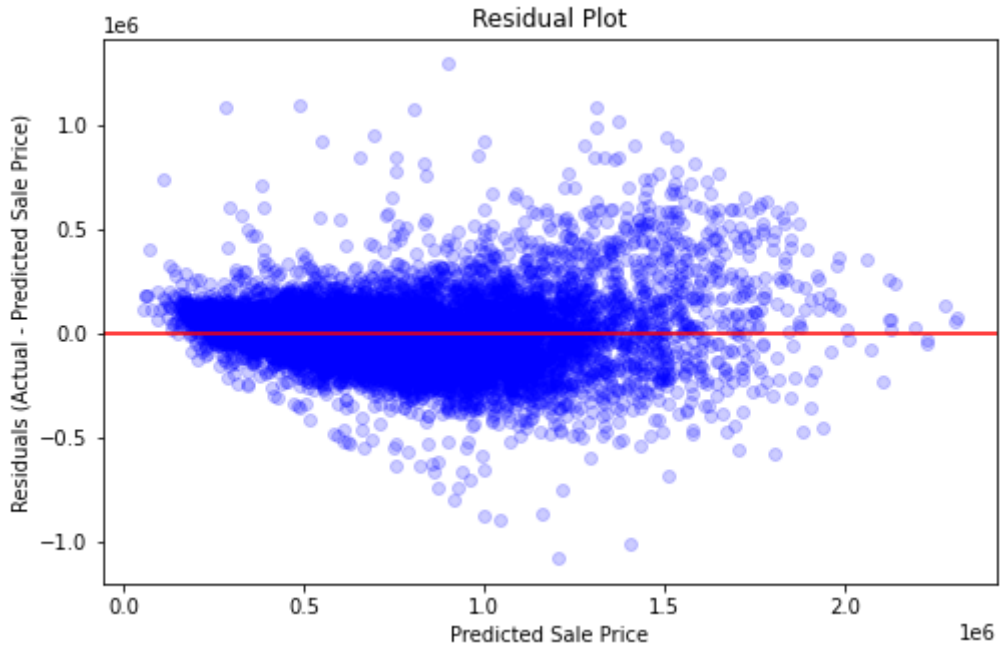
In [614]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg9 = linreg.fit(X_train9, y_train9)

scores9 = cross_val_score(
            linreg9,
            X_train9,
            y_train9,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores9 = np.sqrt(-scores9)

display(rmse_scores9.mean())
display(rmse_scores9.std())
```

```
157905.4822369088

111436.72335630671
```

In [615]:

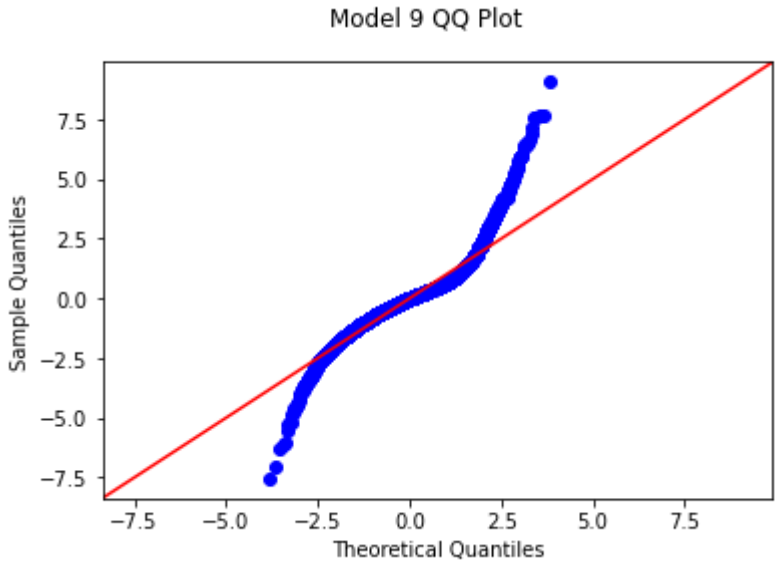
```
get_resid(df9, model9)
```



In [616]:

```
get_qq(model9, 'Model 9');
```

```
..\..\src\useful_functions.py:70: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cann
ot show the figure.
  fig.show()
```



In [617]:

```
summary.loc[8] = ['Multiple Linear Model 7', 'MLP 6 + bathrooms', model9.df_model, round(mo
                  , round(model9.rsquared_adj,3), int(rmse_scores9.mean()), int(rmse_scores
                  round((sms.jarque_bera(model9.resid)[0]),0)]

summary
```

Out[617]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.0	342506.0	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.0	168991.0	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.0	158954.0	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.0	153473.0	16023.0
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.0	146517.0	16416.0
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.0	118312.0	165256.0
6	Multiple Linear Model 5	MLP 4 + SqFt2nd + Clean Cat	123.0	0.823	0.821	164932.0	118703.0	206712.0
7	Multiple Linear Model 6	MLP 5 outliers longitude removed	116.0	0.834	0.833	158260.0	111643.0	32901.0
8	Multiple Linear Model 7	MLP 6 + bathrooms	119.0	0.835	0.834	157905.0	111436.0	32792.0

Still quite a high JB number, it would be preferable to reduce this if possible to try and honour normality assumption as best as possible

Model 10 - Log SalePrice

In [618]:

```
df10 = df9.copy()
```

In [619]:

```
df10['SalePrice_log'] = np.log(df10['SalePrice'])
```

In [620]:

```
#replace saleprice with its log equivalent
features_10 = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'M
               'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
               'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
              'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving', 'SqFt1stFloor
               'dist_to_exp', 'age', 'SalePrice_log', 'SqFt2ndFloor', 'heating', 'bedbath']

#remove bathbins based on its significance in previous model
categoricals10 = ['HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms', 'BldgGrade']
```

In [621]:

```
cat10_ohe = df10[categoricals10]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder10 = encoder.fit(cat10_ohe)
cat10_ohe = encoder10.transform(cat10_ohe)
cat10_ohe = pd.DataFrame(cat10_ohe, columns = encoder10.get_feature_names())
cont10 = df10[features_10]
cat10_ohe.reset_index(drop=True, inplace=True)
cont10.reset_index(drop=True, inplace=True)
preprocessed10 = pd.concat([cont10, cat10_ohe], axis=1)
X_train10 = preprocessed10.drop('SalePrice_log', axis=1)
y_train10 = preprocessed10['SalePrice_log']
```

In [622]:

```
X_int10 = sm.add_constant(X_train10)
model10 = sm.OLS(y_train10, X_int10).fit()
model10.summary()
```

Method:	Least Squares	**F-statistic:**	724.3
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:23:39	**Log-Likelihood:**	4779.2
No. Observations:	14580	**AIC:**	-9322.
Df Residuals:	14462	**BIC:**	-8427.
Df Model:	117		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-11.1531	3.741	-2.981	0.003	-18.486	-3.820
AdjacentGreenbelt	0.0050	0.010	0.516	0.606	-0.014	0.024
excellent_view	0.1410	0.016	8.694	0.000	0.109	0.173
Topography	-0.0113	0.006	-1.921	0.055	-0.023	0.000

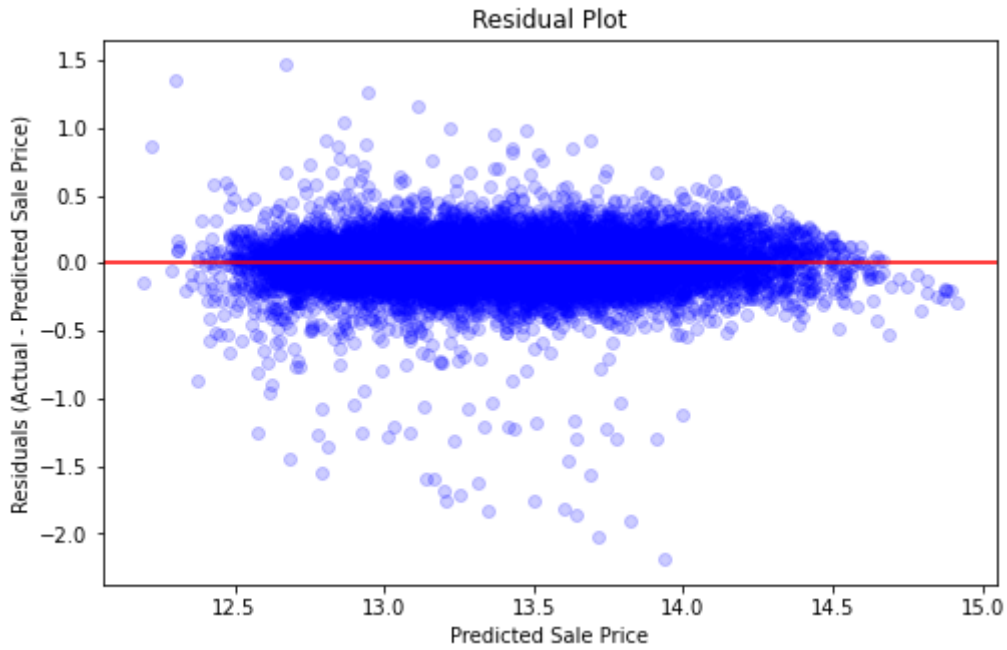
In [623]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg10 = linreg.fit(X_train10, y_train10)

scores10 = cross_val_score(
            linreg10,
            X_train10,
            y_train10,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores10 = np.sqrt(-scores10)

display(rmse_scores10.mean())
display(rmse_scores10.std())
```

```
0.18237320082567346

0.0731371458561881
```

In [624]:

```
get_logresid(preprocessed10, model10, 'SalePrice_log')
```



In [625]:

```
summary.loc[9] = ['Multiple Linear Model 8', 'MLP 7 Log Sale Price', model10.df_model, roun
                  , round(model10.rsquared_adj,3), rmse_scores10.mean(), rmse_scores10.std(
                  round((sms.jarque_bera(model10.resid)[0]),0)]

summary
```

Out[625]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.000000	342506.000000	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.000000	168991.000000	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.000000	158954.000000	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.000000	153473.000000	16023.0
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.000000	146517.000000	16416.0
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.000000	118312.000000	165256.0
6	Multiple Linear Model 5	MLP 4 + SqFt2nd + Clean Cat	123.0	0.823	0.821	164932.000000	118703.000000	206712.0
7	Multiple Linear Model 6	MLP 5 outliers longitude removed	116.0	0.834	0.833	158260.000000	111643.000000	32901.0
8	Multiple Linear Model 7	MLP 6 + bathrooms	119.0	0.835	0.834	157905.000000	111436.000000	32792.0
9	Multiple Linear Model 8	MLP 7 Log Sale Price	117.0	0.854	0.853	0.182373	0.073137	196501.0

In [626]:

```
y_hats = linreg10.predict(X_train10)
```

In [627]:

```
df10['y_hats'] = y_hats
```

In [628]:

```
df10['delta'] = df10['y_hats'] - df10['SalePrice_log']
```

In [629]:

```
df10.sort_values('delta').head(100)
```

Out[629]:

	Address	BuildingNumber	DirectionPrefix	StreetName	StreetType	DirectionSuffix	ZipCode
14558	17432 SE 392ND ST 98092	17432	SE	392ND	ST		98092
11447	25400 173RD AVE SE 98042	25400		173RD	AVE	SE	98042
14785	6526 25TH AVE NW 98117	6526		25TH	AVE	NW	98117
	23860 NE 8TH						

In [630]:

```
large_delta = df10.sort_values('delta')[:200]
```

In [631]:

```
np.exp(14.356089)
```

Out[631]:

```
1716999.7598267233
```

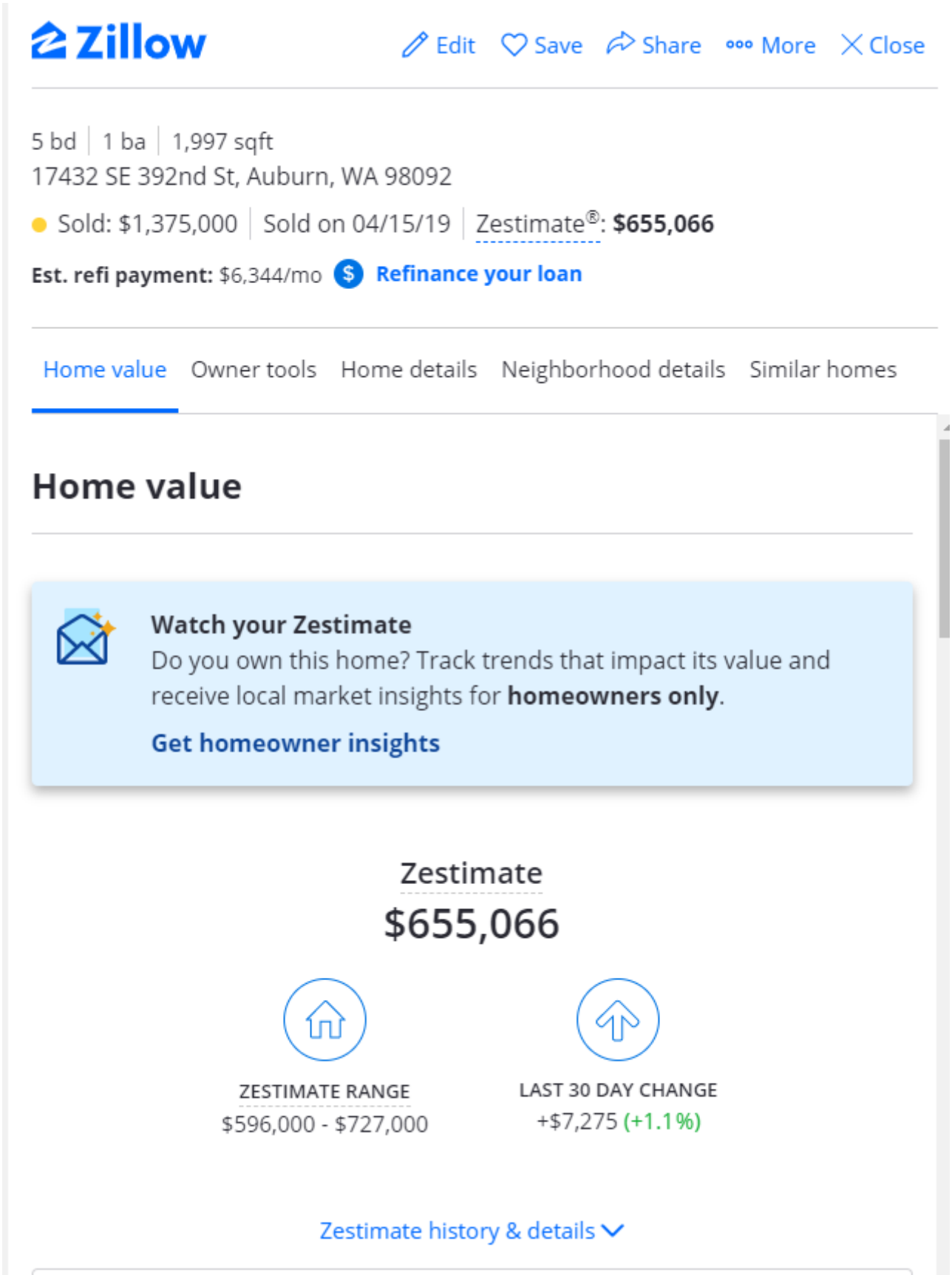
In [632]:

```
np.exp(13.896574)
```

Out[632]:

```
1084439.6934462036
```

out of curiosity I am looking at where I have the biggest discrepancies between the model and reality to see if there is anything I could have missed, a luxury area perhaps. The first property is bamboozling, over 1mil difference, even Zillow says this house is way over priced.



it might be worth plotting these locations on a map to see if there is a pattern

In [633]:

```
get_map(large_delta)
```

Out[633]:

While there is a good spread of locations here, this gives me confidence I haven't missed one location of interest specifically, it is clear however there is a big premium on properties that extremely closed to water, this isn't surprising but something I thought would be captured in the features that determine whether or not you have a view of a location or the wtftlocation feature itself. The model is therefore not capturing this, with more time I would have liked to engineer a feature that estimates proximity to nearest water body. I could do this in a rudimentary fashion with some coordinates but given the size of some of the water bodies in King County, this could still prove difficult. I also don't believe the increase in price will be linear with distance away from water body and suspect the premium can be applied on one street and then the next street back would have a steep drop off i.e a non linear increase as you approach the coast.

In [634]:

```
large_delta.WfntLocation.sum()
```

Out[634]:

```
5
```

The dataframe which contains the top 200 biggest differences between predicted and actual indicates there are only 5 properties on a waterfront location! Looking at the map it looks like this could be an error in the data gathering side which cannot ever be explained by the model, or the criteria for waterfront location is extremely strict i.e you need to be within 5m of the water! Either way, I think a feature that could better capture this would be valuable.

This model looks ok, let's do one last check on multicollinearity

In [635]:

```
get_multicol(preprocessed10)
```

Out[635]:

pairs	cc

No multicollinearity over 0.75, this is good

Model 11 - Square Root SalePrice

In [636]:

```
df11 = df10.copy()
```

In [637]:

```
df11['sale_sqrt'] = np.sqrt(df11['SalePrice'])
```

In [638]:

```
#replace saleprice with its square rroot equivalent
features_11 = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'M
               'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
               'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
              'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving', 'SqFt1stFloor
               'dist_to_exp', 'age', 'sale_sqrt', 'SqFt2ndFloor', 'heating', 'bedbath']


categoricals11 = ['HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms', 'BldgGrade']
```

In [639]:

```
cat11_ohe = df11[categoricals11]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder11 = encoder.fit(cat11_ohe)
cat11_ohe = encoder11.transform(cat11_ohe)
cat11_ohe = pd.DataFrame(cat11_ohe, columns = encoder11.get_feature_names())
cont11 = df11[features_11]
cat11_ohe.reset_index(drop=True, inplace=True)
cont11.reset_index(drop=True, inplace=True)
preprocessed11 = pd.concat([cont11, cat11_ohe], axis=1)
X_train11 = preprocessed11.drop('sale_sqrt', axis=1)
y_train11 = preprocessed11['sale_sqrt']
```

In [640]:

```
X_int11 = sm.add_constant(X_train11)
model11 = sm.OLS(y_train11, X_int11).fit()
model11.summary()
```

Out[640]:

OLS Regression Results

Dep. Variable:	sale_sqrt	**R-squared:**	0.860
Model:	OLS	**Adj. R-squared:**	0.859
Method:	Least Squares	**F-statistic:**	759.4
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:24:04	**Log-Likelihood:**	-83091.
No. Observations:	14580	**AIC:**	1.664e+05
Df Residuals:	14462	**BIC:**	1.673e+05
Df Model:	117		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**

In [641]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg11 = linreg.fit(X_train11, y_train11)

scores11 = cross_val_score(
            linreg11,
            X_train11,
            y_train11,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores11 = np.sqrt(-scores11)

display(rmse_scores11.mean())
display(rmse_scores11.std())
```

```
78.99820839212325

38.394641698327476
```

In [642]:

```
summary.loc[10] = ['Multiple Linear Model 9', 'MLP 7 Sqrt Sale Price', model11.df_model, ro
                   , round(model11.rsquared_adj,3), rmse_scores11.mean(), rmse_scores11.std(
                   round((sms.jarque_bera(model11.resid)[0]),0)]

summary
```

Out[642]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	J
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.000000	342506.000000	9825842.
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.000000	168991.000000	7132.
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.000000	158954.000000	18348.
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.000000	153473.000000	16023.
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.000000	146517.000000	16416.
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.000000	118312.000000	165256.
6	Multiple Linear Model 5	MLP 4 + SqFt2nd + Clean Cat	123.0	0.823	0.821	164932.000000	118703.000000	206712.
7	Multiple Linear Model 6	MLP 5 outliers longitude removed	116.0	0.834	0.833	158260.000000	111643.000000	32901.
8	Multiple Linear Model 7	MLP 6 + bathrooms	119.0	0.835	0.834	157905.000000	111436.000000	32792.
9	Multiple Linear Model 8	MLP 7 Log Sale Price	117.0	0.854	0.853	0.182373	0.073137	196501.
10	Multiple Linear Model 9	MLP 7 Sqrt Sale Price	117.0	0.860	0.859	78.998208	38.394642	35088.

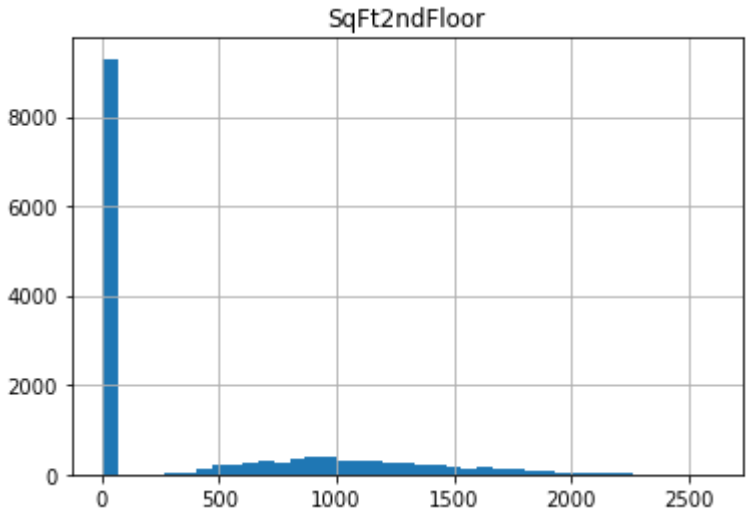
Model 12 - Cube Root other continuous variables

In [643]:

```
df12 = df11.copy()
```

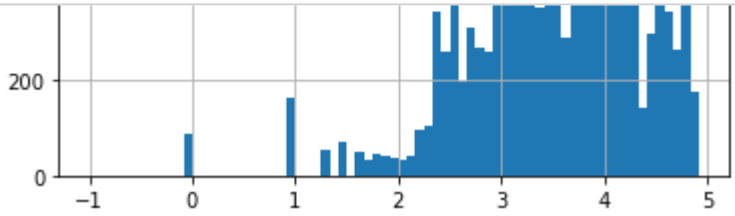
In [644]:

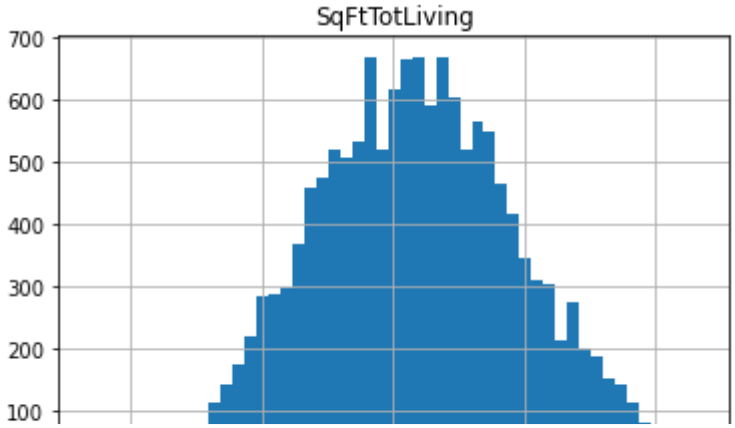
```
cont_12 = ['SqFt2ndFloor', 'age', 'SqFtTotLiving', 'SqFt1stFloor', 'latitude', 'dist_to_exp
for col in cont_12:
    df12[col].hist(bins='auto')
    plt.title(col)
    plt.show()
```



In [645]:

```
for col in cont_12:
    np.cbrt(df12[col]).hist(bins='auto')
    plt.title(col)
    plt.show()
```





In [646]:

```
df12['duplex'] = 0
df12.loc[(df12['SqFt2ndFloor'] > 0),'duplex'] = 1
```

In [647]:

```
# cube root transform those that make sense
df12['SqFtTotLiving_cb'] = np.cbrt(df12['SqFtTotLiving'])
df12['SqFt1stFloor_cb'] = np.cbrt(df12['SqFt1stFloor'])
df12['dist_to_exp_cb'] = np.cbrt(df12['dist_to_exp'])
df12['SalePrice_cb'] = np.cbrt(df12['SalePrice'])
```

In [648]:

```
#remove adjacent greenbelt as it has been consistently insignificant
features_12 = ['excellent_view', 'Topography', 'InadequateParking', 'MtRainier', 'Olympics'
               'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington', 'LakeSammam
               'OtherView', 'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
              'has_basement', 'TrafficNoise', 'has_problem', 'SqFtTotLiving_cb', 'SqFt1stFl
               'dist_to_exp_cb', 'age', 'SalePrice_cb', 'duplex', 'heating', 'bedbath']


categoricals12 = ['HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms', 'BldgGrade']
```

In [649]:

```
cat12_ohe = df12[categoricals12]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder12 = encoder.fit(cat12_ohe)
cat12_ohe = encoder12.transform(cat12_ohe)
cat12_ohe = pd.DataFrame(cat12_ohe, columns = encoder12.get_feature_names())
cont12 = df12[features_12]
cat12_ohe.reset_index(drop=True, inplace=True)
cont12.reset_index(drop=True, inplace=True)
preprocessed12 = pd.concat([cont12, cat12_ohe], axis=1)
X_train12 = preprocessed12.drop('SalePrice_cb', axis=1)
y_train12 = preprocessed12['SalePrice_cb']
```

In [650]:

```
X_int12 = sm.add_constant(X_train12)
model12 = sm.OLS(y_train12, X_int12).fit()
model12.summary()
```

Out[650]:

OLS Regression Results

Dep. Variable:	SalePrice_cb	**R-squared:**	0.860
Model:	OLS	**Adj. R-squared:**	0.859
Method:	Least Squares	**F-statistic:**	766.8
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:24:23	**Log-Likelihood:**	-44288.
No. Observations:	14580	**AIC:**	8.881e+04
Df Residuals:	14463	**BIC:**	8.970e+04
Df Model:	116		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**

In [651]:

```
# I will now check run the model using sk-learn
linreg = LinearRegression()

# Fit on training data
linreg12 = linreg.fit(X_train12, y_train12)

scores12 = cross_val_score(
            linreg12,
            X_train12,
            y_train12,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores12 = np.sqrt(-scores12)

display(rmse_scores12.mean())
display(rmse_scores12.std())
```

```
5.46215914199116

2.371203248306395
```

In [652]:

```
summary.loc[11] = ['Multiple Linear Model 10', 'MLP 7 cbrt Sale Price - adjacentGreen', mod
                   , round(model12.rsquared_adj,3), rmse_scores12.mean(), rmse_scores12.std(
                   round((sms.jarque_bera(model12.resid)[0]),0)]

summary
```

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	JB
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.000000	342506.000000	9825842.0
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.000000	168991.000000	7132.0
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.000000	158954.000000	18348.0
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.000000	153473.000000	16023.0
4	Multiple Linear	continuous +	29.0	0.622	0.622	227275.000000	146517.000000	16416.0

one last check on multicollinearity...

Model 13 - MLM 7b Last attempt to increase Normality Assumption

In [653]:

```
# Create duplex column to remove SqFt2ndFloor
df9['duplex'] = 0
df9.loc[(df9['SqFt2ndFloor'] > 0),'duplex'] = 1
```

In [654]:

```
features_9
```

Out[654]:

```
['AdjacentGreenbelt',
 'excellent_view',
 'Topography',
 'InadequateParking',
 'MtRainier',
 'Olympics',
 'Cascades',
 'Territorial',
 'SeattleSkyline',
 'PugetSound',
 'LakeWashington',
 'LakeSammamish',
 'SmallLakeRiverCreek',
 'OtherView',
 'WfntLocation',
 'Easements',
 'SaleWarning',
 'ViewUtilization',
 'has_porch',
 'has_renovation',
 'has_basement',
 'TrafficNoise',
 'has_problem',
 'SqFtTotLiving',
 'SqFt1stFloor',
 'latitude',
 'dist_to_exp',
 'age',
 'SalePrice',
 'SqFt2ndFloor',
 'heating',
 'bedbath']
```

In [655]:

```
rows1 = df9[['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'MtR
                      'Cascades', 'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashin
                      'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'Sale
                       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_p
                       'SqFt1stFloor', 'latitude', 'dist_to_exp', 'age', 'SalePrice', 'SqFt2

vif_df1 = pd.DataFrame()
vif_df1["VIF"] = [variance_inflation_factor(rows1, i) for i in range(32)]
vif_df1["feature"] = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParki
                      'Cascades', 'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashin
                      'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'Sale
                       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_p
                       'SqFt1stFloor', 'latitude', 'dist_to_exp', 'age', 'SalePrice', 'SqFt2

vif_df1
```

Out[655]:

	VIF	feature
0	1.064298	AdjacentGreenbelt
1	1.432768	excellent_view
2	1.185921	Topography
3	2.568941	InadequateParking
4	1.120000	MtRainier
5	1.794581	Olympics
6	1.522574	Cascades
7	2.515150	Territorial
8	1.171936	SeattleSkyline
9	1.871353	PugetSound

In [656]:

```
# remove latitude,
rows2 = df9[['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'MtR
                      'Cascades', 'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashin
                      'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'Sale
                       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_p
                       'SqFt1stFloor', 'dist_to_exp', 'age', 'SalePrice', 'heating', 'bedbat
                         'duplex']].values

vif_df2 = pd.DataFrame()
vif_df2["VIF"] = [variance_inflation_factor(rows2, i) for i in range(31)]
vif_df2["feature"] = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParki
                      'Cascades', 'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashin
                      'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'Sale
                       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_p
                       'SqFt1stFloor', 'dist_to_exp', 'age', 'SalePrice', 'heating', 'bedbat
                         'duplex']

vif_df2
```

Out[656]:

	VIF	feature
0	1.063141	AdjacentGreenbelt
1	1.431977	excellent_view
2	1.183737	Topography
3	2.478906	InadequateParking
4	1.119793	MtRainier
5	1.794183	Olympics
6	1.519588	Cascades
7	2.514902	Territorial
8	1.171464	SeattleSkyline
9	1.858630	PugetSound

In [657]:

```
rows3 = df9[['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParking', 'MtR
                      'Cascades', 'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashin
                      'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'Sale
                       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_p
                       'SqFt1stFloor', 'dist_to_exp', 'age', 'heating', 'bedbath',\
                         'duplex']].values

vif_df3 = pd.DataFrame()
vif_df3["VIF"] = [variance_inflation_factor(rows3, i) for i in range(29)]
vif_df3["feature"] = ['AdjacentGreenbelt', 'excellent_view', 'Topography', 'InadequateParki
                      'Cascades', 'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashin
                      'SmallLakeRiverCreek', 'OtherView', 'WfntLocation', 'Easements', 'Sale
                       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise', 'has_p
                       'SqFt1stFloor', 'dist_to_exp', 'age', 'heating', 'bedbath',\
                         'duplex']

vif_df3
```

Out[657]:

	VIF	feature
0	1.059418	AdjacentGreenbelt
1	1.426628	excellent_view
2	1.179818	Topography
3	2.461651	InadequateParking
4	1.119747	MtRainier
5	1.790585	Olympics
6	1.518483	Cascades
7	2.510640	Territorial
8	1.165198	SeattleSkyline
9	1.855271	PugetSound

In [658]:

```
vif_df3.feature.values
```

Out[658]:

```
array(['AdjacentGreenbelt', 'excellent_view', 'Topography',
       'InadequateParking', 'MtRainier', 'Olympics', 'Cascades',
       'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington',
       'LakeSammamish', 'SmallLakeRiverCreek', 'OtherView',
       'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise',
       'has_problem', 'SqFt1stFloor', 'dist_to_exp', 'age', 'heating',
       'bedbath', 'duplex'], dtype=object)
```

In [659]:

```
df13 = df9.copy()
```

In [660]:

```
features_13 = ['AdjacentGreenbelt', 'excellent_view', 'Topography',
       'InadequateParking', 'MtRainier', 'Olympics', 'Cascades',
       'Territorial', 'SeattleSkyline', 'PugetSound', 'LakeWashington',
       'LakeSammamish', 'SmallLakeRiverCreek', 'OtherView',
       'WfntLocation', 'Easements', 'SaleWarning', 'ViewUtilization',
       'has_porch', 'has_renovation', 'has_basement', 'TrafficNoise',
       'has_problem', 'SqFt1stFloor','SalePrice', 'dist_to_exp', 'age', 'heating',
       'bedbath', 'duplex']
categoricals13 = ['HeatSource', 'Access', 'ZipCode', 'Condition', 'Bedrooms', 'BldgGrade']
```

In [661]:

```
cat13_ohe = df13[categoricals13]
encoder = OneHotEncoder(drop='first', sparse=False)
encoder13 = encoder.fit(cat13_ohe)
cat13_ohe = encoder13.transform(cat13_ohe)
cat13_ohe = pd.DataFrame(cat13_ohe, columns = encoder13.get_feature_names())
cont13 = df13[features_13]
cat13_ohe.reset_index(drop=True, inplace=True)
cont13.reset_index(drop=True, inplace=True)
preprocessed13 = pd.concat([cont13, cat13_ohe], axis=1)
X_train13 = preprocessed13.drop('SalePrice', axis=1)
y_train13 = preprocessed13['SalePrice']
```

In [662]:

```
X_int13 = sm.add_constant(X_train13)
model13 = sm.OLS(y_train13, X_int13).fit()
model13.summary()
```

Out[662]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.819
Model:	OLS	**Adj. R-squared:**	0.817
Method:	Least Squares	**F-statistic:**	568.0
Date:	Sat, 06 Mar 2021	**Prob (F-statistic):**	0.00
Time:	03:24:41	**Log-Likelihood:**	-1.9447e+05
No. Observations:	14580	**AIC:**	3.892e+05
Df Residuals:	14464	**BIC:**	3.901e+05
Df Model:	115		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.738e+05	2.13e+04	8.158	0.000	1.32e+05	2.16e+05
AdjacentGreenbelt	8757.5503	8414.718	1.041	0.298	-7736.374	2.53e+04
excellent_view	1.651e+05	1.4e+04	11.827	0.000	1.38e+05	1.92e+05
Topography	-1.049e+04	5037.489	-2.083	0.037	-2.04e+04	-616.938
InadequateParking	-7952.4105	3244.484	-2.451	0.014	-1.43e+04	-1592.807
MtRainier	4.223e+04	1.55e+04	2.724	0.006	1.18e+04	7.26e+04
Olympics	6.275e+04	1.15e+04	5.434	0.000	4.01e+04	8.54e+04
Cascades	784.5289	9432.227	0.083	0.934	-1.77e+04	1.93e+04
Territorial	7011.4726	6624.090	1.058	0.290	-5972.591	2e+04
SeattleSkyline	8.344e+04	1.51e+04	5.531	0.000	5.39e+04	1.13e+05
PugetSound	6.608e+04	1.04e+04	6.331	0.000	4.56e+04	8.65e+04
LakeWashington	1.924e+05	1.02e+04	18.883	0.000	1.72e+05	2.12e+05
LakeSammamish	1.794e+05	2.08e+04	8.617	0.000	1.39e+05	2.2e+05
SmallLakeRiverCreek	-5.816e+04	1.94e+04	-3.002	0.003	-9.61e+04	-2.02e+04
OtherView	-7.493e+04	2.28e+04	-3.281	0.001	-1.2e+05	-3.02e+04
WfntLocation	3.101e+05	1.78e+04	17.432	0.000	2.75e+05	3.45e+05
Easements	2.126e+04	9547.595	2.226	0.026	2542.549	4e+04
SaleWarning	-1.003e+05	5464.057	-18.356	0.000	-1.11e+05	-8.96e+04
ViewUtilization	6.722e+04	1.01e+04	6.685	0.000	4.75e+04	8.69e+04
has_porch	2.845e+04	2702.822	10.527	0.000	2.32e+04	3.37e+04
has_renovation	5.459e+04	6187.913	8.822	0.000	4.25e+04	6.67e+04
has_basement	2.946e+04	3365.849	8.751	0.000	2.29e+04	3.61e+04
TrafficNoise	-2.711e+04	3712.427	-7.303	0.000	-3.44e+04	-1.98e+04

has_problem	-2.107e+04	6711.891	-3.140	0.002	-3.42e+04	-7916.377
SqFt1stFloor	140.2987	4.864	28.847	0.000	130.765	149.832
dist_to_exp	-9294.7787	813.729	-11.422	0.000	-1.09e+04	-7699.765
age	830.6921	78.309	10.608	0.000	677.197	984.187
heating	-8617.8699	3864.972	-2.230	0.026	-1.62e+04	-1042.031
bedbath	-6.606e+04	3048.684	-21.670	0.000	-7.2e+04	-6.01e+04
duplex	6.882e+04	4448.021	15.471	0.000	6.01e+04	7.75e+04
x0_Gas	1.483e+04	4127.416	3.594	0.000	6743.640	2.29e+04
x0_Oil	2.199e+04	5471.637	4.018	0.000	1.13e+04	3.27e+04
x1_PUBLIC	-1.026e+04	5694.895	-1.801	0.072	-2.14e+04	907.573
x2_98002	-3.338e+04	1.36e+04	-2.457	0.014	-6e+04	-6754.179
x2_98003	1.043e+04	1.25e+04	0.834	0.404	-1.41e+04	3.49e+04
x2_98004	8.996e+05	1.83e+04	49.271	0.000	8.64e+05	9.35e+05
x2_98005	5.361e+05	1.98e+04	27.142	0.000	4.97e+05	5.75e+05
x2_98006	3.204e+05	1.41e+04	22.694	0.000	2.93e+05	3.48e+05
x2_98007	3.462e+05	1.93e+04	17.980	0.000	3.08e+05	3.84e+05
x2_98008	3.491e+05	1.43e+04	24.338	0.000	3.21e+05	3.77e+05
x2_98010	-6576.4923	2.5e+04	-0.263	0.792	-5.56e+04	4.24e+04
x2_98011	2.024e+05	1.5e+04	13.523	0.000	1.73e+05	2.32e+05
x2_98014	3.03e+05	2.28e+04	13.312	0.000	2.58e+05	3.48e+05
x2_98019	2.657e+05	1.96e+04	13.531	0.000	2.27e+05	3.04e+05
x2_98022	8.385e+04	1.56e+04	5.372	0.000	5.33e+04	1.14e+05
x2_98023	3257.9524	1.18e+04	0.276	0.782	-1.99e+04	2.64e+04
x2_98024	3.716e+05	2.65e+04	14.026	0.000	3.2e+05	4.23e+05
x2_98027	2.477e+05	1.4e+04	17.637	0.000	2.2e+05	2.75e+05
x2_98028	1.794e+05	1.38e+04	13.009	0.000	1.52e+05	2.06e+05
x2_98029	3.119e+05	1.51e+04	20.616	0.000	2.82e+05	3.42e+05
x2_98030	-6.706e+04	1.54e+04	-4.343	0.000	-9.73e+04	-3.68e+04
x2_98031	-5.142e+04	1.42e+04	-3.632	0.000	-7.92e+04	-2.37e+04
x2_98032	-2.988e+04	1.64e+04	-1.822	0.068	-6.2e+04	2261.361
x2_98033	5.275e+05	1.33e+04	39.765	0.000	5.01e+05	5.53e+05
x2_98034	2.607e+05	1.17e+04	22.265	0.000	2.38e+05	2.84e+05
x2_98038	1.558e+04	1.2e+04	1.302	0.193	-7873.459	3.9e+04
x2_98040	6.272e+05	1.69e+04	37.101	0.000	5.94e+05	6.6e+05
x2_98042	-6.468e+04	1.29e+04	-5.015	0.000	-9e+04	-3.94e+04
x2_98045	2.181e+05	1.07e+05	2.032	0.042	7705.962	4.29e+05
x2_98047	4.22e+04	2.93e+04	1.440	0.150	-1.52e+04	9.96e+04
x2_98051	1.349e+05	2.91e+04	4.645	0.000	7.8e+04	1.92e+05
x2_98052	3.815e+05	1.2e+04	31.913	0.000	3.58e+05	4.05e+05
x2_98053	4.124e+05	1.46e+04	28.236	0.000	3.84e+05	4.41e+05

x2_98055	1.939e+04	1.48e+04	1.314	0.189	-9525.882	4.83e+04
x2_98056	6.578e+04	1.39e+04	4.736	0.000	3.86e+04	9.3e+04
x2_98058	-7762.9950	1.2e+04	-0.647	0.518	-3.13e+04	1.57e+04
x2_98059	9.941e+04	1.29e+04	7.734	0.000	7.42e+04	1.25e+05
x2_98065	3.29e+05	1.98e+04	16.631	0.000	2.9e+05	3.68e+05
x2_98070	2.465e+05	2.06e+04	11.941	0.000	2.06e+05	2.87e+05
x2_98072	2.767e+05	1.45e+04	19.112	0.000	2.48e+05	3.05e+05
x2_98074	3.064e+05	1.31e+04	23.451	0.000	2.81e+05	3.32e+05
x2_98075	3.167e+05	1.44e+04	22.051	0.000	2.89e+05	3.45e+05
x2_98077	2.834e+05	1.66e+04	17.071	0.000	2.51e+05	3.16e+05
x2_98092	-3.57e+04	1.27e+04	-2.803	0.005	-6.07e+04	-1.07e+04
x2_98102	5.269e+05	2.37e+04	22.236	0.000	4.8e+05	5.73e+05
x2_98103	4.227e+05	1.35e+04	31.207	0.000	3.96e+05	4.49e+05
x2_98105	4.949e+05	1.69e+04	29.281	0.000	4.62e+05	5.28e+05
x2_98106	1.918e+05	1.3e+04	14.798	0.000	1.66e+05	2.17e+05
x2_98107	4.099e+05	1.58e+04	25.948	0.000	3.79e+05	4.41e+05
x2_98108	1.717e+05	1.58e+04	10.867	0.000	1.41e+05	2.03e+05
x2_98109	6.239e+05	2.09e+04	29.909	0.000	5.83e+05	6.65e+05
x2_98112	6.249e+05	1.65e+04	37.780	0.000	5.92e+05	6.57e+05
x2_98115	3.907e+05	1.36e+04	28.647	0.000	3.64e+05	4.17e+05
x2_98116	4.027e+05	1.35e+04	29.798	0.000	3.76e+05	4.29e+05
x2_98117	4.094e+05	1.24e+04	33.042	0.000	3.85e+05	4.34e+05
x2_98118	1.771e+05	1.35e+04	13.105	0.000	1.51e+05	2.04e+05
x2_98119	5.678e+05	1.75e+04	32.371	0.000	5.33e+05	6.02e+05
x2_98122	3.75e+05	1.57e+04	23.909	0.000	3.44e+05	4.06e+05
x2_98125	2.53e+05	1.39e+04	18.264	0.000	2.26e+05	2.8e+05
x2_98126	2.828e+05	1.31e+04	21.657	0.000	2.57e+05	3.08e+05
x2_98133	2.22e+05	1.23e+04	18.109	0.000	1.98e+05	2.46e+05
x2_98136	3.337e+05	1.47e+04	22.688	0.000	3.05e+05	3.63e+05
x2_98144	3.302e+05	1.52e+04	21.665	0.000	3e+05	3.6e+05
x2_98146	1.511e+05	1.34e+04	11.273	0.000	1.25e+05	1.77e+05
x2_98148	1.276e+05	2.2e+04	5.789	0.000	8.44e+04	1.71e+05
x2_98155	2.242e+05	1.23e+04	18.294	0.000	2e+05	2.48e+05
x2_98166	1.698e+05	1.47e+04	11.542	0.000	1.41e+05	1.99e+05
x2_98168	7.597e+04	1.32e+04	5.773	0.000	5.02e+04	1.02e+05
x2_98177	3.005e+05	1.38e+04	21.810	0.000	2.74e+05	3.28e+05
x2_98178	1.749e+04	1.41e+04	1.238	0.216	-1.02e+04	4.52e+04
x2_98188	6.909e+04	1.66e+04	4.168	0.000	3.66e+04	1.02e+05
x2_98198	3.043e+04	1.34e+04	2.269	0.023	4137.578	5.67e+04
x2_98199	5.17e+05	1.41e+04	36.723	0.000	4.89e+05	5.45e+05

x3_4	3.173e+04	3096.929	10.245	0.000	2.57e+04	3.78e+04
x3_5	6.781e+04	4298.400	15.775	0.000	5.94e+04	7.62e+04
x4_3	1.052e+05	4997.120	21.050	0.000	9.54e+04	1.15e+05
x4_4	1.911e+05	6838.045	27.939	0.000	1.78e+05	2.04e+05
x4_5	2.802e+05	9696.367	28.895	0.000	2.61e+05	2.99e+05
x4_6	2.97e+05	1.65e+04	17.953	0.000	2.65e+05	3.29e+05
x5_6	3316.0503	1.43e+04	0.231	0.817	-2.48e+04	3.14e+04
x5_7	1.448e+04	1.44e+04	1.005	0.315	-1.38e+04	4.27e+04
x5_8	7.334e+04	1.49e+04	4.909	0.000	4.41e+04	1.03e+05
x5_9	2.416e+05	1.57e+04	15.348	0.000	2.11e+05	2.72e+05
x5_10	3.985e+05	1.71e+04	23.308	0.000	3.65e+05	4.32e+05
x5_11	5.758e+05	2.17e+04	26.552	0.000	5.33e+05	6.18e+05
x5_12	5.344e+05	5.8e+04	9.215	0.000	4.21e+05	6.48e+05

Omnibus:	3405.102	**Durbin-Watson:**	1.435
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	28053.321
Skew:	0.893	**Prob(JB):**	0.00
Kurtosis:	9.557	**Cond. No.**	1.13e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.13e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [663]:

```
linreg = LinearRegression()

# Fit on training data
linreg13 = linreg.fit(X_train13, y_train13)

scores13 = cross_val_score(
            linreg13,
            X_train13,
            y_train13,
            cv=10,
            scoring="neg_mean_squared_error"
)

rmse_scores13 = np.sqrt(-scores13)
display(rmse_scores13.mean())
display(rmse_scores13.std())
```

```
163682.63428539137
```

```
115393.45995078197
```

In [664]:

```
summary.loc[12] = ['Multiple Linear Model 7b', 'MLP 7 reduced features due to VIF', model13
                   , round(model13.rsquared_adj,3), rmse_scores13.mean(), rmse_scores13.std(
                   round((sms.jarque_bera(model13.resid)[0]),0)]

summary
```

Out[664]:

	Model	Description	No. Features	R^2	Adj R^2	RMSE	RMSE sd	J
0	Simple Model - one independent variable	Square ft Total living	1.0	0.389	0.389	398611.000000	342506.000000	9825842.
1	Simple Model - outliers removed	Square ft Total living	1.0	0.354	0.354	294579.000000	168991.000000	7132.
2	Multiple Linear Model 1	Continuous features	4.0	0.535	0.535	238663.000000	158954.000000	18348.
3	Multiple Linear Model 2	Continuous features + age/dist_to_exp	6.0	0.567	0.566	236962.000000	153473.000000	16023.
4	Multiple Linear Model 3	continuous + binary features	29.0	0.622	0.622	227275.000000	146517.000000	16416.
5	Multiple Linear Model 4	continuous + binary features + cat	139.0	0.816	0.814	166951.000000	118312.000000	165256.
6	Multiple Linear Model 5	MLP 4 + SqFt2nd + Clean Cat	123.0	0.823	0.821	164932.000000	118703.000000	206712.
7	Multiple Linear Model 6	MLP 5 outliers longitude removed	116.0	0.834	0.833	158260.000000	111643.000000	32901.
8	Multiple Linear Model 7	MLP 6 + bathrooms	119.0	0.835	0.834	157905.000000	111436.000000	32792.
9	Multiple Linear Model 8	MLP 7 Log Sale Price	117.0	0.854	0.853	0.182373	0.073137	196501.
10	Multiple Linear Model 9	MLP 7 Sqrt Sale Price	117.0	0.860	0.859	78.998208	38.394642	35088.
11	Multiple Linear Model 10	MLP 7 cbrt Sale Price - adjacentGreen	116.0	0.860	0.859	5.462159	2.371203	53457.
12	Multiple Linear Model 7b	MLP 7 reduced features due to VIF	115.0	0.819	0.817	163682.634285	115393.459951	28053.

Summary

General Findings

The final model that will be used to advise on the potential home improvements will be MLM7.

With an R-squared of 83.5 it explains 83.5% of the variance in the Sale price. This seems reasonable for relatively simple model.

Other models achieved higher R-squared numbers but the JB number suffered, meaning the coefficients were likely to be less reliable.

With the transformed variables, interpretation of variables can become problematic beyong the standard log transformation and for such marginal improvement in the R-squared, sacrificing a couple % on R-squared for both ease of interpretation and better attempt at honouring normality seems to make sense.

I have worked to reduce the JB number and try to ensure the model honours the normality assumption but clearly this could be improved upon. As such the recommendations for home improvement will have to come with a bold caveat next to them - this model is far from perfect.

Interpretation of Coeffients

Square footage

For every one square foot of total living space you add to a property, its price will increase by 119 USD assuming all other variables are kept constant. i.e if you add 500 square feet of living space this could add up to 60,000USD to the price.

Bathroom Adding

If you increase the bedbath variable by one, this will decrease the house price by 29420 USD. The bed bath was simply the number of bedrooms minus the number of bathrooms. Therefore the intepretation of this is that reducing the difference between the two numbers by one will increase the property price by ~30,000USD.

Renovation

Renovating a house can me expensive but it can be worth it. The coefficient of the renovation feature is 58,220USD. Meaning if you renovate the house on average it will increase the house price by 58,220USD.

Porch

The interpretation of the coefficient for having a porch suggests having a porch will increase the price of your property by 17,560USD.

General condition

This is a slightly ambiguous feature, however it is assumed this means the general condition of the property. The difference here is stark though, assuming all other variables are kept the same, a 'good' condition home will sell for 33,950USD more than an 'average' condition property. for 'very good' this is even more - 70,980USD.

Has Water Problems or other

Assuming all other variables are kept the same, a home with a water issue or some other unspecified will be worth 18,960USD less than a home that is issue free.

Appendix - Images etc.

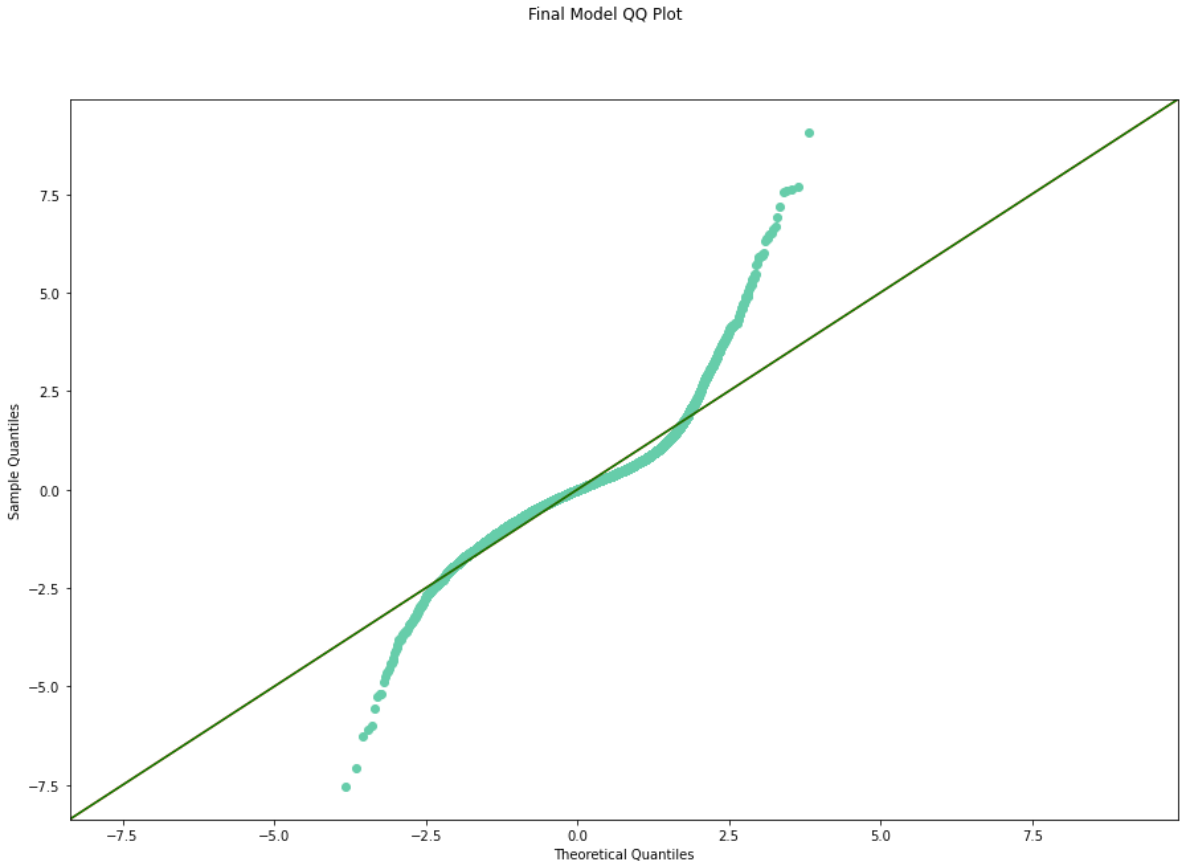
In [665]:

```
# X_int9 = sm.add_constant(X_train9)
# model9 = sm.OLS(y_train9, X_int9).fit()
# model9.summary()
# y_pred9 = linreg9.predict(X_train9)
# # # Create a matplotlib figure
# plt.figure(figsize=(15, 10))
# # Create a scatter plot
# plt.scatter(y_train9, y_pred9, label='Actual', color='mediumaquamarine')
# plt.plot(y_pred9, y_pred9, label='Predicted', color='green')
# plt.xlabel('Actual House Price $MM dollars', fontsize=15)
# plt.ylabel('Predicted House Price ($MM dollars)', fontsize=15)
# plt.title('Final Model Predictions vs. actual', fontsize=15)
# plt.legend()

# plt.show()


plt.figure(figsize=(15,10))
plt.axhline(y = 0, color = 'green', linestyle = '-')
plt.ylabel("Residuals (Actual - Predicted Sale Price $ Mil)", fontsize=13)
plt.xlabel("Predicted Sale Price", fontsize=13)
plt.scatter(x=y_pred9, y=y_train9-y_pred9, alpha=0.4, color='mediumaquamarine');
plt.title('Residual Plot - Final Model', fontsize=15)
plt.savefig('modelvsactual.png', bbox_inches = 'tight')
plt.show()
```

In [666]:

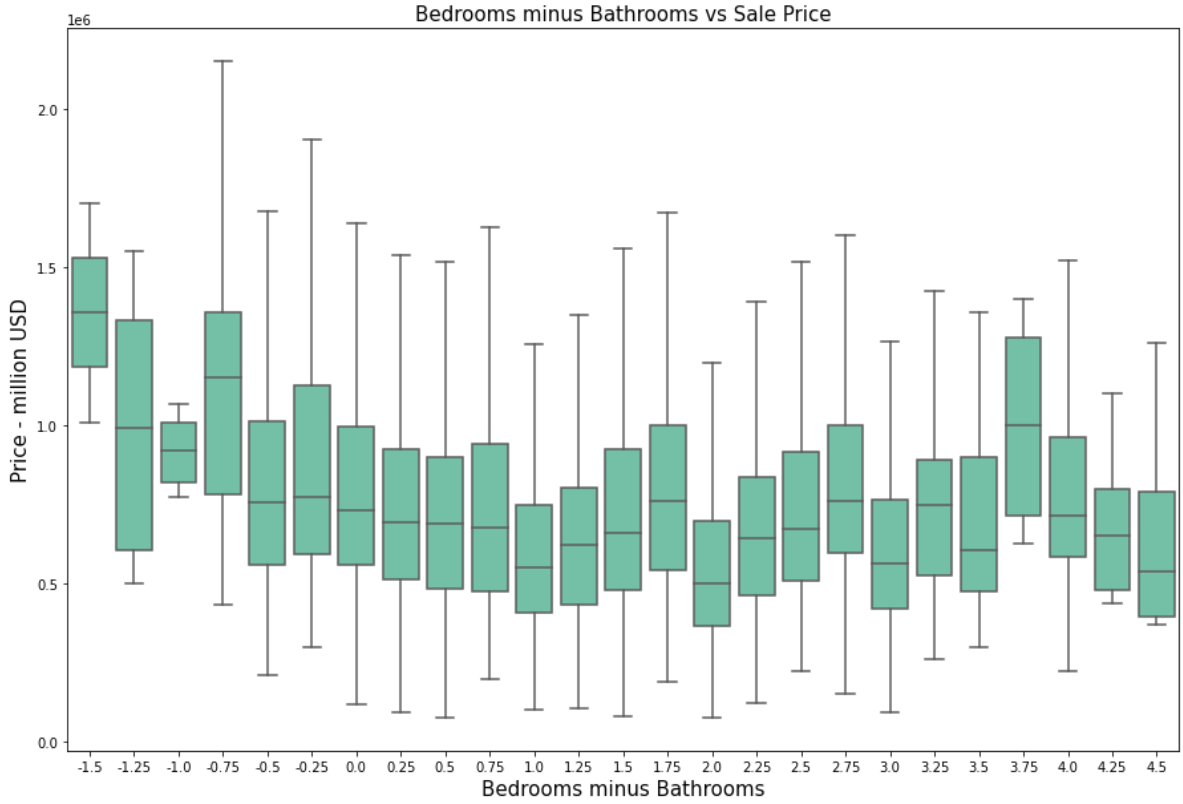
```
fig, ax = plt.subplots(figsize=(15, 10))
residuals9 = model9.resid
fig = sm.graphics.qqplot(residuals9, line='45',markerfacecolor='mediumaquamarine', markered
                         fit=True, ax=ax)
fig.suptitle('Final Model QQ Plot', fontsize=12)
sm.qqline(fig.axes[0], line='45', fmt='green')
fig.show()
fig.savefig('qq.png', bbox_inches = 'tight')
```

```
<ipython-input-666-30adecf57294>:7: UserWarning: Matplotlib is currently usi
ng module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so c
annot show the figure.
  fig.show()
```



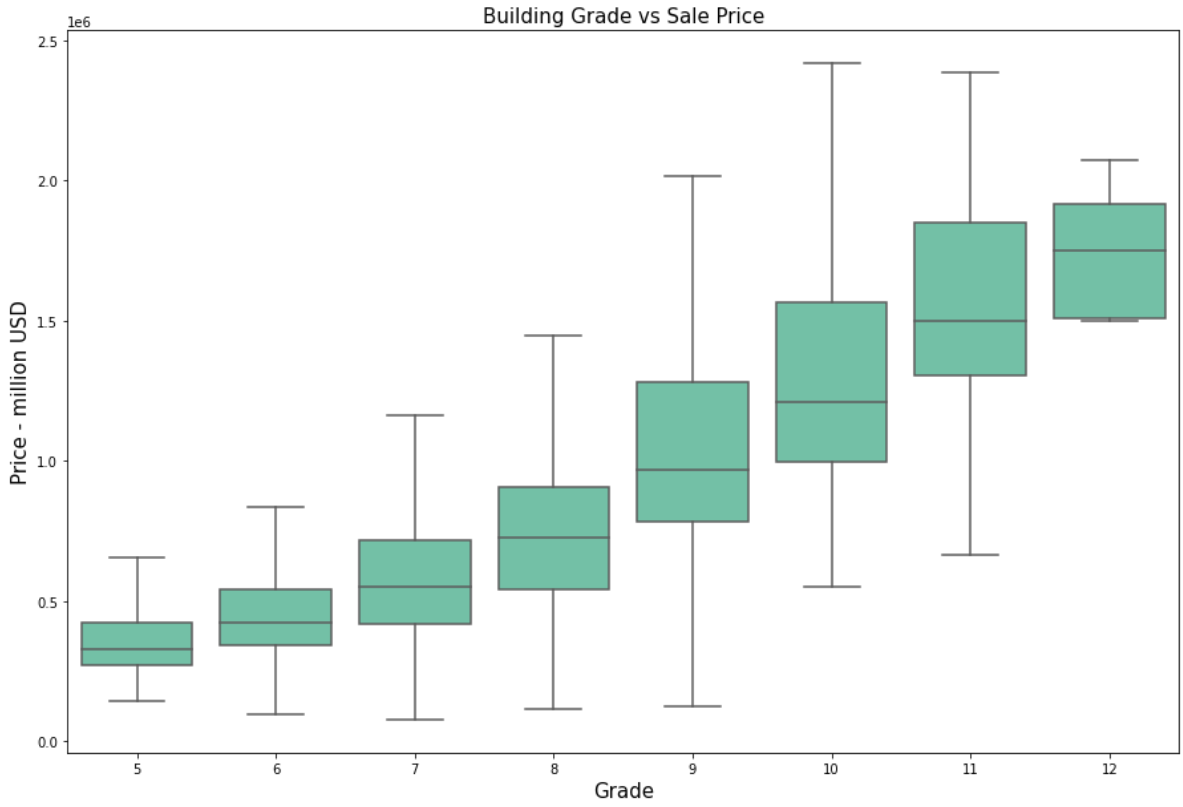
In [667]:

```
plt.figure(figsize=(15,10))
plt.title('Bedrooms minus Bathrooms vs Sale Price', fontsize=15)
plt.ylabel('Price - million USD', fontsize=15)
plt.xlabel('Bedrooms minus Bathrooms', fontsize=15)
boxplot = sns.boxplot(x='bedbath', y="SalePrice", data=df9, showfliers=False, color='medium
boxplot.set(xlabel='Bedrooms minus Bathrooms',ylabel='Price - million USD');
plt.savefig('bedroomsvsbathrooms.png', bbox_inches = 'tight')
```



In [668]:

```
plt.figure(figsize=(15,10))
plt.title('Building Grade vs Sale Price', fontsize=15)
plt.ylabel('Price - million USD', fontsize=15)
plt.xlabel('Bedrooms minus Bathrooms', fontsize=15)
boxplot = sns.boxplot(x='BldgGrade', y="SalePrice", data=df9, color='mediumaquamarine' ,sho
boxplot.set(xlabel='Grade',ylabel='Price - million USD');
plt.savefig('bldggrade.png', bbox_inches = 'tight')
```



In [669]:

```
import json
```

In [670]:

```
data_geo = df12.groupby('ZipCode')[['SalePrice']].mean().reset_index()
data_geo['ZipCode'] = data_geo['ZipCode'].astype(str)
```

In [671]:

```
data_geo = df12.groupby('ZipCode')[['SalePrice']].mean().reset_index()
data_geo['ZipCode'] = data_geo['ZipCode'].astype(str)


# count number of houses grouped by zipcode
df12['count'] = 1
temp = df12.groupby('ZipCode').sum()
temp.reset_index(inplace = True)
temp = temp[['ZipCode', 'count']]
data_geo = pd.merge(data_geo, temp, on='ZipCode')
# drop count from org dataset
df12.drop(['count'], axis = 1, inplace = True)

### download from here https://data-seattlecitygis.opendata.arcgis.com/datasets/zip-codes (
### the GeoJSON link and paste into browser to download.

king_county_geo = '..\..\data\Zip_Codes.geojson'
with open(king_county_geo, 'r') as j:
    geo_data = json.load(j)


tmp = geo_data
# remove ZIP codes not in geo data
geozips = []
for i in range(len(tmp['features'])):
    if tmp['features'][i]['properties']['ZIPCODE'] in list(data_geo['ZipCode'].unique()):
        geozips.append(tmp['features'][i])
# creating new JSON object
new_json = dict.fromkeys(['type','features'])
new_json['type'] = 'FeatureCollection'
new_json['features'] = geozips
# save updated JSON object
open("cleaned_geodata.json", "w").write(json.dumps(new_json, sort_keys=True, indent=4, sepa
```

Out[671]:

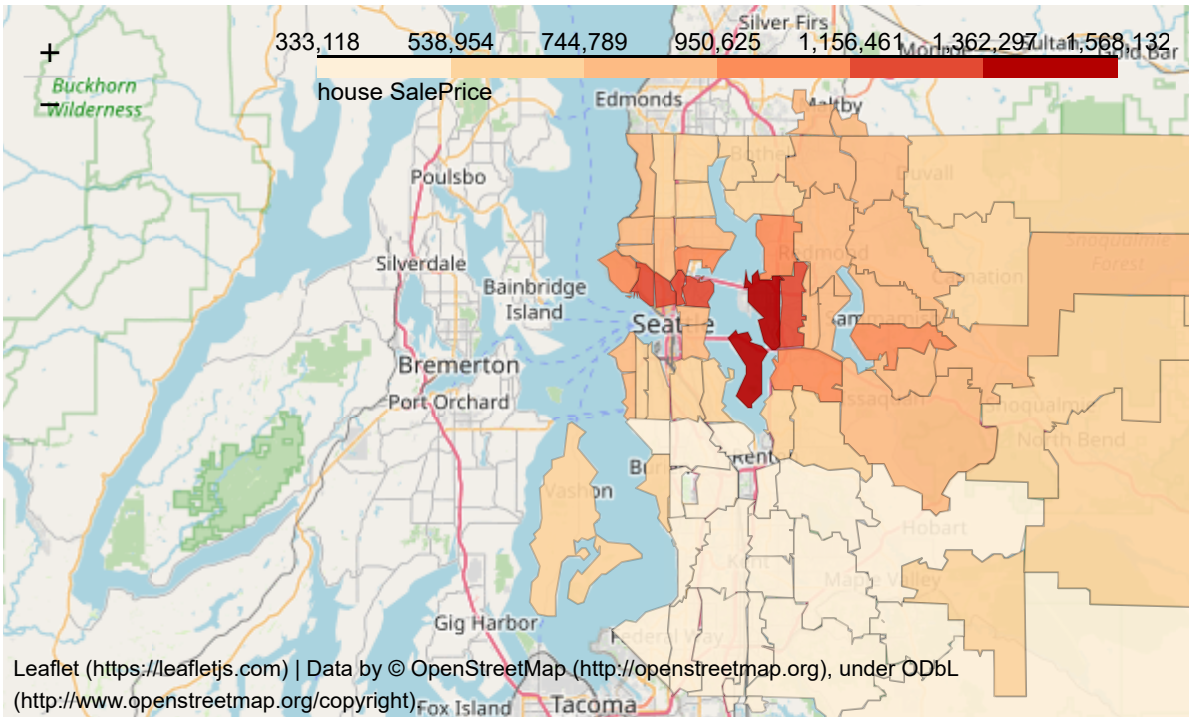
```
14976118
```

In [672]:

```
map_feature_by_zipcode(data_geo, 'SalePrice')
```

```
C:\Users\Andrew\anaconda3\envs\geo-env\lib\site-packages\folium\folium.py:40
9: FutureWarning: The choropleth  method has been deprecated. Instead use th
e new Choropleth class, which has the same arguments. See the example notebo
ok 'GeoJSON_and_choropleth' for how to do this.
  warnings.warn(
```

Out[672]:



In [673]:

```
# map showing the location of the 200 worst predicted house prices
get_map(large_delta)
```

Out[673]:

In [ ]:

```

```