LINEAR REGRESSION MODEL

IRONHACK'S 10.2020
DATA ANALYTICS COHORT
CHARLOTTE VELILLA

STRUCTURE

I. MySQL

II. Understanding Data

III. EDA

IV. Linear Regression Model

V. Tableau

VI. Insights, Challenges, Further possibilities

STRUCTURE

I. MySQL

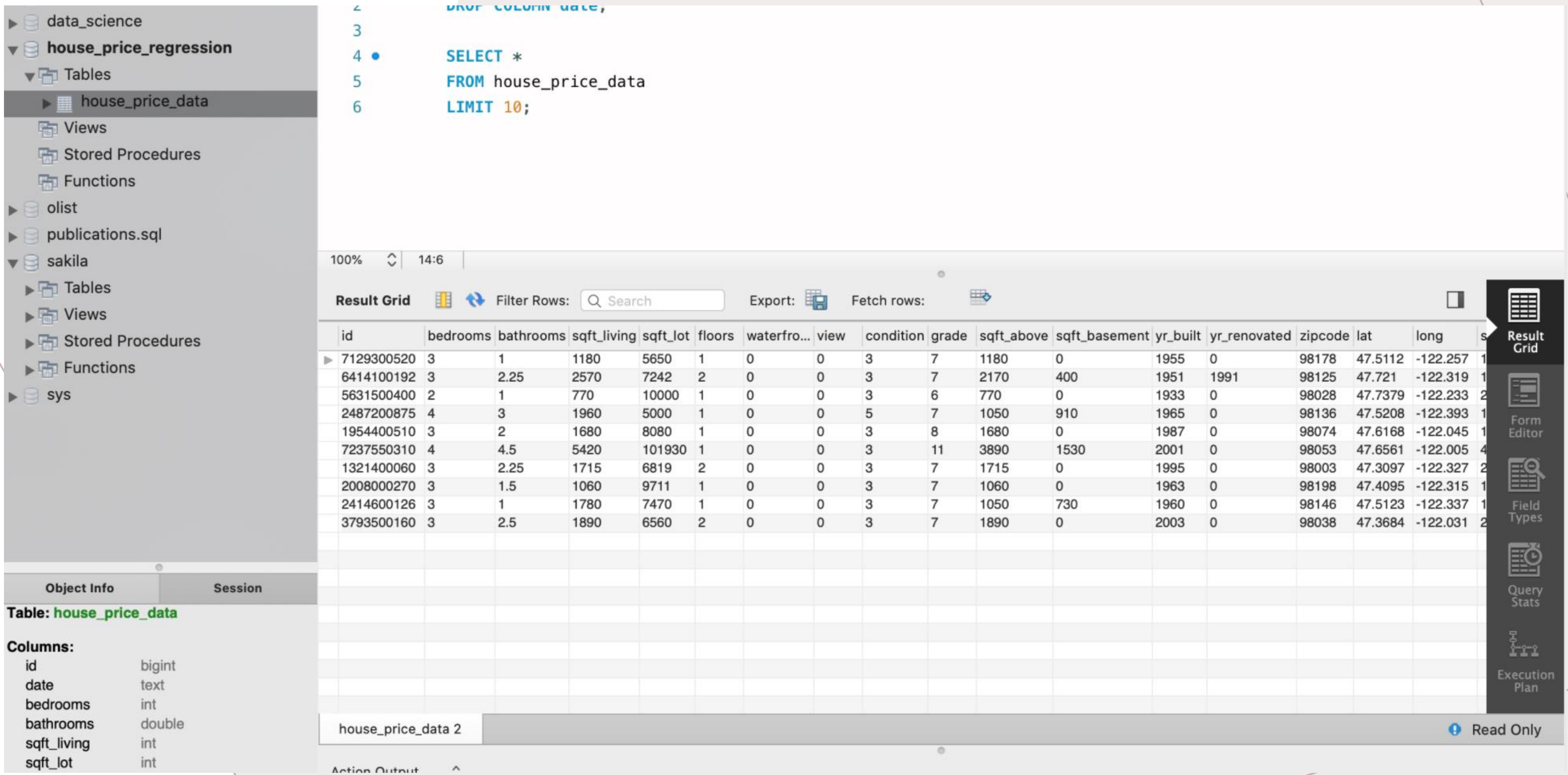
II. Understanding Data

III. EDA

IV. Linear Regression Model

V. Tableau

VI. Insights, Challenges, Further possibilities




```sql
SELECT *
FROM house_price_data
LIMIT 10;
```

id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro...	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long
7129300520	3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257
6414100192	3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	98125	47.721	-122.319
5631500400	2	1	770	10000	1	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233
2487200875	4	3	1960	5000	1	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393
1954400510	3	2	1680	8080	1	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045
7237550310	4	4.5	5420	101930	1	0	0	3	11	3890	1530	2001	0	98053	47.6561	-122.005
1321400060	3	2.25	1715	6819	2	0	0	3	7	1715	0	1995	0	98003	47.3097	-122.327
2008000270	3	1.5	1060	9711	1	0	0	3	7	1060	0	1963	0	98198	47.4095	-122.315
2414600126	3	1	1780	7470	1	0	0	3	7	1050	730	1960	0	98146	47.5123	-122.337
3793500160	3	2.5	1890	6560	2	0	0	3	7	1890	0	2003	0	98038	47.3684	-122.031

11. One of the customers is only interested in the following houses:
    - Number of bedrooms either 3 or 4
    - Bathrooms more than 3
    - One Floor
    - No waterfront
    - Condition should be 3 at least
    - Grade should be 5 at least
    - Price less than 300000

12. For the rest of the things, they are not too concerned. Write a simple query to find what are the options available for them?

In [16]:
```
query = """SELECT *
FROM house_price_data d
WHERE d.bedrooms = 3 OR 4
AND d.bathrooms > 3
AND d.floors = 1
AND d.waterfront = 0
AND d.condition >= 3
AND d.grade >= 5
AND d.price < 300000;"""
answer_11= pd.read_sql_query(query, engine)
answer_11
```

STRUCTURE

I. MySQL

II. Understanding Data

III. EDA

IV. Linear Regression Model

V. Tableau

VI. Insights, Challenges, Further possibilities

In [29]:
```
#The only types of data present in our data frame are: floats and integers
data.dtypes
```

Out[29]:
```
id                 int64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors             int64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
```

In [30]:
```
#It's expected latitude and longitute data to be of float type, bathroom data makes no sense when it's 2.25
#checking values in bathrooms column
data['bathrooms'].unique()
```

Out[30]:
```
array([1.  , 2.25, 3.  , 2.  , 4.5 , 1.5 , 2.5 , 1.75, 2.75, 3.25, 4.  ,
       3.5 , 0.75, 4.75, 5.  , 4.25, 3.75, 1.25, 5.25, 6.  , 0.5 , 5.5 ,
       6.75, 5.75, 8.  , 7.5 , 7.75, 6.25, 6.5 ])
```

```
In [31]: #As expected from .dtypes, all columns have numeric data
         data._get_numeric_data().columns

Out[31]: Index(['id', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
                'waterfront', 'view', 'condition', 'grade', 'sqft_above',
                'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long',
                'sqft_living15', 'sqft_lot15', 'price'],
               dtype='object')
```

```
In [32]: #Checking for Null values, data doesn't contain Null values
         data.isna().sum()

Out[32]: id               0
         bedrooms         0
         bathrooms        0
         sqft_living      0
         sqft_lot         0
         floors           0
         waterfront       0
         view             0
         condition        0
         grade            0
         sqft_above       0
         sqft_basement    0
         yr_built         0
         yr_renovated     0
         zipcode          0
         lat              0
         long             0
         sqft_living15    0
         sqft_lot15       0
         price            0
         dtype: int64
```

```
In [31]: #As expected from .dtypes, all columns have numeric data
         data._get_numeric_data().columns
```

???

```
Out[31]: Index(['id', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
                'waterfront', 'view', 'condition', 'grade', 'sqft_above',
                'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long',
                'sqft_living15', 'sqft_lot15', 'price'],
               dtype='object')
```

```
In [32]: #Checking for Null values, data doesn't contain Null values
         data.isna().sum()
```

```
Out[32]: id               0
         bedrooms         0
         bathrooms        0
         sqft_living      0
         sqft_lot         0
         floors           0
         waterfront       0
         view             0
         condition        0
         grade            0
         sqft_above       0
         sqft_basement    0
         yr_built         0
         yr_renovated     0
         zipcode          0
         lat              0
         long             0
         sqft_living15    0
         sqft_lot15       0
         price            0
         dtype: int64
```

STRUCTURE

I. MySQL

II. Understanding Data

III. EDA

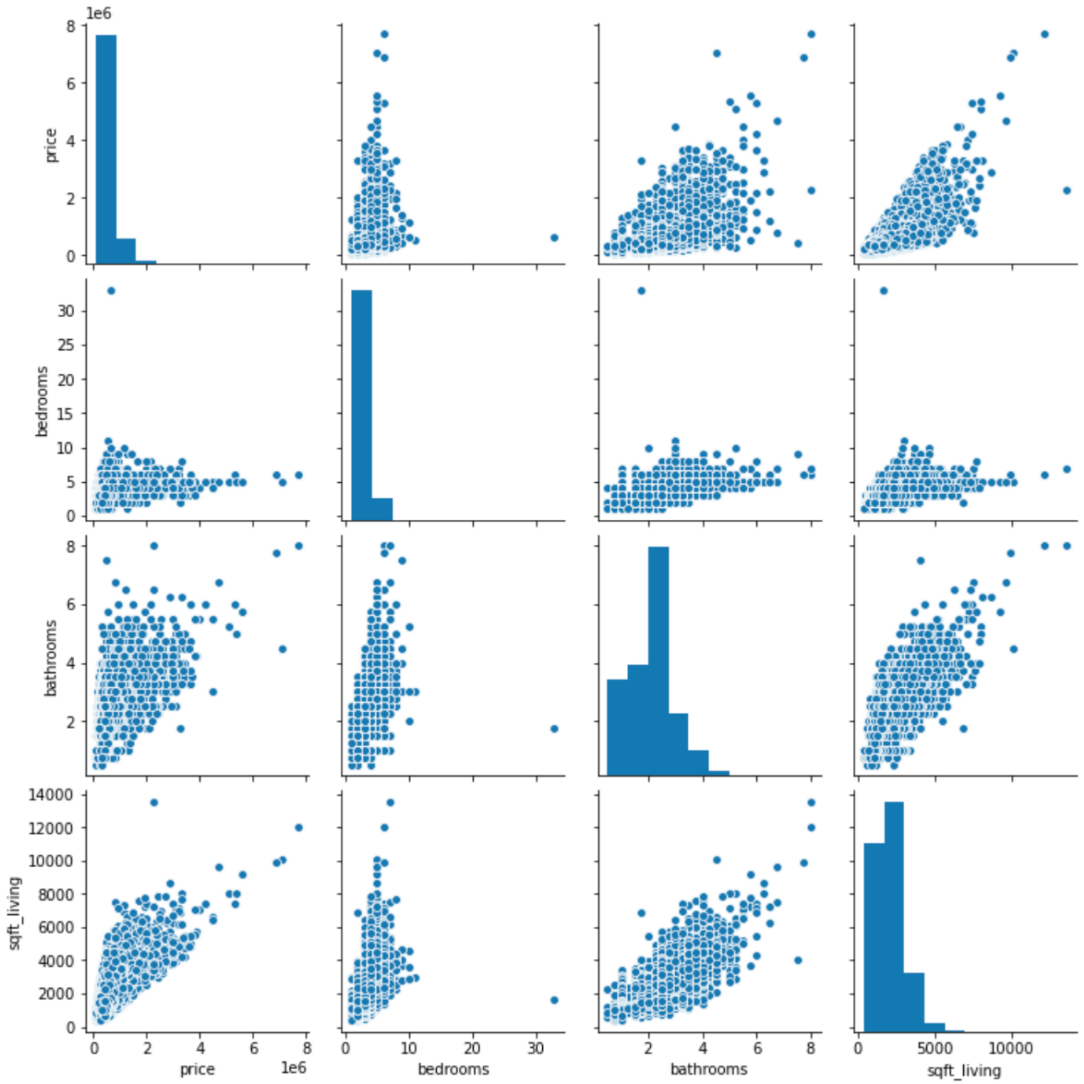
IV. Linear Regression Model

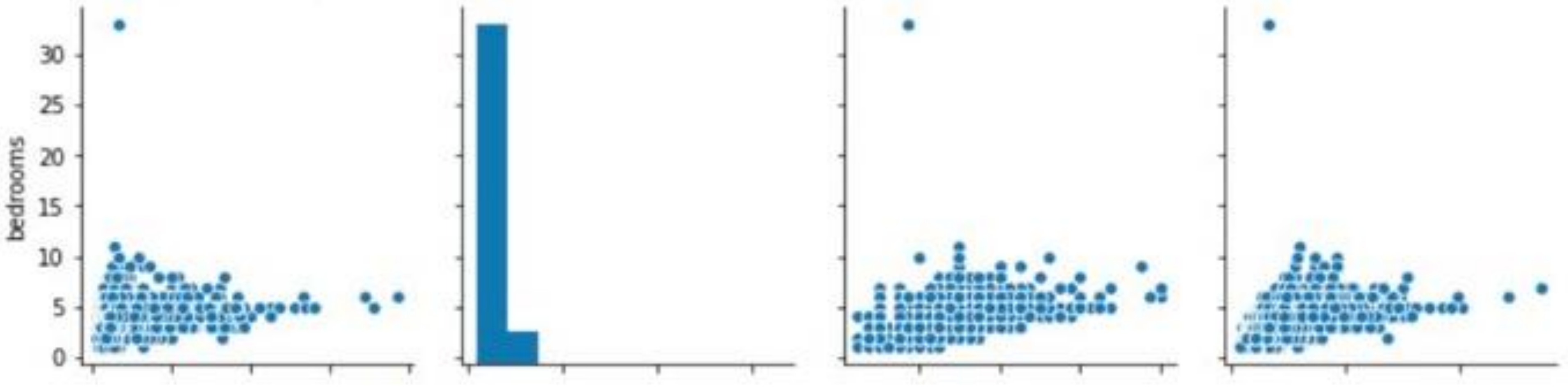
V. Tableau

VI. Insights, Challenges, Further possibilities

OUTLIERS?

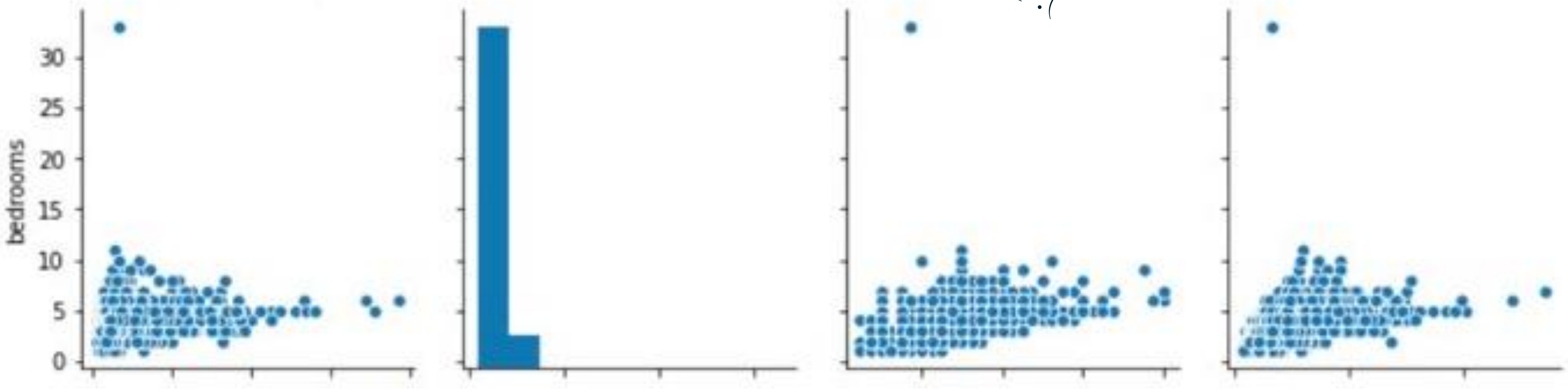
In [7]:
```
data_pairplot1 = sns.pairplot(data, vars=['price','bedrooms','bathrooms', 'sqft_living'])
```



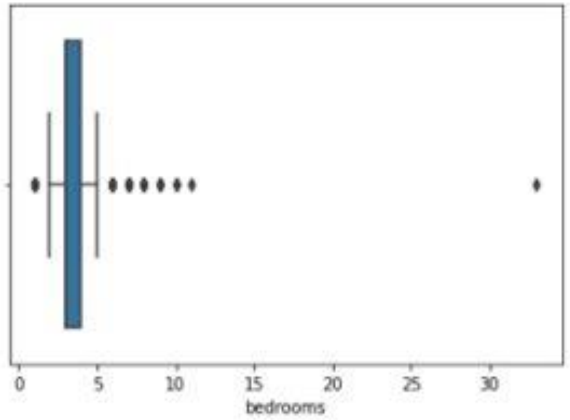


bedrooms
30
25
20
15
10
5
0

LONELY :(



```
In [36]: ax = sns.boxplot(x=data["bedrooms"])
```




```
In [37]: #Understanding outlier, seems to be a data entry error since it's not probable
         #a property of 33 rooms and 6,000ft2 lot has only 1.75 bahtrooms and a price of $640,000
         data.loc[data['bedrooms']== 33]
```

Out[37]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	z
15856	2402100895	33	1.75	1620	6000	1	0	0	5	7	1040	580	1947	0	

```
In [38]: #Comparing outlier with most expensive property on dataset with a price of $7,700,000
         data.loc[data['id']== 6762700020]
```

Out[38]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zip
7245	6762700020	6	8.0	12050	27600	3	0	3	4	13	8570	3480	1910	1987	

In [37]:
```python
#Understanding outlier, seems to be a data entry error since it's not probable
#a property of 33 rooms and 6,000ft2 lot has only 1.75 bahtrooms and a price of $640,000
data.loc[data['bedrooms']== 33]
```

Out[37]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	z
15856	2402100895	33	1.75	1620	6000	1	0	0	5	7	1040	580	1947	0	

In [38]:
```python
#Comparing outlier with most expensive property on dataset with a price of $7,700,000
data.loc[data['id']== 6762700020]
```

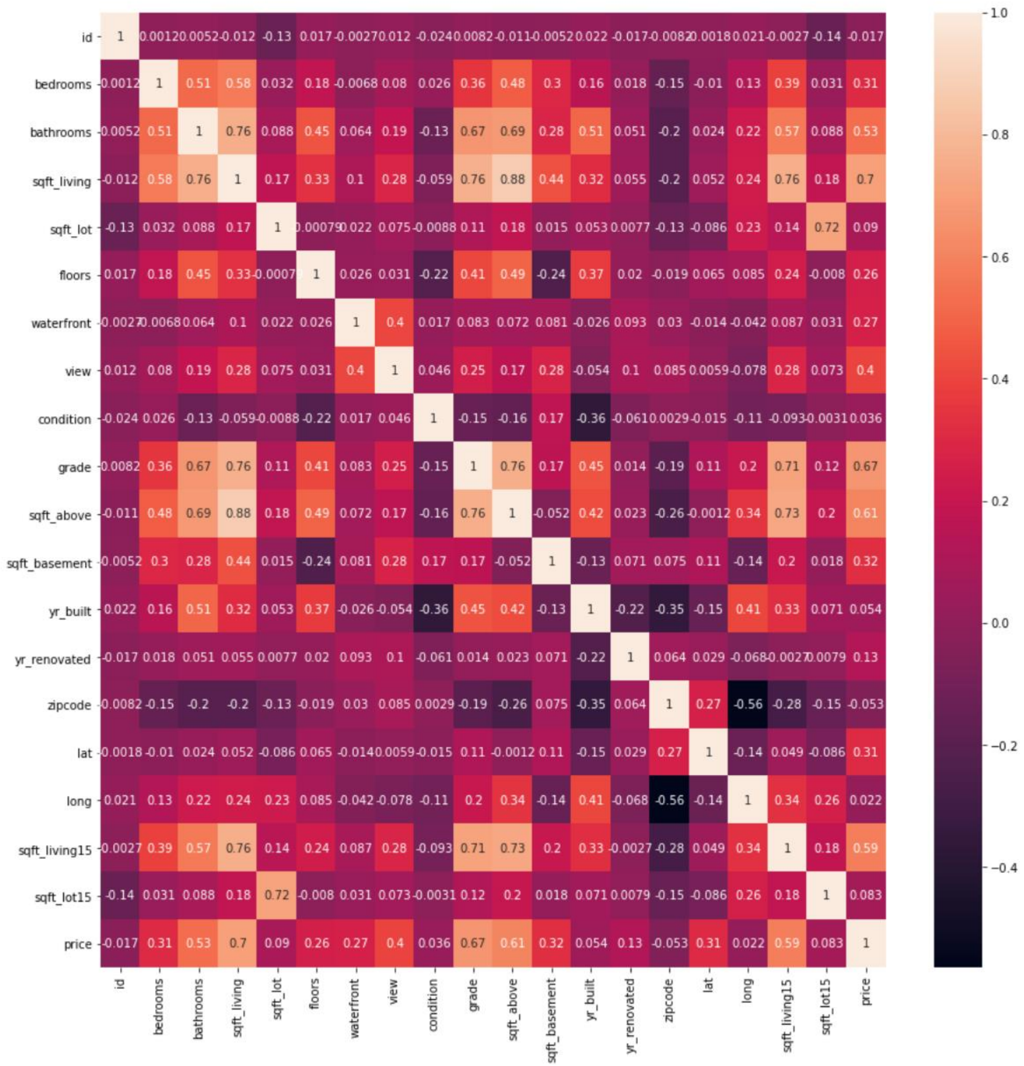
Out[38]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zip
7245	6762700020	6	8.0	12050	27600	3	0	3	4	13	8570	3480	1910	1987	

ANYWAYS…

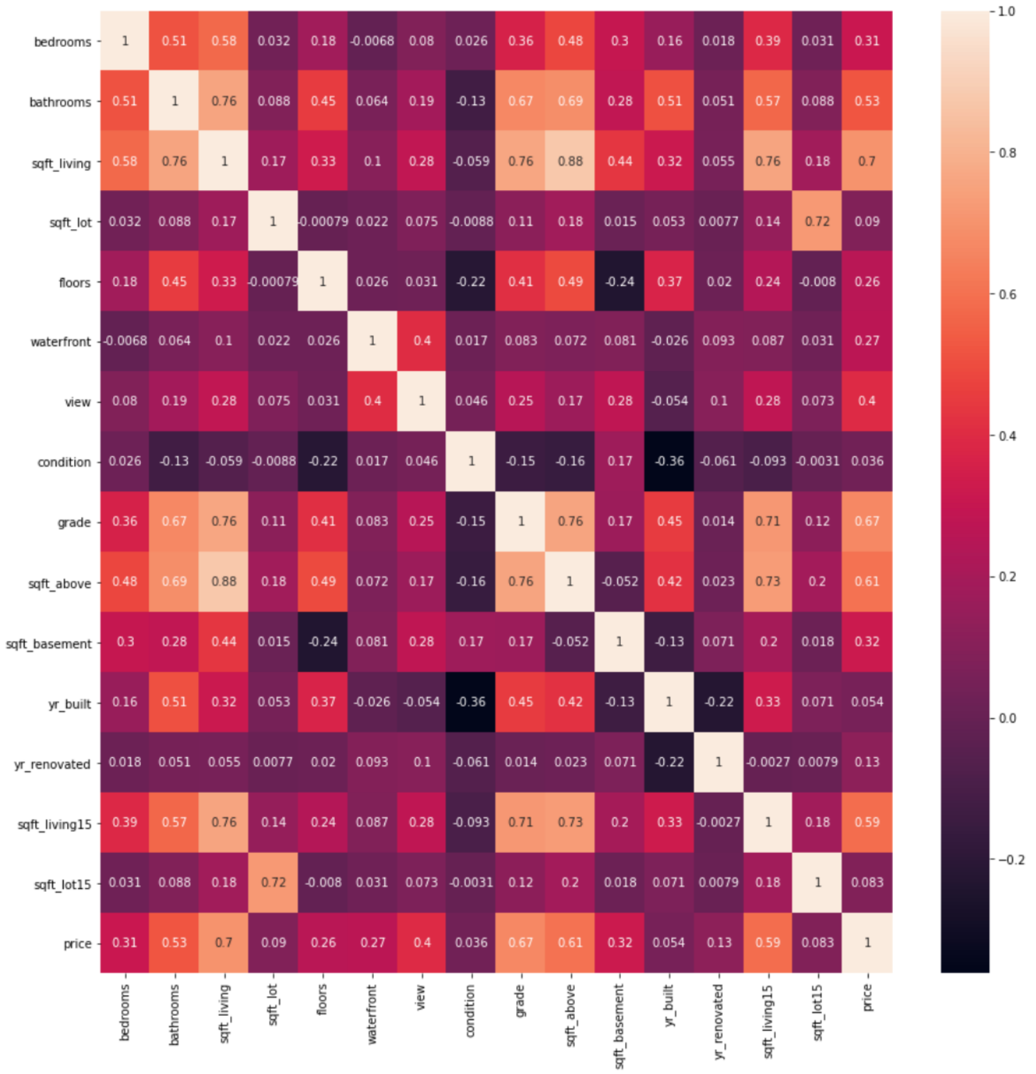
In [56]:
```
#checking correlation matrix for strenght of collinearity of the  data with all comlumns
#(using Pearson correlation)
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(data.corr(), annot=True)
```

Out[56]: <AxesSubplot:>



```python
#(using Pearson correlation)
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(data_model.corr(), annot=True)
```

Out[58]: <AxesSubplot:>



In [77]:
```
#column 'condition'
bins2 = [1,2,3,5]
group_names2 = ['low', 'average', 'high']
```

In [78]:
```
data_model['condition_bin'] = pd.cut(data['condition'], bins2, labels=group_names2)
```

In [79]:
```
data_model['condition_bin'].value_counts()
```

Out[79]:
```
average    14020
high        7378
low          170
Name: condition_bin, dtype: int64
```

```
In [74]: #column 'grade'
         bins = [0, 5, 10, 15]
         group_names = ['low', 'average', 'high']
         data_model['grade_bin'] = pd.cut(data['grade'], bins, labels=group_names)
```

```
In [77]: #column 'condition'
         bins2 = [1,2,3,5]
         group_names2 = ['low', 'average',

In [78]: data_model['condition_bin'] = pd.cut(da                labels=group_names2)

In [79]: data_model['condition_bin'].value_counts()

Out[79]: average    14020
         high        7378
         low          170
         Name: condition_bin, dtype: int64
```

```
In [74]: #column
         bins =
         group_
In [74]: data_
```

```
In [77]: #column 'condition'
         bins2 = [1,2,3,5
         group_names2 =

In [78]: data_mod                                   ames2)

In [79]:

Out[7
```

```
In [81]: #column 'yr_built'
         bins3 = [1900,1950,2000,2050]
         group_names3 = ['old', 'average', 'recent']

In [82]: data_model['yr_built_bin'] = pd.cut(data['yr_built'], bins3, labels=group_names3)

In [83]: data_model['yr_built_bin']

Out[83]: 0        average
         1        average
         2            old
         3        average
         4        average
                   ...
         21592     recent
         21593     recent
         21594     recent
         21595     recent
         21596     recent
         Name: yr_built_bin, Length: 21597, dtype: category
         Categories (3, object): ['old' < 'average' < 'recent']
```

```
], bins, labels=group_names)
```

In [109]:

```
data_model
```

Out[109]:

rice	condition_bin_low	condition_bin_average	condition_bin_high	grade_bin_low	grade_bin_average	grade_bin_high	yr_built_bin_old	yr_built_bin_average	yr_bui
1900	0	1	0	0	1	0	0	1	
3000	0	1	0	0	1	0	0	1	
0000	0	1	0	0	1	0	1	0	
4000	0	0	1	0	1	0	0	1	
0000	0	1	0	0	1	0	0	1	
...	...	...	...	...	...	...	...	...	
0000	0	1	0	0	1	0	0	0	
0000	0	1	0	0	1	0	0	0	
2101	0	1	0	0	1	0	0	0	
0000	0	1	0	0	1	0	0	0	
5000	0	1	0	0	1	0	0	0	

STRUCTURE

I. MySQL

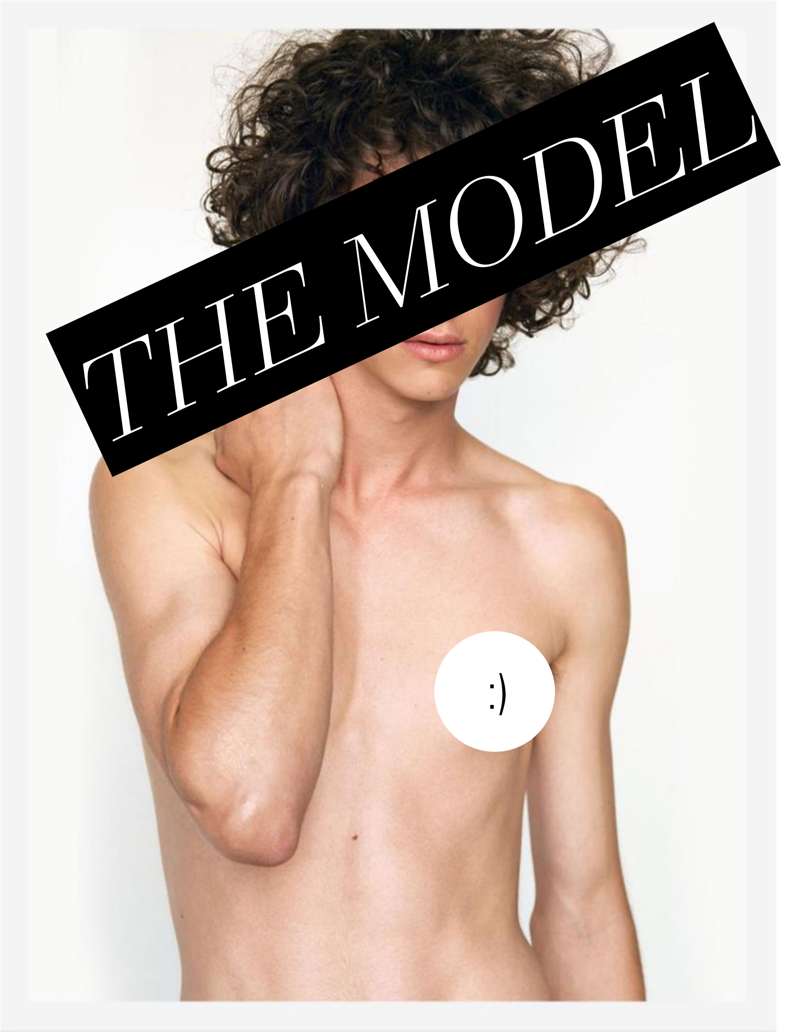
II. Understanding Data

III. EDA

IV. Linear Regression Model

V. Tableau

VI. Insights, Challenges, Further possibilities


THE MODEL
:)

Linear Regression Model

```
#Dependent Variable
Y = data_model['price']
#Independent Variables
X = data_model.drop(['price'], axis=1)
```

```
Y
```

```
0        221900
1        538000
2        180000
3        604000
4        510000
          ...
21592    360000
21593    400000
21594    402101
21595    400000
21596    325000
Name: price, Length: 21597, dtype:
```

```
Y
```

```
0        221900
1        538000
2        180000
3        604000
4        510000
          ...
21592    360000
21593    400000
21594    402101
21595    400000
21596    325000
Name: price, Length: 21597, dtype: int64
```

Y

In [112]:
```
X
```

Out[112]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	...	sqft_lot15	condition_bin_low	condition_bin_avera
0	3	1.00	1180	5650	1	0	0	3	7	1180	...	5650	0	
1	3	2.25	2570	7242	2	0	0	3	7	2170	...	7639	0	
2	2	1.00	770	10000	1	0	0	3	6	770	...	8062	0	
3	4	3.00	1960	5000	1	0	0	5	7	1050	...	5000	0	
4	3	2.00	1680	8080	1	0	0	3	8	1680	...	7503	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21592	3	2.50	1530	1131	3	0	0	3	8	1530	...	1509	0	
21593	4	2.50	2310	5813	2	0	0	3	8	2310	...	7200	0	
21594	2	0.75	1020	1350	2	0	0	3	7	1020	...	2007	0	
21595	3	2.50	1600	2388	2	0	0	3	8	1600	...	1287	0	
21596	2	0.75	1020	1076	2	0	0	3	7	1020	...	1357	0	

21597 rows × 24 columns

X

```python
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_iris
from sklearn import datasets, linear_model, metrics
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

Splitting Train/Test

```python
#Splitting 40% data  as testing data
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.4)
```

```python
#Checking split

for data in (X_train, X_test, Y_train, Y_test):
    print(len(data))
```

```
12958
8639
12958
8639
```

```python
lm.coef_
```

```
array([-2.78407057e+04,  4.27722541e+04,  9.70174841e+01, -1.19350573e-02,
        9.91650853e+03,  5.65247862e+05,  4.14125609e+04,  3.39871683e+04,
        1.14645304e+05,  3.98078311e+01,  5.72096532e+01, -3.46770235e+03,
        1.80309655e+01,  2.76998317e+01, -4.58697647e-01, -1.22073152e+04,
       -3.31683841e+04, -4.18102601e+04, -3.36938522e+04, -1.31214740e+05,
        1.64908592e+05,  6.20452044e+04,  3.24100601e+04,  1.08626776e+05])
```

Normalizing Data

In [133]:
```
transformer = Normalizer().fit(X)
x_normalized = transformer.transform(X)
x = pd.DataFrame(x_normalized)
```

```
lm = linear_model.LinearRegression()
model = lm.fit(X,Y)
lm.score(X,Y)
```

0.6715839085011694

Normalizing Data

In [133]:

```
transformer = Normalizer().fit(X)
x_normalized = transform
x = pd.DataFram
```

In [126]:

```
#apply the machine learn model with normalized
lm = linear_model.LinearRegression()
model = lm.fit(x_train,y_train)
predictions  = lm.predict(x_test)
r2_score(y_test, predictions)
```

Out[126]: 0.4560691494926651

STRUCTURE

I. MySQL

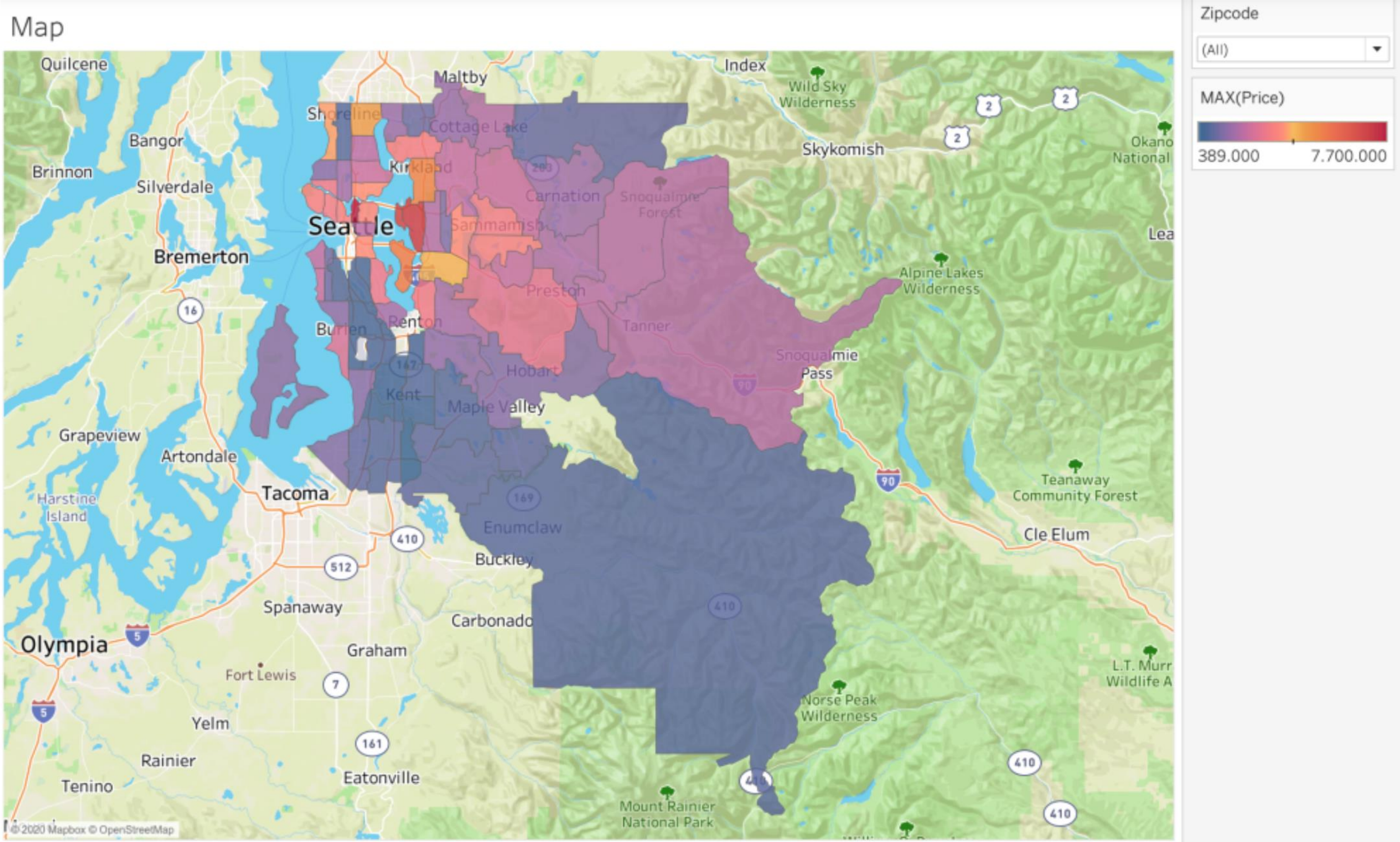
II. Understanding Data

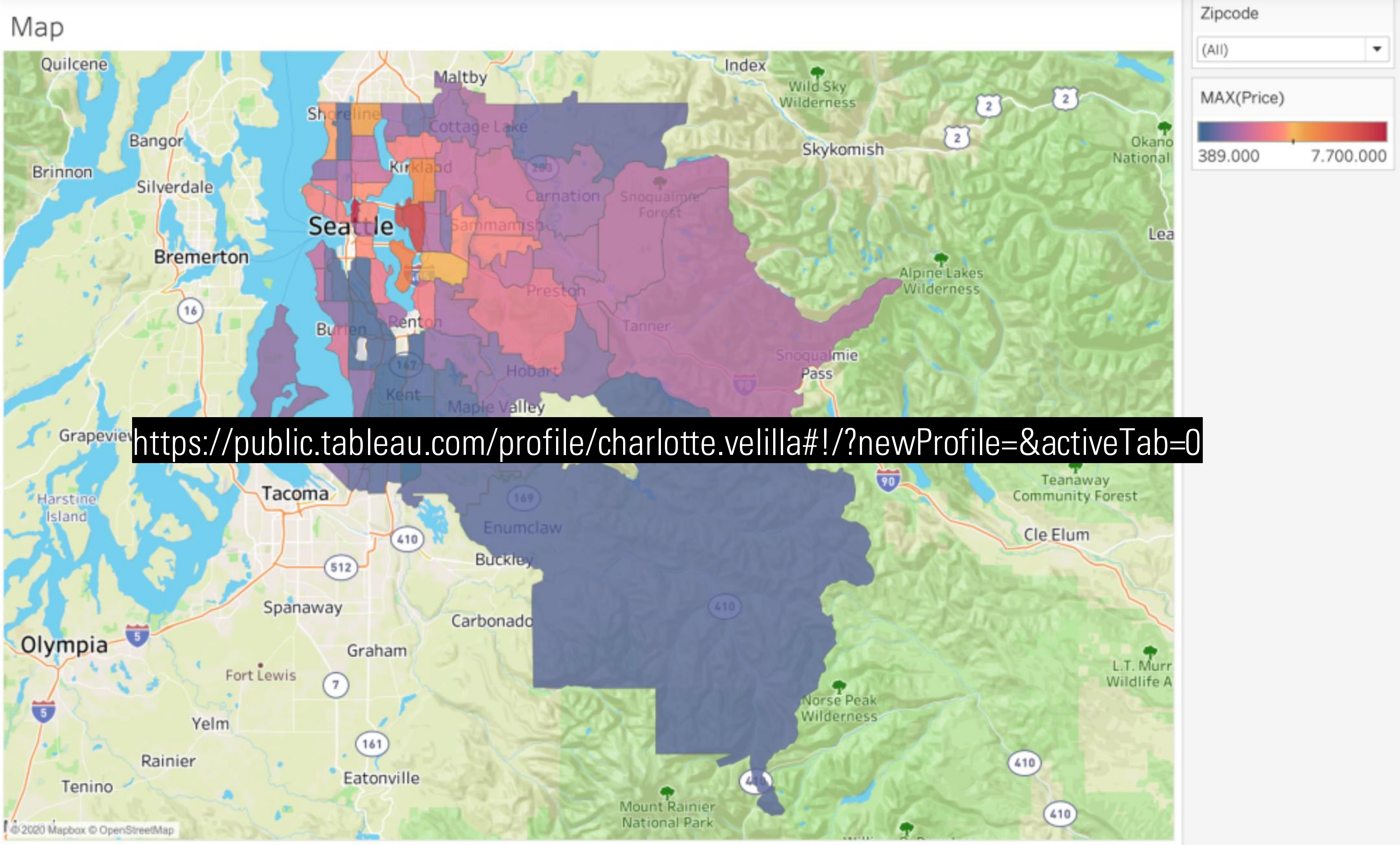
III. EDA

IV. Linear Regression Model

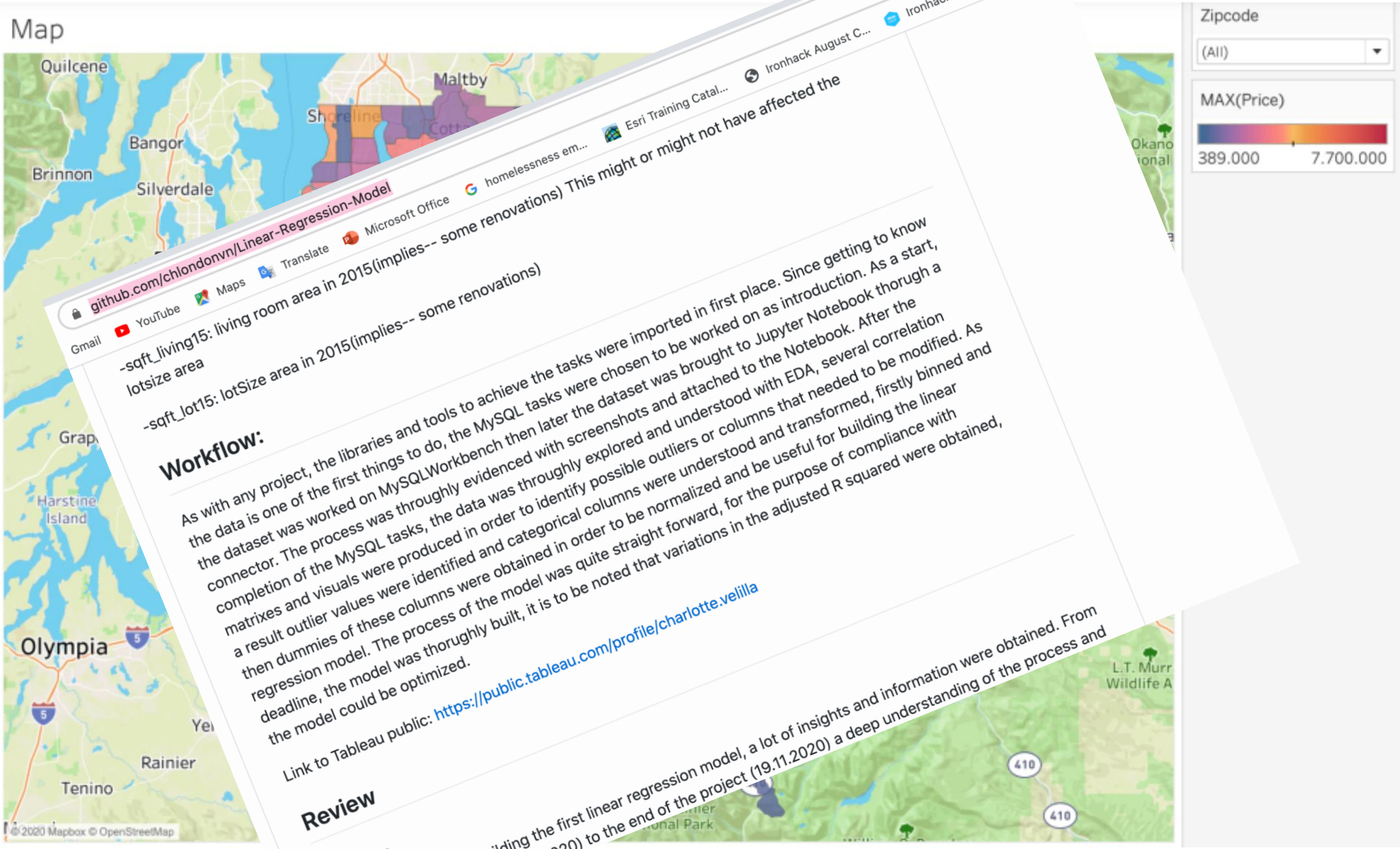
V. Tableau

VI. Insights, Challenges, Further possibilities


Map
Zipcode
(All)
MAX(Price)
389.000
7.700.000
Quilcene
Maltby
Index
Wild Sky Wilderness
Shoreline
Cottage Lake
Bangor
Skykomish
Brinnon
Silverdale
Kirkland
Carnation
Snoqualmie Forest
Okano National
Seattle
Sammamish
Lea
Bremerton
Alpine Lakes Wilderness
Preston
Burien
Renton
Tanner
Snoqualmie Pass
Hobart
Kent
Maple Valley
Grapeview
Artondale
Teanaway Community Forest
Tacoma
Harstine Island
Enumclaw
Cle Elum
Buckley
Spanaway
Carbonado
Olympia
Graham
L.T. Murr Wildlife A
Fort Lewis
Norse Peak Wilderness
Yelm
Rainier
Eatonville
Tenino
Mount Rainier National Park
© 2020 Mapbox © OpenStreetMap



https://public.tableau.com/profile/charlotte.velilla#!/?newProfile=&activeTab=0



github.com/chlondonvn/Linear-Regression-Model

Gmail YouTube Maps Translate Microsoft Office homelessness em... Esri Training Catal... Ironhack August C... Ironhack October...

-sqft_living15: living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area

-sqft_lot15: lotSize area in 2015(implies-- some renovations)

Workflow:

As with any project, the libraries and tools to achieve the tasks were imported in first place. Since getting to know the data is one of the first things to do, the MySQL tasks were chosen to be worked on as introduction. As a start, the dataset was worked on MySQLWorkbench then later the dataset was brought to Jupyter Notebook thorough a connector. The process was throughly evidenced with screenshots and attached to the Notebook. After the completion of the MySQL tasks, the data was throughly explored and understood with EDA, several correlation matrixes and visuals were produced in order to identify possible outliers or columns that needed to be modified. As a result outlier values were identified and categorical columns were understood and transformed, firstly binned and then dummies of these columns were obtained in order to be normalized and be useful for building the linear regression model. The process of the model was quite straight forward, for the purpose of compliance with deadline, the model was thoroughly built, it is to be noted that variations in the adjusted R squared were obtained, the model could be optimized.

Link to Tableau public: https://public.tableau.com/profile/charlotte.velilla

Review

Insights

From the process of building the first linear regression model, a lot of insights and information were obtained. From the start of day one on (16.11.2020) to the end of the project (19.11.2020) a deep understanding of the process and

STRUCTURE

I. MySQL

II. Understanding Data

III. EDA

IV. Linear Regression Model

V. Tableau

VI. Insights, Challenges, Further possibilities