In [347]:
```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *
import statsmodels.api as sm
pd.options.mode.chained_assignment = None  # default='warn'

from sklearn.linear_model import LinearRegression, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_
error
from sklearn.metrics import accuracy_score, confusion_matrix, plot_confu
sion_matrix

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict

%matplotlib inline
```

In [203]:
```
house = pd.read_csv("https://raw.githubusercontent.com/jacobpappa/kc_hou
se/main/kc_house_data.csv")
house.head()
```

Out[203]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

1. Which variable has the greatest effect on the sales price of a house in King County?

In [204]:
```
zScore = StandardScaler()
```

In [205]:
```
zScore.fit(house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot","floors"]])

z = zScore.transform(house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot","floors"]])

house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot","floors"]] = z

house.head()
```

Out[205]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	20141013T000000	221900.0	-0.398737	-1.447464	-0.979835	-0.228321	-0.91542
1	6414100192	20141209T000000	538000.0	-0.398737	0.175607	0.533634	-0.189885	0.9365(
2	5631500400	20150225T000000	180000.0	-1.473959	-1.447464	-1.426254	-0.123298	-0.91542
3	2487200875	20141209T000000	604000.0	0.676485	1.149449	-0.130550	-0.244014	-0.91542
4	1954400510	20150218T000000	510000.0	-0.398737	-0.149007	-0.435422	-0.169653	-0.91542

5 rows × 21 columns

In [206]:
```
kf = KFold (n_splits = 100)
```

In [207]:
```
predictors = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "waterfront", "condition", "grade"]
X = house[predictors]
y = house["price"]
```

In [208]:
```
LR = LinearRegression()
mse = []
r2 = []
```

In [209]:
```
for train,test in kf.split(X):
    X_train = X.iloc[train]
    X_test = X.iloc[test]
    y_train = y[train]
    y_test = y[test]

    model = LR.fit(X_train, y_train)

    mse.append(mean_squared_error(y_test, model.predict(X_test)))
    r2.append(r2_score(y_test,model.predict(X_test)))
```

In [210]:
```
np.mean(mse)
```

Out[210]: 55293167267.167366

In [211]:
```
np.mean(r2)
```

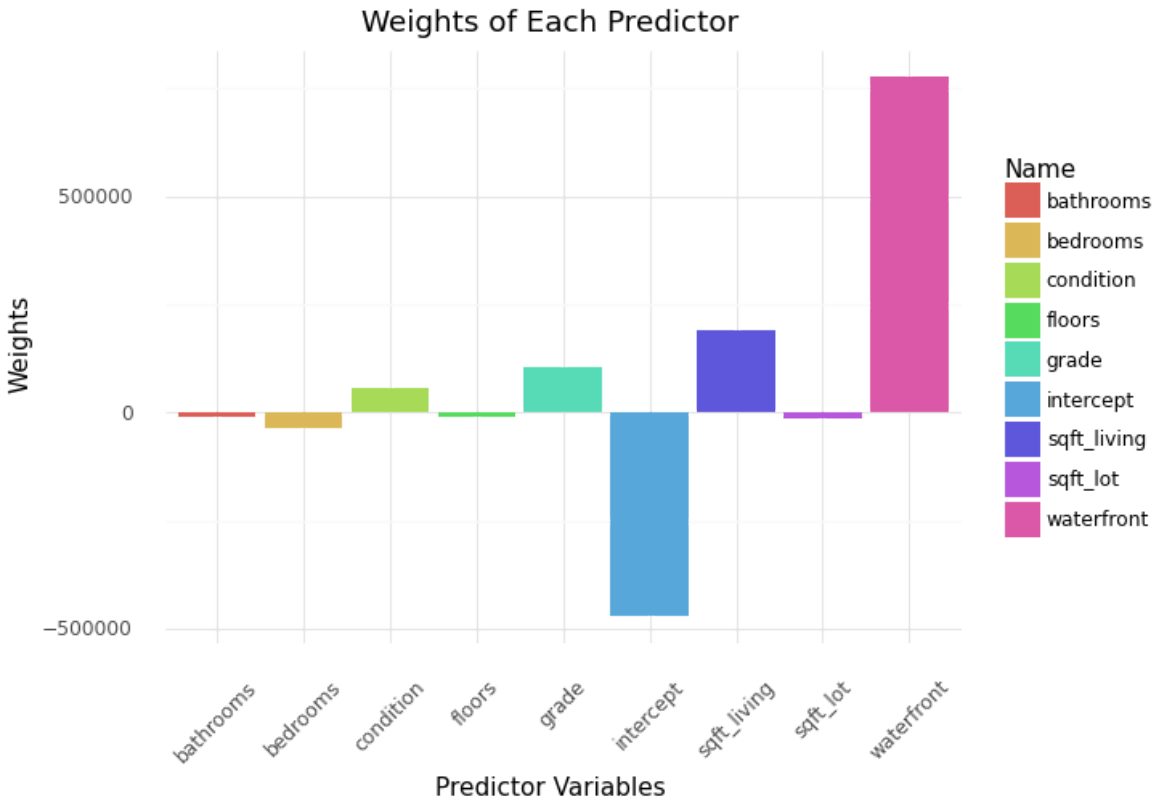
Out[211]: 0.575414103810196

In [212]:
```
coefficients = pd.DataFrame({"Coef":LR.coef_, "Name": predictors})

coefficients = coefficients.append({"Coef": LR.intercept_, "Name": "inte
rcept"}, ignore_index = True)

coefficients
```

Out[212]:

	Coef	Name
0	-35192.258011	bedrooms
1	-9550.403207	bathrooms
2	192244.157989	sqft_living
3	-13679.179094	sqft_lot
4	-11140.174024	floors
5	775866.564204	waterfront
6	58737.445190	condition
7	105141.902813	grade
8	-471081.132921	intercept

In [213]:
```
(ggplot(coefficients, aes(x = "Name", y = "Coef", fill = "Name")) +
 stat_summary(geom = "bar") + ggtitle("Weights of Each Predictor") +
 xlab("Predictor Variables") +
 ylab("Weights") +
 theme_minimal() +
 theme(axis_text_x = element_text(angle = 45)))
```

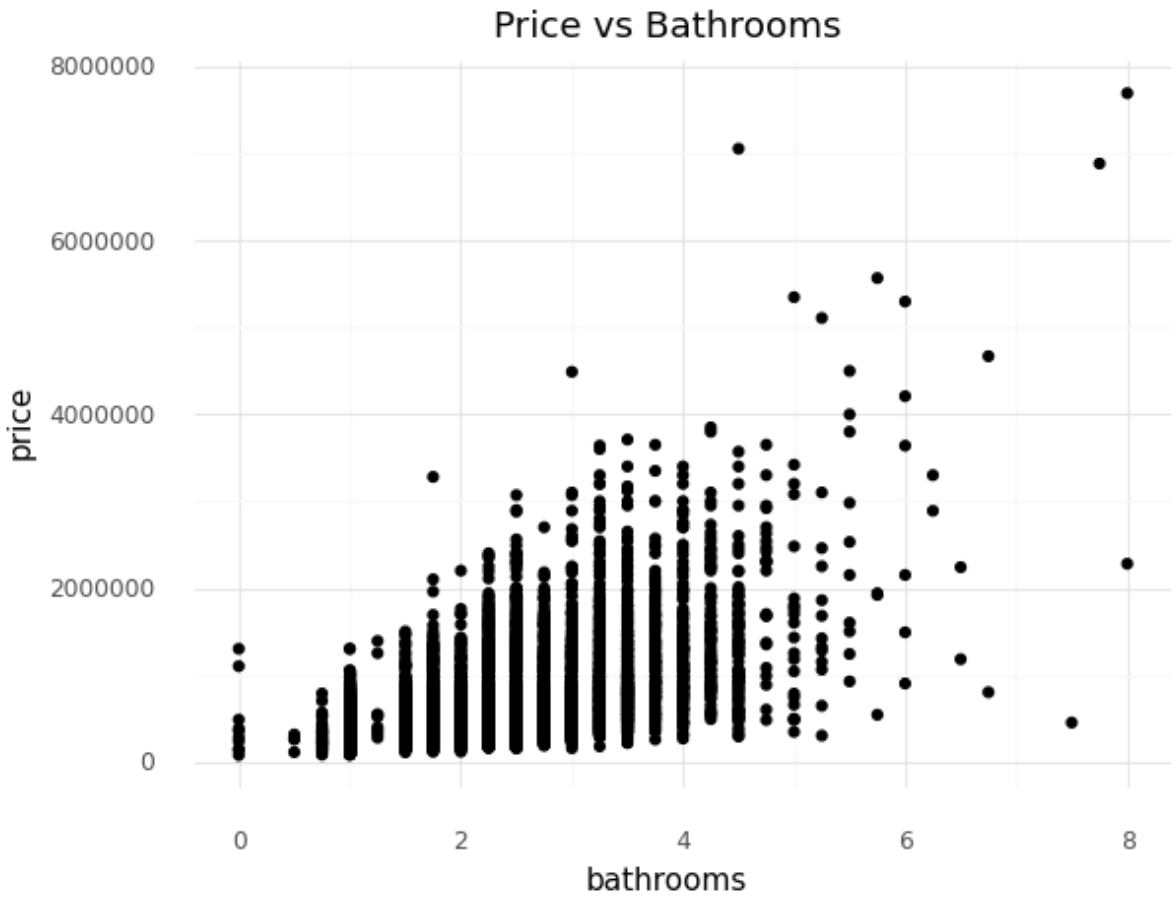


Out[213]: <ggplot: (8771759285071)>

This bar graph shows the weights of each predictor variable in the linear regression model to predict the price of a house.

In [214]:
```
house1 = pd.read_csv("https://raw.githubusercontent.com/jacobpappa/kc_house/main/kc_house_data.csv")
```

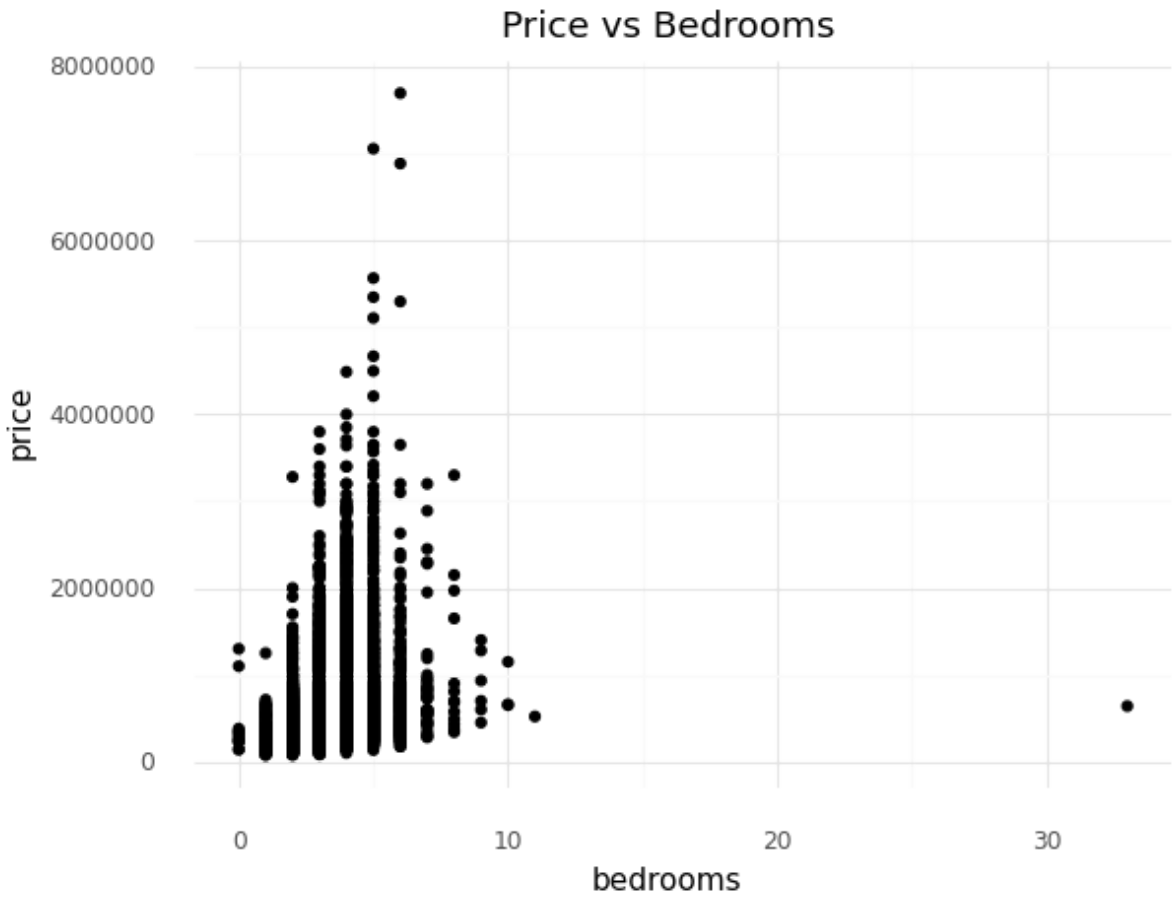
In [215]:
```
(ggplot(house1, aes(x = "bathrooms", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Bathrooms"))
```



Out[215]: <ggplot: (8771736467655)>

This scatterplot shows the price of the house based on the amount of bathrooms in the house.

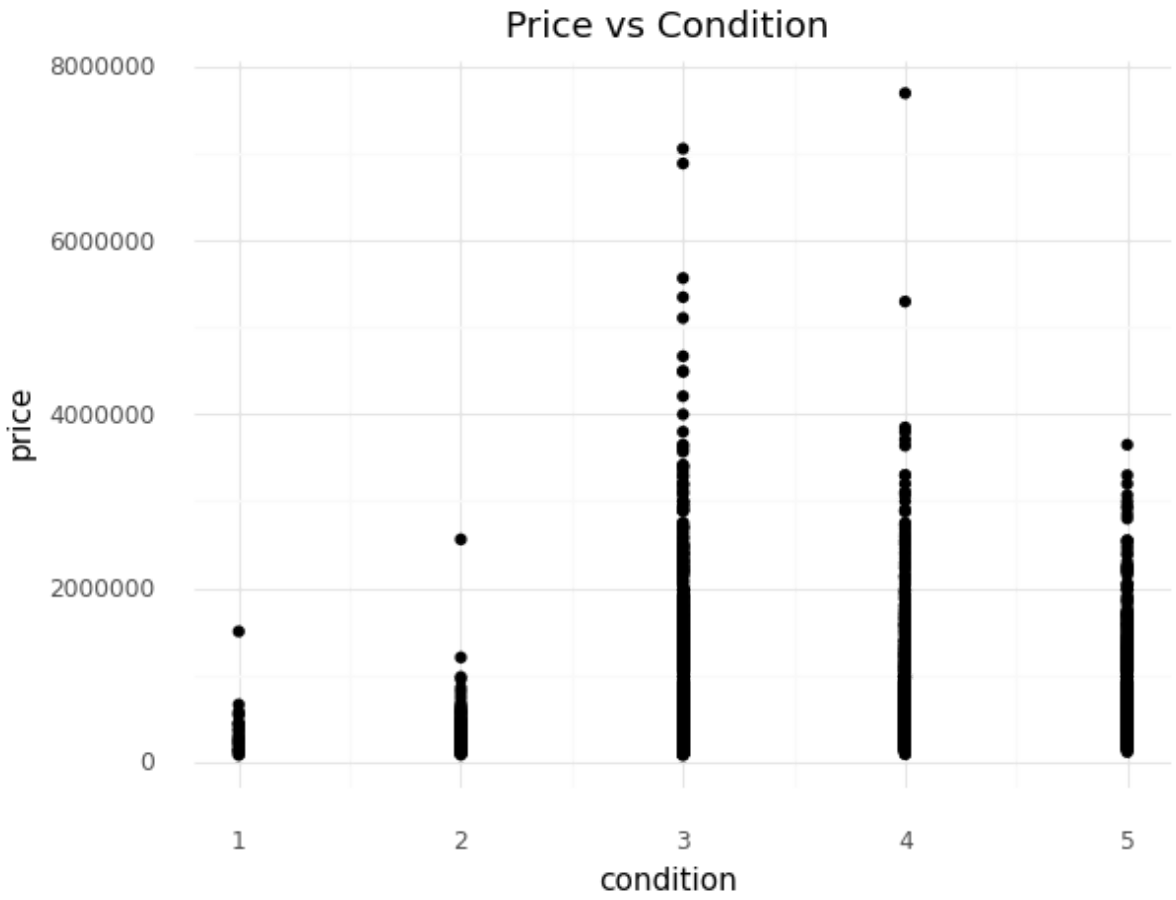
In [216]:
```
(ggplot(house1, aes(x = "bedrooms", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Bedrooms"))
```



Out[216]: <ggplot: (8771740091642)>

This scatterplot shows the price of the house based on the amount of bathrooms in the house.

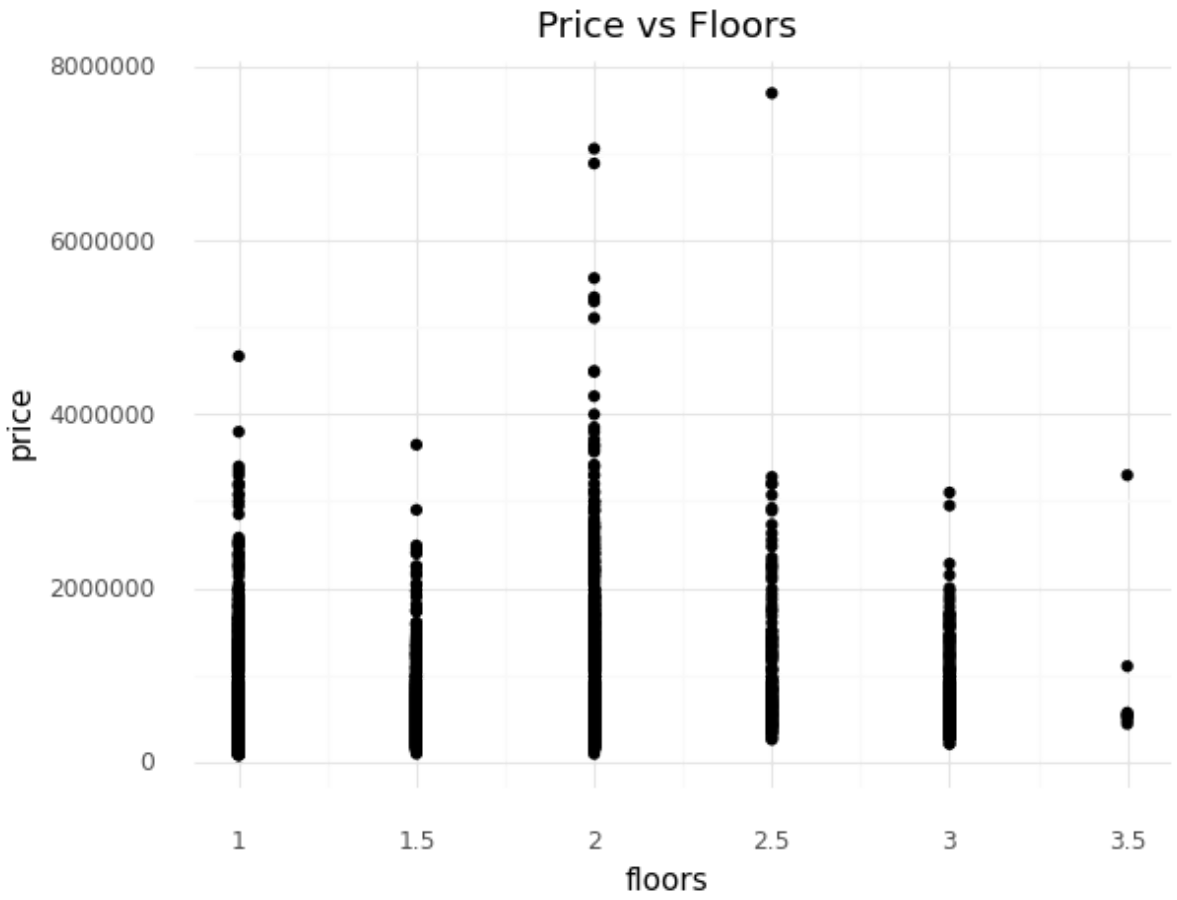
In [217]:
```
(ggplot(house1, aes(x = "condition", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Condition"))
```



Out[217]: <ggplot: (8771760937087)>

This scatterplot shows the price of the house based on the condition of the house.

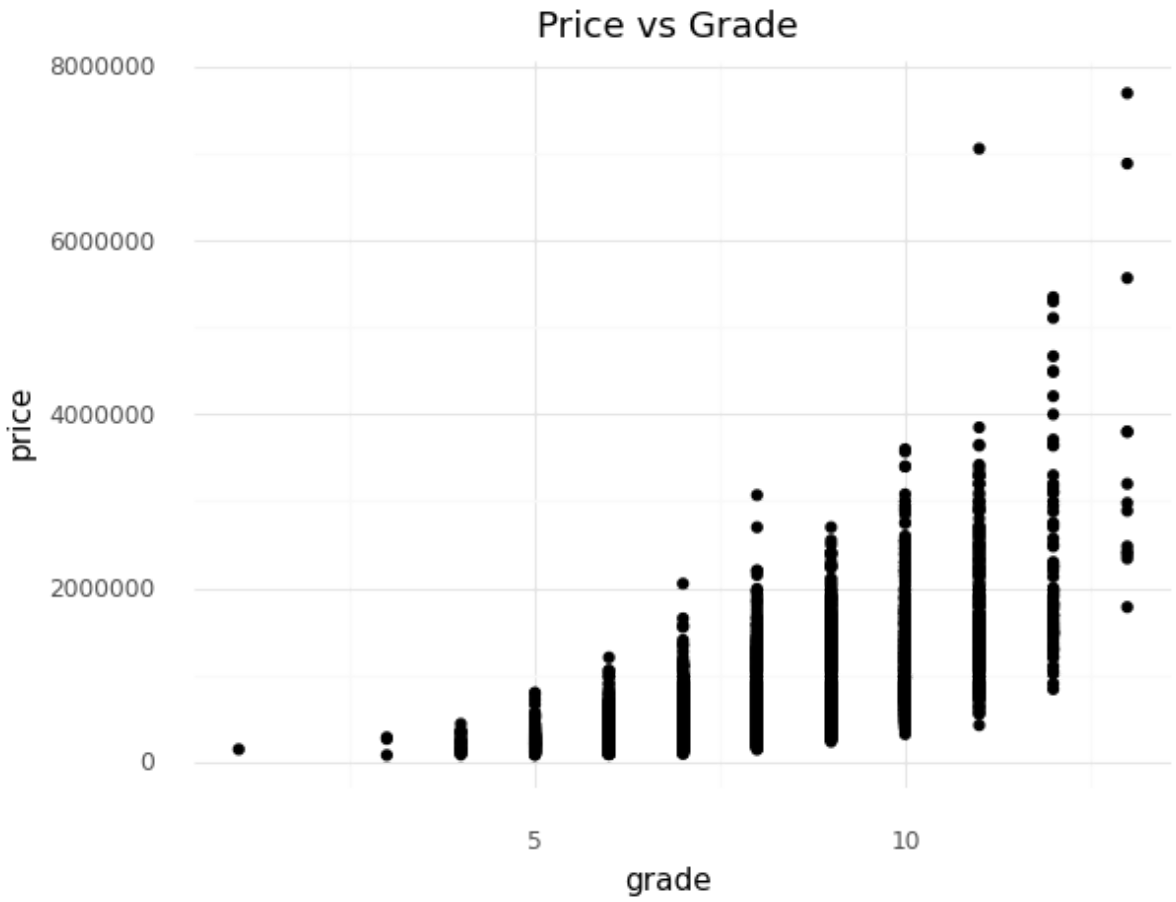
In [218]:
```
(ggplot(house1, aes(x = "floors", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Floors"))
```



Out[218]: <ggplot: (8771757697774)>

This scatterplot shows the price of the house based on the amount of floors in the house.

In [219]:
```
(ggplot(house1, aes(x = "grade", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Grade"))
```



Out[219]: <ggplot: (8771734025430)>

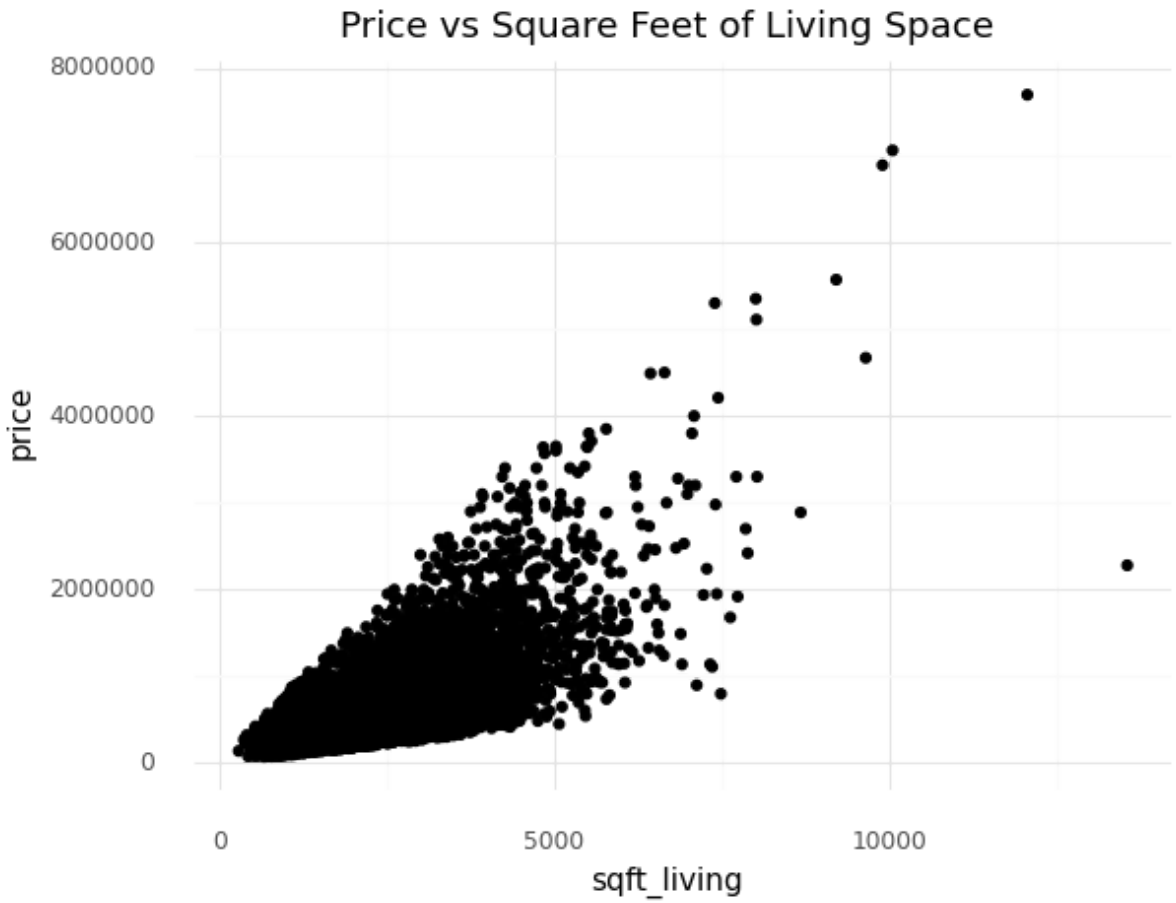
This scatterplot shows the price of the house based on the grade of the house.

In [220]:
```
(ggplot(house1, aes(x = "sqft_lot", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Square Feet of Lot"))
```

Out[220]: <ggplot: (8771757697558)>

This scatterplot shows the price of the house based on the square feet of the lot.

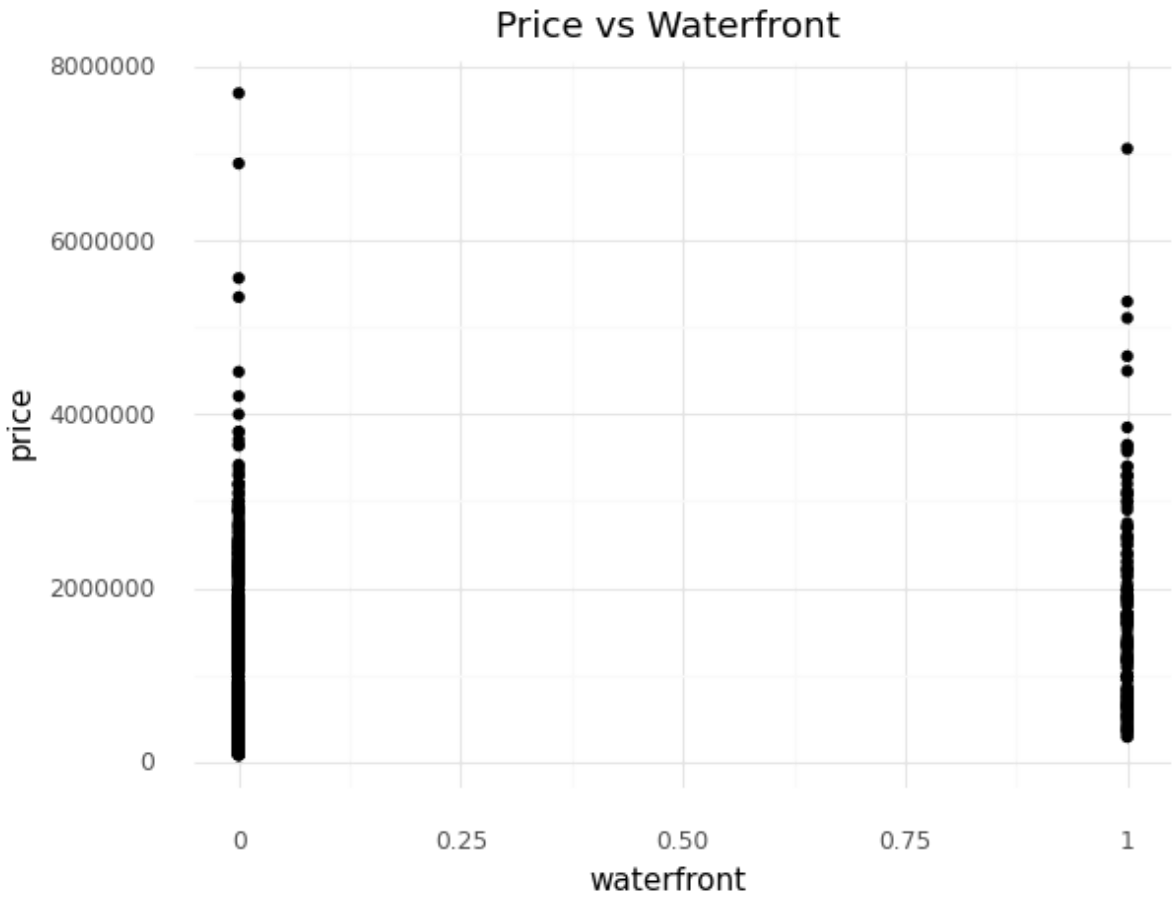
In [221]:
```
(ggplot(house1, aes(x = "sqft_living", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Square Feet of Living Space"))
```



Out[221]: <ggplot: (8771733106532)>

This scatterplot shows the price of the house based on the square feet of the living area.

In [222]:
```
(ggplot(house1, aes(x = "waterfront", y = "price"))+
geom_point()+
theme_minimal()+
ggtitle("Price vs Waterfront"))
```



Out[222]: <ggplot: (8771733850528)>

This scatterplot shows the price of the house based on if the house has a waterfront view or not.

The question that we had to answer was which variable has the greatest effect on the sales price of a house in King County. We can see that when we print out the coeffient weights of each predictor variable. When looking at the bar graph of all the coefficent weights, we can see the differences between. The greatest effect that we can see is the variable waterfront. The value came out to be around 775866, which came out to be one of the very few variables that had a positive coefficent weight. This means that when a house had a waterfront, it would expect the house price to increase by 775866. Also, we can see all the features individually compared to the price of the house. Usually, when it had more of that feature or a higher quality, the price of the house increased. In the waterfront graph, we see that there is a smaller range in price compared to not having a waterfront.

1. Based on the house, can we cluster based on price, bedrooms, bathrooms and floors?

In [287]:
```
feature = ["price","bedrooms", "bathrooms", "floors"]

X = house[feature]

z = StandardScaler()

X[feature] = z.fit_transform(X)

house.head()
```

Out[287]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	20141013T000000	221900.0	-0.398737	-1.447464	-0.979835	-0.228321	-0.91542
1	6414100192	20141209T000000	538000.0	-0.398737	0.175607	0.533634	-0.189885	0.9365(
2	5631500400	20150225T000000	180000.0	-1.473959	-1.447464	-1.426254	-0.123298	-0.91542
3	2487200875	20141209T000000	604000.0	0.676485	1.149449	-0.130550	-0.244014	-0.91542
4	1954400510	20150218T000000	510000.0	-0.398737	-0.149007	-0.435422	-0.169653	-0.91542

5 rows × 21 columns

In [296]:
```
EM = GaussianMixture(n_components = 3)

EM.fit(X)

cluster = EM.predict(X)
```

In [297]:
```
print("SILHOUETTE:", silhouette_score(X, cluster))

X["cluster"] = cluster
```

```
SILHOUETTE: 0.4164498948714565
```

In [302]:
```
(ggplot(X, aes(x = "bedrooms", y = "price", color = "factor(cluster)"))
+
 geom_point() +
 ggtitle("Price and Bedrooms"))
```

Out[302]: <ggplot: (8771751575703)>

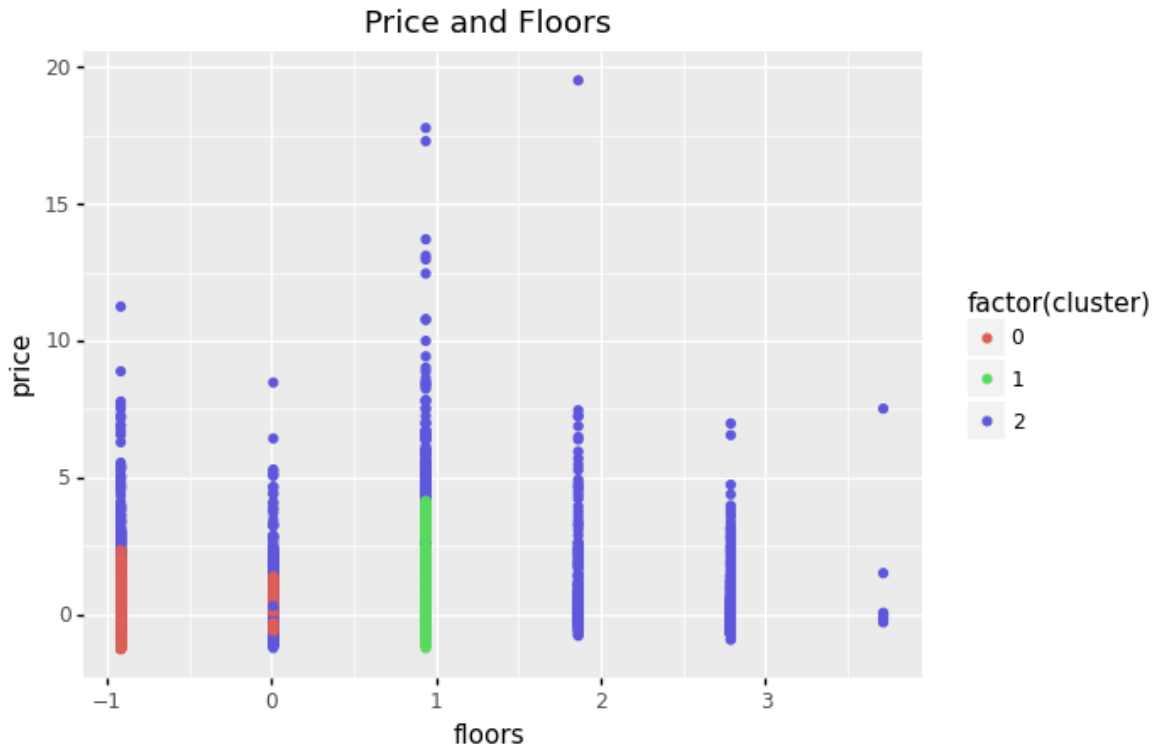
This scatterplot shows the 3 different cluster groups of homes based on the price of the house and the number of bedrooms. The blue cluster represents a house that has relatively more bedrooms. The green cluster represents a house with an average amount of bedrooms. The red cluster represents houses with below average bedrooms.

In [303]:
```
(ggplot(X, aes(x = "bathrooms", y = "price", color = "factor(cluster)"))
+ geom_point()+
ggtitle("Price and Bathrooms"))
```

Out[303]: <ggplot: (8771759285745)>

This scatterplot shows the 3 different cluster groups of homes based on the price of the house and the number of bathrooms. The blue cluster represents a house that has relatively more bathrooms. The green cluster represents a house with an average amount of bathrooms. The red cluster represents houses with below average bathrooms.

In [304]:
```
(ggplot(X, aes(x = "floors", y = "price", color = "factor(cluster)")) +
geom_point() +
 ggtitle("Price and Floors"))
```



Out[304]: <ggplot: (8771760239050)>

This scatterplot shows the 3 different cluster groups of homes based on the price of the house and the number of floors.

We are trying to see if we can cluster houses based on the price of the house, bedrooms, bathrooms and number of floors. As we can see, we are using 3 different clusters to group the houses. This gives the best silhoutte score of 0.4164498948714565. This is a pretty good silhoutte score as it shows pretty good cohesion and separation between the clusters. Although, it came out with a good score, the graphs above tell a different story than the calculated number. When looking at the graphs price vs. bedrooms, bathrooms, and floors, you can't really group them into cluster. In a cluster you want to able to recognize the group and be able to describe it. You can't really describe each group because they are so close together with data points from different clusters overlapping. This shows that we can't really cluster these variables together in order to group different houses.

1. Exclusively using the variables Bedroom, Bathroom, and Sqft_lv, how accurately can we predict the price of a house compared to using all the other variables in the dataset?

In [223]:
```
kf = KFold (n_splits = 100)
```

In [224]:
```
predictors = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floor
s", "waterfront", "condition", "grade"]
X = house[predictors]
y = house["price"]
```

In [225]:
```
LR = LinearRegression()
r2 = []
```

In [226]:
```
for train,test in kf.split(X):
    X_train = X.iloc[train]
    X_test = X.iloc[test]
    y_train = y[train]
    y_test = y[test]

    model = LR.fit(X_train, y_train)

    r2.append(r2_score(y_test,model.predict(X_test)))
```

In [227]:
```
np.mean(r2)
```

Out[227]:
```
0.575414103810196
```

In [228]:
```
coefficients = pd.DataFrame({"Coef":LR.coef_, "Name": predictors})

coefficients = coefficients.append({"Coef": LR.intercept_, "Name": "inte
rcept"}, ignore_index = True)

coefficients
```

Out[228]:

	Coef	Name
0	-35192.258011	bedrooms
1	-9550.403207	bathrooms
2	192244.157989	sqft_living
3	-13679.179094	sqft_lot
4	-11140.174024	floors
5	775866.564204	waterfront
6	58737.445190	condition
7	105141.902813	grade
8	-471081.132921	intercept

In [229]:
```
predictor = ["bedrooms", "bathrooms", "sqft_living"]
X = house[predictor]
y = house["price"]
```

In [188]:
```
lr = LinearRegression()
mse = []
r1 = []
```

In [189]:
```
for train,test in kf.split(X):
    X_train = X.iloc[train]
    X_test = X.iloc[test]
    y_train = y[train]
    y_test = y[test]

    model = lr.fit(X_train, y_train)

    r1.append(r2_score(y_test,model.predict(X_test)))
```

In [190]:
```
np.mean(r1)
```

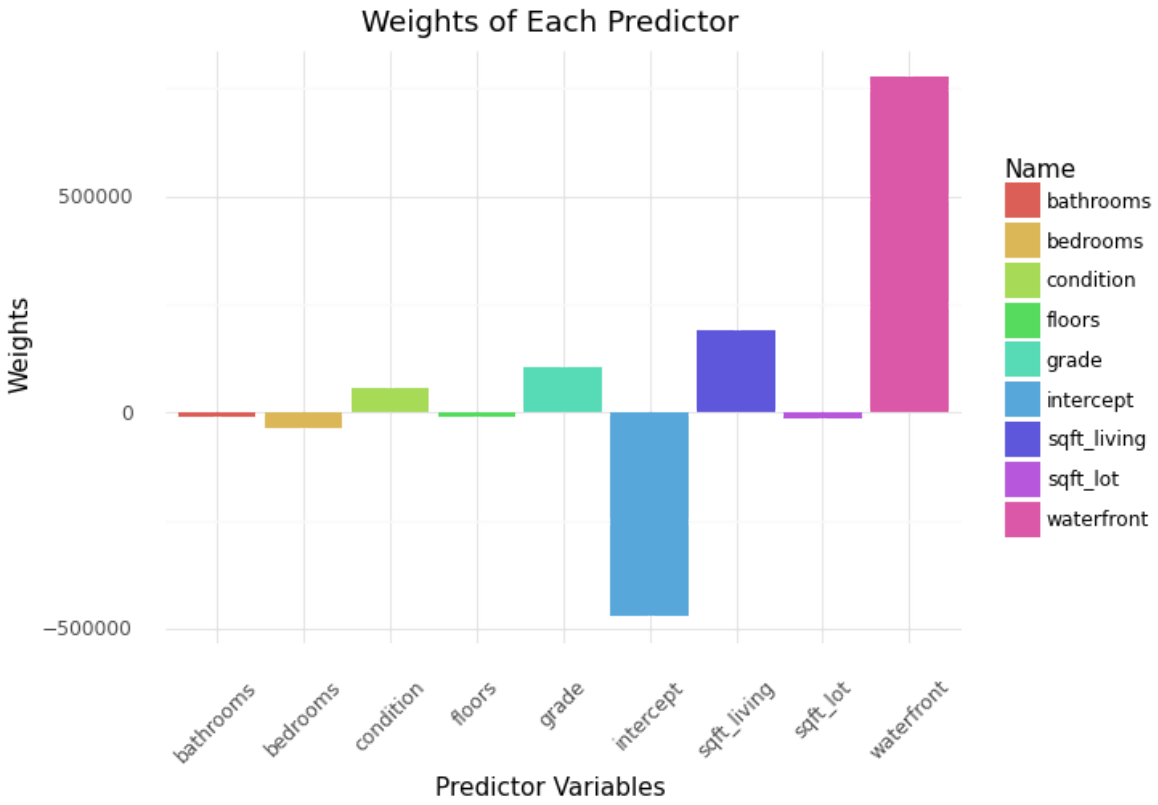
Out[190]:
```
0.48720662837631906
```

In [198]:
```
coefficient = pd.DataFrame({"Coef":lr.coef_, "Name": predictor})

coefficient = coefficient.append({"Coef": lr.intercept_, "Name": "interc
ept"}, ignore_index = True)

coefficient
```

Out[198]:

	Coef	Name
0	-53487.602507	bedrooms
1	5323.878735	bathrooms
2	284348.566599	sqft_living
3	540074.411425	intercept

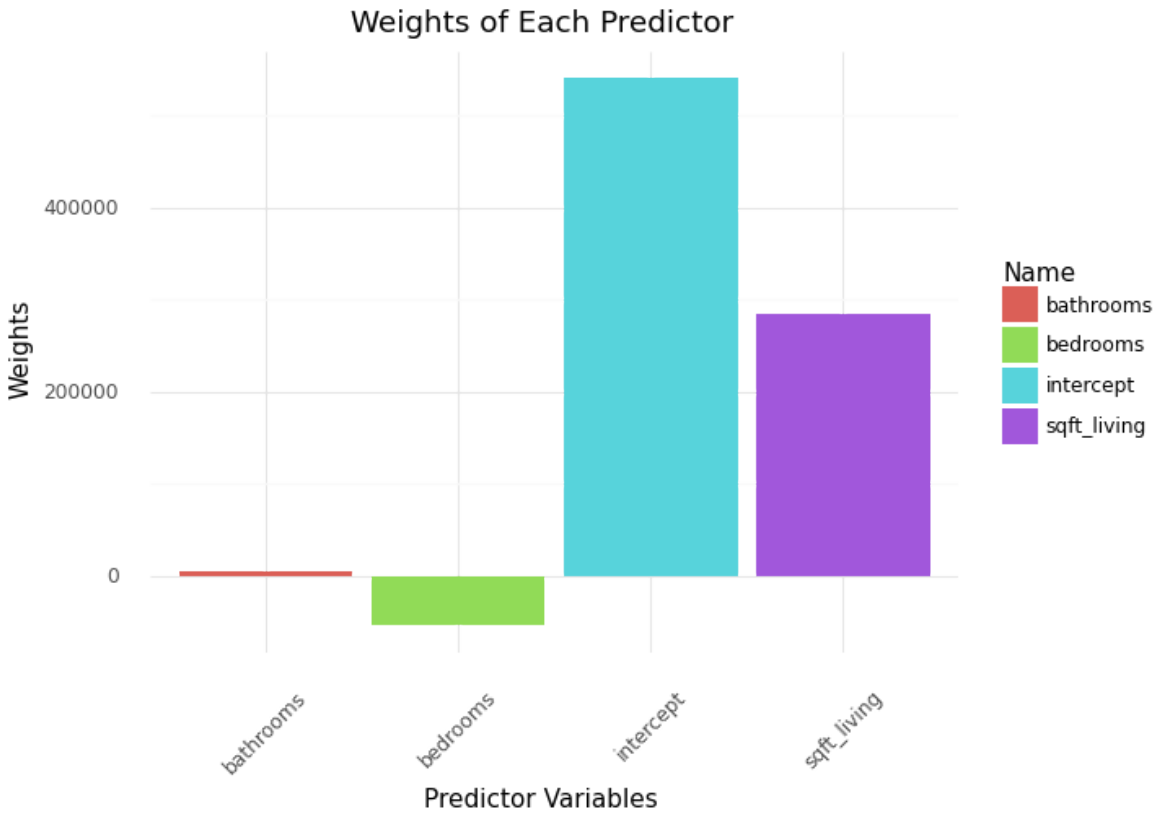
In [232]:
```
(ggplot(coefficients, aes(x = "Name", y = "Coef", fill = "Name")) +
 stat_summary(geom = "bar") + ggtitle("Weights of Each Predictor") +
 xlab("Predictor Variables") +
 ylab("Weights") +
 theme_minimal() +
 theme(axis_text_x = element_text(angle = 45)))
```



Out[232]: <ggplot: (8771741746056)>

This graph shows the coeffienct weights of all available predictor variables for a linear regression model predicting price of a house.

In [233]:
```
(ggplot(coefficient, aes(x = "Name", y = "Coef", fill = "Name")) +
 stat_summary(geom = "bar") + ggtitle("Weights of Each Predictor") +
 xlab("Predictor Variables") +
 ylab("Weights") +
 theme_minimal() +
 theme(axis_text_x = element_text(angle = 45)))
```



Out[233]: <ggplot: (8771733852365)>

This graph shows the coeffienct weights of the predictor variables, exceusively using bathrooms, bedrooms and square feet living, for a linear regression model predicting price of a house.

We are trying to see if we can accurately predict a house price based on the amount of bedrooms, bathrooms and square feet living space compared to a model that is using the amount of bedrooms, bathrooms, floors, square feet of lot, square feet of living space, condition of the house, grade of the house and if it has waterfront or not. We outputted the r2 squares of both models. R2 is the proportion of the variance in the dependent variable (price of the house) that is explained from the independent variables (variables used to predict the price of the house).As we see using all of the variables, we get a r2 score of 0.575414103810196. This means that 57.41% of the house prices is explained by using all of the variables. Using only the 3 variables, we get an r2 of 0.48720662837631906. This means that 48.72% of the house prices is explained by using only bedrooms, bathrooms and sqaure feet of living space. Ultimately, this shows that using more variables is more accurate in predicting a house price. This makes sense because there is usually more features of a house than just bedrooms, bathrooms and square feet. A house might have other features that increase the price of the house or might be located in a particularly nice area.

1. If we are given future data, can we accurately predict the price based off of the model created?

In [234]:
```
house = pd.read_csv("https://raw.githubusercontent.com/JasonCabardo/kc_house_data/main/kc_house_data.csv")

zScore = StandardScaler()

house.head()
```

Out[234]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [235]:
```
zScore.fit(house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]])

z = zScore.transform(house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]])

house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]] = z

house.head()
```

Out[235]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	20141013T000000	221900.0	-0.398737	-1.447464	-0.979835	-0.228321	-0.91542
1	6414100192	20141209T000000	538000.0	-0.398737	0.175607	0.533634	-0.189885	0.9365(
2	5631500400	20150225T000000	180000.0	-1.473959	-1.447464	-1.426254	-0.123298	-0.91542
3	2487200875	20141209T000000	604000.0	0.676485	1.149449	-0.130550	-0.244014	-0.91542
4	1954400510	20150218T000000	510000.0	-0.398737	-0.149007	-0.435422	-0.169653	-0.91542

5 rows × 21 columns

In [236]:
```
kf = KFold(n_splits = 100)
```

In [237]:
```
predictors = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floor
s", "waterfront", "condition", "grade"]
x = house[predictors]
y = house["price"]

lr = LinearRegression()
mse = []
r2 = []
```

In [238]:
```
for train,test in kf.split(x):
    x_train = x.iloc[train]
    x_test = x.iloc[test]
    y_train = y[train]
    y_test = y[test]

    #model
    model = lr.fit(x_train, y_train)

    #record accuracy
    mse.append(mean_squared_error(y_test, model.predict(x_test)))
    r2.append(r2_score(y_test, model.predict(x_test)))
```

In [239]:
```
np.mean(mse)
```

Out[239]:
```
55225782353.21309
```

In [240]:
```
np.mean(r2)
```

Out[240]:
```
0.5753881324676033
```

In [241]:
```
coefficients = pd.DataFrame({"Coef":model.coef_, "Name": predictors})

coefficients = coefficients.append({"Coef": model.intercept_, "Name": "i
ntercept"}, ignore_index = True)

coefficients
```

Out[241]:

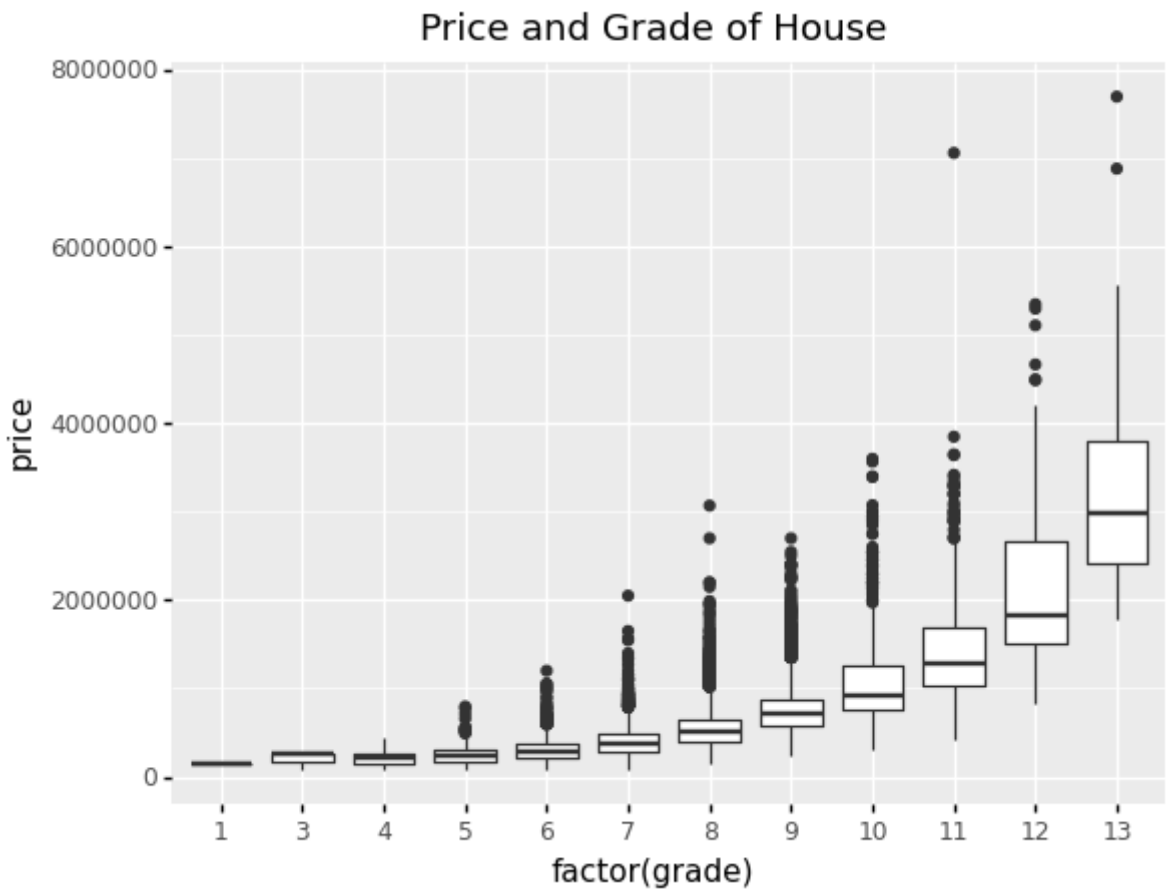
	Coef	Name
0	-35162.066309	bedrooms
1	-9535.518615	bathrooms
2	192121.499103	sqft_living
3	-13666.848416	sqft_lot
4	-11128.335198	floors
5	775558.315284	waterfront
6	58680.089010	condition
7	105054.789596	grade
8	-470309.671345	intercept

In [342]:
```
(ggplot(house, aes(x = "sqft_living", y = "price")) + geom_point() + sta
t_smooth(method = "lm")+
ggtitle("Price and Square Feet Living"))
```

Out[342]: <ggplot: (8771734048615)>

This graph shows us a relationship between the price of a house and its square foot of the living area. In the front area of the graph we can see that it follows the trend line meaning that price increases with the square foot living area increase. It relatively follows the trend through the whole graph although houses with 6+ square foot living areas tended be higher in price than the trend line predicts.

In [341]:
```
(ggplot(house, aes(x = "factor(grade)", y = "price")) + geom_boxplot() +
ggtitle("Price and Grade of House"))
```



Out[341]: <ggplot: (8771758266384)>

As you can tell in the graph above the grade has an effect on how much a house is being sold for in King County, Seattle. As the grade begins at 0 and reaches a grade of 5 we can see the price begin to inflate slowly. Once we reach the grades that are over 5 and increase to around 13, the price is significantly increasing. Each time the grade increases by one the mean price of the houses for that grade also increases.

In this model we are evaluating if the model created is accurate and could be used to predict future data sets. We created a model that used the variables "bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "waterfront", "condition", and "grade" to see the effect they had on the price of each house. For this model we outputted the MSE along with the R2 to test the effectiveness and error. Although we got a relatively good R2 of 0.575388 meaning that 57.5% of house prices are determined by these predictors, we also got a large MSE of 55225782353.21309. The large MSE we got tells us that there was too much error and the model was underfitted meaning it didnt predict the train seet accuratly. Because we recieved so much error and the model was underfitted it is evident that it would not be useful in predicting other future data sets.

15.Does the square foot of the basement and above ground of a house directly affect the price of a house?

In [244]:
```
house = pd.read_csv("https://raw.githubusercontent.com/JasonCabardo/kc_house_data/main/kc_house_data.csv")

house.head()
```

Out[244]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [336]:
```
features = ["price", "sqft_above", "sqft_basement"]

X = house[features]

z = StandardScaler()

X[features] = z.fit_transform(X)

house.head()
```

Out[336]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

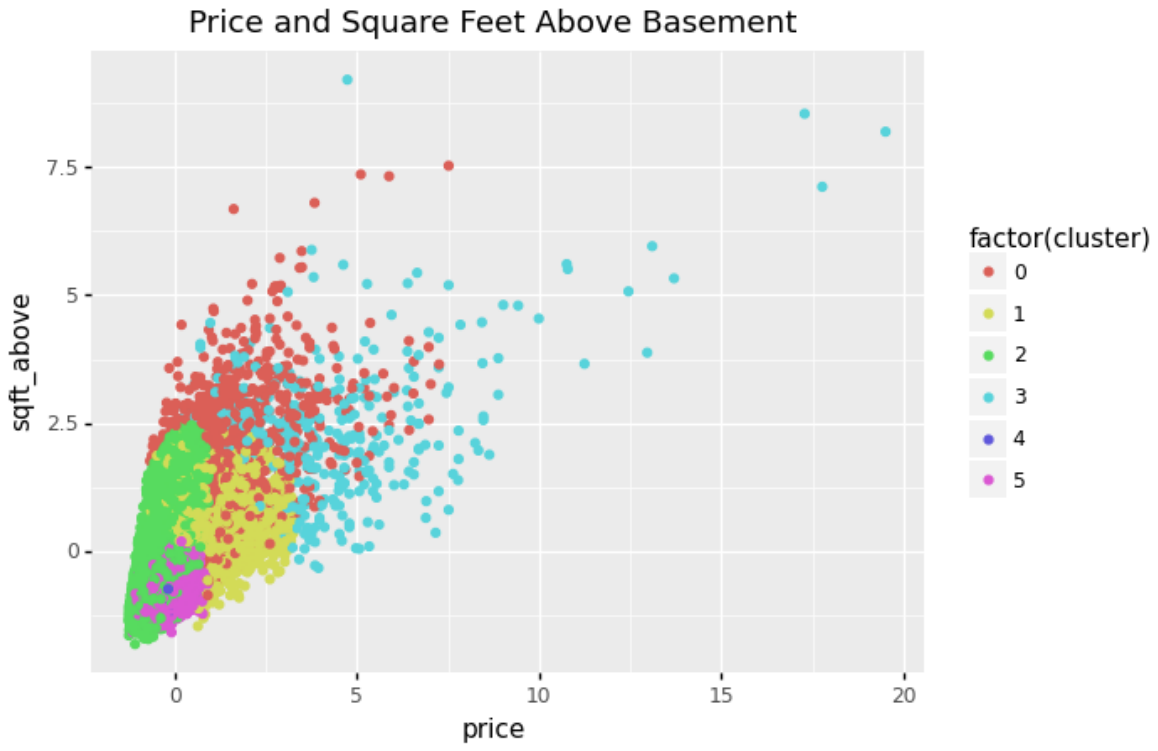
5 rows × 21 columns

In [337]:
```
EM = GaussianMixture(n_components = 6)

EM.fit(X)

cluster = EM.predict(X)
```

In [274]:
```
print("SILHOUETTE:", silhouette_score(X, cluster))

X["cluster"] = cluster
```

```
SILHOUETTE: 0.5961175707151629
```

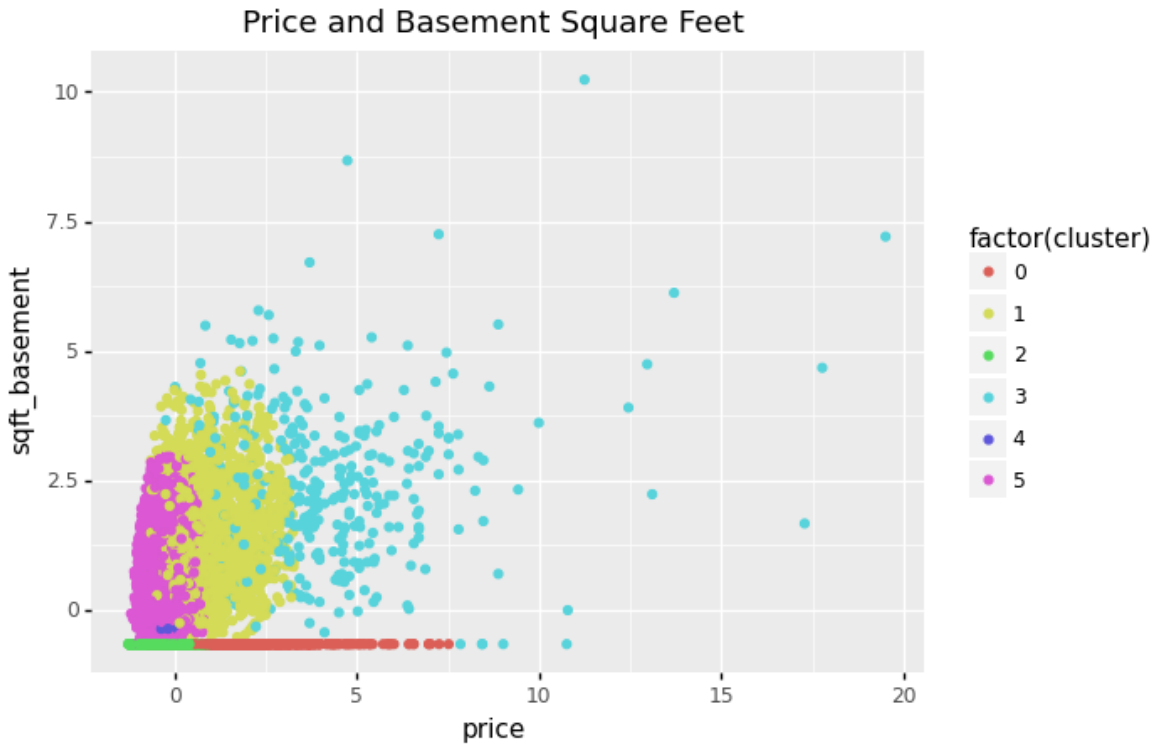
In [339]:
```
(ggplot(X, aes(x = "price", y = "sqft_above", color = "factor(cluster)"
)) + geom_point() +
 ggtitle("Price and Square Feet Above Basement"))
```



Out[339]: <ggplot: (8771761289328)>

This scatter plot shows the different clusters based on price and their square foot above ground. The red cluster represents houses that are priced lower because they have less square feet above ground. The dark purple cluster at the top right represents houses that have relatively large square feet above ground but also could have other factors contributing to the higher price.

In [340]:
```
(ggplot(X, aes(x = "price", y = "sqft_basement", color = "factor(cluste
r)")) + geom_point()+
ggtitle("Price and Basement Square Feet"))
```



Out[340]: <ggplot: (8771732047273)>

This scatter plot shows the different clusters based on price and the square feet of the basement. The red cluster represents houses with no basement that are valued at low prices probably because they are also small given no basements. The blue cluster represents houses that have basements, although relatively small, that area also smaller houses with overall lower prices.

In this model we were trying to find out if we the square foot of the above ground and the square foot of the basement had an affect on the price of a house. When creating the clusters we found that we got a silhouette score of 0.5961175707151629 which is relatively high given the scale is -1 to 1. We chose to use six clusters because it gave us the highest silhouette score meaning our clusters were cohesive and separate. Once producing the graphs we can see that although the clusters are close they have little overlap and confusion among the clusters. In the graphs we can tell that houses that have less square feet for either the basement or above ground also have lower prices. Similarly we can see that the houses that have larger area above ground and in the basement have higher prices.

10.What effect does waterfront pose on sales price?

In [253]:
```
house = pd.read_csv("https://raw.githubusercontent.com/JasonCabardo/kc_h
ouse_data/main/kc_house_data.csv")

zScore = StandardScaler()
```

In [254]:
```
zScore.fit(house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]])

z = zScore.transform(house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]])

house[["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]] = z

house.head()
```

Out[254]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	20141013T000000	221900.0	-0.398737	-1.447464	-0.979835	-0.228321	-0.91542
1	6414100192	20141209T000000	538000.0	-0.398737	0.175607	0.533634	-0.189885	0.9365(
2	5631500400	20150225T000000	180000.0	-1.473959	-1.447464	-1.426254	-0.123298	-0.91542
3	2487200875	20141209T000000	604000.0	0.676485	1.149449	-0.130550	-0.244014	-0.91542
4	1954400510	20150218T000000	510000.0	-0.398737	-0.149007	-0.435422	-0.169653	-0.91542

5 rows × 21 columns

In [255]:
```
kf = KFold(n_splits = 100)
```

In [256]:
```
predictors = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "waterfront", "condition", "grade"]
x = house[predictors]
y = house["price"]

lr = LinearRegression()
mse = []
r2 = []
```

In [257]:
```
for train,test in kf.split(x):
    x_train = x.iloc[train]
    x_test = x.iloc[test]
    y_train = y[train]
    y_test = y[test]

    #model
    model = lr.fit(x_train, y_train)

    #record accuracy
    mse.append(mean_squared_error(y_test, model.predict(x_test)))
    r2.append(r2_score(y_test, model.predict(x_test)))
```

In [258]:
```
np.mean(mse)
```

Out[258]: 55225782353.21309

In [259]:
```
np.mean(r2)
```

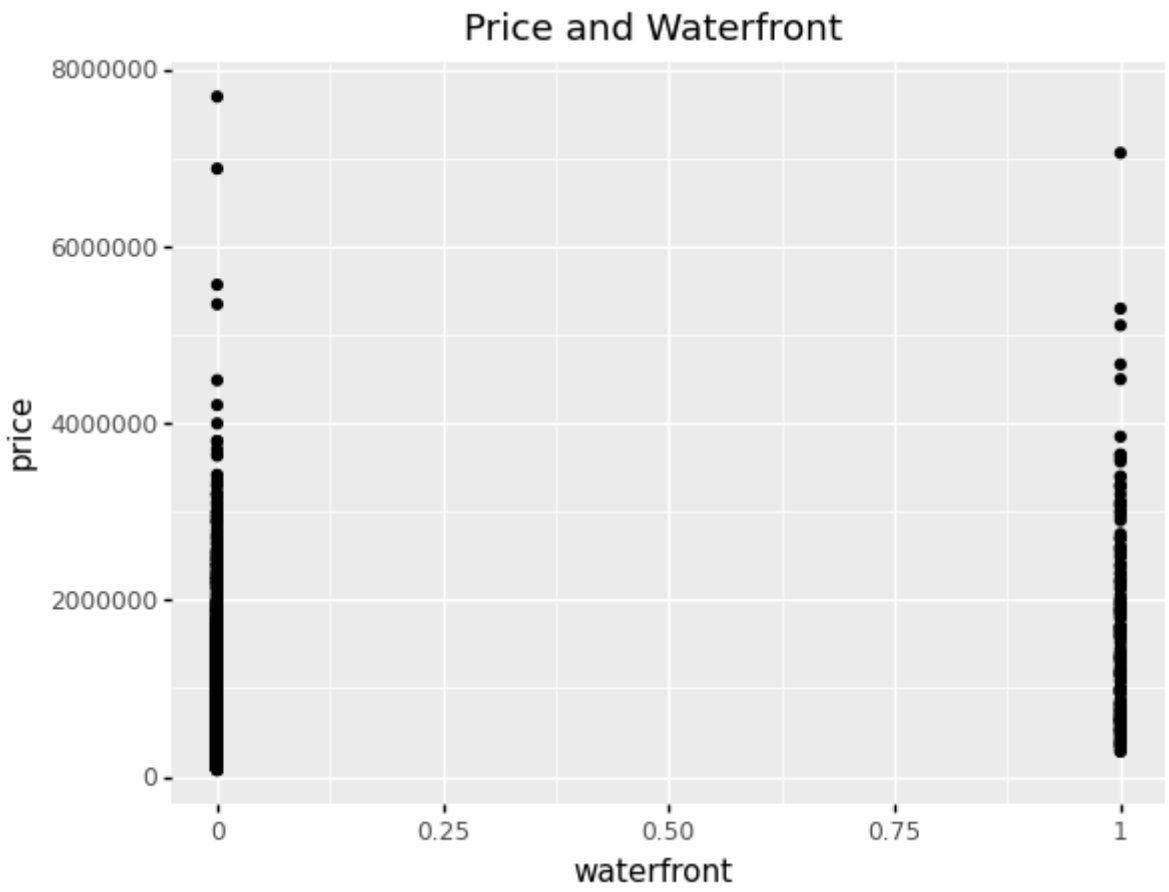
Out[259]:
```
0.5753881324676033
```

In [260]:
```
coefficients = pd.DataFrame({"Coef":model.coef_, "Name": predictors})

coefficients = coefficients.append({"Coef": model.intercept_, "Name": "i
ntercept"}, ignore_index = True)

coefficients
```

Out[260]:

	Coef	Name
0	-35162.066309	bedrooms
1	-9535.518615	bathrooms
2	192121.499103	sqft_living
3	-13666.848416	sqft_lot
4	-11128.335198	floors
5	775558.315284	waterfront
6	58680.089010	condition
7	105054.789596	grade
8	-470309.671345	intercept

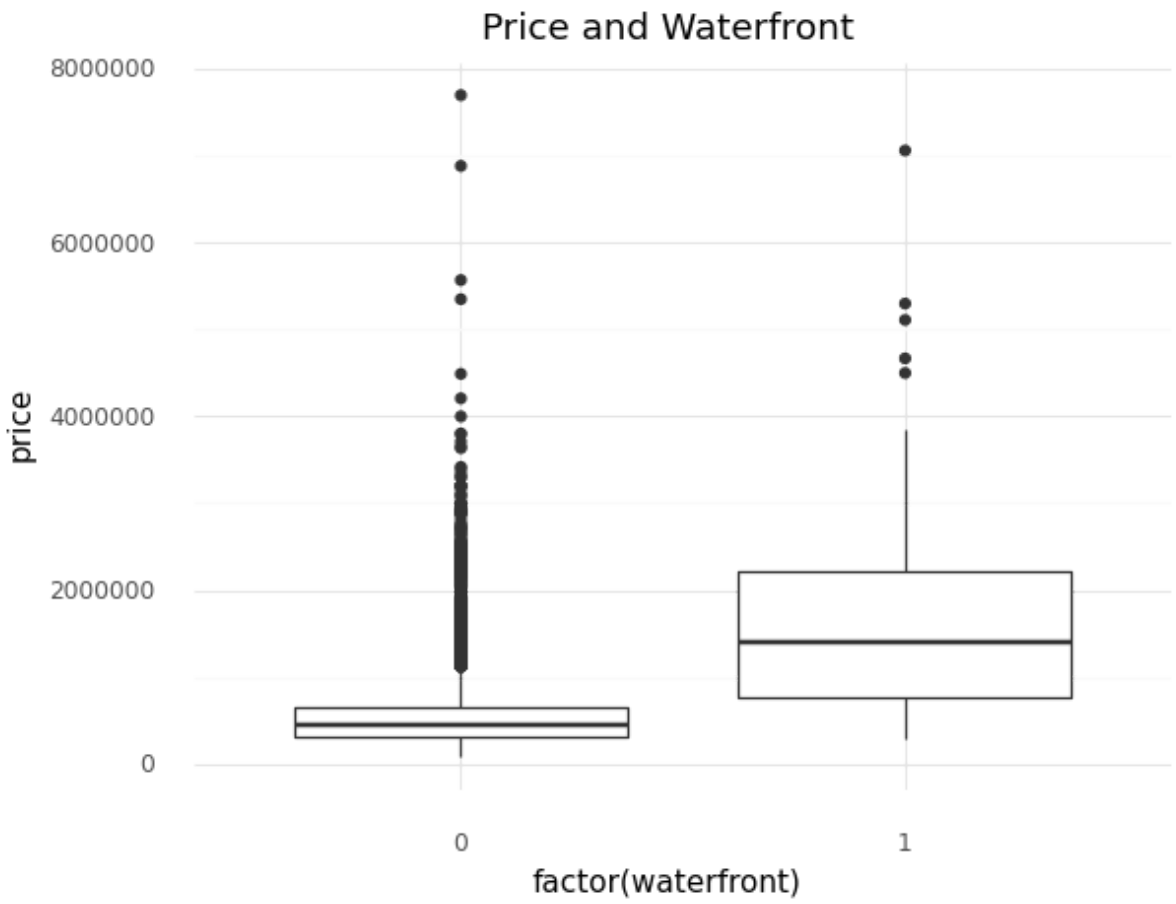
In [332]:
```
(ggplot(house, aes(x = "waterfront", y = "price")) + geom_point()+
ggtitle("Price and Waterfront"))
```



Out[332]: <ggplot: (8771757412846)>

This graph shows that there might not be a correlation between the waterfront of a house and the price of the house. It seems as though the prices are relatively similar between the waterfront with the houses being on the water having more density in prices closer to 200,000 dollars.

In [331]:
```
(ggplot(house, aes(x = "factor(waterfront)", y = "price")) + geom_boxplo
t() + theme_minimal()+
ggtitle("Price and Waterfront"))
```



Out[331]: <ggplot: (8771736115833)>

This graph shows a better evaluation of the breakdown of the prices for each house. Waterfront houses have a mean price closer to 175,000 dollars and non-waterfront houses have a mean price closer to 50,000 dollars.

In this model we are evaluating if waterfront and nonwaterfront houses have an affect on the price of the home. We created a model that used the variables "bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "waterfront", "condition", and "grade" to see the effect they had on the price of each house. For this model we outputted the MSE along with the R2 to test the effectiveness and error. Although we got a relatively good R2 of 0.5753881324676033 meaning that 57.5% of house prices are determined by these predictors, we also got a large MSE of 55225782353.21309. The large MSE we got tells us that there was too much error and the model was underfitted meaning it didnt predict the train seet accuratly. Although the graphs showed the difference in average price for houses on the water and not on the water, we recieved so much error and the model was underfitted telling us that the model was not necessarily as accurate as we assumed.

In [305]:
```
house = pd.read_csv("https://raw.githubusercontent.com/JasonCabardo/kc_house_data/main/kc_house_data.csv")
house.head()
```

Out[305]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [306]:
```
house.shape
```

Out[306]:
```
(21613, 21)
```

1. How accurate are the variables in predicting the sales price of a house in King County?

In [307]:
```
predictors = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "waterfront", "condition", "grade"]

X_train, X_test, y_train, y_test = train_test_split(house[predictors], house["price"], test_size=0.3)

#continuous variables only
#not including categorical variables of "waterfront", "condition", "grade"
continuous = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors"]

zscore = StandardScaler()
zscore.fit(X_train[continuous])
X_train[continuous] = zscore.transform(X_train[continuous])
X_test[continuous] = zscore.transform(X_test[continuous])

Xz_train = X_train
Xz_test = X_test

LR_Model = LinearRegression()
LR_Model.fit(Xz_train,y_train)
```

Out[307]:
```
LinearRegression()
```

In [308]:
```
price_pred = LR_Model.predict(Xz_test)
price_pred[1:10]
```

Out[308]:
```
array([ 309519.31948767,  406513.15053821,  549768.00081539,
        704558.65867547, 1346957.20709364,  521167.87112714,
        370727.19169645,  675742.69855383,  493551.69600378])
```

In [309]:
```
coefficients = pd.DataFrame({"Coef":LR_Model.coef_,
                "Name": predictors})
coefficients = coefficients.append({"Coef": LR_Model.intercept_,
                     "Name": "intercept"}, ignore_index = True)
coefficients
```

Out[309]:

	Coef	Name
0	-33518.758681	bedrooms
1	-8883.970732	bathrooms
2	189761.780530	sqft_living
3	-11105.657653	sqft_lot
4	-10332.796752	floors
5	795149.750705	waterfront
6	60684.090739	condition
7	103450.715916	grade
8	-465452.980544	intercept

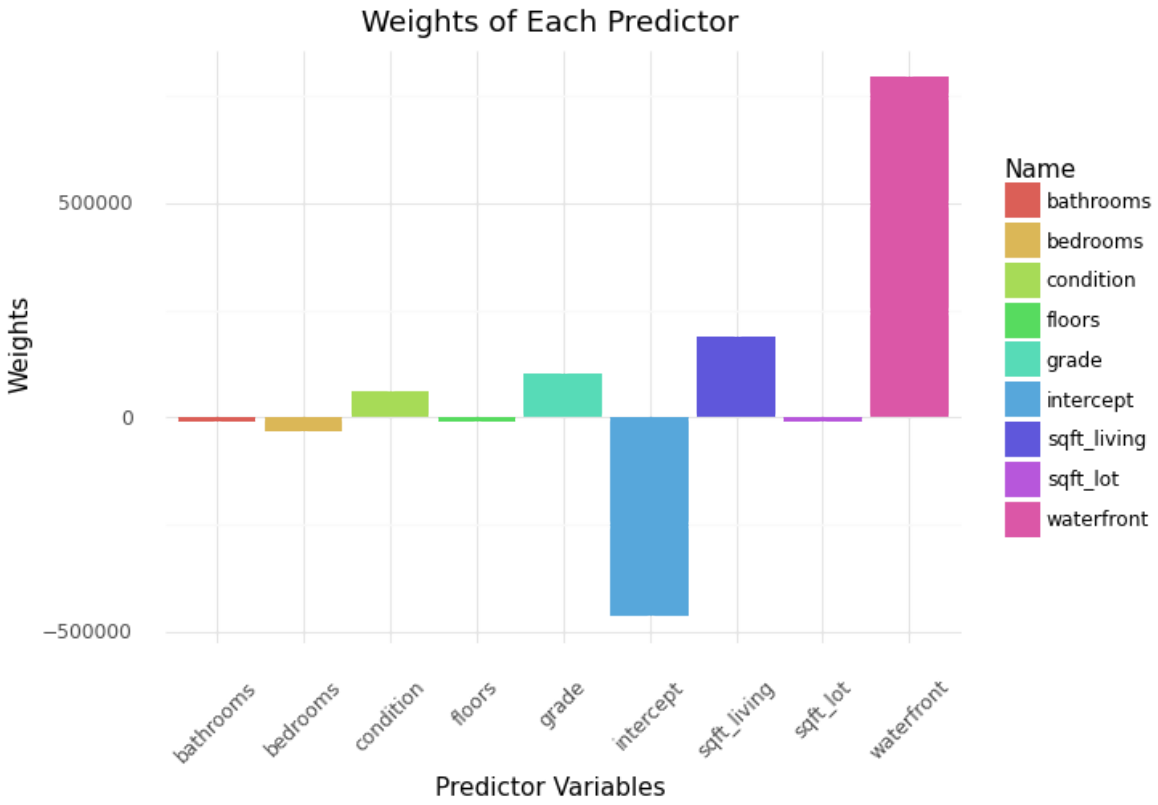
In [310]:
```
##This high mean_squared_error value indicates high bias in our model (u
nderfitting).
mean_squared_error(y_test, price_pred)
```

Out[310]:
```
57523446054.44221
```

In [311]:
```
r2_score(y_test, price_pred)
```

Out[311]:
```
0.5884174495869205
```

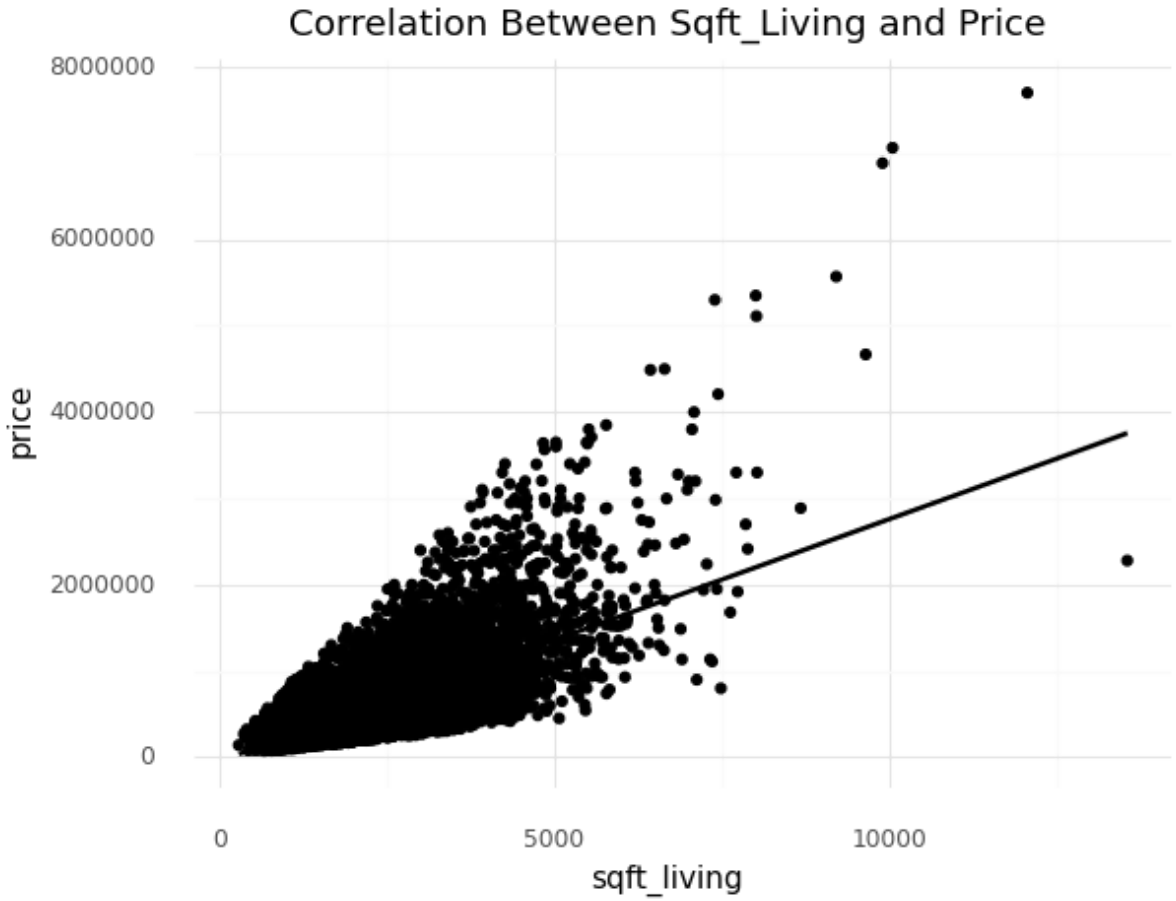
In [312]:
```
(ggplot(coefficients, aes(x = "Name", y = "Coef", fill = "Name")) +
                        stat_summary(geom = "bar") +
                        ggtitle("Weights of Each Predictor") +
                        xlab("Predictor Variables") +
                        ylab("Weights") +
                        theme_minimal() +
                        theme(axis_text_x = element_text(angle = 45)))
```



Out[312]: <ggplot: (8771473460351)>

This bar chart displays the variables' influence and weight in predicting house price.

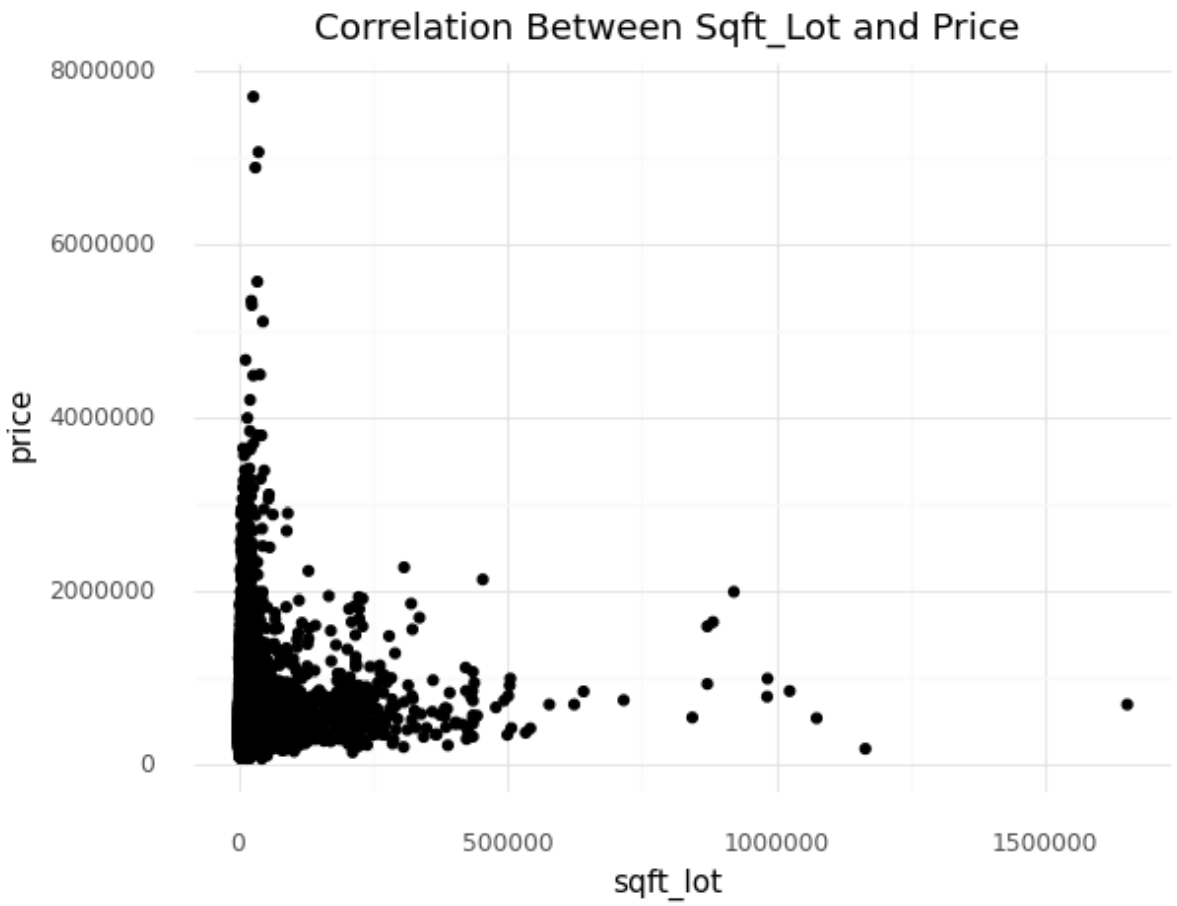
In [314]:
```
(ggplot(house, aes(x = "sqft_living", y = "price")) +
 geom_point() + theme_minimal() + stat_smooth(method = 'lm')+ ggtitle("C
orrelation Between Sqft_Living and Price"))
```



Out[314]: <ggplot: (8771475558623)>

This scatterplot shows the correlation between the predictor variable sqft_living and price.

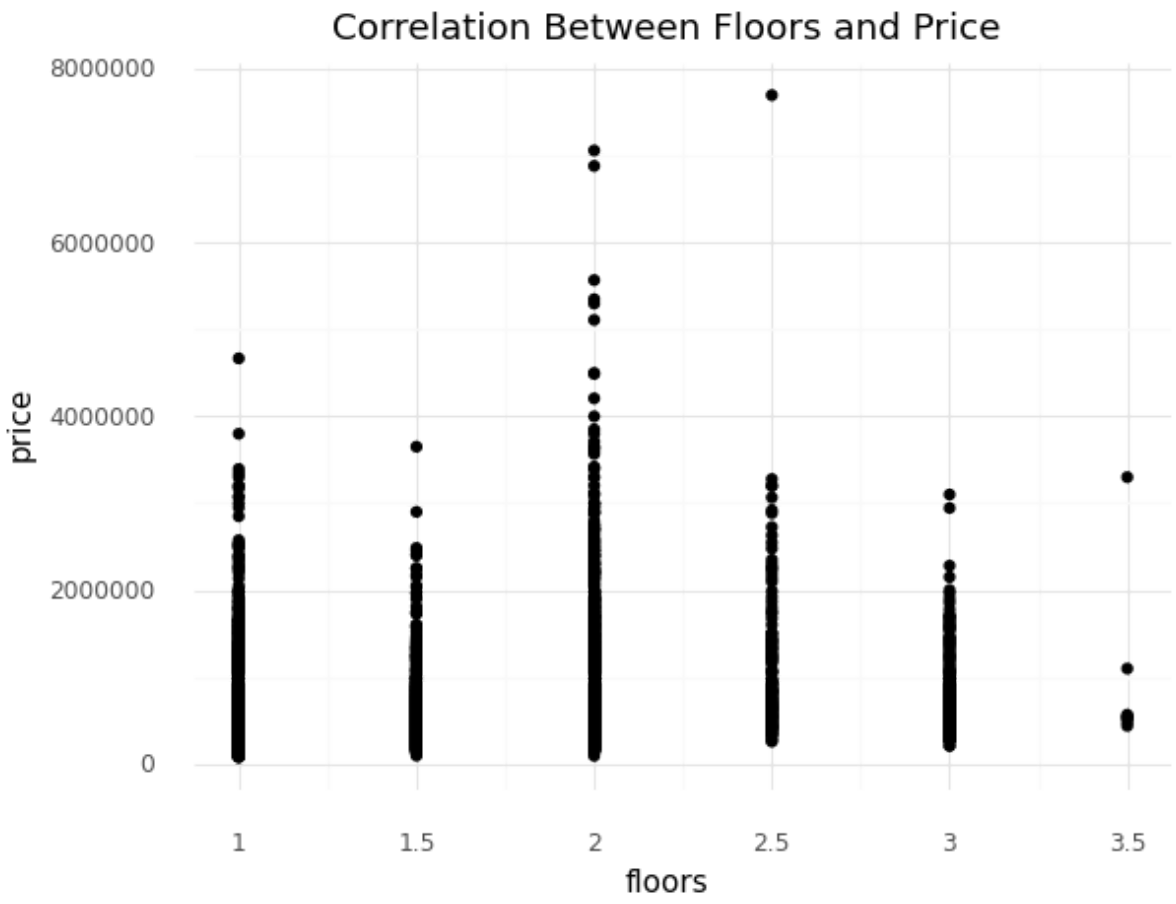
In [315]:
```
(ggplot(house, aes(x = "sqft_lot", y = "price")) +
 geom_point() + theme_minimal() + ggtitle("Correlation Between Sqft_Lot
 and Price"))
```



Out[315]: <ggplot: (8771474120473)>

This scatterplot shows the correlation between the predictor variable sqft_lot and price.

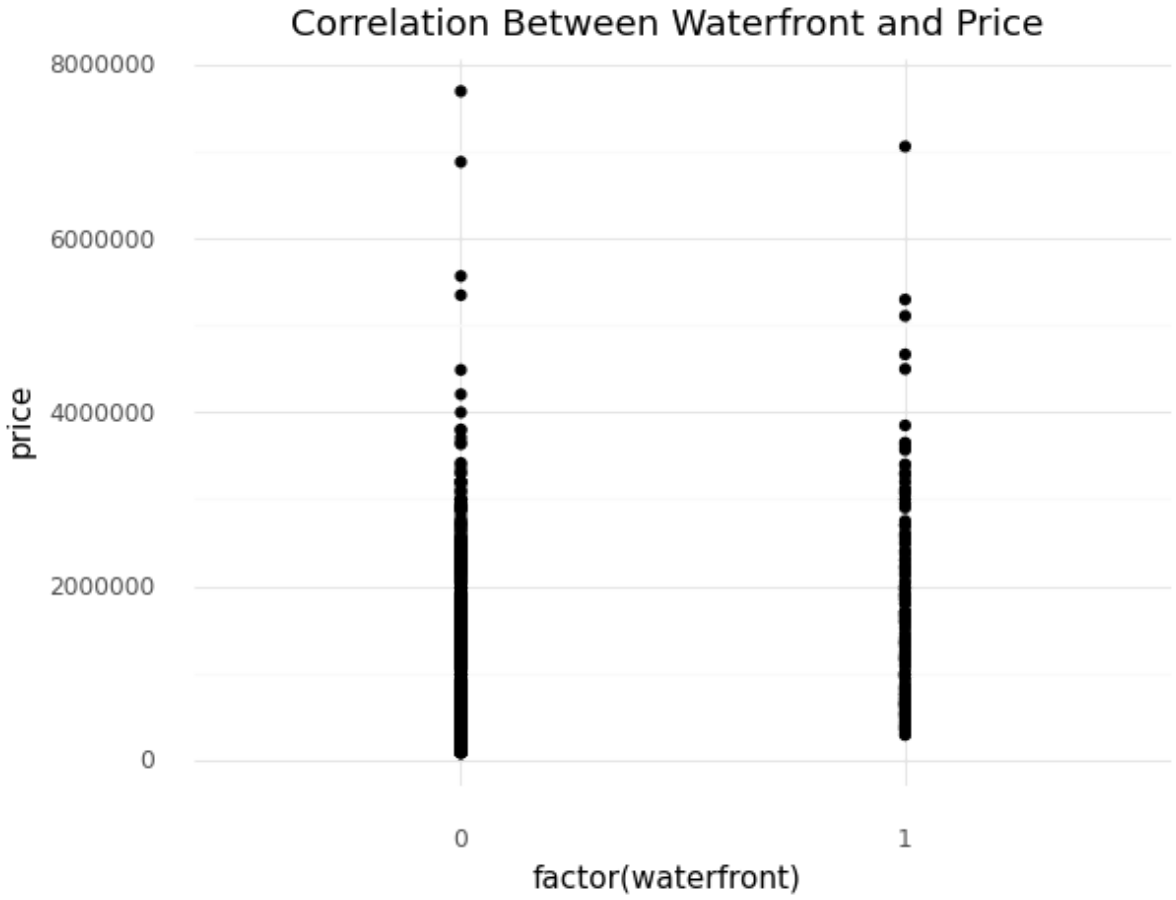
In [317]:
```
(ggplot(house, aes(x = "floors", y = "price")) +
 geom_point() + theme_minimal() + ggtitle("Correlation Between Floors an
d Price"))
```



Out[317]: <ggplot: (8771477541449)>

This scatterplot shows the correlation between the predictor variable floors and price.

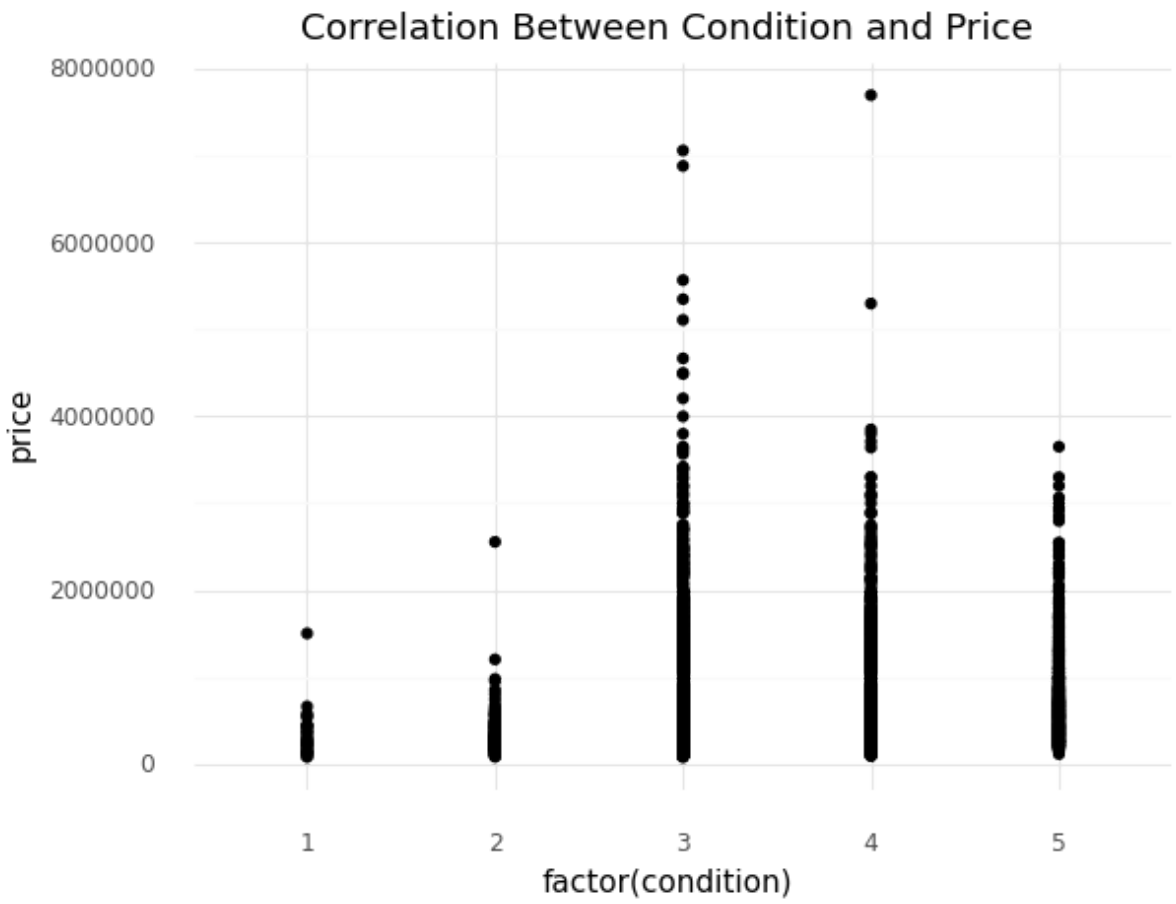
In [319]:
```
(ggplot(house, aes(x = "factor(waterfront)", y = "price")) +
 geom_point() + theme_minimal() + ggtitle("Correlation Between Waterfron
t and Price"))
```



Out[319]: <ggplot: (8771477542997)>

This scatterplot shows the correlation between the predictor variable waterfront and price.

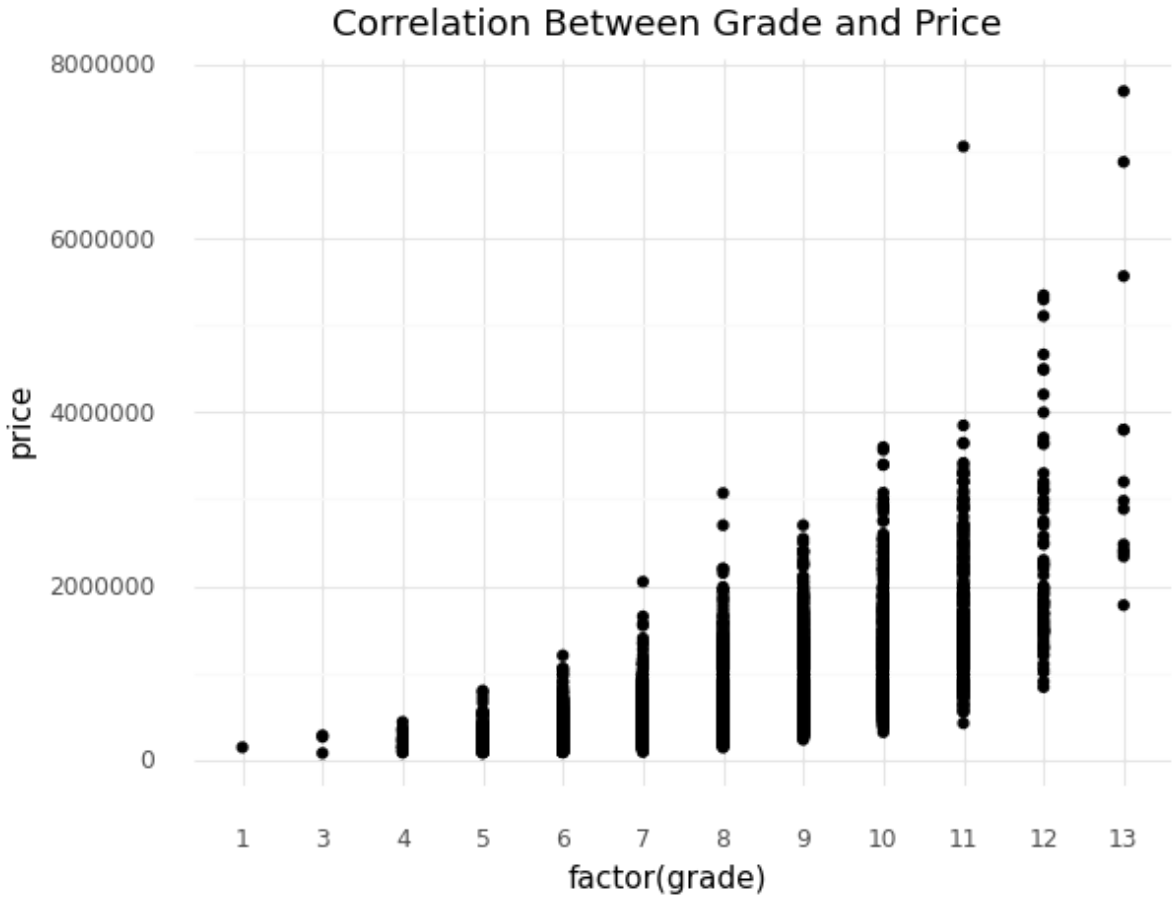
In [321]:
```
(ggplot(house, aes(x = "factor(condition)", y = "price")) +
 geom_point() + theme_minimal() + ggtitle("Correlation Between Condition
and Price"))
```



Out[321]: <ggplot: (8771473465323)>

This scatterplot shows the correlation between the predictor variable condition and price.

In [323]:
```
(ggplot(house, aes(x = "factor(grade)", y = "price")) +
 geom_point() + theme_minimal() + ggtitle("Correlation Between Grade and
Price"))
```



Out[323]: <ggplot: (8771474118536)>

This scatterplot shows the correlation between the predictor variable grade and price.

For this analysis, we will be determining first if the variable “Price” can be predicted using all the other variables in the dataset. The predictor variables we will be using are:"bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "waterfront", "condition", and "grade". To determine how accurate these variables are in predicting “Price”, we will first standardize solely the continuous variables by z-scoring them. Before z-scoring, each predictor variable will be measured in different metrics, making it really difficult to compare them to each other. By standardization, we are able to put these metrics that were originally on very different scales on a scale that is similar. Furthermore, z-scoring is what allows us to make the comparison between coefficients. As for the method of model validation, we will decide to use Train Test Split. Through this, we are taking the sample of data that we have now and we are making a split (70% to train the model and we use the remaining portion (30%) when we're done training the model to see how well it would do on unseen data). Working with this very large dataset, we are able to use the training data to fit the machine learning model, and then we use the test data to evaluate its performance.

However, my Linear Regression Model did not perform well based on my Mean Squared Error and r2 values. The average squared error between each data point and the predicted value for that data point is 57523446054.44221. By calculating the mean squared error (MSE) on the validation set, this really high value indicates a lesser likelihood of the model generalizing correctly from the training data, implying that the model is underfitting. With 21,613 rows Even with a moderately high r2 value of 0.58841744958692053 (58.84% of variation of the data was explained in the model), with 21,613 rows, the model can neither model the training data nor generalize to new data due to its significant MSE. Thus, the variables are not accurate in predicting the sales price of a house in King County as it is underfitting.

9.Can we cluster houses based off of sales price and the square feet of the living area/lot of the house?

In [325]:
```
features = ["price", "sqft_living","sqft_lot"]

X_gm = house[features]

z = StandardScaler()

X_gm[features] = z.fit_transform(X_gm)
```

In [326]:
```
EM = GaussianMixture(n_components = 3)

EM.fit(X_gm)
```

Out[326]:
```
GaussianMixture(n_components=3)
```

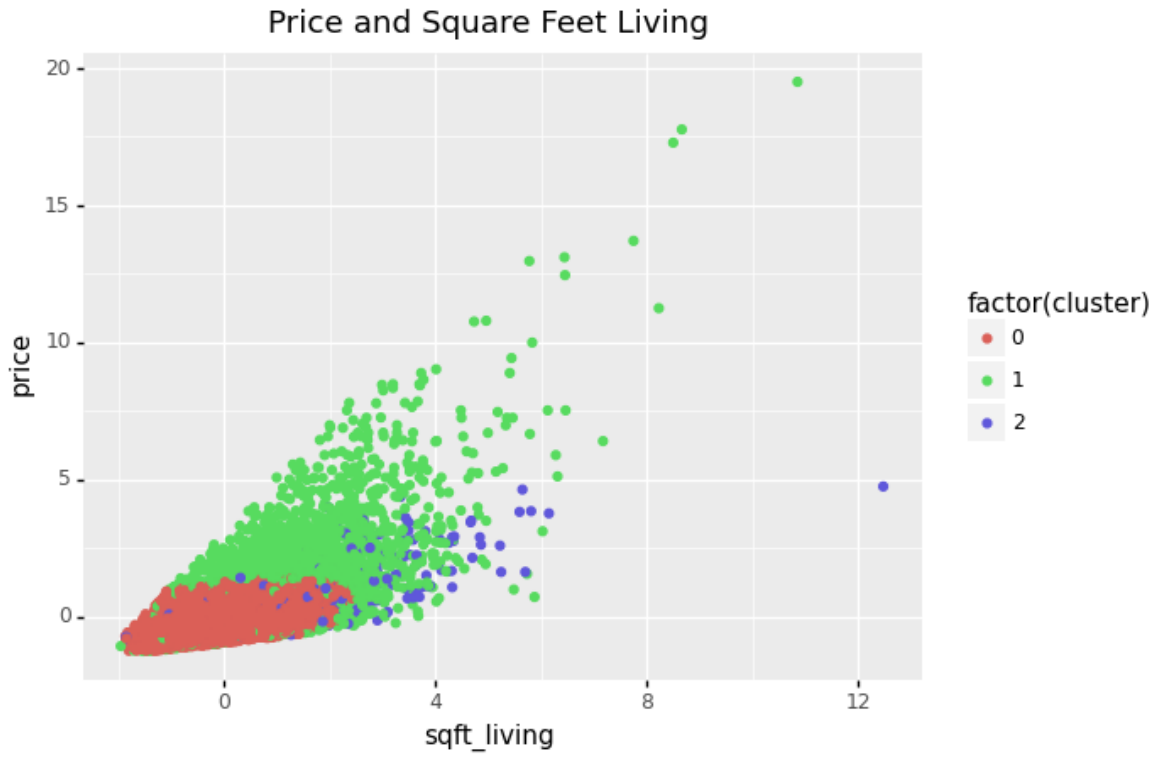
In [327]:
```
cluster = EM.predict(X_gm)
cluster
```

Out[327]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [328]:
```
silhouette_score(X_gm, cluster)
```

Out[328]:
```
0.42632621159027
```

In [330]:
```
X_gm["cluster"] = cluster

(ggplot(X_gm, aes(x = "sqft_living", y = "price", color = "factor(cluste
r)")) + geom_point() +
ggtitle("Price and Square Feet Living"))
```



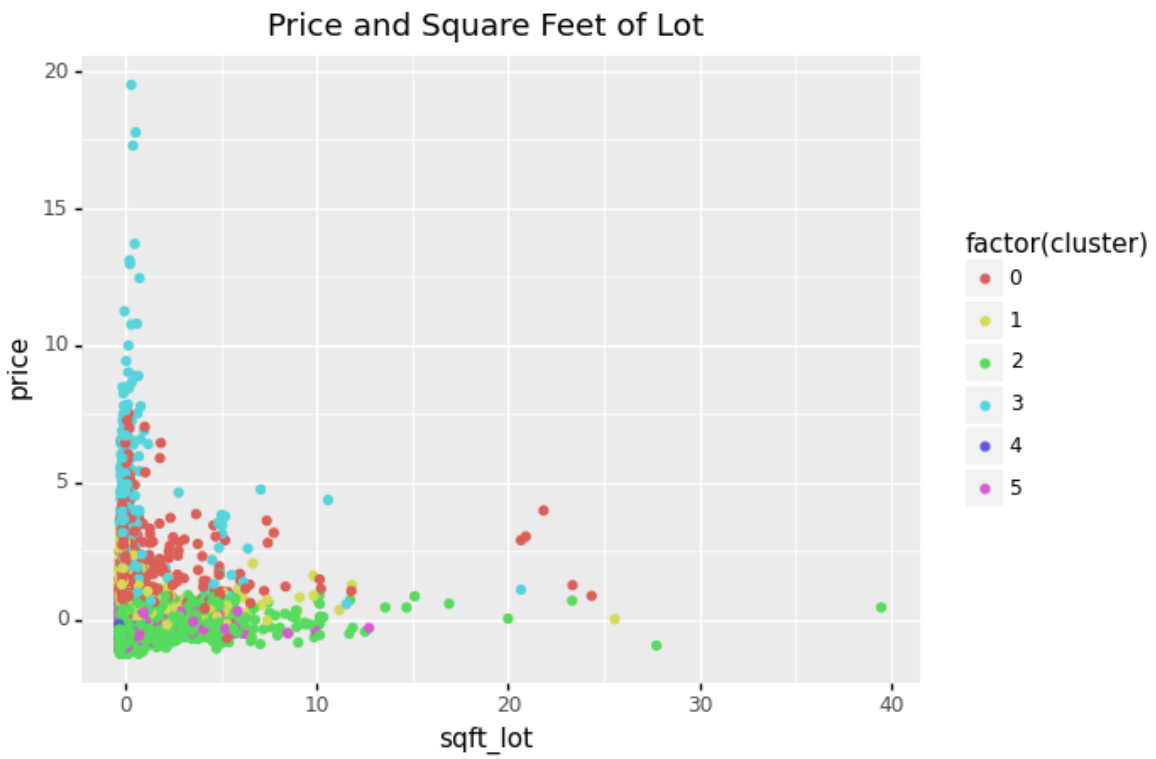
Out[330]: <ggplot: (8771467478285)>

-The red cluster in the bottom left shows houses that are low in price and low in both sqft_lot and sqft_living area. These are likely cheap properties that are very small and minimal for lower class citizens.

-The blue cluster shows real estate with a way more space in sqft_lot and sqft_living area for a little more of the price. These can be small houses rather than meager apartments/condos.

-The green cluster shows the most expensive houses with the most sqft_lot and Sqft_living area. These can range from very large mansions to residential places with a waterfront view.

In [357]:
```
X_gm["cluster"] = cluster

(ggplot(X_gm, aes(x = "sqft_lot", y = "price", color = "factor(cluster)"
)) + geom_point() +
 ggtitle("Price and Square Feet of Lot"))
```



Out[357]: <ggplot: (8771747521805)>

-The red cluster in the bottom left shows houses that are low in price and low in both sqft_lot and sqft_living area. These are likely cheap properties that are very small and minimal for lower class citizens.

-The blue cluster shows real estate with a way more space in sqft_lot and sqft_living area for a little more of the price. These can be small houses rather than meager apartments/condos.

-The green cluster shows the most expensive houses with the most sqft_lot and Sqft_living area. These can range from very large mansions to residential places with a waterfront view.

Yes, we are able to cluster houses based off of sales price and the square feet of the living area/lot of the house. We will utilize a Gaussian Mixture Model. As there were no explicitly labeled groups in this data, we wanted to confirm my assumptions about what types of groups exist and identify unknown groups in this King County House dataset. Gaussian is an iterative process to create "k" groups from this property data based on the values of the variables, and through this, we are able to see the clusters converge to a particular membership so that we can identify different groups in the data. When clustering my data, we only want to use 3 features for better visualization in two/three dimensions. We will decide to cluster the data utilizing only “price”, "Sqft_lv", and “Sqft_lot”, to assign membership for this King County House dataset. Gaussian assigns each data point a probability of being assigned to each cluster. So rather it being a hard assignment and assuming that all clusters are spherical with the same variance in every direction; Gaussian has a soft-probabilistic assignment that allows more elliptical shapes.

-The red cluster in the bottom left shows houses that are low in price and low in both sqft_lot and sqft_living area. These are likely cheap properties that are very small and minimal for lower class citizens.

-The blue cluster shows real estate with a way more space in sqft_lot and sqft_living area for a little more of the price. These can be small houses rather than meager apartments/condos.

-The green cluster shows the most expensive houses with the most sqft_lot and Sqft_living area. These can range from very large mansions to residential places with a waterfront view.

14.Taking some of the coefficients and putting them to zero, is the model as accurate when cutting out some variables (dimensionality reduction)?

In [345]:
```
feat = ["bedrooms", "bathrooms", "sqft_living", "sqft_lot", "floors", "w
aterfront", "condition", "grade"]
X = house[feat]
y = house["price"]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3
)

z = StandardScaler()

X_train[feat] = z.fit_transform(X_train[feat])
X_test[feat] = z.transform(X_test[feat])

X_train.head()
```

Out[345]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	condition	grade
14396	0.668853	0.498223	0.056981	-0.286318	0.006223	-0.088282	0.912716	-0.560968
11275	-0.402516	0.171453	0.200120	0.952894	-0.916860	-0.088282	-0.628237	-0.560968
13820	-1.473885	-0.808860	-1.297327	-0.229248	-0.916860	-0.088282	-0.628237	-0.560968
268	-0.402516	0.498223	-0.009083	-0.127601	-0.916860	-0.088282	-0.628237	0.292843
10154	-0.402516	-0.808860	-1.121157	-0.199759	-0.916860	-0.088282	0.912716	-1.414780

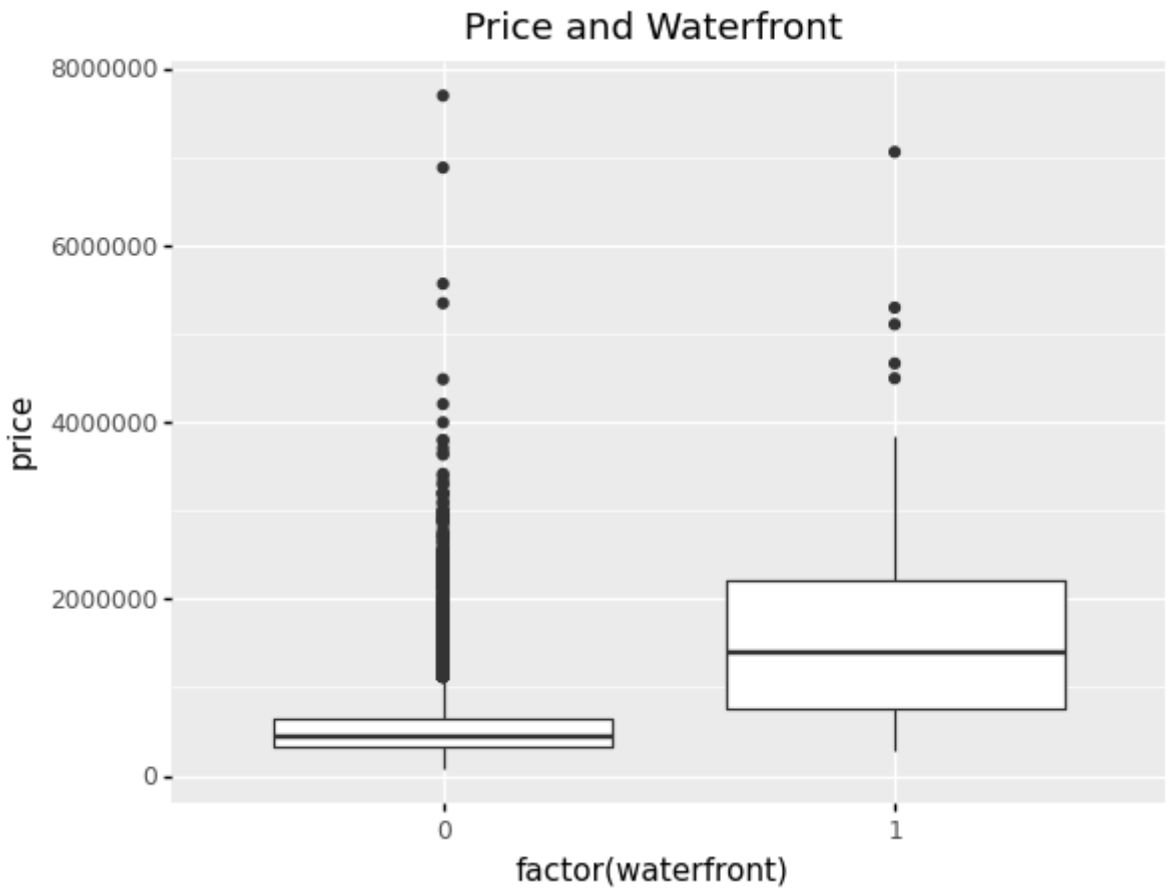
In [348]:
```
lr = LinearRegression()
lr.fit(X_train,y_train)

print("TRAIN: ", mean_absolute_error(y_train, lr.predict(X_train)))
print("TEST : ", mean_absolute_error(y_test, lr.predict(X_test)))
```

```
TRAIN:  155094.94019675173
TEST :  157590.33771304155
```

In [349]:
```
lsr = Lasso()

lsr.fit(X_train,y_train)

print("TRAIN: ", mean_absolute_error(y_train, lsr.predict(X_train)))
print("TEST : ", mean_absolute_error(y_test, lsr.predict(X_test)))
```

```
TRAIN:  155094.75944668357
TEST :  157590.1693831657
```

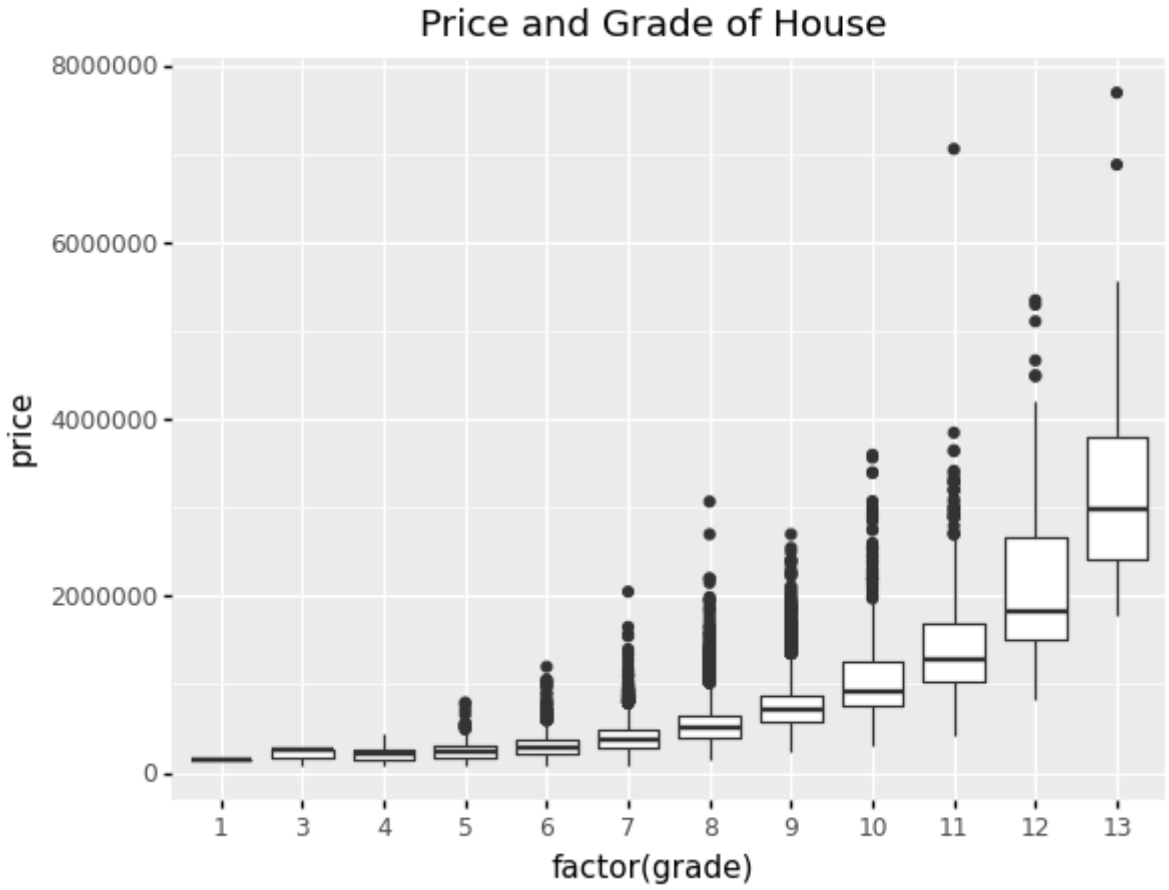
In [354]:
```
(ggplot(house, aes(x = "factor(waterfront)", y = "price"))+ geom_boxplot
()+ ggtitle("Price and Waterfront"))
```



Out[354]: <ggplot: (8771469104083)>

The scatterplot shows the correlation between the variable grade and waterfront.

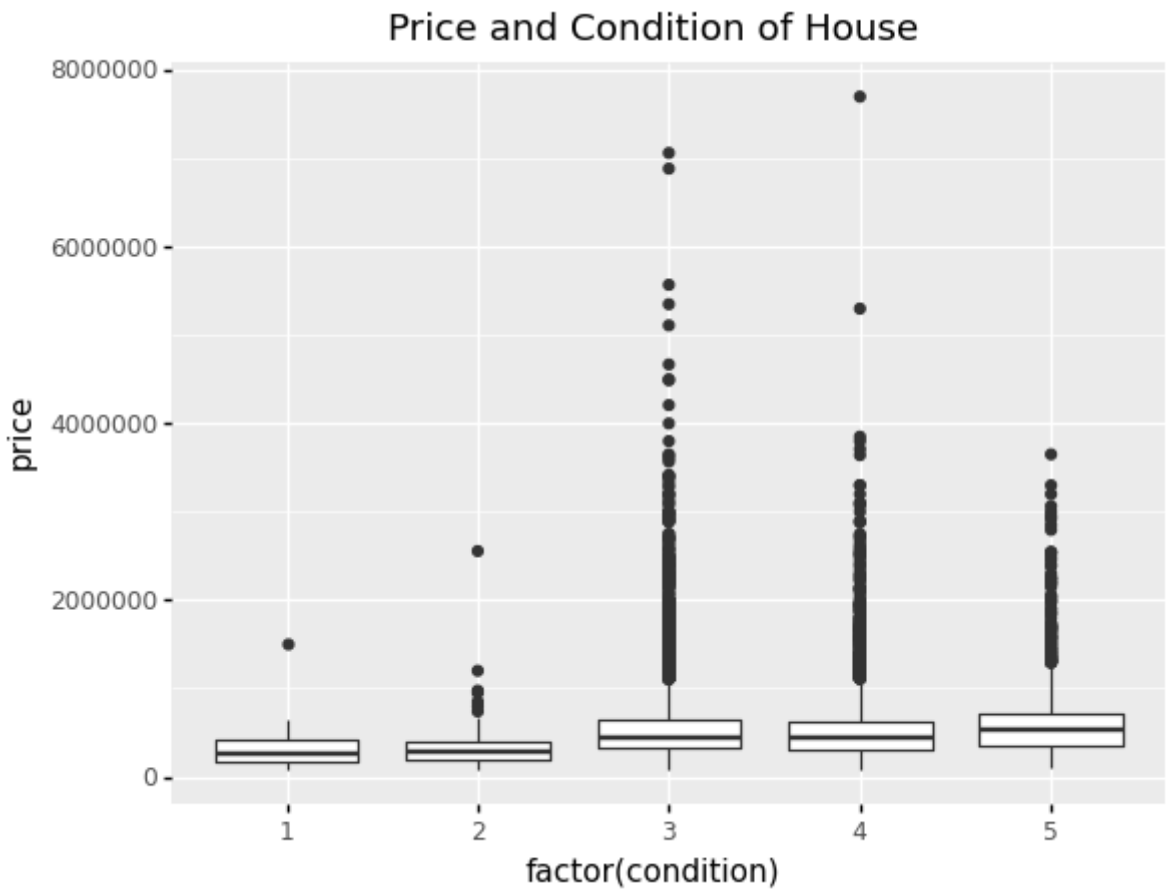
In [355]:
```
ggplot(house, aes(x = "factor(grade)", y = "price"))+ geom_boxplot() + g
gtitle("Price and Grade of House")
```



Out[355]: <ggplot: (8771760236054)>

The scatterplot shows the correlation between the variable grade and price. Through observation, we are able to predict that when the grade of the house increases, so does the price of the property.

In [356]:
```
(ggplot(house, aes(x = "factor(condition)", y = "price"))+ geom_boxplot
() + ggtitle("Price and Condition of House"))
```



Out[356]: <ggplot: (8771469107125)>

The scatterplot shows the correlation between the variable condition and price.

By utilizing Lasso, it pushes the small coefficients to be exactly zero. This is a way of improving our model generalization - avoiding overfitting by assuming the small relationships are actually just noise from our sample and do not apply to the population of our data as a whole. In this way, we can choose which variables to include in the model. Lasso has slightly less error between the train and test set. However, comparing Lasso where it essentially removes variables from the model to Linear Regression where it accounts for all of them... it is very similar and probably just as accurate.