Name:-Priansh Madan

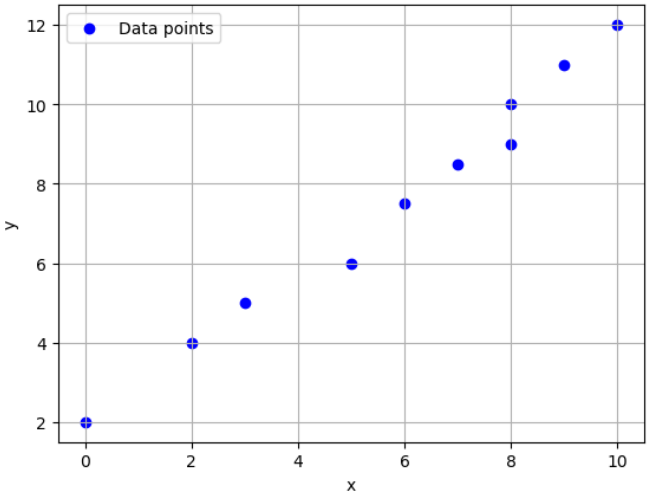
Rollno:-58

PART(A): Linear Regression

1. Consider a set of points x and the predicted values y

a) plot the data

In [4]:
```
import matplotlib.pyplot as plt

x = [0, 2, 3, 5, 6, 7, 8, 8, 9, 10]
y = [2, 4, 5, 6, 7.5, 8.5, 9, 10, 11, 12]

plt.scatter(x, y, color='blue', label='Data points')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```



(b) Use LinearRegression model from sklearn library to perform linear regression.

In [5]:
```
from sklearn.linear_model import LinearRegression
import numpy as np

x = np.array(x).reshape(-1, 1)
y = np.array(y)

model = LinearRegression()

# Fit the model to the data
model.fit(x, y)

slope = model.coef_[0]
intercept = model.intercept_

print("Slope:", slope)
print("Intercept:", intercept)
```

```
Slope: 0.9780334728033473
Intercept: 1.8274058577405858
```

(c) Consider a data value for x and predict the value of y using the above model.

In [6]:
```
# Choose a data value for x
new_x = np.array([[4]])  # Predicting y for x = 4

# Predict the corresponding value of y
predicted_y = model.predict(new_x)

print("Predicted y:", predicted_y[0])
```

```
Predicted y: 5.739539748953975
```

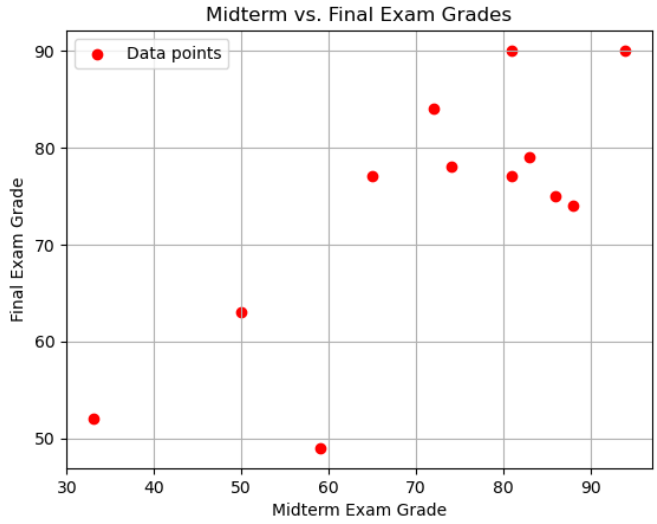
Q.2

2. The given table shows the midterm and final exam grades obtained for students in a

database course.

a) plot

In [7]:
```
import matplotlib.pyplot as plt

x = [72, 50, 81, 74, 94, 86, 59, 83, 65, 33, 88, 81]
y = [84, 63, 77, 78, 90, 75, 49, 79, 77, 52, 74, 90]

plt.scatter(x, y, color='red', label='Data points')
plt.xlabel('Midterm Exam Grade')
plt.ylabel('Final Exam Grade')
plt.title('Midterm vs. Final Exam Grades')
plt.legend()
plt.grid()
plt.show()
```



(b) Use the method of least squares to find an equation for the prediction of a student's

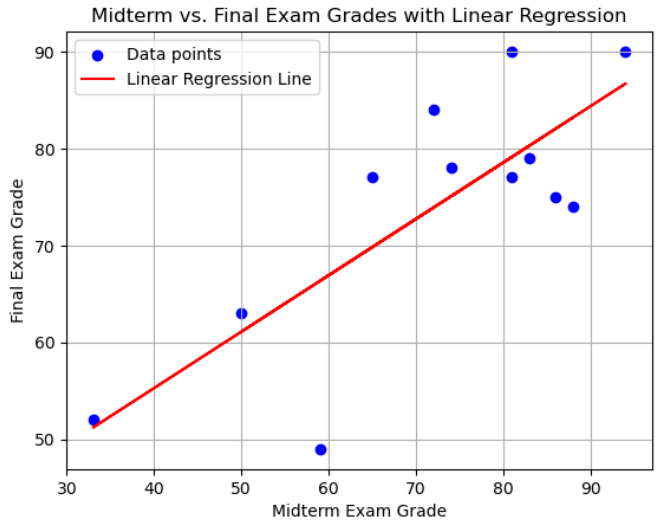
final exam grade based on the student's midterm grade in the course. Write a function in python to compute the coefficients and equation. [Do not use the inbuild library method.]

In [8]:
```
def least_squares(x, y):
    n = len(x)
    xy_sum = sum([xi * yi for xi, yi in zip(x, y)])
    x_sum = sum(x)
    y_sum = sum(y)
    x_squared_sum = sum([xi ** 2 for xi in x])

    m = (n * xy_sum - x_sum * y_sum) / (n * x_squared_sum - x_sum ** 2)
    b = (y_sum - m * x_sum) / n

    return m, b

m, b = least_squares(x, y)
print("Slope (m):", m)
print("Intercept (b):", b)
print(f"y = {m}x + {b}")
```

```
Slope (m): 0.5816000773918932
Intercept (b): 32.027861081551706
y = 0.5816000773918932x + 32.027861081551706
```

(c) Also show the plot with the datapoints and the obtained linear equation line.

In [9]:
```
plt.scatter(x, y, color='blue', label='Data points')
plt.plot(x, [m * xi + b for xi in x], color='red', label='Linear Regression Line')
plt.xlabel('Midterm Exam Grade')
plt.ylabel('Final Exam Grade')
plt.title('Midterm vs. Final Exam Grades with Linear Regression')
plt.legend()
plt.grid()
plt.show()
```



(d) Predict the final exam grade of student who received an 86 in the midterm exam based on the equation of least squares.

In [10]:
```
midterm_grade = 86
predicted_final_grade = m * midterm_grade + b
print("Predicted Final Exam Grade:", predicted_final_grade)
```

```
Predicted Final Exam Grade: 82.04546773725453
```

Q.3

Multiple Linear regression

Perform Multiple Linear regression on cars.csv dataset.

(a) Analyse each column of the datset using appropriate visualization technique.

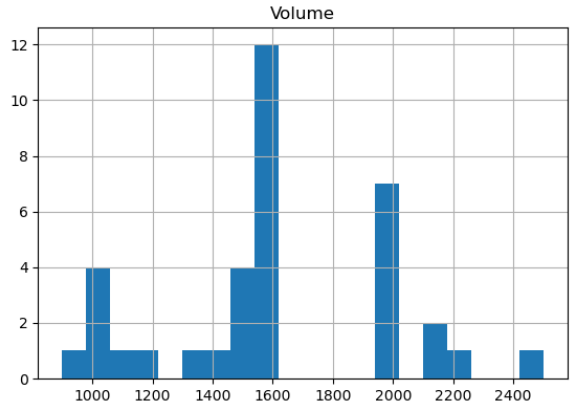
In [11]:
```
import pandas as pd
# Load the dataset
data = pd.read_csv("cars - cars.csv")
data
```

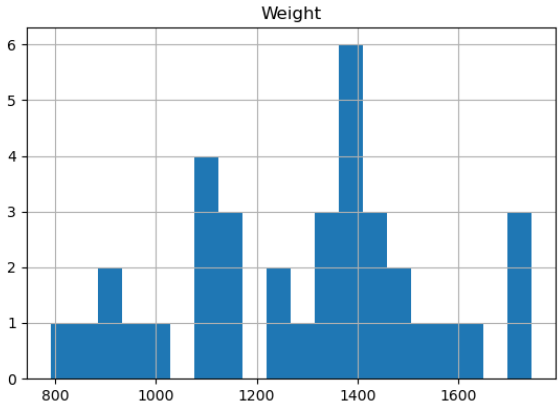
Out[11]:

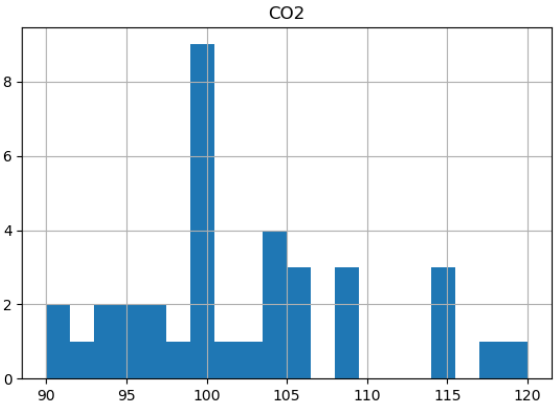
	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1000	790	99
1	Mitsubishi	Space Star	1200	1160	95
2	Skoda	Citigo	1000	929	95
3	Fiat	500	900	865	90
4	Mini	Cooper	1500	1140	105
5	VW	Up!	1000	929	105
6	Skoda	Fabia	1400	1109	90
7	Mercedes	A-Class	1500	1365	92
8	Ford	Fiesta	1500	1112	98
9	Audi	A1	1600	1150	99
10	Hyundai	I20	1100	980	99
11	Suzuki	Swift	1300	990	101
12	Ford	Fiesta	1000	1112	99
13	Honda	Civic	1600	1252	94
14	Hundai	I30	1600	1326	97
15	Opel	Astra	1600	1330	97
16	BMW	1	1600	1365	99
17	Mazda	3	2200	1280	104
18	Skoda	Rapid	1600	1119	104
19	Ford	Focus	2000	1328	105
20	Ford	Mondeo	1600	1584	94
21	Opel	Insignia	2000	1428	99
22	Mercedes	C-Class	2100	1365	99
23	Skoda	Octavia	1600	1415	99
24	Volvo	S60	2000	1415	99
25	Mercedes	CLA	1500	1465	102
26	Audi	A4	2000	1490	104
27	Audi	A6	2000	1725	114
28	Volvo	V70	1600	1523	109
29	BMW	5	2000	1705	114
30	Mercedes	E-Class	2100	1605	115
31	Volvo	XC70	2000	1746	117
32	Ford	B-Max	1600	1235	104
33	BMW	216	1600	1390	108
34	Opel	Zafira	1600	1405	109
35	Mercedes	SLK	2500	1395	120

In [12]:

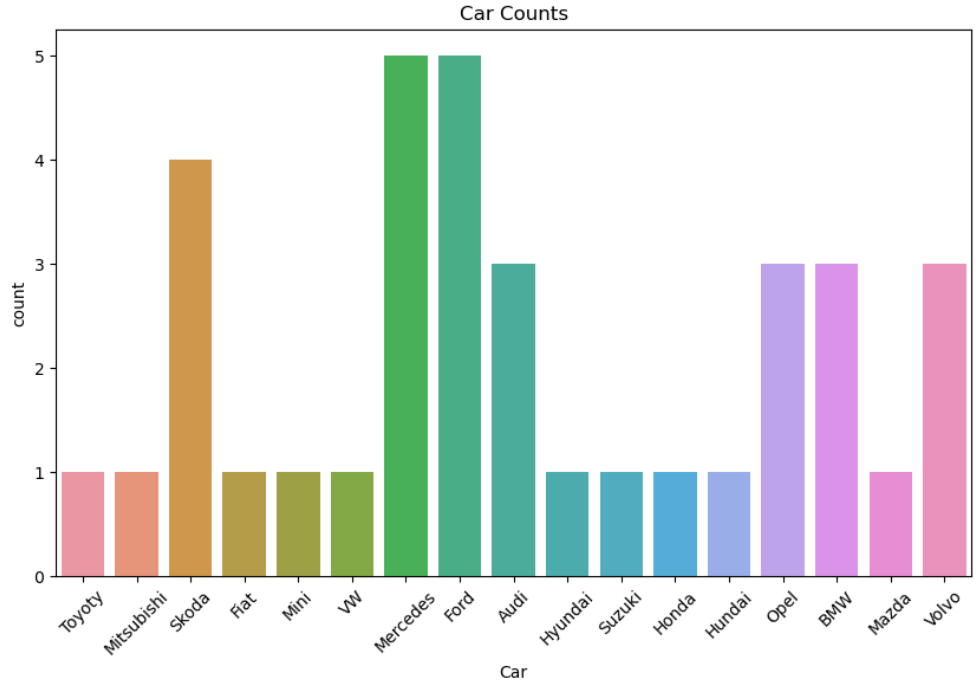
```
import matplotlib.pyplot as plt

# Analyze columns using appropriate visualization techniques
data.hist(bins=20, figsize=(15, 10))
plt.show()
```

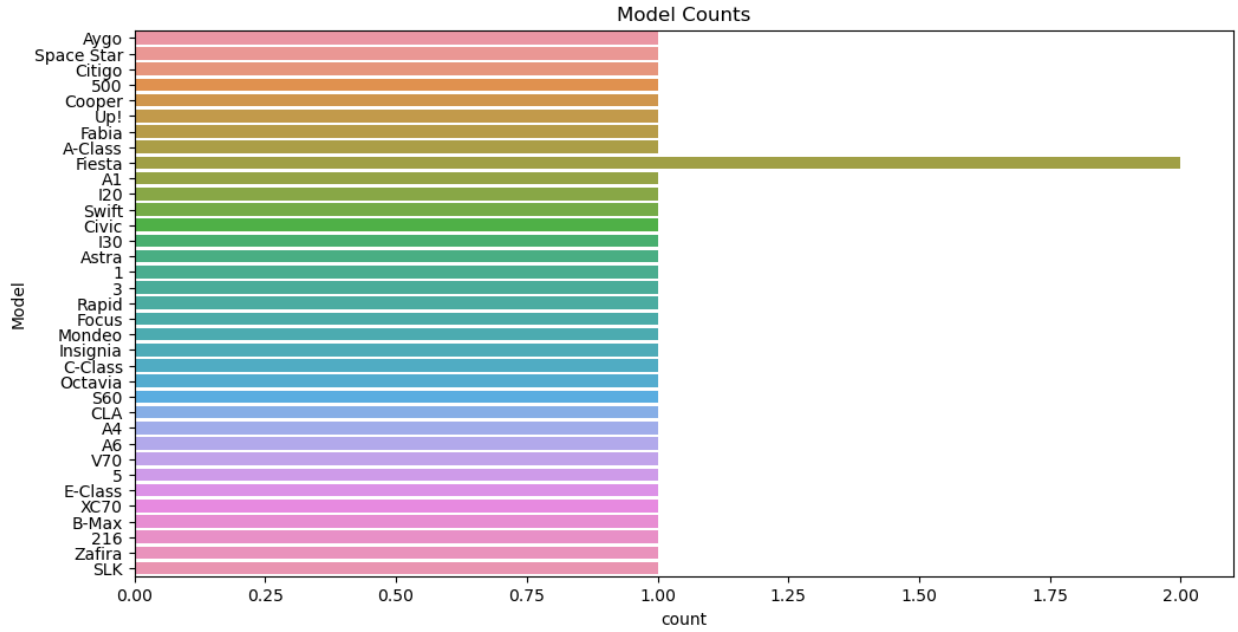






In [13]:
```
import seaborn as sns

# Analyze the "cars" column
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='Car')
plt.xticks(rotation=45)
plt.title('Car Counts')
plt.show()

# Analyze the "models" column
plt.figure(figsize=(12, 6))
sns.countplot(data=data, y='Model')
plt.title('Model Counts')
plt.show()
```





(b) Consider only weight and volume columns as multiple variables to predict the CO2 emission.

In [14]:
```
from sklearn.linear_model import LinearRegression

# Considering only weight and volume columns for prediction
X = data[['Weight', 'Volume']]
y = data['CO2']

# Create a Linear Regression model
model = LinearRegression()

# Fit the model to the data
model.fit(X, y)
```

Out[14]:
```
LinearRegression()
```

(c) Predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300ccm. Also conclude about the obtained result.

In [15]:
```
weight = 2300
volume = 1300
predicted_co2 = model.predict([[weight, volume]])
print("Predicted CO2 Emission:", predicted_co2[0])
```

```
Predicted CO2 Emission: 107.20873279892223

C:\Users\acer\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was f
itted with feature names
  warnings.warn(
```

(d) Perform prediction on few more values.

In [16]:
```
# More values for prediction
test_data = [
    [2100, 1200],
    [2600, 1400],
    [1800, 1100]
]

predicted_co2_values = model.predict(test_data)
for i, prediction in enumerate(predicted_co2_values):
    print(f"Predicted CO2 Emission for test {i + 1}: {prediction}")
```

```
Predicted CO2 Emission for test 1: 104.91801759208738
Predicted CO2 Emission for test 2: 110.25454273278714
Predicted CO2 Emission for test 3: 101.87220765822246

C:\Users\acer\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was f
itted with feature names
  warnings.warn(
```

(e) Show the coefficient obtained and conclude.

In [17]:
```
# Coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

```
Coefficients: [0.00755095 0.00780526]
Intercept: 79.69471929115939
```

Q.4

Perform linear regression on the dataset [use dataset: kc_house_data.csv]

(a) Load the dataset, display it, visualize various columns and explain the dataset

composition

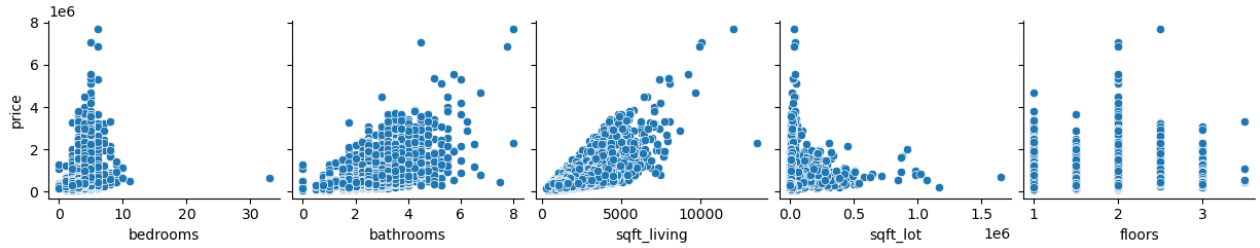
In [20]:
```
data = pd.read_csv("kc_house_data.csv")
data
```

Out[20]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renova
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	1530	0	2009	
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	2310	0	2014	
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	1020	0	2009	
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	1600	0	2004	
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	1020	0	2008	

21613 rows × 21 columns

In [21]:
```
# Visualize various columns
sns.pairplot(data, x_vars=['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors'],
             y_vars='price', kind='scatter')
plt.show()
```



(b) Describe the dataset

In [22]:
```
data.describe()
```

Out[22]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_al
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.00
mean	4.580302e+09	5.401822e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.656873	1788.39
std	2.876566e+09	3.673622e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.175459	828.09
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.000000	290.00
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.000000	1190.00
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.000000	1560.00
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.000000	2210.00
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.000000	9410.00

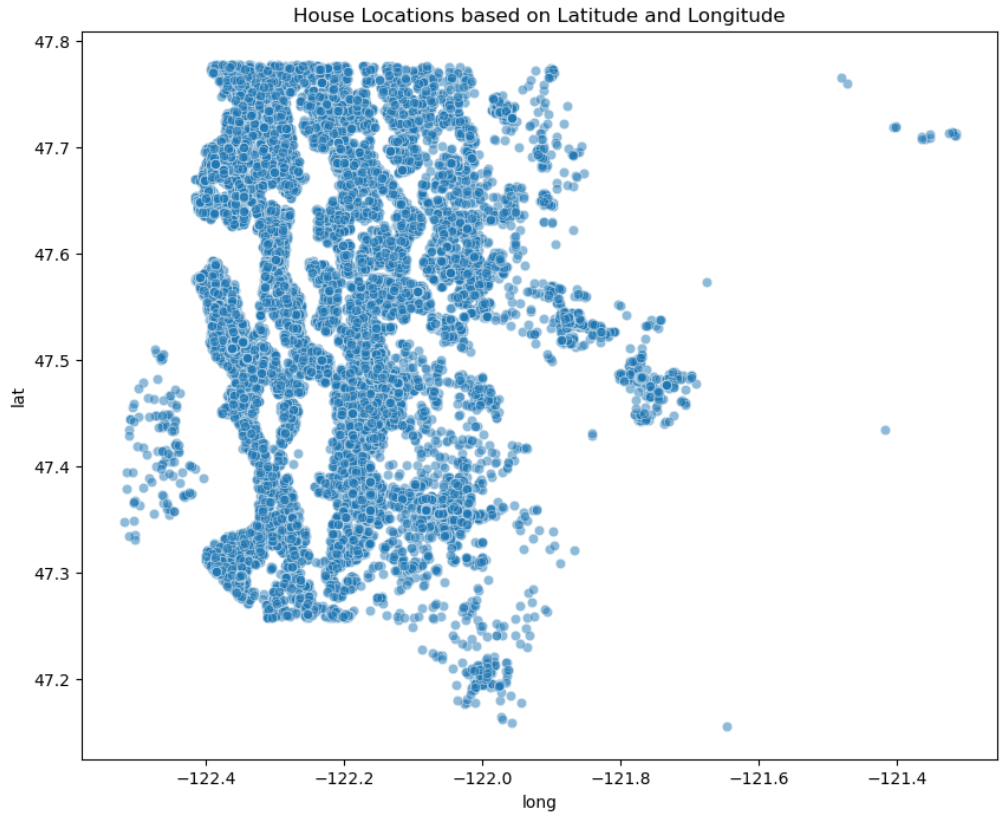
(c) Houses with how many bedrooms are most sold?

In [23]:
```
most_common_bedrooms = data['bedrooms'].value_counts().idxmax()
print("Houses with", most_common_bedrooms, "bedrooms are most sold.")
```

```
Houses with 3 bedrooms are most sold.
```

(d) Visualizing the location of the houses based on latitude and longitude. Use appropriate plot.

In [24]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(data=data, x='long', y='lat', palette='coolwarm', alpha=0.5)
plt.title('House Locations based on Latitude and Longitude')
plt.show()
```



(e) Find the correlated features/ columns

In [25]:
```
correlation_matrix = data.corr()
correlation_matrix['price'].sort_values(ascending=False)
```

Out[25]:
```
price            1.000000
sqft_living      0.702044
grade            0.667463
sqft_above       0.605566
sqft_living15    0.585374
bathrooms        0.525134
view             0.397346
sqft_basement    0.323837
bedrooms         0.308338
lat              0.306919
waterfront       0.266331
floors           0.256786
yr_renovated     0.126442
sqft_lot         0.089655
sqft_lot15       0.082456
yr_built         0.053982
condition        0.036392
long             0.021571
id              -0.016797
zipcode         -0.053168
Name: price, dtype: float64
```

(f) Find null values and fill with mean value for all columns

In [26]:
```
data.fillna(data.mean(), inplace=True)
data
```

```
C:\Users\acer\AppData\Local\Temp\ipykernel_13772\3452971976.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  data.fillna(data.mean(), inplace=True)
```

Out[26]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renova
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	1530	0	2009	
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	2310	0	2014	
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	1020	0	2009	
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	1600	0	2004	
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	1020	0	2008	

21613 rows × 21 columns

(g) Find dependant and independent data (place in X and y)

In [28]:
```
X = data[['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'long', 'lat']]
y = data['price']
```

(h) Split train and test data

In [29]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [30]:
```
X_train
```

Out[30]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	long	lat
6325	3	1.75	1780	13095	1.0	-122.152	47.3670
13473	2	1.00	1000	3700	1.0	-122.290	47.5520
17614	3	1.00	1080	7486	1.5	-122.335	47.4838
16970	3	2.25	2090	7500	1.0	-122.172	47.3951
20868	2	2.50	1741	1439	2.0	-122.209	47.7043
...	...	...	...	...	...	...	...
11964	3	1.50	1000	6914	1.0	-122.319	47.7144
21575	3	2.50	3087	5002	2.0	-122.349	47.2974
5390	3	2.50	2120	4780	2.0	-122.032	47.6810
860	1	0.75	380	15000	1.0	-122.323	47.4810
15795	4	2.50	3130	5999	2.0	-122.099	47.3837

17290 rows × 7 columns

(i) Train the model and test it. Find the accuracy.

In [31]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print("Root Mean Squared Error:", rmse)

test_score = model.score(X_test, y_test)
print("Test score: ", test_score)
```

```
Root Mean Squared Error: 249585.26277177373
Test score:  0.5884316637460509
```

(j) Test the model using some arbitrary input.

In [32]:
```
arbitrary_input = [[3, 2, 1800, 4000, 2, -122.1, 47.6]]
predicted_price = model.predict(arbitrary_input)
print("Predicted Price:", predicted_price[0])
```

```
Predicted Price: 457200.8450436592

C:\Users\acer\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was f
itted with feature names
  warnings.warn(
```

In [ ]: