Author: Jonathan Ibifubara Pollyn
Course: DSC-540

Assignment: calculate the appreciation of a real estate property over time.

Model: Multiple Regression model

College of Science, Engineering and Technology, Grand Canyon University

In [1]:
```
#Importing the required packages
import pandas as pd
import numpy as np
import statsmodels.api as sm
from scipy import stats
import statsmodels.tools.tools as stattools
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
import sklearn.metrics as met
from sklearn.linear_model import LinearRegression
%matplotlib inline
import math
```

In [2]:
```
#Reading housing data
house = pd.read_csv('C:/School/DSC-540/Topic 1 - Machine Learning Packages in R and Python/housing.csv')
```

Checking the housing data by checking using the head, info() and describe() methods.

In [3]:
```
house.head()
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	

In [4]:
```
house.describe()
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_ho
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	206
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	2068
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	1153
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	149
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	1196
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	1797
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	2647
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	5000

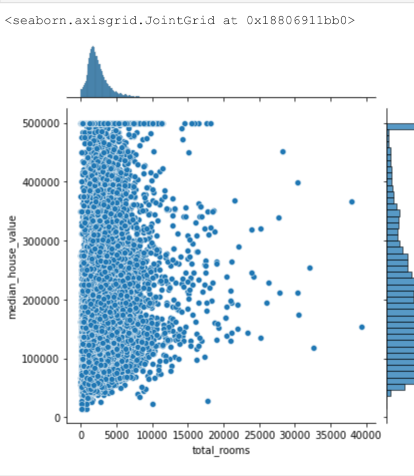
In [5]:
```
house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Performing Data exploration

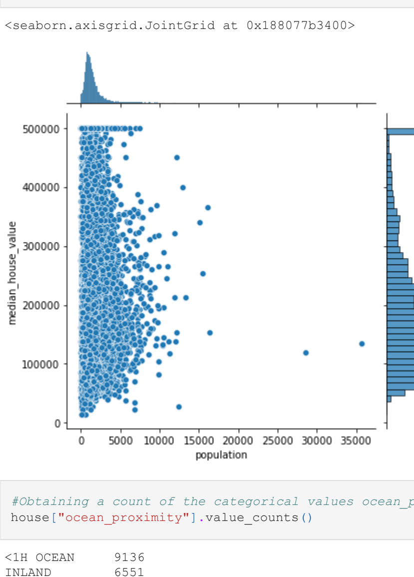
In [6]:
```
#Exploring total_rooms and median_house_value
sns.jointplot(x='total_rooms',y='median_house_value',data=house)
```

Out[6]: `<seaborn.axisgrid.JointGrid at 0x18806911bb0>`



In [7]:
```
#Exploring population and median_house_value
sns.jointplot(x='population',y='median_house_value',data=house)
```

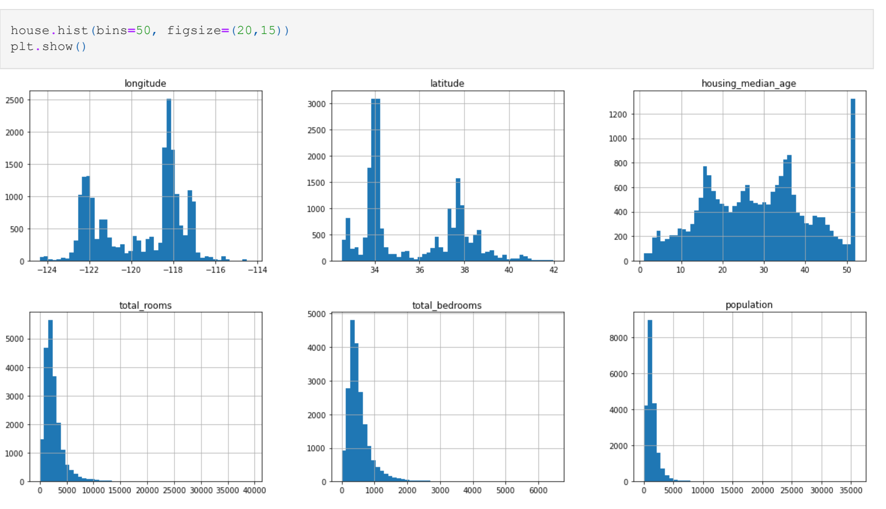
Out[7]: `<seaborn.axisgrid.JointGrid at 0x188077b3400>`



In [8]:
```
#Obtaining a count of the categorical values ocean_proximity
house["ocean_proximity"].value_counts()
```

Out[8]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

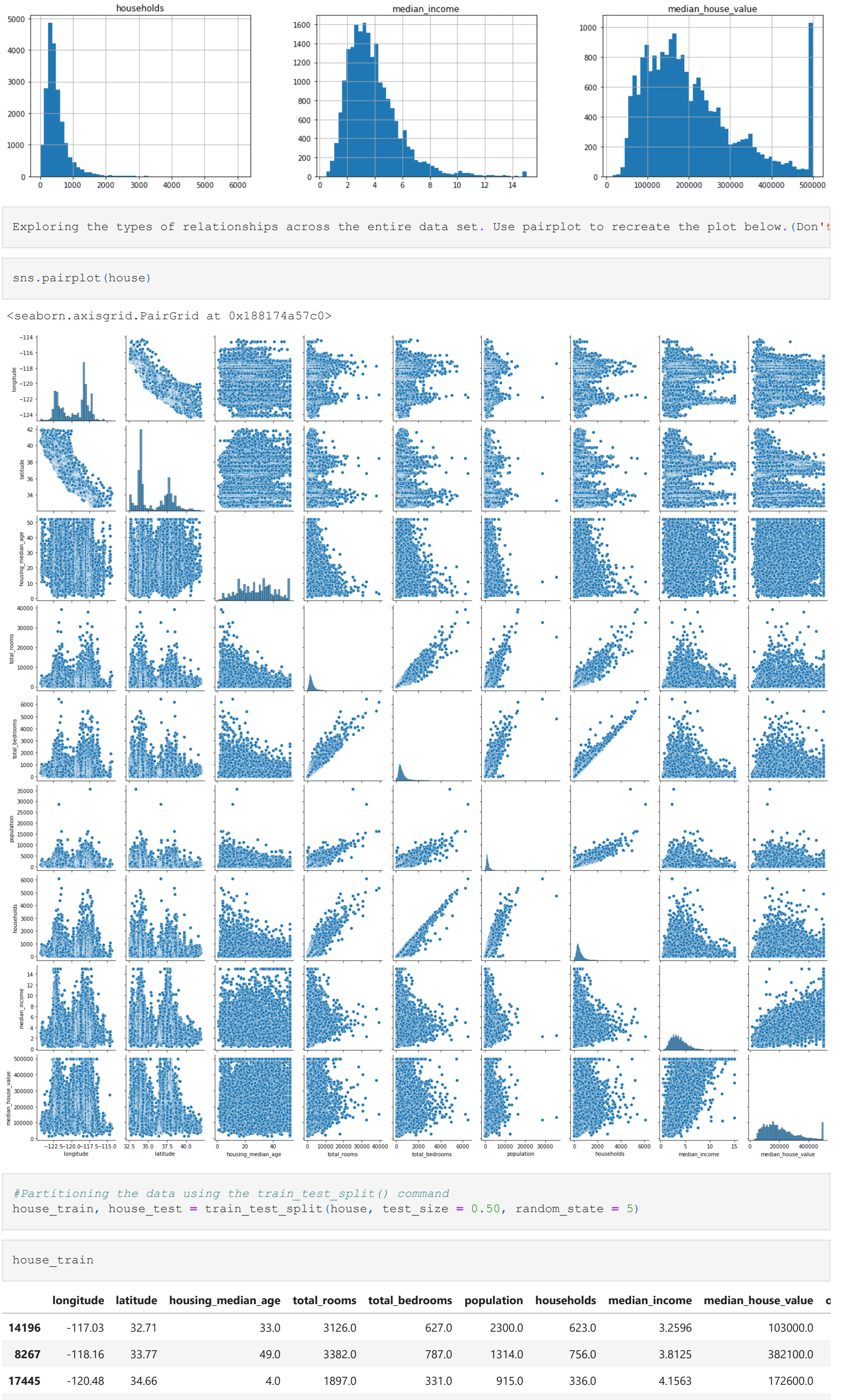
In [9]:
```
house.hist(bins=50, figsize=(20,15))
plt.show()
```



In [ ]:
```
Exploring the types of relationships across the entire data set. Use pairplot to recreate the plot below.(Don't
```

In [36]:
```
sns.pairplot(house)
```

Out[36]: `<seaborn.axisgrid.PairGrid at 0x188174a57c0>`



In [ ]:
```
#Partitioning the data using the train_test_split() command
house_train, house_test = train_test_split(house, test_size = 0.50, random_state = 5)
```

In [52]:
```
house_train
```

Out[52]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	o
14196	-117.03	32.71	33.0	3126.0	627.0	2300.0	623.0	3.2596	103000.0	
8267	-118.16	33.77	49.0	3382.0	787.0	1314.0	756.0	3.8125	382100.0	
17445	-120.48	34.66	4.0	1897.0	331.0	915.0	336.0	4.1563	172600.0	
14265	-117.11	32.69	36.0	1421.0	367.0	1418.0	355.0	1.9425	93400.0	
2271	-119.80	36.78	43.0	2382.0	431.0	874.0	380.0	3.5542	96500.0	
...	...	...	...	...	...	...	...	...	...	
11284	-117.96	33.78	35.0	1330.0	201.0	658.0	217.0	6.3700	229200.0	
11964	-117.43	34.02	33.0	3084.0	570.0	1753.0	449.0	3.0500	97800.0	
5390	-118.38	34.03	36.0	2101.0	569.0	1756.0	527.0	2.9344	222100.0	
860	-121.96	37.58	15.0	3575.0	597.0	1777.0	559.0	5.7192	283500.0	
15795	-122.42	37.77	52.0	4226.0	1315.0	2619.0	1242.0	2.5755	325000.0	

16512 rows × 10 columns

In [63]:
```
#Separating the training data using data frame into predictors and target variables.
pred_train = pd.DataFrame(house_train[['housing_median_age','population','total_rooms']])
pred_train = sm.add_constant(pred_train)
target_train = pd.DataFrame(house_train[['median_house_value']])
```

In [57]:
```
pred_train
```

Out[57]:

	const	housing_median_age	population	total_rooms
14196	1.0	33.0	2300.0	3126.0
8267	1.0	49.0	1314.0	3382.0
17445	1.0	4.0	915.0	1897.0
14265	1.0	36.0	1418.0	1421.0
2271	1.0	43.0	874.0	2382.0
...	...	...	...	...
11284	1.0	35.0	658.0	1330.0
11964	1.0	33.0	1753.0	3084.0
5390	1.0	36.0	1756.0	2101.0
860	1.0	15.0	1777.0	3575.0
15795	1.0	52.0	2619.0	4226.0

16512 rows × 4 columns

In [58]:
```
target_train
```

Out[58]:

	median_house_value
14196	103000.0
8267	382100.0
17445	172600.0
14265	93400.0
2271	96500.0
...	...
11284	229200.0
11964	97800.0
5390	222100.0
860	283500.0
15795	325000.0

16512 rows × 1 columns

In [66]:
```
#Now running the multiple regression model
model0_train = sm.OLS(target_train, pred_train).fit()
```

In [67]:
```
model0_train.summary2()
```

Out[67]:

Model:	OLS	Adj. R-squared:	0.123
Dependent Variable:	median_house_value	AIC:	429691.1943
Date:	2021-11-02 10:34	BIC:	429722.0417
No. Observations:	16512	Log-Likelihood:	-2.1484e+05
Df Model:	3	F-statistic:	773.4
Df Residuals:	16508	Prob (F-statistic):	0.00
R-squared:	0.123	Scale:	1.1723e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	143607.2145	2795.4275	51.3722	0.0000	138127.8756	149086.5534
housing_median_age	1704.1759	71.7420	23.7542	0.0000	1563.5538	1844.7980
population	-55.5579	1.4434	-38.4912	0.0000	-58.3871	-52.7287
total_rooms	35.6111	0.7739	46.0143	0.0000	34.0941	37.1280

Omnibus:	1625.157	Durbin-Watson:	1.982
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2265.964
Skew:	0.789	Prob(JB):	0.000
Kurtosis:	3.895	Condition No.:	12743

Validating the model

In [64]:
```
#Separating the test data using data frame into predictors and target variables.
pred_test = pd.DataFrame(house_test[['housing_median_age','population','total_rooms']])
pred_test = sm.add_constant(pred_test)
target_test = pd.DataFrame(house_test[['median_house_value']])
```

In [68]:
```
model0_test = sm.OLS(target_test, pred_test).fit()
```

In [69]:
```
model0_test.summary2()
```

Out[69]:

Model:	OLS	Adj. R-squared:	0.114
Dependent Variable:	median_house_value	AIC:	107387.0995
Date:	2021-11-02 10:35	BIC:	107412.4017
No. Observations:	4128	Log-Likelihood:	-53690.
Df Model:	3	F-statistic:	177.9
Df Residuals:	4124	Prob (F-statistic):	1.85e-108
R-squared:	0.115	Scale:	1.1614e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	144482.5362	5622.6270	25.6966	0.0000	133459.1545	155505.9180
housing_median_age	1696.0575	143.7947	11.7950	0.0000	1414.1423	1977.9727
population	-50.8251	2.8976	-17.5405	0.0000	-56.5059	-45.1443
total_rooms	32.3592	1.4892	21.7292	0.0000	29.4396	35.2789

Omnibus:	336.587	Durbin-Watson:	2.008
Prob(Omnibus):	0.000	Jarque-Bera (JB):	422.416
Skew:	0.753	Prob(JB):	0.000
Kurtosis:	3.435	Condition No.:	12839

Predicting the error

In [70]:
```
np.sqrt(model0_train.scale)
```

Out[70]: `108274.26125413286`

Finding the MAE regression and MAE Baseline

In [71]:
```
#Get the predicted target values
ypred_train = model0_train.predict(pred_train)
```

In [73]:
```
#Get the actual target values
ytrue_train = house_train[['median_house_value']]
```

In [74]:
```
#Calculate the MAE Regression Vales
met.mean_absolute_error(y_true=ytrue_train, y_pred=ypred_train)
```

Out[74]: `84442.49105327472`

In [76]:
```
#Get the MAE Baseline
ypred_test = model0_test.predict(pred_test)
ytrue_test = house_test[['median_house_value']]
```

In [77]:
```
#Calculate the MAE Baseline values
met.mean_absolute_error(y_true=ytrue_test, y_pred=ypred_test)
```

Out[77]: `83987.30606597457`

Summaries and interpretation are in the technical documentation.

In [ ]: