Data Exploration and Preparation - Housing Problem

Business Objective: Exploratory data analysis of housing prices.

1. Data Import: The very first step is to import the data:

(would be including some important libraries to be used in the program)

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

import os

# Listing the files in the given directory
print(os.listdir("C:/R/Housing"))

# Reading the housing data
data_path = "C:/R/Housing/housing.csv"
housing = pd.read_csv(data_path)
```

2. Understanding the data

Gives the number of data points (number of rows in the dataset) and no of fields (dimensions)

housing.shape
Out[3]: (20640, 10)

Now we try to understand the overall data: Below output shows us each of the fields along with the data type, number of null values, memory usage etc.

```
In [6]: housing.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

We can observe that total_bedrooms has some missing values here itself which we would be dealing with later.

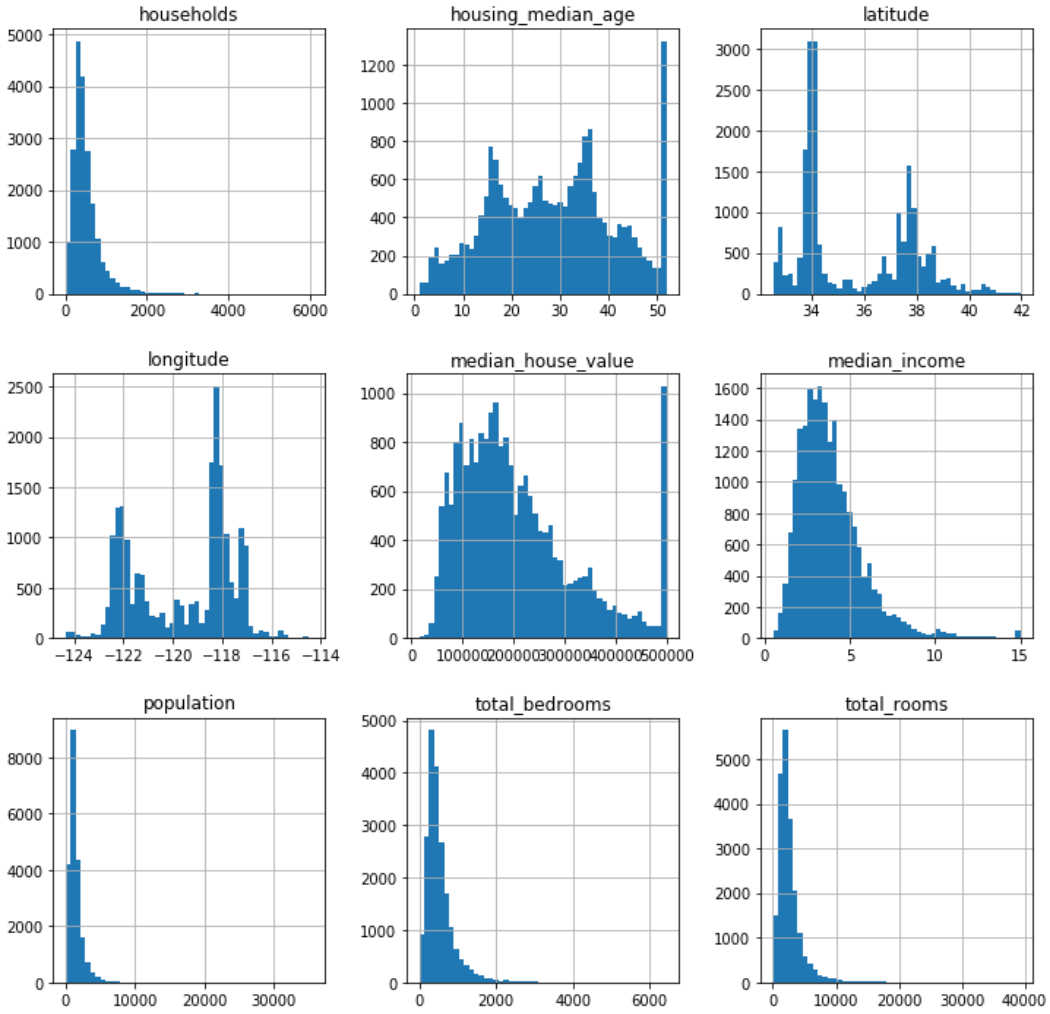
S. No.	Variable	Variable Category	Dependent/Independent	Type
1	longitude	Continuous	I	float64
2	latitude	Continuous	I	float64
3	housing_median_age	Continuous	I	float64
4	total_rooms	Continuous	I	float64
5	total_bedrooms	Continuous	I	float64
6	population	Continuous	I	float64
7	households	Continuous	I	float64
8	median_income	Continuous	I	float64
9	median_house_value	Continuous	D	float64
10	ocean_proximity	Categorical	I	object

Univariate analysis: For **continuous variable**, we try to understand the basic statistics central tendency (mean, median, mode, min, max) and measure of dispersion (range, IQR, variance, std dev) followed by graphs and visualizations.

```
In [8]: housing.describe()
Out[8]:
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

```
In [14]: descr_housing = housing.aggregate([np.median, np.std, np.mean, np.var, np.std]).reset_index()
    ...: descr_housing
Out[14]:
    index   longitude   latitude  housing_median_age   total_rooms  \
0  median -118.490000  34.260000           29.000000  2.127000e+03
1     std    2.003532   2.135952           12.585558  2.181615e+03
2    mean -119.569704  35.631861           28.639486  2.635763e+03
3     var    4.014139   4.562293          158.396260  4.759445e+06
4     std    2.003532   2.135952           12.585558  2.181615e+03

   total_bedrooms    population     households  median_income  \
0      435.000000  1.166000e+03     409.000000       3.534800
1      421.385070  1.132462e+03     382.329753       1.899822
2      537.870553  1.425477e+03     499.539680       3.870671
3   177565.377281  1.282470e+06  146176.039900       3.609323
4      421.385070  1.132462e+03     382.329753       1.899822

   median_house_value
0        1.797000e+05
1        1.153956e+05
2        2.068558e+05
3        1.331615e+10
4        1.153956e+05
```

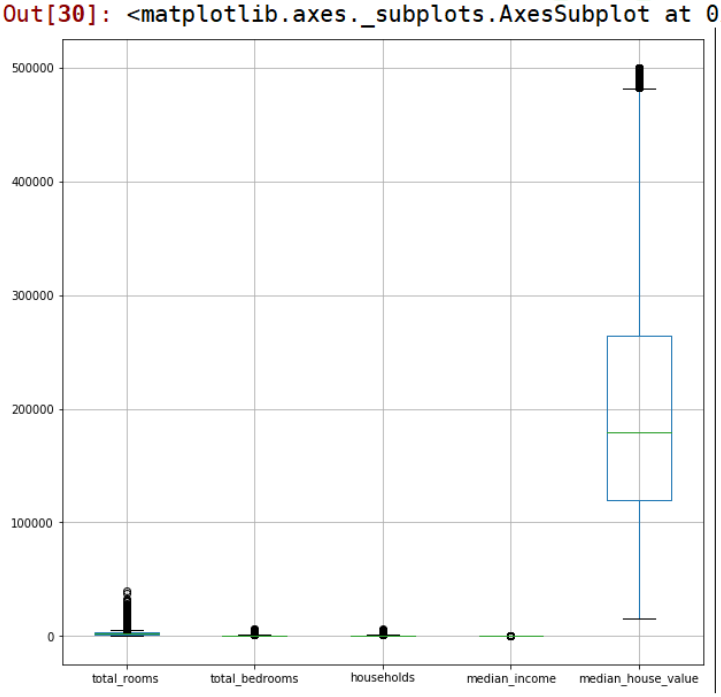
We would be looking at various graphs to understand the variables -



Above graphs shows the distribution of the variable and we can clearly see that many of them are skewed. In many cases, we need to transform the data so that they have more bell-shaped distributions.

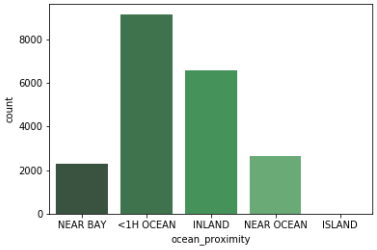
Outliers are detected by the boxplots -

```
In [30]: housing.boxplot(column=[ 'total_rooms',
    ...:                          'total_bedrooms',
    ...:                          'households',
    ...:                          'median_income',
    ...:                          'median_house_value'], figsize=(10, 10))
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1aa7784deb8>
```



As shown above, outliers from median_house_value needs to be taken care of. For **categorical variable** we look at distribution tables and graphs -

```
In [33]: housing['ocean_proximity'].value_counts()
Out[33]:
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64

In [34]: housing['ocean_proximity'].value_counts()
    ...: sns.countplot(x="ocean_proximity", data=housing,
palette="Greens_d");
```



Bivariate analysis:

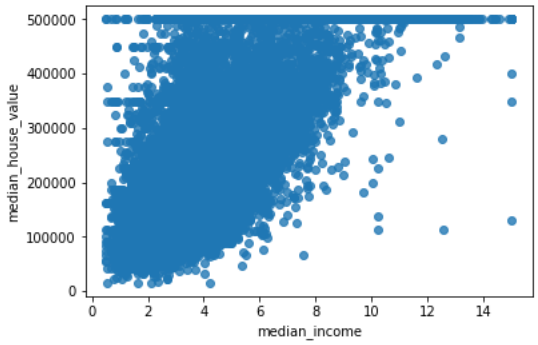
It is performed to understand relationship between two variables. Pearson's r or standard correlation coefficient for every pair indicates how far away all these data points are to this line of best fit. Can take a range of values from +1 to -1.

```
In [44]: corr_matrix = housing.corr()
    ...: # Correlation of each variable w.r.t the median house value
    ...: corr_matrix['median_house_value'].sort_values(ascending=False)
Out[44]:
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

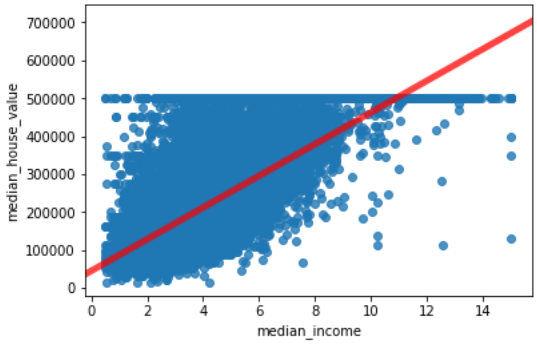
0 indicates that there is no association
-1 means that for every positive increase in one variable, there is a positive increase of a fixed proportion in the other.
+1 means that for every positive increase in one variable, there is a negative decrease of a fixed proportion in the other

We can see that in below plots that median_income is a field which can predict median_house_value -

```
In [12]: sns.regplot(x=housing["median_income"],
y=housing["median_house_value"], fit_reg=False)
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x20e129aa9b0>
```



```
In [13]: sns.regplot(x=housing["median_income"],
y=housing["median_house_value"], line_kws={"color":"r","alpha":0.7,"lw":
5})
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x20e107c8208>
```




```
In [43]: from pandas.plotting import scatter_matrix
    ...: fields = ['median_house_value', 'median_income', 'total_rooms',
'housing_median_age']
    ...: scatter_matrix(housing[fields], alpha=0.2, figsize=(12, 12), diagonal='kde')
    ...: plt.show()
```