In [1]:
```
from google.colab import drive
drive.mount('/content/MyDrive/')
```

```
Mounted at /content/MyDrive/
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
%matplotlib inline
```

1. Load the data :

Read the “housing.csv” file from the folder into the program. Print first few rows of this data. Extract input (X) and output (Y) data from the dataset.

In [3]:
```
data = pd.read_excel('/content/MyDrive/MyDrive/Datasets/1553768847_housin
g.xlsx')
```

In [4]:
```
data.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  int64
 3   total_rooms         20640 non-null  int64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64
 6   households          20640 non-null  int64
 7   median_income       20640 non-null  float64
 8   ocean_proximity     20640 non-null  object
 9   median_house_value  20640 non-null  int64
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

In [5]:
```
data.shape
```

Out[5]:
```
(20640, 10)
```

In [6]:
```
data.head(2)
```

Out[6]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
0	-122.23	37.88	41	880	129.0	322	
1	-122.22	37.86	21	7099	1106.0	2401	

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

In [9]:
```
for column in data:
  print(f"{column}: {data[column].unique()}")
  print
('-----------------------------------------------------------------------
------------------------')
  print(f'{column}: {data[column].nunique()}')
```

```
longitude: [-122.23 -122.22 -122.24 -122.25 -122.26 -122.27 -122.28 -12
2.29 -122.3
 -122.21 -122.2  -122.19 -122.18 -122.13 -122.16 -122.17 -122.15 -122.14
 -122.12 -122.33 -122.34 -122.06 -122.07 -122.08 -122.09 -122.1  -122.11
 -122.03 -121.97 -122.02 -122.04 -122.05 -121.99 -122.01 -121.96 -121.98
 -122.   -121.93 -121.94 -121.95 -121.92 -121.89 -121.91 -121.9  -121.88
 -121.87 -121.85 -121.86 -121.84 -121.82 -121.77 -121.62 -121.61 -121.72
 -121.73 -121.75 -121.8  -121.76 -121.78 -121.79 -119.78 -119.93 -120.
 -120.56 -120.59 -120.55 -120.25 -120.79 -120.8  -120.65 -120.76 -120.88
 -120.69 -120.93 -120.97 -120.87 -120.98 -120.72 -120.77 -120.66 -120.62
 -120.71 -121.83 -121.81 -121.74 -121.68 -121.54 -121.51 -121.59 -121.58
 -121.6  -121.63 -121.57 -121.65 -121.64 -121.71 -121.66 -121.56 -121.5
 -121.41 -121.39 -121.24 -121.19 -121.36 -121.46 -121.49 -121.44 -121.47
 -121.53 -121.52 -121.55 -121.67 -121.69 -121.7  -120.46 -120.54 -120.67
 -120.9  -120.91 -120.57 -120.43 -120.42 -120.41 -120.36 -120.34 -120.33
 -120.37 -120.27 -120.19 -122.51 -122.32 -122.36 -122.31 -122.39 -122.37
 -122.41 -122.35 -122.38 -122.42 -124.17 -124.3  -124.23 -124.21 -124.19
 -124.22 -124.16 -124.14 -124.15 -123.91 -123.83 -123.92 -119.94 -119.95
 -119.97 -119.98 -119.96 -119.99 -120.01 -120.02 -119.92 -120.04 -120.03
 -120.13 -120.16 -120.06 -120.1  -121.04 -120.92 -120.84 -120.81 -120.5
 -120.3  -121.09 -121.08 -121.07 -121.06 -121.   -121.01 -120.99 -121.02
 -120.95 -120.96 -120.86 -120.83 -120.78 -120.7  -120.58 -120.6  -120.63
 -120.44 -120.32 -120.08 -120.85 -119.81 -119.79 -119.8  -119.77 -119.82
 -119.83 -119.74 -119.76 -119.75 -119.69 -119.67 -119.73 -119.72 -119.71
 -119.63 -119.65 -119.68 -119.89 -119.87 -119.85 -119.84 -119.7  -119.86
 -119.9  -120.21 -120.05 -120.07 -119.91 -119.88 -119.64 -119.53 -119.58
 -119.59 -119.5  -119.57 -119.56 -119.55 -119.54 -119.52 -119.47 -119.41
 -119.43 -119.39 -119.4  -119.49 -119.61 -119.48 -119.46 -119.33 -119.21
 -118.94 -119.34 -119.28 -119.32 -118.91 -119.24 -119.25 -119.12 -119.31
 -119.44 -119.45 -119.6  -119.62 -120.09 -120.38 -120.35 -120.31 -120.18
 -120.22 -120.51 -120.39 -120.45 -122.74 -122.53 -124.18 -124.11 -124.13
 -124.06 -124.05 -124.02 -124.08 -124.09 -124.07 -124.1  -123.74 -123.76
 -123.85 -123.72 -123.63 -123.66 -123.52 -124.01 -124.   -123.98 -123.88
 -124.27 -123.96 -123.73 -124.03 -124.26 -124.35 -124.25 -123.84 -123.68
 -123.82 -123.75 -123.78 -115.52 -115.51 -115.46 -115.6  -115.73 -115.62
 -115.41 -115.59 -115.53 -115.54 -115.55 -115.56 -115.32 -115.39 -115.4
 -115.37 -115.38 -115.57 -115.49 -115.64 -115.69 -115.72 -115.58 -115.48
 -115.5  -116.05 -116.   -115.88 -115.9  -116.01 -115.99 -115.94 -115.98
 -115.91 -115.96 -115.95 -115.8  -114.73 -114.98 -114.65 -114.55 -114.63
 -114.66 -118.18 -118.43 -118.6  -118.45 -118.42 -118.4  -118.39 -118.3
 -117.9  -118.31 -118.05 -117.69 -117.02 -116.22 -119.02 -119.03 -119.05
 -119.04 -119.08 -119.07 -119.09 -119.11 -119.01 -118.99 -119.   -118.97
 -118.98 -118.96 -118.95 -118.92 -118.9  -118.87 -118.88 -118.93 -119.06
 -119.1  -119.13 -119.15 -119.16 -119.42 -119.19 -119.2  -119.18 -119.27
 -119.26 -119.38 -119.36 -119.35 -119.14 -119.29 -119.22 -119.23 -118.06
 -118.44 -118.47 -118.5  -118.59 -118.23 -118.33 -118.41 -118.48 -118.61
 -117.73 -117.66 -117.67 -117.68 -117.7  -117.64 -117.76 -117.81 -117.87
 -117.84 -117.74 -118.   -117.82 -117.79 -118.01 -117.98 -117.95 -117.99
 -118.27 -118.34 -117.65 -118.15 -118.17 -118.19 -118.16 -118.51 -118.66
 -118.46 -118.83 -118.85 -118.82 -119.66 -120.14 -120.12 -122.89 -122.9
 -122.91 -122.88 -123.07 -122.95 -122.92 -122.94 -122.99 -122.7  -122.87
 -122.86 -122.83 -122.79 -122.8  -122.78 -122.69 -122.73 -122.66 -122.65
 -122.52 -122.68 -122.63 -122.62 -122.61 -122.6  -122.64 -122.75 -122.71
 -122.85 -122.84 -122.77 -122.72 -122.48 -122.59 -122.5  -122.55 -121.11
 -121.03 -120.64 -120.49 -120.2  -118.28 -118.29 -118.35 -118.32 -118.36
 -118.38 -118.49 -118.52 -118.54 -118.55 -118.57 -118.53 -118.63 -118.62
 -118.64 -118.56 -118.58 -118.37 -118.65 -118.22 -118.2  -118.21 -118.24
 -118.25 -118.26 -117.71 -117.78 -117.8  -117.83 -117.93 -117.91 -117.89
 -117.88 -117.85 -117.86 -117.77 -117.75 -117.72 -117.92 -117.94 -117.97
 -117.96 -118.02 -118.03 -118.04 -118.07 -118.08 -118.09 -118.1  -118.11
```

```
 -118.12 -118.13 -118.14 -118.69 -118.67 -118.68 -118.76 -118.75 -118.72
 -118.78 -118.8  -118.84 -118.79 -118.74 -118.86 -118.7  -119.51 -120.26
 -120.29 -120.11 -122.49 -122.54 -122.58 -122.57 -122.56 -122.44 -122.45
 -122.47 -122.46 -122.93 -122.96 -122.81 -120.15 -123.15 -123.24 -123.23
 -123.47 -123.71 -123.58 -123.5  -123.64 -123.79 -123.8  -123.34 -123.4
 -123.32 -123.38 -123.35 -123.37 -123.36 -123.1  -123.11 -123.18 -123.22
 -123.21 -123.19 -123.2  -123.81 -123.7  -123.53 -123.69 -123.59 -123.54
 -123.39 -123.17 -123.16 -120.68 -120.75 -120.74 -120.73 -120.94 -120.89
 -120.61 -120.48 -120.47 -120.4  -120.24 -120.82 -121.16 -121.18 -119.3
 -121.43 -121.45 -121.42 -121.48 -121.31 -121.32 -121.33 -121.4  -121.23
 -121.25 -121.26 -121.12 -121.13 -121.2  -122.4  -121.1  -121.05 -121.22
 -121.14 -121.21 -121.15 -120.17 -120.23 -117.62 -117.6  -117.63 -117.55
 -117.59 -117.58 -117.49 -117.53 -117.61 -121.17 -121.27 -121.28 -121.3
 -121.29 -120.53 -117.35 -117.36 -117.37 -117.38 -117.39 -117.41 -117.4
 -117.44 -117.43 -117.42 -117.45 -117.5  -117.48 -117.47 -117.51 -117.52
 -117.56 -117.57 -117.46 -117.54 -117.33 -117.34 -117.26 -117.3  -117.28
 -117.32 -117.31 -117.29 -117.24 -117.23 -117.25 -117.21 -117.14 -117.27
 -117.22 -117.16 -117.13 -117.19 -117.17 -117.2  -117.07 -117.11 -117.08
 -117.18 -117.06 -117.09 -117.15 -117.12 -117.05 -116.96 -117.1  -116.99
 -116.91 -116.89 -116.95 -116.9  -116.92 -116.93 -116.94 -116.97 -116.98
 -117.01 -117.   -116.87 -116.88 -116.86 -117.04 -117.03 -116.79 -116.77
 -116.81 -116.75 -116.8  -116.71 -116.68 -116.74 -116.72 -116.48 -116.57
 -116.76 -116.42 -116.6  -116.69 -116.39 -116.51 -116.61 -116.44 -116.36
 -116.52 -116.53 -116.5  -116.47 -116.63 -116.54 -116.55 -116.49 -116.56
 -116.46 -116.43 -116.45 -116.4  -116.38 -116.33 -116.31 -116.37 -116.41
 -116.29 -116.3  -116.26 -116.25 -116.24 -116.21 -116.23 -116.2  -116.15
 -116.11 -116.17 -116.12 -115.84 -116.16 -116.19 -116.18 -116.08 -115.22
 -114.67 -114.49 -114.68 -114.56 -114.57 -114.59 -114.61 -114.6  -114.58
 -114.62 -121.38 -121.37 -121.35 -121.34 -115.93 -115.75 -116.14 -116.32
 -116.27 -116.35 -116.62 -116.73 -116.06 -116.09 -116.02 -115.85 -114.94
 -114.47 -114.31 -114.64 -116.85 -116.83 -116.82 -116.84 -116.78 -116.58
 -116.66 -116.67 -116.34 -116.28 -122.43 -120.52 -122.76 -123.26 -123.41
 -123.08 -122.67 -122.82 -123.04 -123.02 -122.98 -123.01 -123.   -122.97
 -123.03 -123.49 -123.25 -123.48 -123.28 -123.13 -123.12 -123.43 -119.37
 -118.73 -120.28 -119.17 -118.89 -118.81 -118.77 -118.71]
------------------------------------------------------------------------
----------------------
longitude: 844
latitude: [37.88 37.86 37.85 37.84 37.83 37.82 37.81 37.8  37.79 37.77 3
7.78 37.76
 37.75 37.74 37.73 37.9  37.89 37.87 37.72 37.71 37.7  37.69 37.68 37.64
 37.63 37.66 37.65 37.67 37.61 37.62 37.6  37.59 37.58 37.57 37.49 37.52
 37.56 37.55 37.54 37.53 37.51 37.48 37.47 37.5  38.69 38.72 38.52 38.48
 38.45 38.46 38.43 38.55 38.54 38.51 38.5  38.47 38.44 38.42 38.37 38.34
 38.32 38.26 38.38 38.4  38.39 38.36 38.31 39.76 39.78 39.77 39.74 39.75
 39.73 39.71 39.72 39.7  39.82 39.79 39.68 39.64 39.66 39.59 39.88 40.06
 39.97 39.86 39.83 39.8  39.69 39.61 39.65 39.55 39.52 39.53 39.6  39.54
 39.5  39.49 39.51 39.48 39.47 39.45 39.44 39.43 39.4  39.39 39.33 39.37
 39.35 39.34 39.32 39.36 39.38 39.42 39.41 38.15 38.12 38.09 38.07 37.97
 38.24 38.2  38.16 38.11 38.28 38.19 38.25 38.41 38.33 38.35 38.21 38.23
 38.29 39.03 38.99 39.   39.15 39.22 39.25 39.1  39.13 39.31 39.3  39.21
 38.03 38.04 38.   37.98 37.99 37.93 37.94 37.95 37.96 37.91 38.01 38.02
 38.05 37.92 38.06 41.8  41.75 41.77 41.78 41.73 41.76 41.74 41.95 41.92
 41.84 41.81 41.68 41.88 41.54 38.96 38.95 38.94 38.93 38.92 38.91 38.89
 38.9  38.88 38.87 38.84 38.76 38.86 39.06 39.04 39.01 38.81 38.83 38.85
 38.68 38.67 38.66 38.7  38.58 38.73 38.65 38.62 38.79 38.77 38.71 38.75
 38.74 38.6  38.61 38.57 38.53 38.8  38.63 36.73 36.74 36.72 36.75 36.71
 36.7  36.68 36.69 36.65 36.64 36.62 36.63 36.59 36.77 36.76 36.8  36.79
 36.81 36.78 36.66 36.83 36.85 36.82 36.84 36.86 36.91 36.87 36.94 36.88
 36.89 37.1  37.02 37.   37.25 37.13 37.09 37.12 37.11 37.06 36.61 36.6
```

```
 36.58 36.57 36.56 36.55 36.52 36.51 36.53 36.44 36.46 36.43 36.54 36.45
 36.34 36.16 36.21 36.19 36.29 36.18 36.14 36.15 36.13 36.49 36.97 40.8
 40.79 40.78 40.77 40.75 40.76 40.81 40.86 40.85 40.87 40.88 40.9  40.66
 40.91 41.03 41.32 41.09 41.11 41.3  41.01 41.36 41.13 41.06 41.04 40.97
 40.92 40.89 40.93 41.02 40.99 40.95 40.94 40.72 40.73 40.74 40.67 40.69
 40.62 40.6  40.59 40.55 40.57 40.48 40.58 40.5  40.44 40.47 40.45 40.54
 40.28 40.22 40.24 40.16 40.12 40.11 40.05 33.12 33.13 33.19 33.24 33.2
 33.09 33.04 32.99 32.96 32.98 32.97 32.82 32.76 32.86 32.81 32.85 32.84
 32.83 32.87 32.8  32.79 32.75 32.77 32.73 32.78 32.69 32.7  32.67 32.68
 33.33 32.93 32.74 33.41 33.4  33.38 33.32 33.36 33.34 33.3  33.28 33.26
 33.43 33.07 33.35 37.35 37.4  37.39 37.37 37.36 37.17 36.95 36.4  36.
 35.42 35.45 35.44 35.43 35.41 35.4  35.39 35.38 35.37 35.34 35.36 35.35
 35.32 35.33 35.3  35.31 35.27 35.22 35.24 35.28 34.82 34.83 34.81 34.95
 35.06 35.07 35.19 35.17 35.16 35.14 35.15 35.13 35.5  35.52 35.49 35.51
 35.6  35.55 35.58 35.59 35.62 35.65 35.76 35.64 35.68 35.67 35.78 35.77
 35.75 35.79 35.74 35.47 35.72 35.7  35.48 35.63 35.61 35.73 35.54 35.05
 35.03 35.21 35.12 35.08 35.1  34.92 34.86 34.99 35.   34.87 35.04 35.2
 35.26 35.23 36.41 36.37 36.38 36.32 36.31 36.3  36.33 36.27 36.28 36.35
 36.36 36.25 36.11 36.09 36.1  36.08 36.2  35.99 36.02 36.04 36.01 35.87
 35.91 39.23 39.18 39.17 39.14 39.12 39.11 39.09 39.08 39.02 39.07 39.05
 38.98 38.97 38.82 38.78 41.07 41.12 40.98 40.63 40.65 40.51 40.35 40.43
 40.42 40.41 40.36 40.37 40.38 40.29 40.31 40.17 40.26 39.92 34.27 34.26
 34.25 34.24 34.22 34.21 34.28 34.29 34.32 34.33 34.31 34.3  34.23 34.2
 34.19 34.18 34.17 34.16 34.15 34.14 34.13 34.12 34.11 34.1  34.09 34.08
 34.07 34.06 34.05 34.04 34.03 34.02 34.01 34.   33.99 33.98 33.97 33.94
 33.95 33.96 33.93 33.92 33.91 33.9  33.88 33.89 33.87 33.86 33.85 33.84
 33.83 33.82 33.81 33.8  33.79 33.78 33.77 33.76 33.75 33.74 33.73 33.72
 33.69 33.71 33.67 33.68 34.71 34.63 34.58 34.74 34.68 34.69 34.65 34.7
 34.67 34.72 34.76 34.77 34.66 34.62 34.73 34.59 34.57 34.53 34.61 34.6
 34.56 34.64 34.55 34.51 34.43 34.41 34.52 34.5  34.46 34.44 34.49 34.48
 34.45 34.4  34.47 34.42 34.39 34.38 34.37 34.36 37.34 37.32 37.29 37.21
 37.38 37.31 37.33 37.23 37.19 37.27 37.04 36.99 36.93 36.98 36.96 36.9
 36.92 38.1  38.14 38.13 38.08 38.18 37.43 37.46 39.81 39.67 39.46 39.26
 39.28 39.2  39.16 37.44 37.42 37.41 37.2  37.26 37.16 37.3  37.28 37.18
 37.01 37.08 37.07 37.05 37.03 41.31 41.35 41.82 41.79 41.4  41.61 36.67
 36.5  36.47 36.48 36.42 36.24 35.95 35.94 36.06 38.3  38.27 38.22 38.17
 38.49 38.56 38.59 39.24 39.19 39.27 39.29 33.61 33.62 33.6  33.63 33.59
 33.58 33.57 33.65 33.64 33.54 33.55 33.56 33.51 33.52 33.66 33.53 33.42
 33.47 33.45 33.46 33.44 33.48 33.49 33.5  33.7  39.93 39.9  39.95 39.96
 40.02 40.08 40.14 40.01 40.13 40.27 40.23 40.19 40.3  40.25 38.64 34.54
 34.34 34.91 34.9  34.89 34.35 34.85 34.84 34.98 34.75 34.88 34.94 34.96
 32.72 32.71 32.66 32.88 32.92 32.89 32.91 32.9  32.94 32.95 32.59 32.58
 32.57 32.56 32.55 32.54 32.63 32.65 32.64 32.62 32.61 32.6  33.01 33.
 33.02 33.03 33.05 33.06 33.08 33.17 33.16 33.15 33.1  33.11 33.18 33.14
 33.21 33.23 33.22 33.25 33.31 33.29 33.39 33.37 33.27 35.69 35.56 35.29
 35.25 35.11 35.18 35.02 35.46 37.45 37.24 34.93 34.97 37.22 37.15 37.14
 40.56 40.61 40.64 40.52 40.53 40.49 40.71 40.68 40.46 40.4  40.39 40.82
 39.56 39.63 41.86 41.66 41.6  41.7  41.69 41.72 41.63 41.53 41.48 41.46
 41.26 41.43 41.41 41.38 41.28 41.21 41.23 41.2  41.5  40.09 40.2  40.34
 40.32 40.18 40.15 40.07 40.03 39.94 39.91 41.15 36.39 36.22 36.23 36.12
 35.82 35.9  36.05 36.07 35.97 35.96 35.86 35.85 35.89 35.88]
----------------------------------------------------------------------------
----------------------
latitude: 862
housing_median_age: [41 21 52 42 50 40 49 48 51 43  2 46 26 20 17 36 19
23 38 35 10 16 27 39
 31 29 22 37 28 34 32 47 44 30 18 45 33 24 15 14 13 25  5 12  6  8  9  7
  3  4 11  1]
----------------------------------------------------------------------------
----------------------
```

```
housing_median_age: 52
total_rooms: [  880  7099  1467 ...  4598   272 10035]
--------------------------------------------------------------------------
-----------------------
total_rooms: 5926
total_bedrooms: [ 129. 1106.  190. ... 3008. 1857. 1052.]
--------------------------------------------------------------------------
-----------------------
total_bedrooms: 1923
population: [ 322 2401  496 ... 3060 2707 6912]
--------------------------------------------------------------------------
-----------------------
population: 3888
households: [ 126 1138  177 ... 1767 1832 1818]
--------------------------------------------------------------------------
-----------------------
households: 1815
median_income: [8.3252 8.3014 7.2574 ... 2.3598 2.3661 2.0943]
--------------------------------------------------------------------------
-----------------------
median_income: 12928
ocean_proximity: ['NEAR BAY' '<1H OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
--------------------------------------------------------------------------
-----------------------
ocean_proximity: 5
median_house_value: [452600 358500 352100 ... 425800 200700  47000]
--------------------------------------------------------------------------
-----------------------
median_house_value: 3842
```

1. Handle missing values :

   Fill the missing values with the mean of the respective column.

In [10]:
```
data['total_bedrooms'].mean()
```

Out[10]:
```
537.8705525375618
```

In [12]:
```
data['total_bedrooms'].fillna(data['total_bedrooms'].mean(),inplace=True)
```

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
ocean_proximity       0
median_house_value    0
dtype: int64
```

In [14]:
```
data.ocean_proximity.value_counts()
```

Out[14]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [17]:
```
data.columns
```

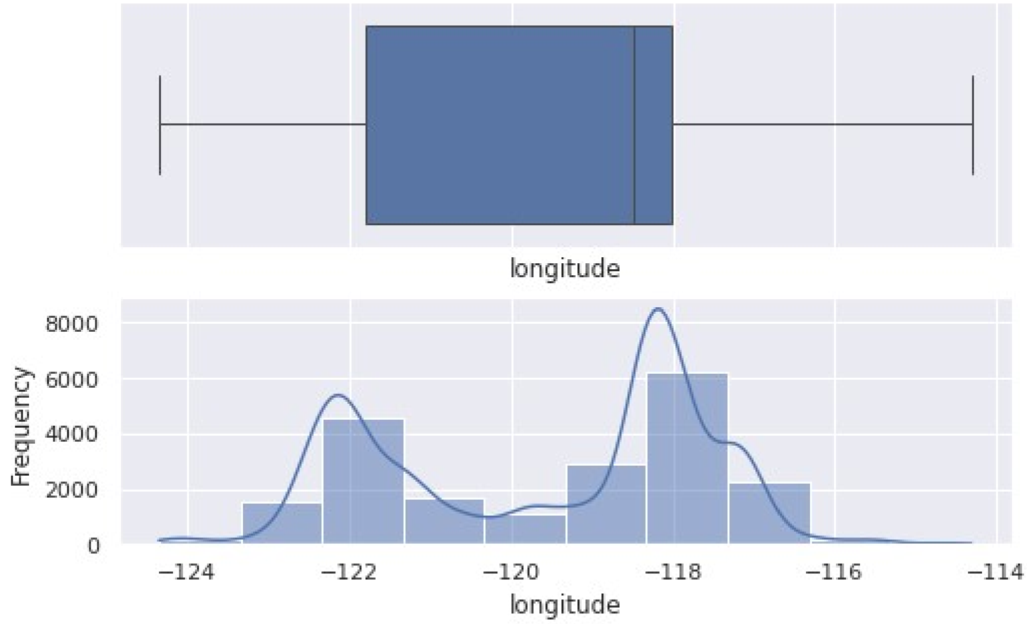
Out[17]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity', 'median_house_value'],
      dtype='object')
```

In [24]:
```
col = ['longitude', 'latitude', 'housing_median_age', 'total_rooms','tota
l_bedrooms', 'population', 'households', 'median_income','median_house_va
lue']
```

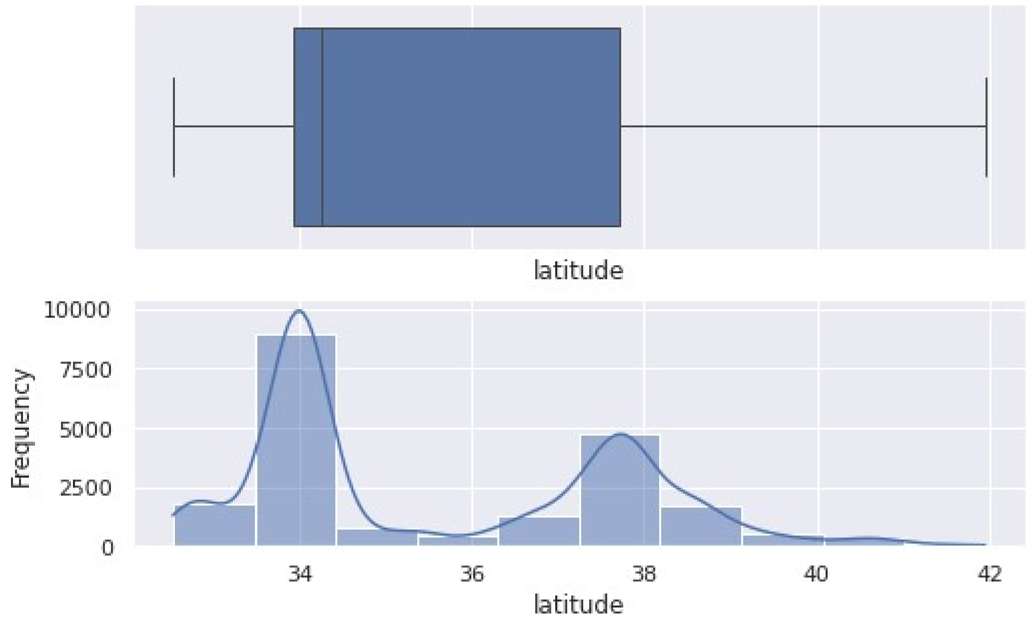
In [26]:
```
for i in col:
    plt.figure()
    plt.tight_layout()
    sns.set(rc={"figure.figsize":(8, 5)})

    f, (ax_box, ax_hist) = plt.subplots(2, sharex=True)
    plt.gca().set(xlabel= i,ylabel='Frequency')
    sns.boxplot(data[i], ax=ax_box , linewidth= 1.0)
    sns.histplot(data[i], ax=ax_hist , bins = 10,kde=True)
```

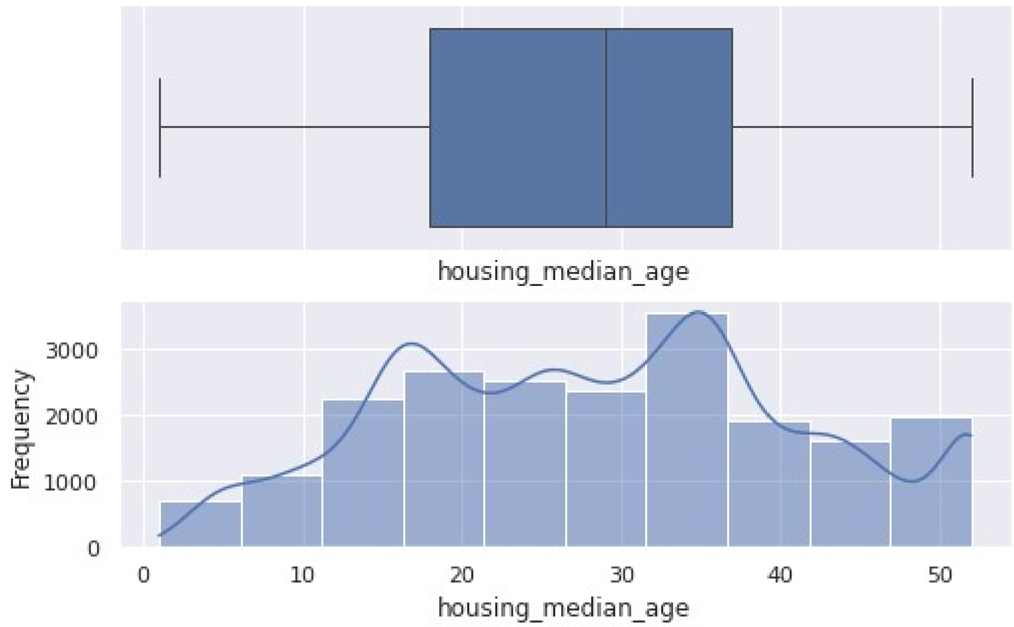
```
<Figure size 432x288 with 0 Axes>
```



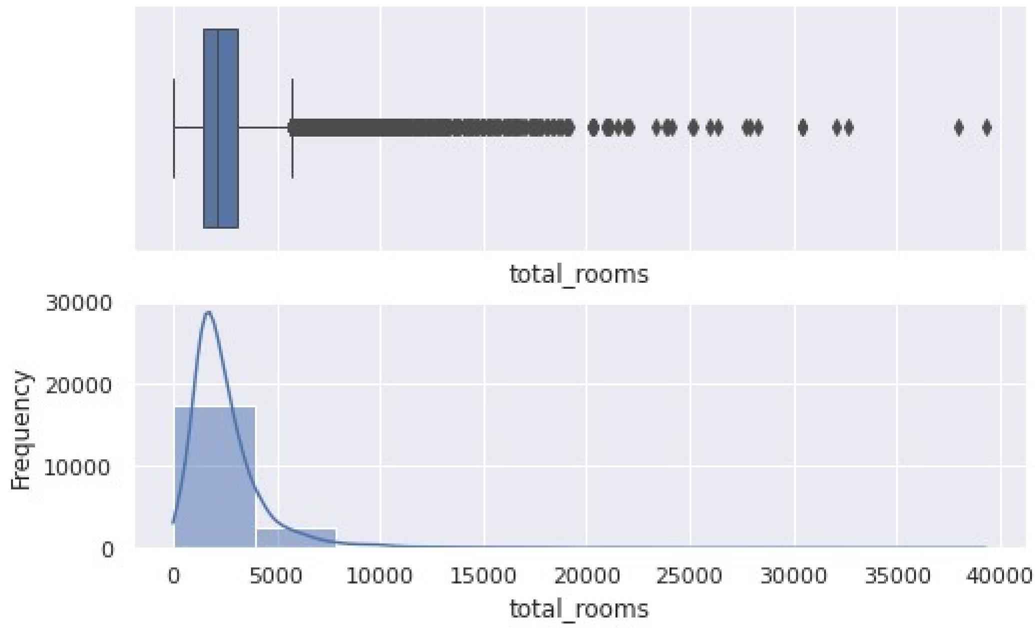

```
<Figure size 576x360 with 0 Axes>
```



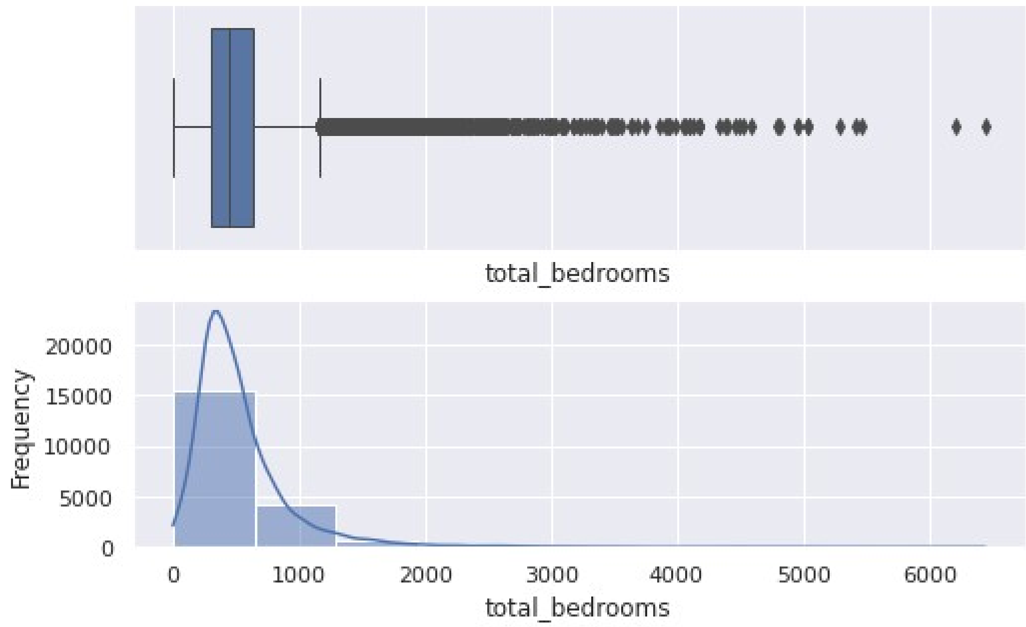

```
<Figure size 576x360 with 0 Axes>
```



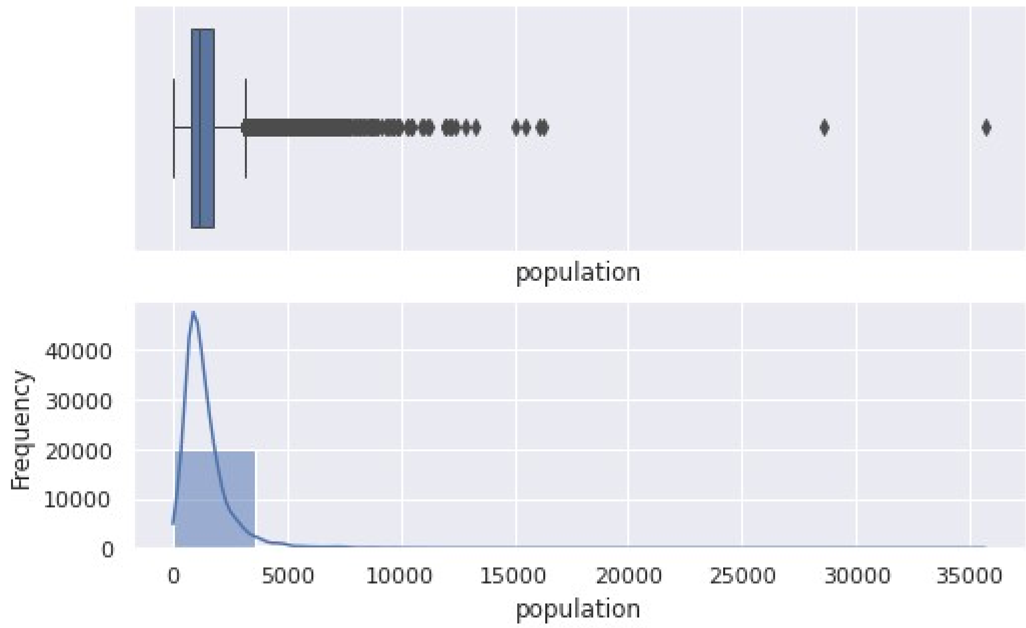
<Figure size 576x360 with 0 Axes>



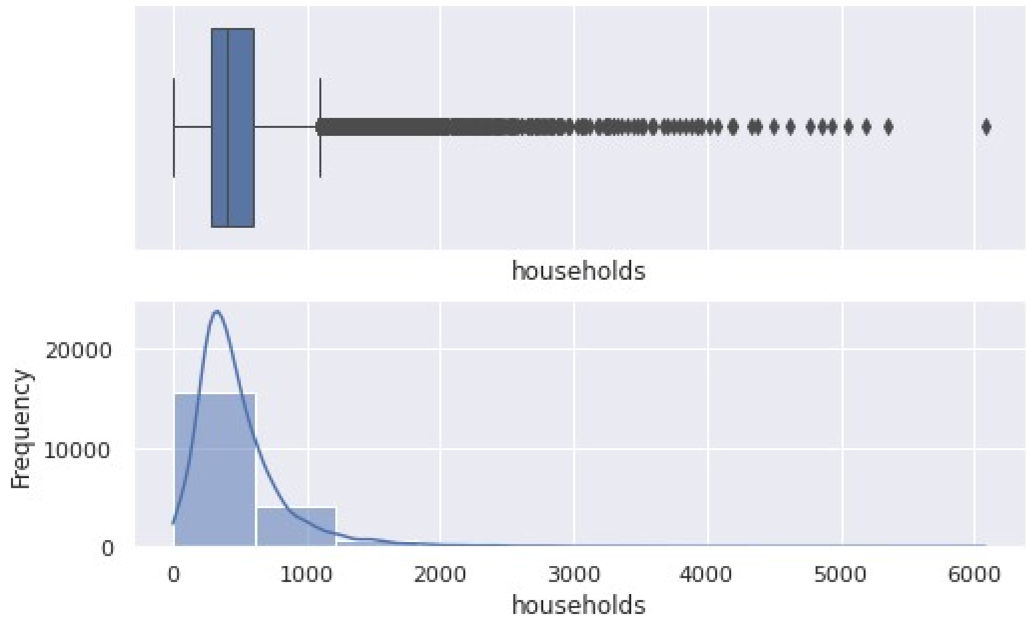
<Figure size 576x360 with 0 Axes>



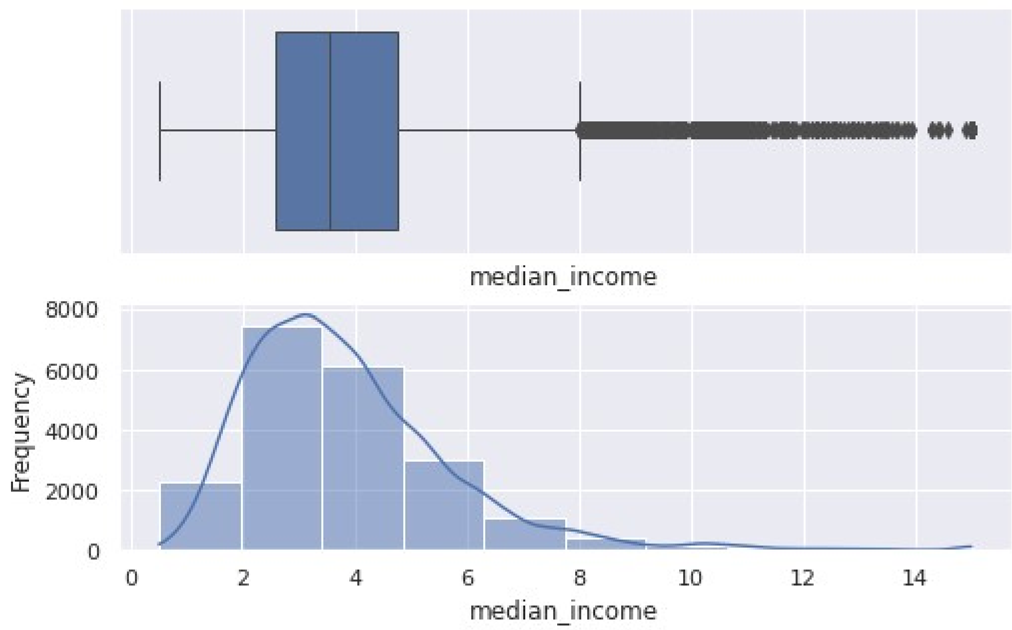

```
<Figure size 576x360 with 0 Axes>
```



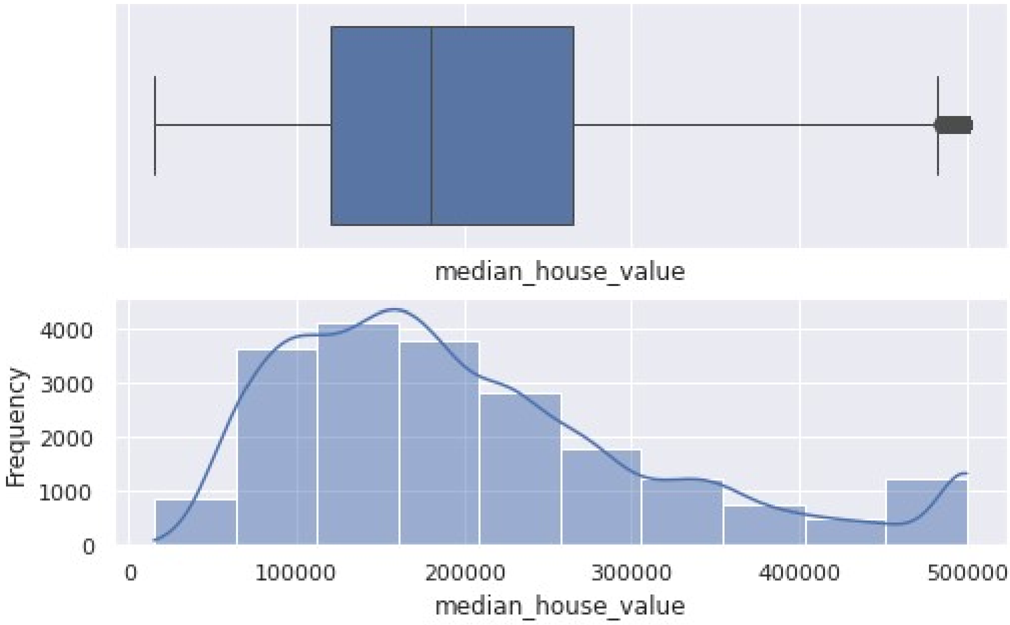

```
<Figure size 576x360 with 0 Axes>
```




```
<Figure size 576x360 with 0 Axes>
```




```
<Figure size 576x360 with 0 Axes>
```



1. Encode categorical data :

Convert categorical column in the dataset to numerical data.

In [30]:
```
from sklearn.preprocessing import LabelEncoder
```

In [32]:
```
X = data.drop(['median_house_value'],axis=1)
```

In [33]:
```
y = data.median_house_value
```

In [34]:
```
LE = LabelEncoder()
```

In [35]:
```
LE.fit(X['ocean_proximity'])
```

Out[35]:
```
LabelEncoder()
```

In [36]:
```
X['ocean_proximity'] = LE.fit_transform(X['ocean_proximity'])
```

In [37]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 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  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20640 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   ocean_proximity     20640 non-null  int64  
dtypes: float64(4), int64(5)
memory usage: 1.4 MB
```

1. Split the dataset :

   Split the data into 80% training dataset and 20% test dataset.

In [40]:
```
from sklearn.model_selection import train_test_split
```

In [41]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

In [42]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(16512, 9)
(4128, 9)
(16512,)
(4128,)
```

1. Standardize data :

   Standardize training and test datasets

In [43]:
```
from sklearn.preprocessing import StandardScaler
```

In [44]:
```
SC = StandardScaler()
```

In [45]:
```
SC.fit(X_train,X_test)
```

Out[45]:
```
StandardScaler()
```

In [46]:
```
X_train = SC.fit_transform(X_train)
X_test = SC.fit_transform(X_test)
```

In [47]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(16512, 9)
(4128, 9)
```

In [48]:
```
X_train
```

Out[48]:
```
array([[-1.33741695,  1.27101975,  0.18902153, ...,  0.09956798,
         0.88138919,  1.30166291],
       [-1.31237313,  1.01308135,  1.70239009, ..., -0.56544365,
        -0.29933418,  1.30166291],
       [ 0.5859483 , -0.75965892,  0.90588032, ..., -0.60749972,
         0.7055783 , -0.81629708],
       ...,
       [ 0.56090448, -0.67524272,  0.58727641, ...,  0.07591144,
        -0.86152815, -0.81629708],
       [-1.20218033,  0.7785919 , -0.0499314 , ...,  1.31130852,
         0.07992784, -0.81629708],
       [-1.46263604,  1.01777114,  1.86169204, ..., -0.7152684 ,
         2.80739673,  1.30166291]])
```

1. Perform Linear Regression :

Perform Linear Regression on training data. Predict output for test dataset using the fitted model. Print root mean squared error (RMSE) from Linear Regression.

`[ HINT: Import mean_squared_error from sklearn.metrics ]`

In [49]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,accuracy_score
from math import sqrt
```

In [50]:
```
LR = LinearRegression()
```

In [51]:
```
LR.fit(X_train,y_train)
```

Out[51]:
```
LinearRegression()
```

In [56]:
```
predict = LR.predict(X_test)
predict
```

Out[56]:
```
array([107699.19147644, 309142.2006786 , 162932.4329406 , ...,
       224910.24553595, 252001.55827049, 143627.788912  ])
```

In [54]:
```
print(LR.intercept_)
print(LR.coef_)
```

```
206620.9759568804
[-84765.67944816 -90137.21656387  14649.03732568 -16808.11270103
  43415.24539556 -42326.6922323   20477.18332267  75971.11557135
   -208.48533812]
```

In [61]:
```
X_test
```

Out[61]:
```
array([[-1.4974493 ,  1.53049515,  0.26050655, ..., -0.53683036,
        -1.40883376, -0.1421567 ],
       [ 0.71361666, -0.71178304, -0.762607  , ..., -0.57275269,
         1.41644561, -0.83826922],
       [ 0.72346551, -0.77691146,  1.04751698, ...,  0.00200459,
        -0.70855664, -0.83826922],
       ...,
       [-1.28077468,  0.9862077 ,  1.67712532, ...,  0.13029863,
        -0.52046222,  1.25006834],
       [ 1.31439628, -1.29793875, -0.36910179, ..., -0.3315599 ,
         0.71225312, -0.83826922],
       [ 0.65944801, -0.88390813, -0.60520492, ...,  1.56206008,
        -0.71189775, -0.83826922]])
```

In [55]:
```
print(sqrt(mean_squared_error(y_test,predict)))
```

```
69640.92784540239
```

In [65]:
```
LR.predict([X_test[0]])
```

Out[65]:
```
array([107699.19147644])
```

1. Perform Decision Tree Regression :

Perform Decision Tree Regression on training data. Predict output for test dataset using the fitted model. Print root mean squared error from Decision Tree Regression.

In [66]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [67]:
```
DR = DecisionTreeRegressor(random_state=0)
```

In [68]:
```
DR.fit(X_train,y_train)
```

Out[68]:
```
DecisionTreeRegressor(random_state=0)
```

In [69]:
```
DR_predict = DR.predict(X_test)
DR_predict
```

Out[69]:
```
array([156700., 266700., 196000., ..., 218400., 167500., 113700.])
```

In [71]:
```
print(sqrt(mean_squared_error(y_test,DR_predict)))
```

```
82322.42765969141
```

1. Perform Random Forest Regression :

Perform Random Forest Regression on training data. Predict output for test dataset using the fitted model. Print RMSE (root mean squared error) from Random Forest Regression.

In [72]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [73]:
```
RF = RandomForestRegressor(n_estimators=100,random_state=0)
```

In [74]:
```
RF.fit(X_train,y_train)
```

Out[74]:
```
RandomForestRegressor(random_state=0)
```

In [76]:
```
RF_predicts = RF.predict(X_test)
RF_predicts
```

Out[76]:
```
array([ 91121.   , 338639.06, 166656.   , ..., 190879.   , 172548.   ,
       189832.   ])
```

In [77]:
```
print(sqrt(mean_squared_error(y_test,RF_predicts)))
```

```
60403.898959786944
```

1. Bonus exercise: Perform Linear Regression with one independent variable :

   Extract just the median_income column from the independent variables (from X_train and X_test). Perform Linear Regression to predict housing values based on median_income. Predict output for test dataset using the fitted model. Plot the fitted model for training data as well as for test data to check if the fitted model satisfies the test data.

In [85]:
```
x = np.array(data['median_income']).reshape(-1,1)
```

In [86]:
```
x.shape
```

Out[86]:
```
(20640, 1)
```

In [89]:
```
x
```

Out[89]:
```
array([[8.3252],
       [8.3014],
       [7.2574],
       ...,
       [1.7   ],
       [1.8672],
       [2.3886]])
```

In [87]:
```
x_new_train,x_new_test,y_train,y_test = train_test_split(x,y,test_size=0.
2)
```

In [88]:
```
print(x_new_train.shape)
print(x_new_test.shape)
```

```
(16512, 1)
(4128, 1)
```

In [90]:
```
LR.fit(x_new_train,y_train)
```

Out[90]:
```
LinearRegression()
```

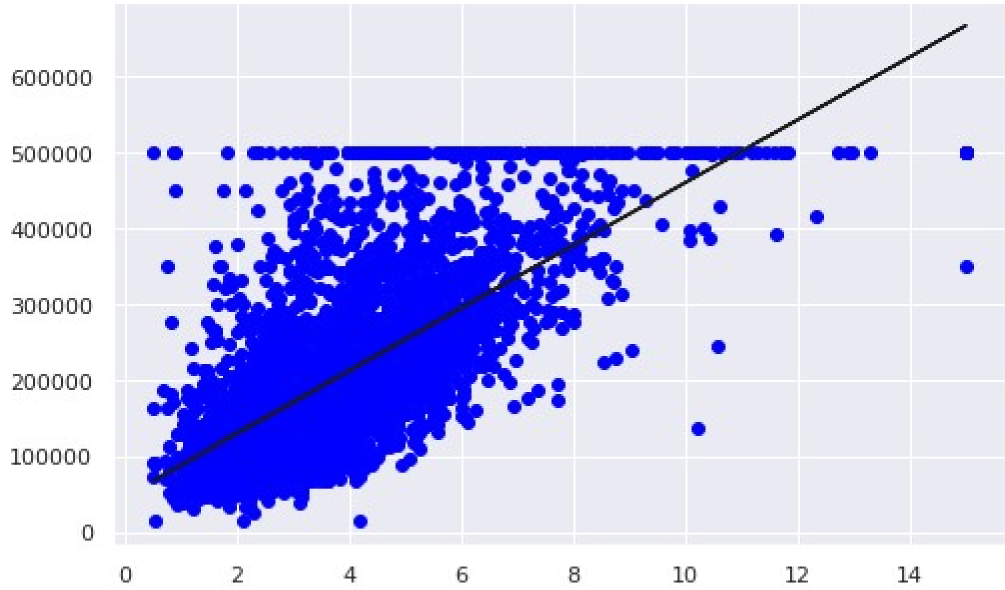
In [91]:
```
new_predicts = LR.predict(x_new_test)
new_predicts
```

Out[91]:
```
array([116505.77594462, 256554.44392199, 222331.63612798, ...,
       102771.13276828, 180068.33068659, 364689.41253934])
```

In [92]:
```
print(sqrt(mean_squared_error(y_test,new_predicts)))
```

```
83080.01475483587
```

In [94]:
```
plt.scatter(x_new_test,y_test,color='blue')
plt.plot(x_new_test,new_predicts,color='k')
plt.show()
```



In [ ]: