In [1]:
```
# import libraries
import pandas as pd
import os
```

In [2]:
```
os.getcwd()
```

Out[2]:
```
'C:\\Users\\utpala mohapatra\\Documents\\python_folder'
```

In [3]:
```
os.chdir('C:\\Users\\utpala mohapatra\\Documents\\python_folder\\python datasets')
```

In [4]:
```
#1. Load the data :
#-----------------------------

#Read the "housing.csv" file from the folder into the program.
#Print first few rows of this data.

house_df = pd.read_excel('1553768847_housing.xlsx')
house_df.head()
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	median_h
0	-122.23	37.88	41	880	129.0	322	126	8.3252	NEAR BAY	
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	NEAR BAY	
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	NEAR BAY	
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	NEAR BAY	
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	NEAR BAY	

In [5]:
```
house_df.columns
```

Out[5]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity', 'median_house_value'],
      dtype='object')
```

In [6]:
```
#2. Handle missing values :
#-----------------------------

#checking for missing values
house_df.isnull().any(axis=1).sum()
```

Out[6]:
```
207
```

In [9]:
```
# If there is  any missing value , replace it with the Column mean.
house_df_means = house_df.mean()
house_df = house_df.fillna(house_df_means)
house_df
```

Out[9]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	medi
0	-122.23	37.88	41	880	129.0	322	126	8.3252	NEAR BAY	
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	NEAR BAY	
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	NEAR BAY	
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	NEAR BAY	
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	NEAR BAY	
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25	1665	374.0	845	330	1.5603	INLAND	
20636	-121.21	39.49	18	697	150.0	356	114	2.5568	INLAND	
20637	-121.22	39.43	17	2254	485.0	1007	433	1.7000	INLAND	
20638	-121.32	39.43	18	1860	409.0	741	349	1.8672	INLAND	
20639	-121.24	39.37	16	2785	616.0	1387	530	2.3886	INLAND	

20640 rows × 10 columns

In [20]:
```
#3. Encode categorical data :
#-----------------------------
#check the types of categorical variable

house_df.ocean_proximity.unique()
```

Out[20]:
```
array(['NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND'],
      dtype=object)
```

In [11]:
```
# create new numeric variable for ocean_proximity

house_df['ocean_proximity_num']=house_df.ocean_proximity.map({'NEAR BAY':0, '<1H OCEAN':1, 'INLAND':2, 'NEAR OC
house_df
```

Out[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	medi
0	-122.23	37.88	41	880	129.0	322	126	8.3252	NEAR BAY	
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	NEAR BAY	
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	NEAR BAY	
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	NEAR BAY	
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	NEAR BAY	
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25	1665	374.0	845	330	1.5603	INLAND	
20636	-121.21	39.49	18	697	150.0	356	114	2.5568	INLAND	
20637	-121.22	39.43	17	2254	485.0	1007	433	1.7000	INLAND	
20638	-121.32	39.43	18	1860	409.0	741	349	1.8672	INLAND	
20639	-121.24	39.37	16	2785	616.0	1387	530	2.3886	INLAND	

20640 rows × 11 columns

In [14]:
```
#Extract input (X) and output (Y) data from the dataset.

from sklearn.linear_model import LinearRegression
```

In [12]:
```
x = pd.DataFrame(house_df.drop(['median_house_value','ocean_proximity'],axis=1))
x.head(10)
```

Out[12]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity_num
0	-122.23	37.88	41	880	129.0	322	126	8.3252	0
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	0
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	0
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	0
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	0
5	-122.25	37.85	52	919	213.0	413	193	4.0368	0
6	-122.25	37.84	52	2535	489.0	1094	514	3.6591	0
7	-122.25	37.84	52	3104	687.0	1157	647	3.1200	0
8	-122.26	37.84	42	2555	665.0	1206	595	2.0804	0
9	-122.25	37.84	52	3549	707.0	1551	714	3.6912	0

In [13]:
```
y = house_df['median_house_value']
y
```

Out[13]:
```
0        452600
1        358500
2        352100
3        341300
4        342200
          ...
20635     78100
20636     77100
20637     92300
20638     84700
20639     89400
Name: median_house_value, Length: 20640, dtype: int64
```

In [15]:
```
#4. Split the dataset :
#-----------------------------------

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

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.20)
```

In [16]:
```
x_train.shape
```

Out[16]:
```
(16512, 9)
```

In [17]:
```
y_train.shape
```

Out[17]:
```
(16512,)
```

In [18]:
```
x_test.shape
```

Out[18]:
```
(4128, 9)
```

In [19]:
```
y_test.shape
```

Out[19]:
```
(4128,)
```

In [21]:
```
x_train.isnull().any(axis=1).sum()
```

Out[21]:
```
0
```

In [24]:
```
x_test.isnull().any(axis=1).sum()
```

Out[24]:
```
0
```

In [30]:
```
#5. Standardize data :
#-----------------------------------

#Standardize training and test datasets.

from sklearn import preprocessing
train_scaler = preprocessing.StandardScaler().fit(x_train)
x_train_scaled = train_scaler.transform(x_train)
x_train_scaled
x_test_scaled = train_scaler.transform(x_test)
x_test_scaled
```

Out[30]:
```
array([[ 1.15249782, -0.84026056,  1.31758597, ...,  0.8925193 ,
         -0.60357393,  0.63657238],
        [-1.19195455,  0.75973767,  0.50403201, ..., -0.85319078,
          1.20093205, -0.53523176],
        [ 0.25961049,  0.21545675, -1.55518831, ...,  0.17983825,
          0.07725795,  0.63657238],
        ...,
        [-0.67318212,  1.84360744,  1.53366217, ..., -0.92922591,
         -1.0483037 ,  0.63657238],
        [-0.13903342,  0.37967944, -0.84238128, ..., -0.73258333,
         -0.88990608,  0.63657238],
        [-0.59337097,  1.39316806, -1.1591844 , ..., -0.1636308 ,
         -0.67524873,  0.63657238]])
```

In [31]:
```
#6. Perform Linear Regression :
#-----------------------------------

#Perform Linear Regression on training data.

lm_house = LinearRegression().fit(x_train_scaled,y_train)
```

In [32]:
```
print(house_df.columns,lm_house.coef_)
print(lm_house.intercept_)
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity', 'median_house_value'],
      dtype='object') [-83185.52054642 -88823.95175485  13993.06753142 -14718.49542039
  37076.88130873 -44114.9577891   24443.2703696   75482.82626535
  -3019.08042946] 206921.81274227448
```

In [34]:
```
#Predict output for test dataset using the fitted model.

pred_house_price = lm_house.predict(x_test_scaled)
pred_house_price
```

Out[34]:
```
array([112048.24441979, 330161.02133485, 136606.42026473, ...,
        43864.31676901,  98421.17311607,  72529.66350603])
```

In [35]:
```
#Print root mean squared error (RMSE) from Linear Regression.

from sklearn.metrics import mean_squared_error
from math import sqrt
print(sqrt(mean_squared_error(y_test,pred_house_price)))
```

```
68167.11862510696
```

In [46]:
```
#7. Bonus exercise: Perform Linear Regression with one independent variable :
#-----------------------------------------------------------------------------
#Extract just the median_income column from the independent variables

feature_col=['median_income']
x= house_df[feature_col]
y=house_df['median_house_value']
X_train,X_test,Y_train,Y_test= train_test_split(x,y,test_size=0.2)
```

In [49]:
```
#Perform Linear Regression to predict housing values based on median_income.
lm= LinearRegression()
lm.fit(X_train,Y_train)
print(lm.coef_)
print(lm.intercept_)
```

```
[41740.10981192]
45547.763493947978
```

In [49]:
```
#Predict output for test dataset using the fitted model.

pred_house_pricel = lm.predict(X_test)
pred_house_pricel
```

Out[49]:
```
array([322514.26215048, 162941.8223395 , 160333.04547626, ...,
       242322.16317981, 214553.46812194, 183377.78010342])
```

In [68]:
```
#Plot the fitted model for training data as well as for test data to check if the fitted model satisfies the te

import matplotlib.pyplot as plt

plt.figure(figsize=(10,10))
plt.plot(X_test,pred_house_pricel,'r')
plt.scatter(X_train,Y_train,color='blue')
plt.title('Training & Test ')
plt.xlabel('Median Income')
plt.ylabel('Median House Value')
plt.show()
```