In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

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

Out[2]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41	880	129.0	322	12
1	-122.22	37.86	21	7099	1106.0	2401	113
2	-122.24	37.85	52	1467	190.0	496	17
3	-122.25	37.85	52	1274	235.0	558	21
4	-122.25	37.85	52	1627	280.0	565	25

In [3]:

```
data.columns
```

Out[3]:

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

In [4]:

```
data.shape
```

Out[4]:

```
(20640, 10)
```

In [5]:

```
data.isnull().any()
```

Out[5]:

```
longitude             False
latitude              False
housing_median_age    False
total_rooms           False
total_bedrooms         True
population            False
households            False
median_income         False
ocean_proximity       False
median_house_value    False
dtype: bool
```

In [6]:

```
data.isna().sum()
```

Out[6]:

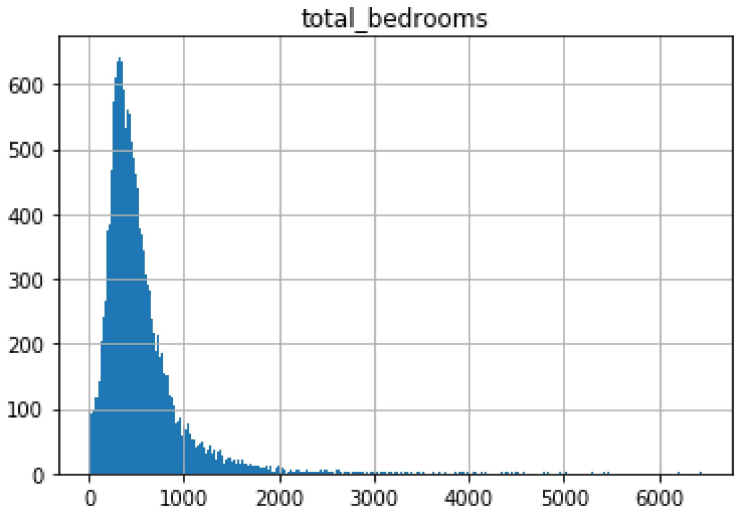
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

In [7]:

```
#Hist plot
data.hist(column='total_bedrooms',bins=400)
```

Out[7]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000028E800C7B00
>]],
      dtype=object)
```



In [8]:

```
data[data['total_bedrooms'].isnull()]
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	media
290	-122.16	37.77	47	1256	NaN	570	218	
341	-122.17	37.75	38	992	NaN	732	259	
538	-122.28	37.78	29	5154	NaN	3741	1273	
563	-122.24	37.75	45	891	NaN	384	146	
696	-122.10	37.69	41	746	NaN	387	161	
738	-122.14	37.67	37	3342	NaN	1635	557	
1097	-121.77	39.66	20	3759	NaN	1705	600	
1350	-121.95	38.03	5	5526	NaN	3207	1012	
1456	-121.98	37.96	22	2987	NaN	1420	540	
1493	-122.01	37.94	23	3741	NaN	1339	499	

In [9]:

```
from sklearn.preprocessing import Imputer
imputer_mean = Imputer(missing_values='NaN',strategy='mean',axis=0)
data['total_bedrooms']=imputer_mean.fit_transform(data[['total_bedrooms']])
```

```
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:58: DeprecationWarning: Class Imputer is deprecated; Imputer was deprecate
d in version 0.20 and will be removed in 0.22. Import impute.SimpleImputer f
rom sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

In [10]:

```
data.isnull().sum()
```

Out[10]:

```
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 [11]:

```
data.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 int64
total_rooms           20640 non-null int64
total_bedrooms        20640 non-null float64
population            20640 non-null int64
households            20640 non-null int64
median_income         20640 non-null float64
ocean_proximity       20640 non-null object
median_house_value    20640 non-null int64
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

In [12]:

```
data['ocean_proximity'].unique()
```

Out[12]:

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

In [13]:

```
data=pd.get_dummies(data,drop_first=True)
data
```

Out[13]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	media
0	-122.23	37.88	41	880	129.0	322	126	
1	-122.22	37.86	21	7099	1106.0	2401	1138	
2	-122.24	37.85	52	1467	190.0	496	177	
3	-122.25	37.85	52	1274	235.0	558	219	
4	-122.25	37.85	52	1627	280.0	565	259	
5	-122.25	37.85	52	919	213.0	413	193	
6	-122.25	37.84	52	2535	489.0	1094	514	
7	-122.25	37.84	52	3104	687.0	1157	647	
8	-122.26	37.84	42	2555	665.0	1206	595	

In [14]:

```
data.columns
```

Out[14]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity_INLAND',
       'ocean_proximity_ISLAND', 'ocean_proximity_NEAR BAY',
       'ocean_proximity_NEAR OCEAN'],
      dtype='object')
```

In [15]:

```
X=data[['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income', 'ocean_proximity_INLA
       'ocean_proximity_ISLAND', 'ocean_proximity_NEAR BAY',
       'ocean_proximity_NEAR OCEAN']]
Y=data[['median_house_value']]
```

In [16]:

```
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=0)
```

In [17]:

```
X_train.head()
```

Out[17]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	house
12069	-117.55	33.83	6	502	76.0	228	
15925	-122.44	37.73	52	2381	492.0	1485	
11162	-118.00	33.83	26	1718	385.0	1022	
4904	-118.26	34.01	38	697	208.0	749	
4683	-118.36	34.08	52	2373	601.0	1135	

In [18]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
Y_train=sc.fit_transform(Y_train)
Y_test=sc.transform(Y_test)
```

```
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\preprocessing\data.
py:625: DataConversionWarning: Data with input dtype uint8, int64, float64 w
ere all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\base.py:462: DataCo
nversionWarning: Data with input dtype uint8, int64, float64 were all conver
ted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: Da
taConversionWarning: Data with input dtype uint8, int64, float64 were all co
nverted to float64 by StandardScaler.
  after removing the cwd from sys.path.
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\preprocessing\data.
py:625: DataConversionWarning: Data with input dtype int64 were all converte
d to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\base.py:462: DataCo
nversionWarning: Data with input dtype int64 were all converted to float64 b
y StandardScaler.
  return self.fit(X, **fit_params).transform(X)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Da
taConversionWarning: Data with input dtype int64 were all converted to float
64 by StandardScaler.
```

Linear Regression :

In [19]:

```
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(X_train,Y_train)
```

Out[19]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [20]:

```
pd.DataFrame(lin_reg.predict(X_test))
```

Out[20]:

	0
0	0.080653
1	0.693263
2	-0.240235
3	-1.043881
4	0.730474
5	-0.038912
6	0.645055
7	0.933200
8	0.864888
9	0.299548

In [21]:

```
#Accurecy Of Train data
lin_reg.score(X_train,Y_train)
```

Out[21]:

```
0.6471730344800684
```

In [22]:

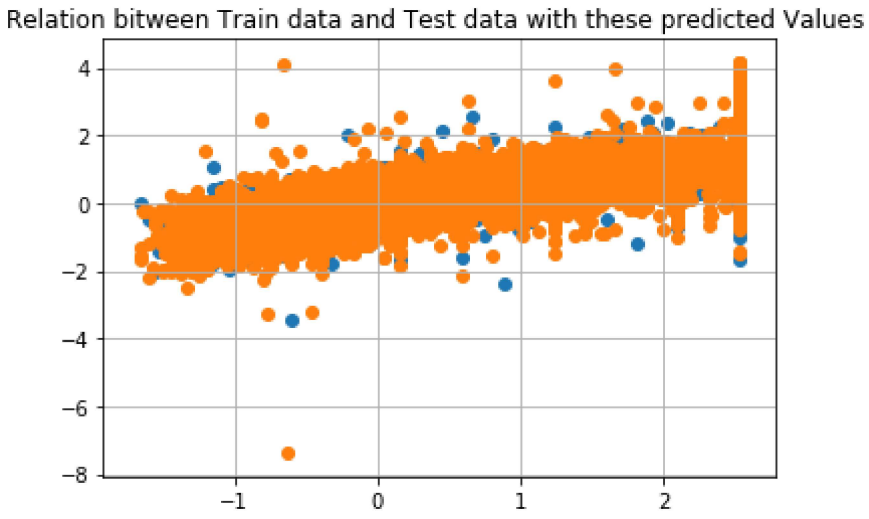
```
#Accurecy Of Test data
lin_reg.score(X_test,Y_test)
```

Out[22]:

```
0.6381617983930402
```

In [23]:

```
plt.scatter(Y_test,lin_reg.predict(X_test))
plt.scatter(Y_train,lin_reg.predict(X_train))
plt.title('Relation bitween Train data and Test data with these predicted Values ')
plt.grid()
plt.show()
```



In [24]:

```
from sklearn.metrics import mean_squared_error
linear_RMSE=mean_squared_error(Y_train,lin_reg.predict(X_train))
linear_RMSE
```

Out[24]:

```
0.3528269655199316
```

Decision Tree Regression :

In [25]:

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
dt.fit(X_train,Y_train)
```

Out[25]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [26]:

```
pd.DataFrame(dt.predict(X_test))
```

Out[26]:

	0
0	-0.583033
1	1.937540
2	-0.405832
3	-0.959044
4	2.530521
5	-1.272819
6	0.252836
7	1.653154
8	0.405834
9	0.517341

In [27]:

```
#Accurecy Of Train data
dt.score(X_train,Y_train)
```

Out[27]:

```
0.9999999999999637
```

In [28]:

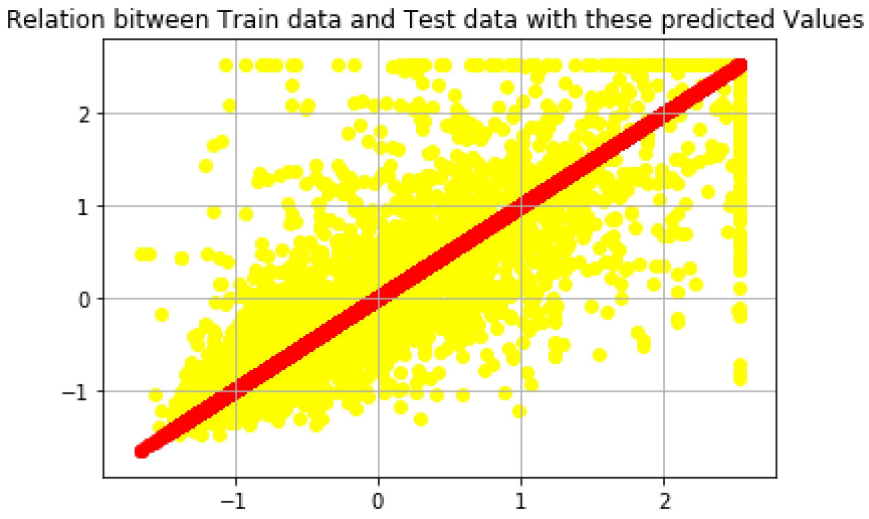
```
#Accurecy Of Test data
dt.score(X_test,Y_test)
```

Out[28]:

```
0.6362711896456079
```

In [29]:

```
plt.scatter(Y_test,dt.predict(X_test),c='yellow')
plt.scatter(Y_train,dt.predict(X_train),c='red')
plt.title('Relation bitween Train data and Test data with these predicted Values ')
plt.grid()
plt.show()
```



In [30]:

```
from sklearn.metrics import mean_squared_error
dt_RMSE=mean_squared_error(Y_train,dt.predict(X_train))
dt_RMSE
```

Out[30]:

```
3.6273880121276177e-14
```

Random Forest Regression :

In [31]:

```
from sklearn.ensemble import RandomForestRegressor
reg_svm=RandomForestRegressor()
reg_svm.fit(X_train,Y_train)
```

```
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:
246: FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Da
taConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports
until
```

Out[31]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [32]:

```
pd.DataFrame(reg_svm.predict(X_test))
```

18	-0.306772
19	1.879799
20	-0.514832
21	-1.265817
22	0.249033
23	0.651928
24	-0.081598
25	-0.364514
26	0.098801
27	-0.074856
28	-0.746922
29	-1.183614

In [33]:

```
#Accurecy Of Train data
reg_svm.score(X_train,Y_train)
```

Out[33]:

```
0.9652231267374025
```

In [34]:

```
#Accurecy Of Test data
reg_svm.score(X_test,Y_test)
```

Out[34]:

```
0.8075864957872383
```

In [35]:

```
plt.scatter(Y_test,reg_svm.predict(X_test))
plt.scatter(Y_train,reg_svm.predict(X_train))
plt.title('Relation bitween Train data and Test data with these predicted Values ')
plt.grid()
plt.show()
```

In [36]:

```
from sklearn.metrics import mean_squared_error
rf_RMSE=mean_squared_error(Y_train,reg_svm.predict(X_train))
rf_RMSE
```

Out[36]:

```
0.03477687326259753
```

RMSE for Liner Regression, Decission Tree and Random Forrest as follows

In [37]:

```
## RMSE for Liner Regression, Decission Tree and Random Forrest as follows
print("RMSE for Linear Regression:{}".format(linear_RMSE))
print("RMSE for Decission Tree:{}".format(dt_RMSE))
print("RMSE for Random Forrest:{}".format(rf_RMSE))
```

```
RMSE for Linear Regression:0.3528269655199316
RMSE for Decission Tree:3.6273880121276177e-14
RMSE for Random Forrest:0.03477687326259753
```

Bonus exercise:

In [38]:

```
## Importing dataset
mydata = pd.read_excel("1553768847_housing.xlsx")
```

In [39]:

```
mydata.head()
```

Out[39]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41	880	129.0	322	12
1	-122.22	37.86	21	7099	1106.0	2401	113
2	-122.24	37.85	52	1467	190.0	496	17
3	-122.25	37.85	52	1274	235.0	558	21
4	-122.25	37.85	52	1627	280.0	565	25

In [40]:

```
mydata.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 int64
total_rooms           20640 non-null int64
total_bedrooms        20433 non-null float64
population            20640 non-null int64
households            20640 non-null int64
median_income         20640 non-null float64
ocean_proximity       20640 non-null object
median_house_value    20640 non-null int64
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

In [41]:

```
#Get number of Null Values
mydata.isnull().sum()
```

Out[41]:

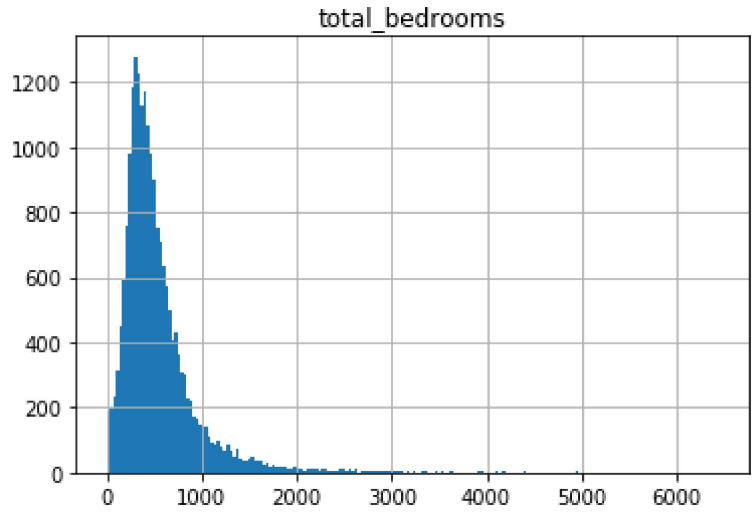
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

In [42]:

```
#Hist plot
mydata.hist(column='total_bedrooms',bins=200)
```

Out[42]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000028E8547B438
>]],
      dtype=object)
```



In [43]:

```
#Imputation
#From hist plot we could say data is right sweked, so it's better to apply Median to replac
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values="NaN",strategy="mean",axis=0)
mydata[['total_bedrooms']]=imputer.fit_transform(mydata[['total_bedrooms']])
```

```
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:58: DeprecationWarning: Class Imputer is deprecated; Imputer was deprecate
d in version 0.20 and will be removed in 0.22. Import impute.SimpleImputer f
rom sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

In [44]:

```
mydata.isnull().sum()
```

Out[44]:

```
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 [45]:

```
iv = mydata[['median_income']]
dv = mydata[['median_house_value']]
```

In [46]:

```
from sklearn.model_selection import train_test_split
iv_train_med,iv_test_med,dv_train_med,dv_test_med=train_test_split(iv,dv,test_size=0.2,rand
```

In [47]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
iv_train_med=sc.fit_transform(iv_train_med)
iv_test_med=sc.transform(iv_test_med)
dv_train_med=sc.fit_transform(dv_train_med)
dv_test_med=sc.transform(dv_test_med)
```

```
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\preprocessing\data.
py:625: DataConversionWarning: Data with input dtype int64 were all converte
d to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\sklearn\base.py:462: DataCo
nversionWarning: Data with input dtype int64 were all converted to float64 b
y StandardScaler.
  return self.fit(X, **fit_params).transform(X)
C:\Users\RAJ KHATANA\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Da
taConversionWarning: Data with input dtype int64 were all converted to float
64 by StandardScaler.
```

In [48]:

```
from sklearn.linear_model import LinearRegression
med_linear = LinearRegression()
med_linear.fit(iv_train_med,dv_train_med)
```

Out[48]:

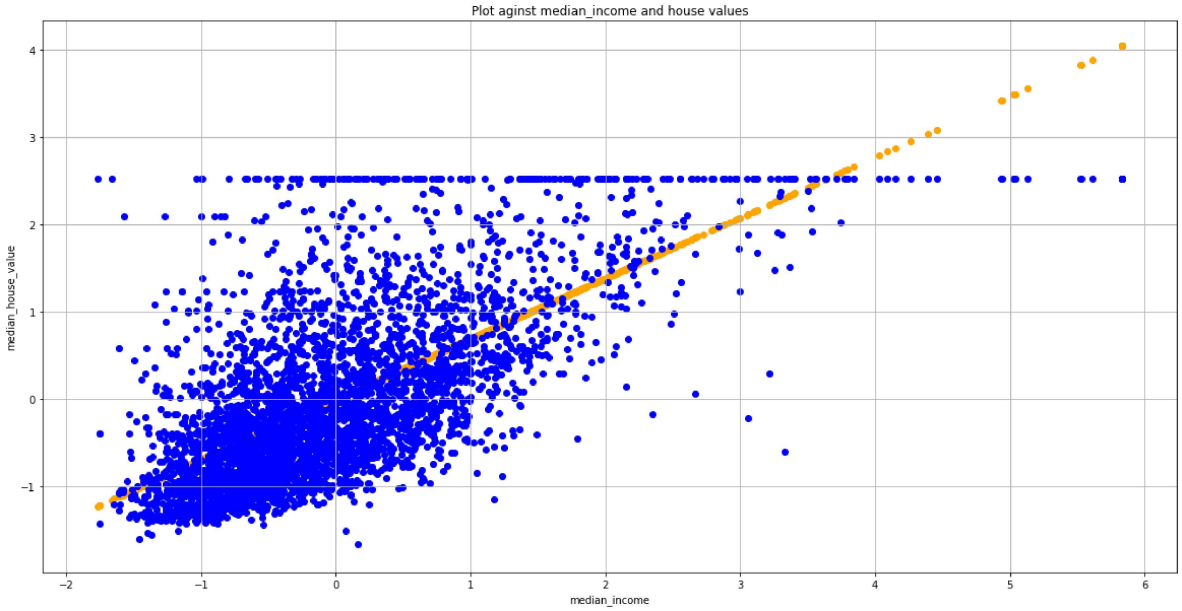
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [49]:

```
y_pred_med = med_linear.predict(iv_test_med)
```

In [50]:

```
#Plotting Test data and Predicted data
get_ipython().run_line_magic('matplotlib', 'inline')
plt.figure(figsize=(20,10))
plt.scatter(iv_test_med, y_pred_med, color='orange')
plt.scatter(iv_test_med, dv_test_med, color = 'blue' )
plt.xlabel('median_income')
plt.ylabel('median_house_value')
plt.title('Plot aginst median_income and house values')
plt.grid()
plt.show()
```



In [51]:

```
#Accurecy Of Train data
med_linear.score(iv_train_med,dv_train_med)
```

Out[51]:

```
0.47991412719941506
```

In [52]:

```
#Accurecy Of Test data
med_linear.score(iv_test_med,dv_test_med)
```

Out[52]:

```
0.4466846804895944
```

In [54]:

```
from sklearn.metrics import mean_squared_error
medlin_RMSE=mean_squared_error(dv_train_med,med_linear.predict(iv_train_med))
medlin_RMSE
```

Out[54]:

```
0.5200858728005849
```

In [55]:

```
print("RMSE  aginst median_income for Linear Regression:{}".format(medlin_RMSE))
```

```
RMSE  aginst median_income for Linear Regression:0.5200858728005849
```