In [3]:
```python
#importing needed libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import seaborn as sns
```

In [17]:
```python
df = pd.read_csv("Boston.csv")
```

In [18]:
```python
df.head()
```

Out[18]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [6]:
```
#Checking for null values
df.isnull().sum()
```

Out[6]:
```
Unnamed: 0    0
crim          0
zn            0
indus         0
chas          0
nox           0
rm            0
age           0
dis           0
rad           0
tax           0
ptratio       0
black         0
lstat         0
medv          0
dtype: int64
```

Q4) Use the boston.csv dataset and determine the best 5 features to predict ‘MEDV’

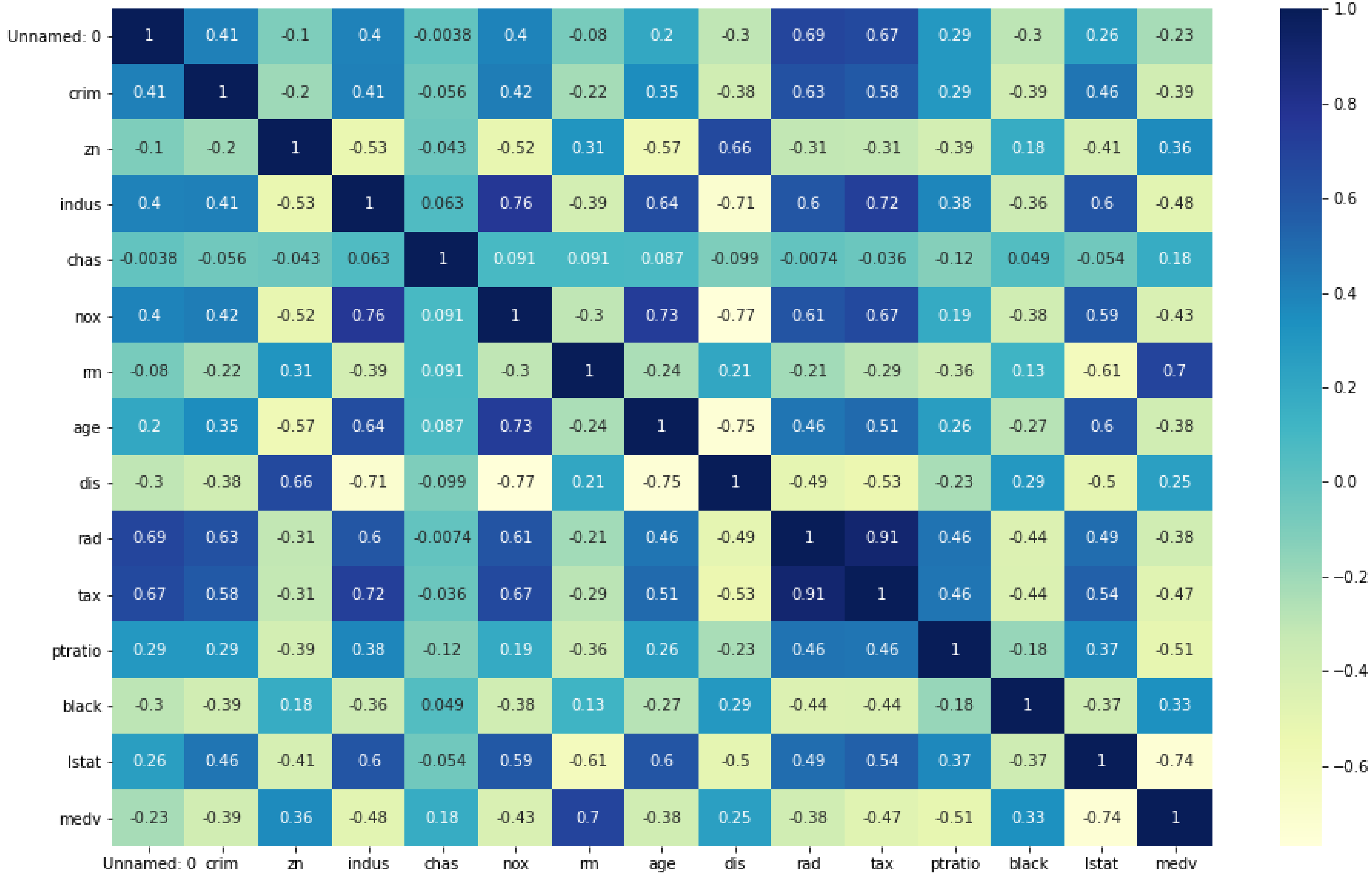
In [7]:
```
#Checking the correlation of other variables with MEDV
df.corr().tail(1)
```

Out[7]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	l
medv	-0.226604	-0.388305	0.360445	-0.483725	0.17526	-0.427321	0.69536	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737

In [8]:

```python
#Plotting correlation heatmap for a better understanding of interdependence of variables
plt.figure(figsize=(16,10))
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



Top correlated features with MEDV are:

1. LSTAT(-0.74)
2. RM(0.7)
3. PTRATIO(-0.51)
4. INDUS(-0.48)
5. TAX(-0.47)
6. NOX(-0.43)
7. CRIM(-0.39)

Here, (INDUS-TAX), (INDUS-NOX) are highly correlated among themselves, hence both can't be selected together

So we can take up the next correlated variable in order, 'CRIM'

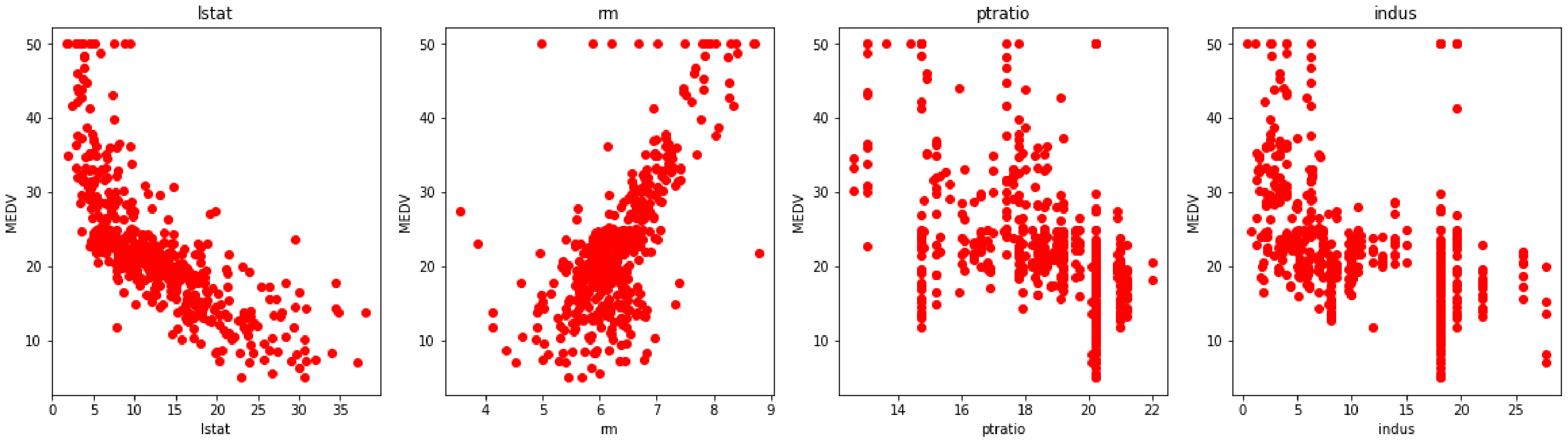
Q5) Using sklearn.linear_model, find the multiple regression model for the boston.csv dataset using the best 4 features. (from sklearn.linear_model import LinearRegression)

In [9]:
```
# Top 4 features selected - LSTAT, RM, PTRATIO, INDUS

#Plotting these features for a visual ally

plt.figure(figsize=(20, 5))
features = ['lstat', 'rm','ptratio','indus']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = df[col]
    y = df['medv']
    plt.scatter(x, y, color = 'r')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



From the above plots, we can infer that LSTAT and RM have a few outliers but mostly follow a linear distribution, on the other hand, PTRATIO and INDUS have quite a scattered composition.

In [10]:
```python
#Alloting values for x and y respectively

X = pd.DataFrame(np.c_[df['lstat'], df['rm'], df['ptratio'],df['indus']], columns = ['lstat','rm', 'ptratio', 'rm']) #Tr
Y = df['medv']
```

In [11]:
```python
print(X.head())
print("\n")
print(y.head())
```

```
   lstat     rm  ptratio    rm
0   4.98  6.575     15.3  2.31
1   9.14  6.421     17.8  7.07
2   4.03  7.185     17.8  7.07
3   2.94  6.998     18.7  2.18
4   5.33  7.147     18.7  2.18


0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: medv, dtype: float64
```

In [12]:
```python
from sklearn.model_selection import train_test_split

# Splitting the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=42)

print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 4)
(102, 4)
(404,)
(102,)
```

In [13]:
```python
#Implementing the linear regression model for the training dataset from the sklearn library
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```python
# Assigning values to the coefficients and intercept
B0 = lin_model.intercept_
B1 = lin_model.coef_[0]
B2 = lin_model.coef_[1]
B3 = lin_model.coef_[2]
B4 = lin_model.coef_[3]
```

In [15]:
```python
#Printing the regression equation

print('The regression model is: y = {} + {} x1 + {} x2 + {} x3 + {} x4'
      .format(round(B0,4), round(B1, 4), round(B2, 4), round(B3, 4), round(B4, 4)))
```

```
The regression model is: y = 14.663 + -0.5809 x1 + 4.9376 x2 + -0.8791 x3 + 0.0262 x4
```

Q6) Find the accuracy of the model using appropriate metrics using 80, 20 split for training and test.

In [16]:
```
#Importing required functions
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

#Getting the y predicted values
y_train_predicted = lin_model.predict(X_train)
y_test_predicted = lin_model.predict(X_test)

#Defining r2 and rmse
r2 = r2_score(Y_train, y_train_predicted)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predicted)))

#Printing the values
print('R2 score is', r2)
print('RMSE is', rmse)
```

```
R2 score is 0.6866881152922459
RMSE is 5.220087893560882
```