Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 04

Submitted by:

Name: Zannat Hossain Tamim

Roll No:1970

Lab Date:08.08.23

Submission Date: 10.08.23

Lab Report # Day 04

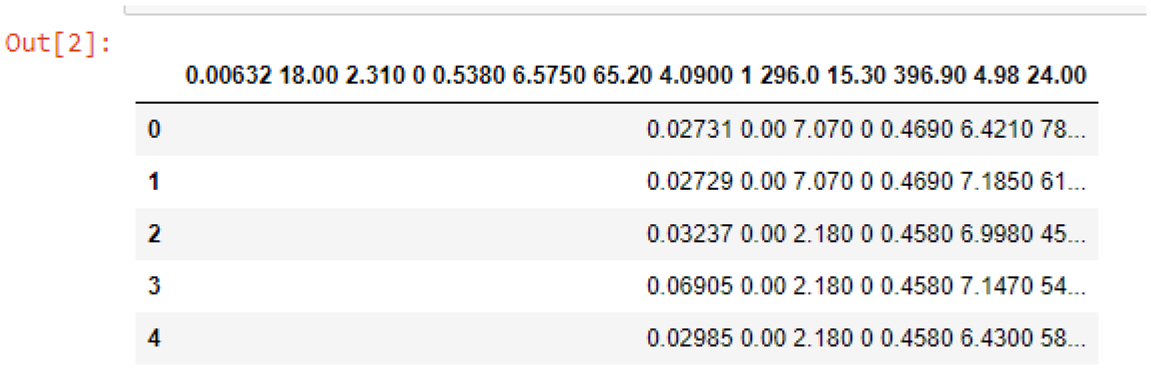
Query 1:

Read CSV file and print the top 5 rows.

Clause:

```
import numpy as np
import pandas as pd
df =pd.read_csv('housing.csv')
df.head()
```

Result :



Out[2]:

	0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
0	0.02731 0.00 7.070 0 0.4690 6.4210 78...
1	0.02729 0.00 7.070 0 0.4690 7.1850 61...
2	0.03237 0.00 2.180 0 0.4580 6.9980 45...
3	0.06905 0.00 2.180 0 0.4580 7.1470 54...
4	0.02985 0.00 2.180 0 0.4580 6.4300 58...

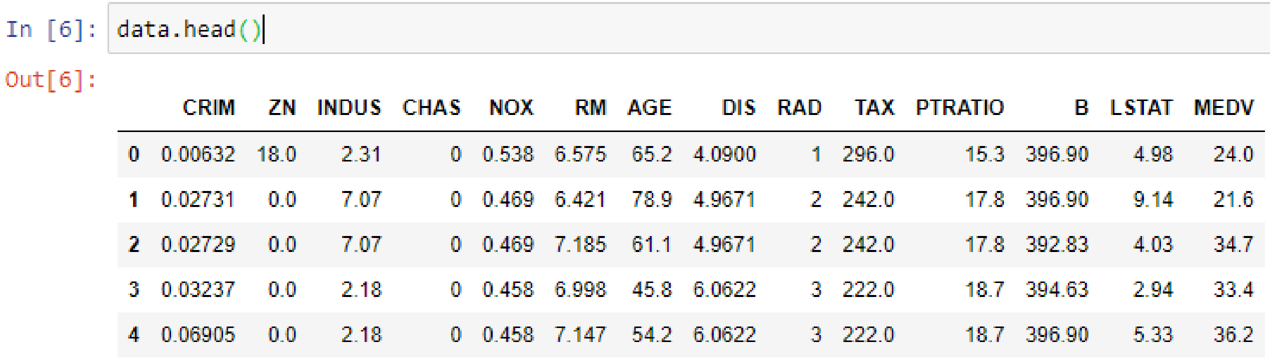
Query 2:

To create a list of column names and assign them to the variable column and print top 5 rows.

Clause:

```
column= ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO',
'B', 'LSTAT', 'MEDV']
data = pd.read_csv('housing.csv', header=None, delimiter=r"\s+", names=column)
data.head()
```

Result :



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

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

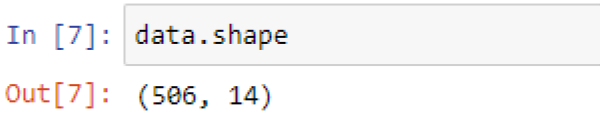
Query 3:

To get the shape of the DataFrame

Clause:

data.shape

Result :



```
In [7]: data.shape
Out[7]: (506, 14)
```

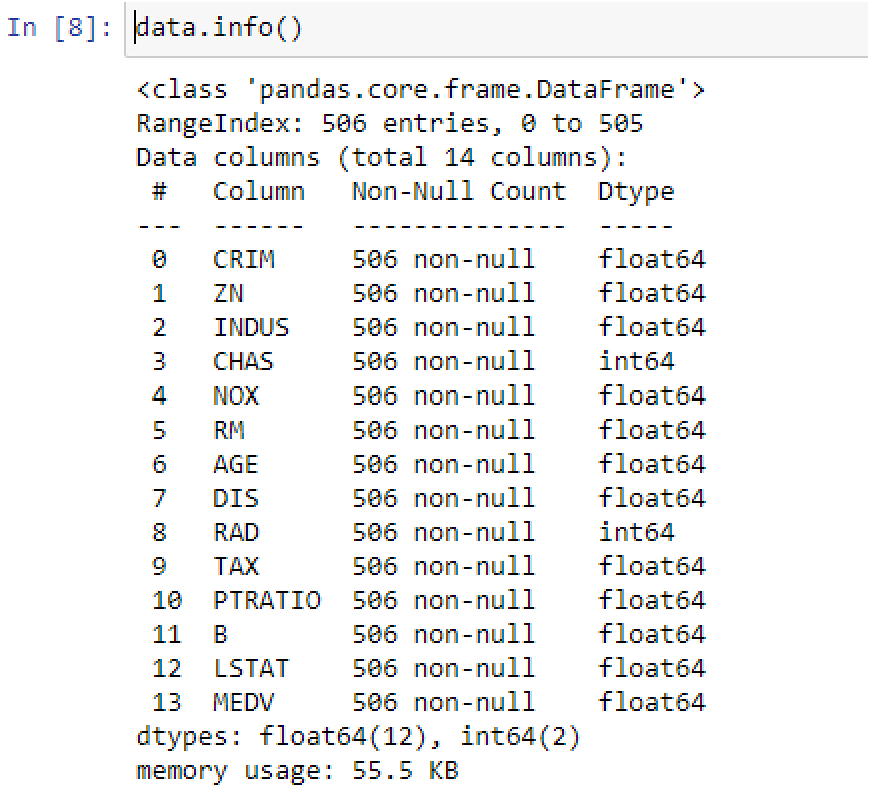
Query 4:

To get the information of the DataFrame

Clause:

data.info()

Result :



```
In [8]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

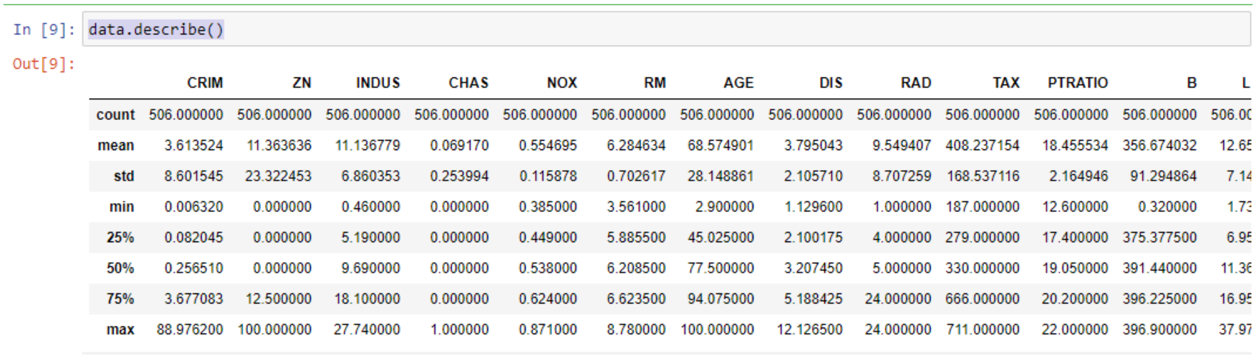
Query 5:

To calculate and print a summary of statistical data for each column in the DataFrame

Clause:

```
data.describe()
```

Result :



```
In [9]: data.describe()
Out[9]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.65
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.14
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.73
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.95
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.36
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.95
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.97

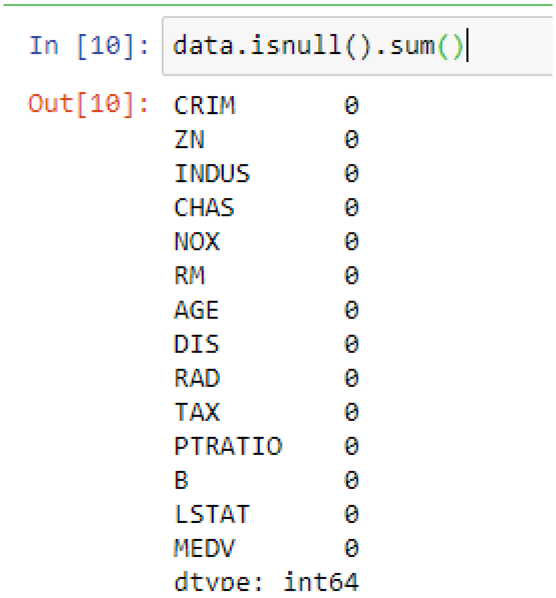
Query 6:

To count the number of missing values in each column of a DataFrame

Clause:

data.isnull().sum()

Result:



```
In [10]: data.isnull().sum()
```

```
Out[10]: CRIM       0
         ZN         0
         INDUS      0
         CHAS       0
         NOX        0
         RM         0
         AGE        0
         DIS        0
         RAD        0
         TAX        0
         PTRATIO    0
         B          0
         LSTAT      0
         MEDV       0
         dtype: int64
```

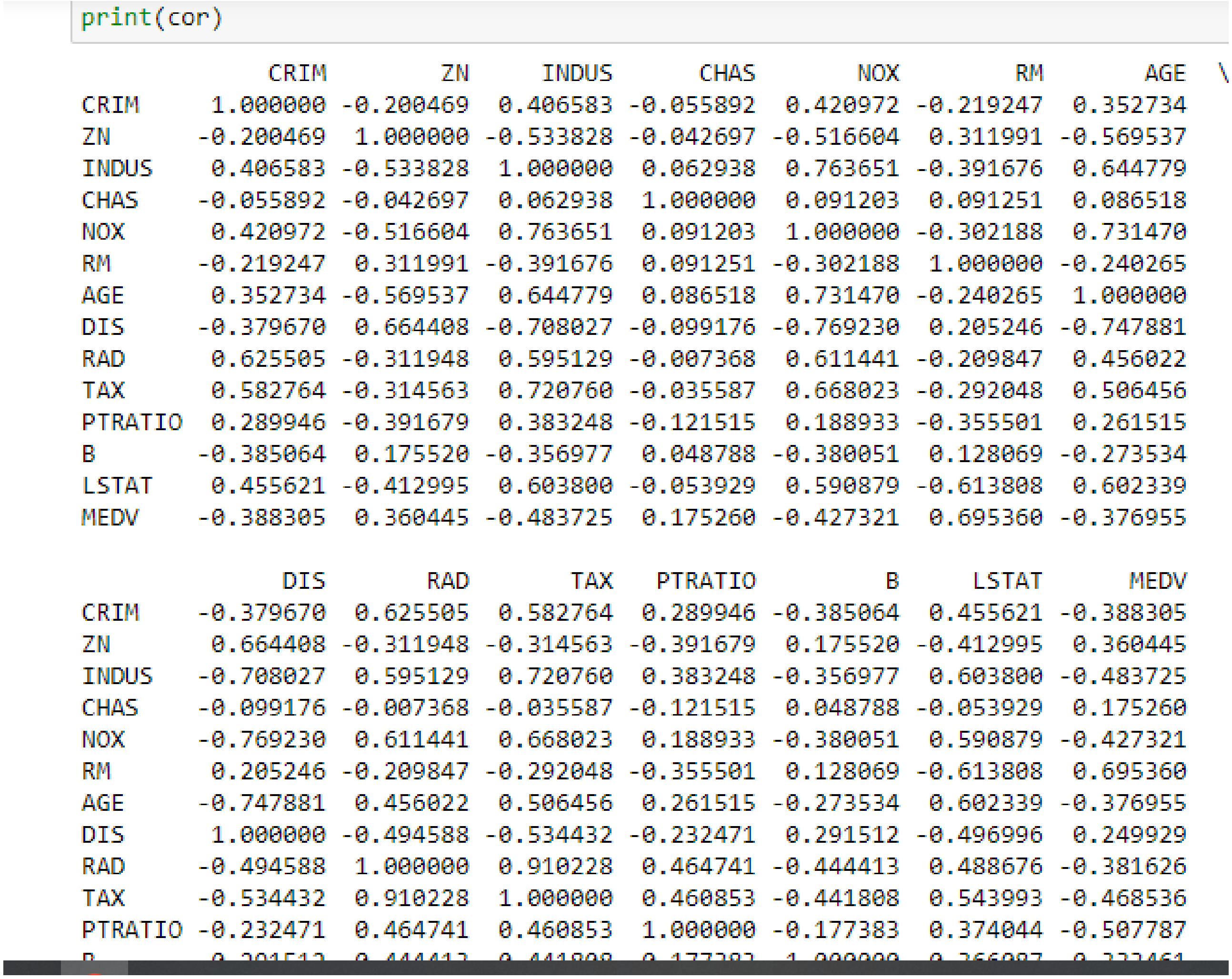
Query 7:

Calculate the correlation coefficient between each pair of columns in the DataFrame and store it in the variable cor

Clause:

cor=data.corr()
print(cor)

Result:



```
print(cor)
```

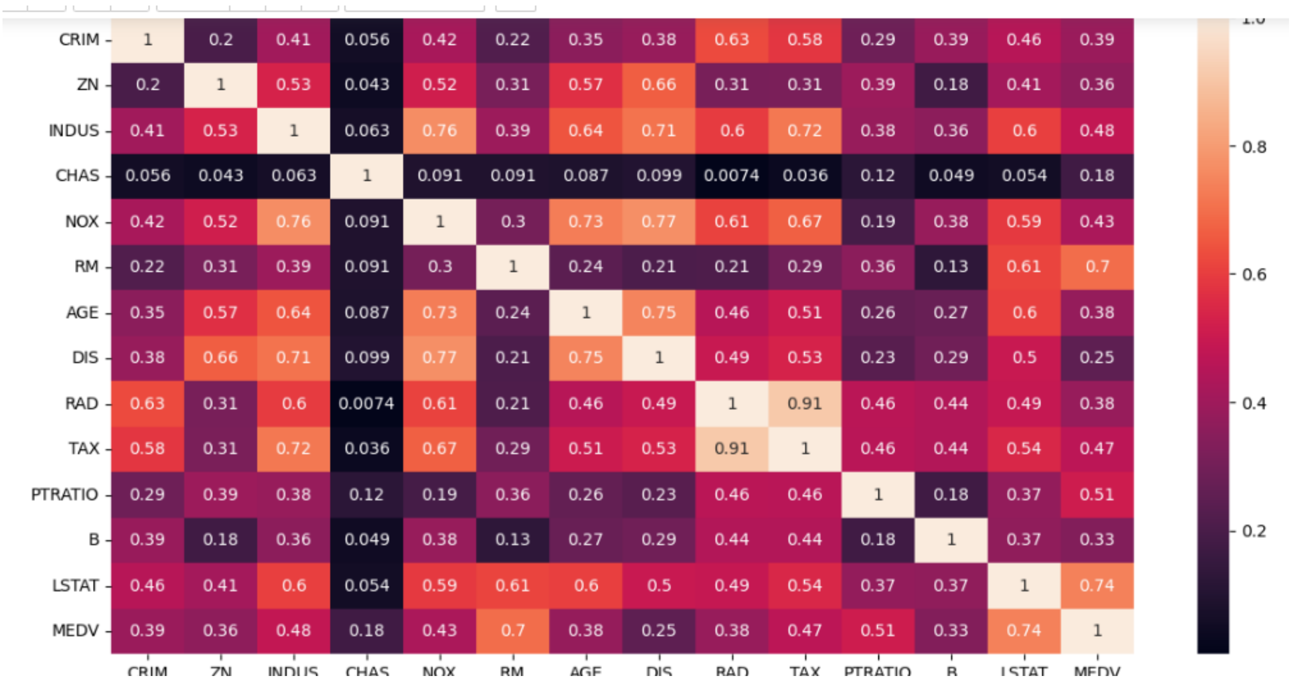
```
             CRIM        ZN     INDUS      CHAS       NOX        RM       AGE  \
CRIM     1.000000 -0.200469  0.406583 -0.055892  0.420972 -0.219247  0.352734
ZN      -0.200469  1.000000 -0.533828 -0.042697 -0.516604  0.311991 -0.569537
INDUS    0.406583 -0.533828  1.000000  0.062938  0.763651 -0.391676  0.644779
CHAS    -0.055892 -0.042697  0.062938  1.000000  0.091203  0.091251  0.086518
NOX      0.420972 -0.516604  0.763651  0.091203  1.000000 -0.302188  0.731470
RM      -0.219247  0.311991 -0.391676  0.091251 -0.302188  1.000000 -0.240265
AGE      0.352734 -0.569537  0.644779  0.086518  0.731470 -0.240265  1.000000
DIS     -0.379670  0.664408 -0.708027 -0.099176 -0.769230  0.205246 -0.747881
RAD      0.625505 -0.311948  0.595129 -0.007368  0.611441 -0.209847  0.456022
TAX      0.582764 -0.314563  0.720760 -0.035587  0.668023 -0.292048  0.506456
PTRATIO  0.289946 -0.391679  0.383248 -0.121515  0.188933 -0.355501  0.261515
B       -0.385064  0.175520 -0.356977  0.048788 -0.380051  0.128069 -0.273534
LSTAT    0.455621 -0.412995  0.603800 -0.053929  0.590879 -0.613808  0.602339
MEDV    -0.388305  0.360445 -0.483725  0.175260 -0.427321  0.695360 -0.376955

              DIS       RAD       TAX   PTRATIO         B     LSTAT      MEDV
CRIM    -0.379670  0.625505  0.582764  0.289946 -0.385064  0.455621 -0.388305
ZN       0.664408 -0.311948 -0.314563 -0.391679  0.175520 -0.412995  0.360445
INDUS   -0.708027  0.595129  0.720760  0.383248 -0.356977  0.603800 -0.483725
CHAS    -0.099176 -0.007368 -0.035587 -0.121515  0.048788 -0.053929  0.175260
NOX     -0.769230  0.611441  0.668023  0.188933 -0.380051  0.590879 -0.427321
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -0.376955
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -0.381626
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.543993 -0.468536
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.374044 -0.507787
```

Query 8: To create a heatmap of the correlation matrix:

Clause:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.figure(figsize=(14,7))
sns.heatmap(data.corr().abs(),annot=True)
```

Result:

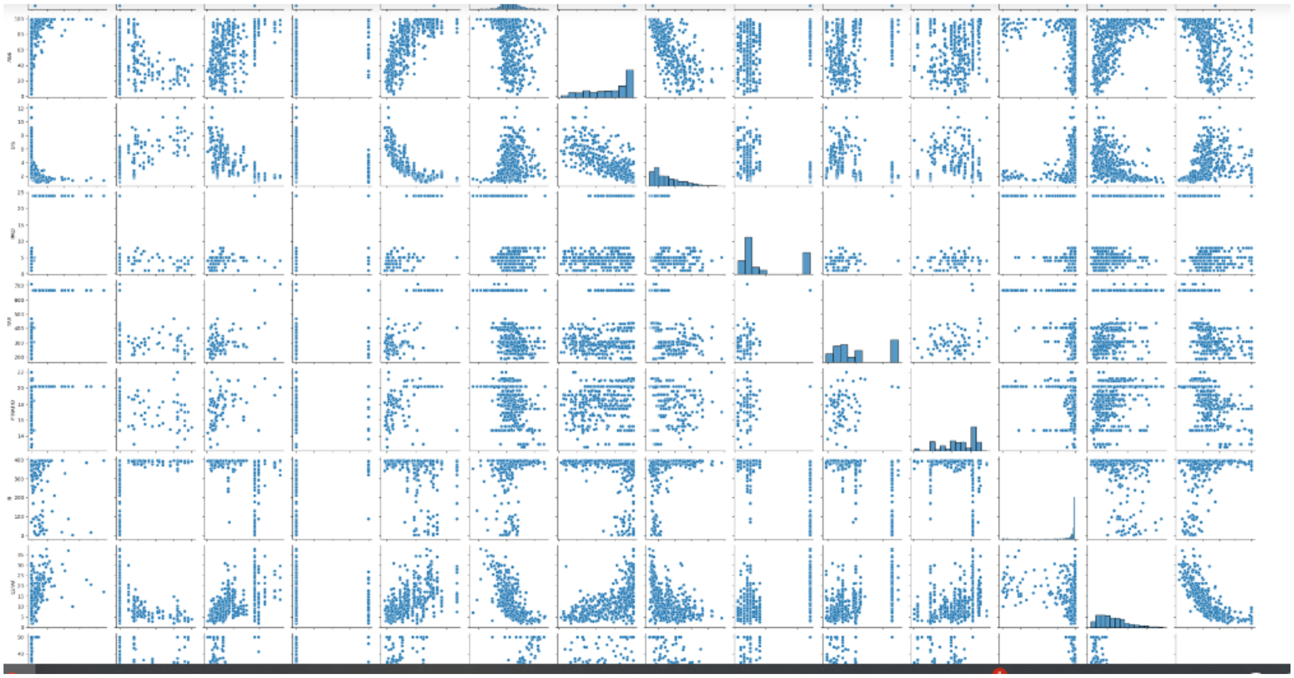


Query 9:

Clause:

```
sns.pairplot(data)
```

Result:

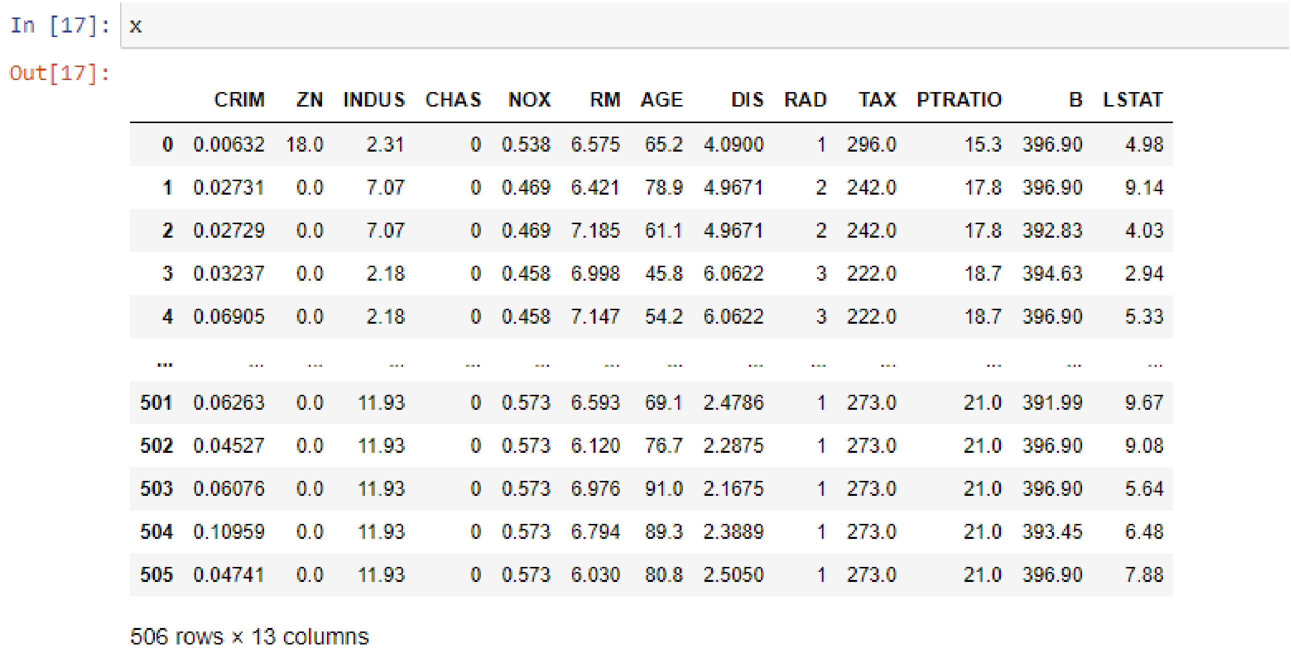


Query 10:

Clause:

```
x = data.drop('MEDV',axis=1)
x
```

Result:



```
In [17]: x
Out[17]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88

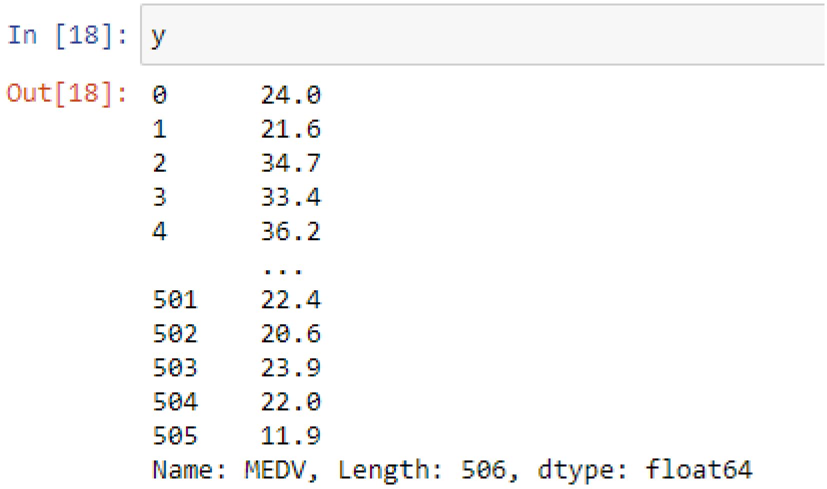
506 rows × 13 columns

Query 11:

Clause:

```
y=data['MEDV']
y
```

Result:



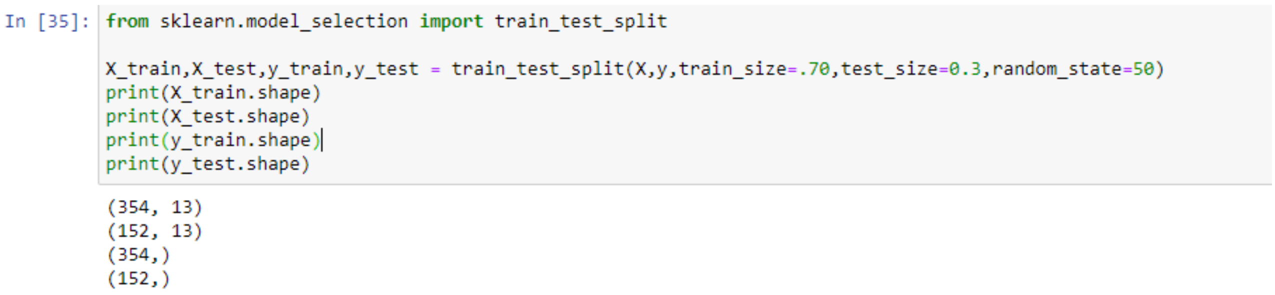
```
In [18]: y

Out[18]: 0      24.0
         1      21.6
         2      34.7
         3      33.4
         4      36.2
                ... 
         501    22.4
         502    20.6
         503    23.9
         504    22.0
         505    11.9
         Name: MEDV, Length: 506, dtype: float64
```

Query 12:

Clause:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =
train_test_split(X,y,train_size=.70,test_size=0.3,random_state=50)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Result:



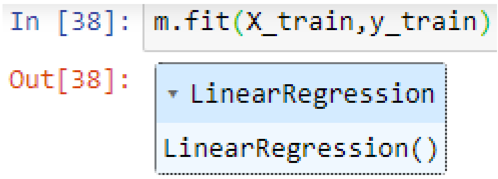
```
In [35]: from sklearn.model_selection import train_test_split

         X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=.70,test_size=0.3,random_state=50)
         print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)

         (354, 13)
         (152, 13)
         (354,)
         (152,)
```

Query 13:

Clause:

```
from sklearn.linear_model import LinearRegression
 m = LinearRegression()
 m.fit(X_train,y_train)
```

Result:



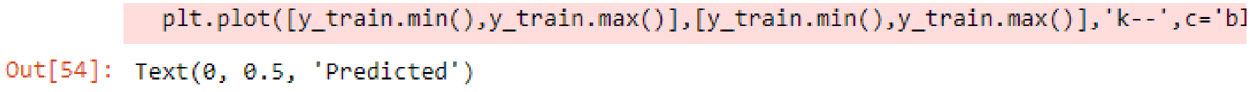
```
In [38]: m.fit(X_train,y_train)

Out[38]: ▾ LinearRegression
         LinearRegression()
```

Query 14:

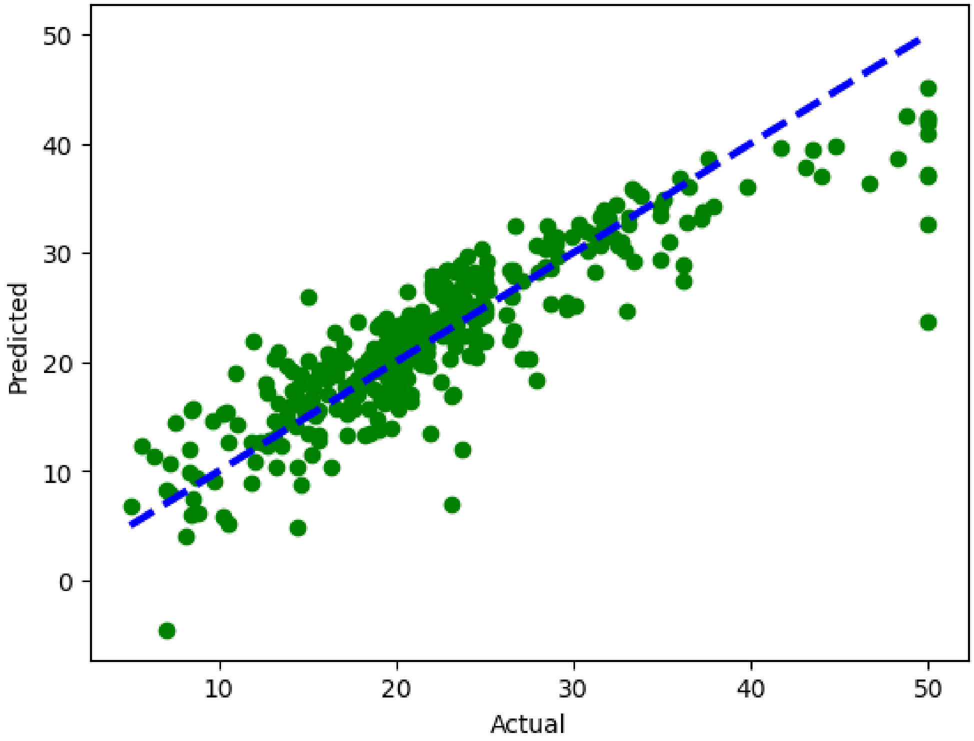
Clause:

```
y_predict = m.predict(X_test)
y_pre_train= m.predict(X_train)
plt.scatter(y_train,y_pre_train,c='green',lw=1)
plt.plot([y_train.min(),y_train.max()],[y_train.min(),y_train.max()],'k--',c='blue',lw=3)
plt.xlabel('Actual')
plt.ylabel('Predicted')
```

Result:



```
          plt.plot([y_train.min(),y_train.max()],[y_train.min(),y_train.max()],'k--',c='b]

Out[54]: Text(0, 0.5, 'Predicted')
```

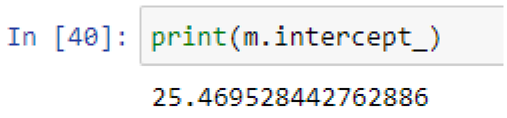


Query 15:

Clause:

```
print(m.intercept_)
```

Result:



```
In [40]: print(m.intercept_)

         25.469528442762886
```

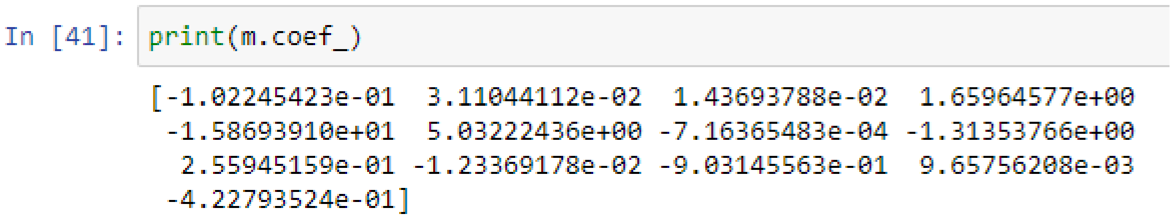
Query 16:

To print coefficient

Clause:

```
print(m.coef_)
```

Result:



```
In [41]: print(m.coef_)

         [-1.02245423e-01  3.11044112e-02  1.43693788e-02  1.65964577e+00
          -1.58693910e+01  5.03222436e+00 -7.16365483e-04 -1.31353766e+00
           2.55945159e-01 -1.23369178e-02 -9.03145563e-01  9.65756208e-03
          -4.22793524e-01]
```

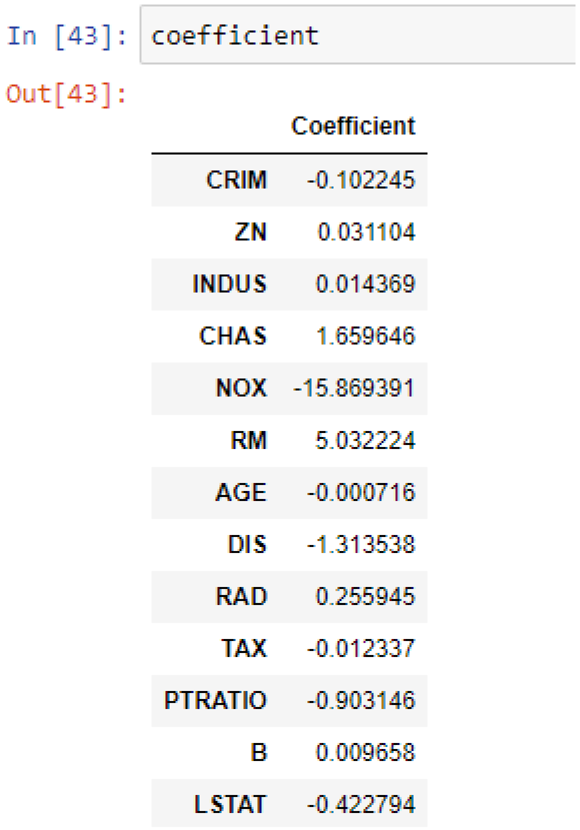
Query 17:

To create a Pandas DataFrame that shows the coefficients of the machine learning model

Clause:

```
coefficient = pd.DataFrame(m.coef_,x.columns,columns=["Coefficient"])
coefficient
```

Result:



```
In [43]: coefficient
```

Out[43]:

	Coefficient
CRIM	-0.102245
ZN	0.031104
INDUS	0.014369
CHAS	1.659646
NOX	-15.869391
RM	5.032224
AGE	-0.000716
DIS	-1.313538
RAD	0.255945
TAX	-0.012337
PTRATIO	-0.903146
B	0.009658
LSTAT	-0.422794

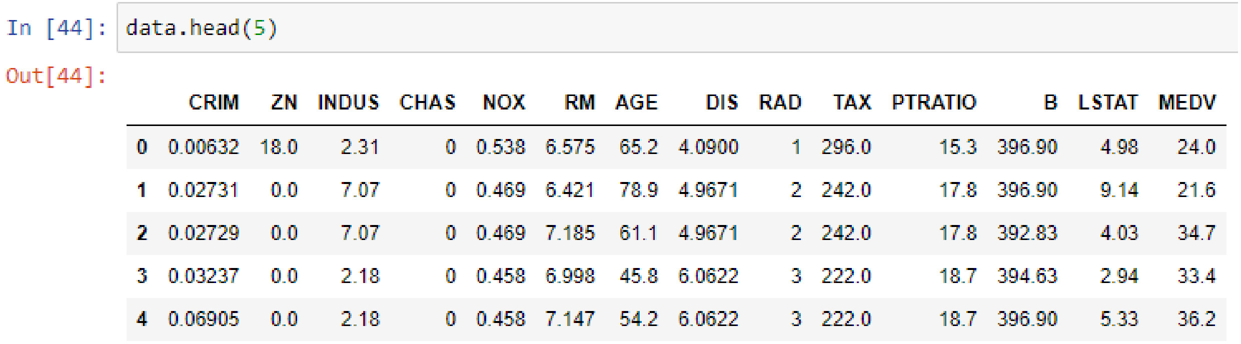
Query 18:

To print the top 5 rows

Clause:

data.head(5)

Result:



```
In [44]: data.head(5)
```

Out[44]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

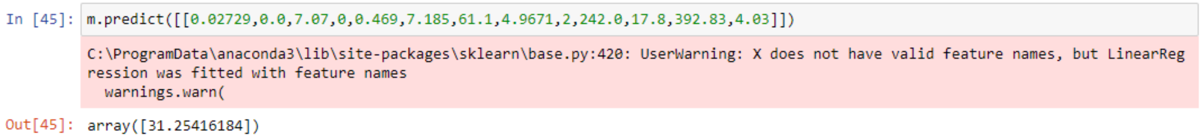
Query 19:

To predict the median house price for the house

Clause:

m.predict([[0.02729,0.0,7.07,0,0.469,7.185,61.1,4.9671,2,242.0,17.8,392.83,4.03]])

Result:



```
In [45]: m.predict([[0.02729,0.0,7.07,0,0.469,7.185,61.1,4.9671,2,242.0,17.8,392.83,4.03]])

         C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but LinearReg
         ression was fitted with feature names
           warnings.warn(

Out[45]: array([31.25416184])
```

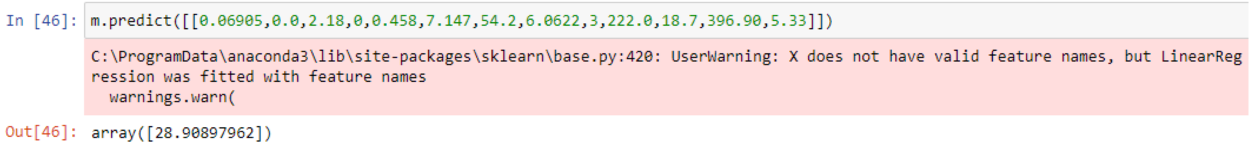
Query 20:

To predict the median house price for the house

Clause:

m.predict([[0.06905,0.0,2.18,0,0.458,7.147,54.2,6.0622,3,222.0,18.7,396.90,5.33]])

Result:



```
In [46]: m.predict([[0.06905,0.0,2.18,0,0.458,7.147,54.2,6.0622,3,222.0,18.7,396.90,5.33]])

         C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but LinearReg
         ression was fitted with feature names
           warnings.warn(

Out[46]: array([28.90897962])
```

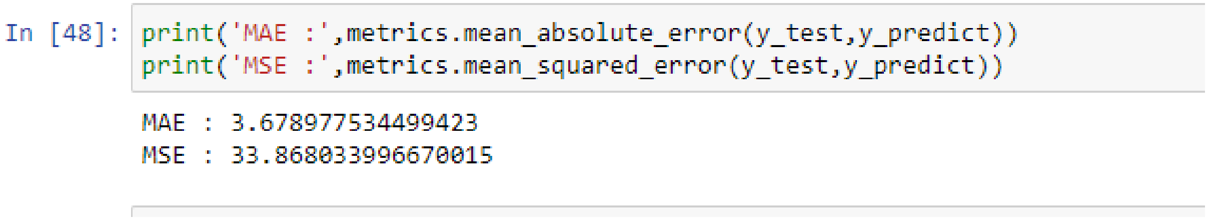
Query 21:

Evaluate the performance of the model

Clause:

from sklearn import metrics
print('MAE :',metrics.mean_absolute_error(y_test,y_predict))
print('MSE :',metrics.mean_squared_error(y_test,y_predict))

Result:



```
In [48]: print('MAE :',metrics.mean_absolute_error(y_test,y_predict))
         print('MSE :',metrics.mean_squared_error(y_test,y_predict))

         MAE : 3.678977534499423
         MSE : 33.868033996670015
```