EXPLORATORY DATA ANALYSIS ON BOSTON HOUSE PRICES

BOSTON-HOUSING DATASET

*Content Each record in the database describes a Boston suburb or town. The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. The attributes are defined as follows :

CRIM: per capita crime rate by town ZN: proportion of residential land zoned for lots over 25,000 sq.ft. INDUS: proportion of non-retail business acres per town. CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise) NOX: nitric oxides concentration (parts per 10 million) RM: average number of rooms per dwelling AGE: proportion of owner-occupied units built prior to 1940 DIS: weighted distances to five Boston employment centres RAD: index of accessibility to radial highways TAX: full-value property-tax rate per 10,000 PTRATIO: pupil-teacher ratio by town LSTAT: lower status of the population MEDV: Median value of owner-occupied homes in 1000s

```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

# Data Reading
data=pd.read_csv("/content/housing.csv")
data.head()
```

	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...

Next steps: Generate code with data View recommended plots

```
# DimensionS of the data
data.shape
```

```
(505, 1)
```

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

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

Next steps: Generate code with data View recommended plots

```
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
```

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

```
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
```

Data Exploration

```
data.describe()
```

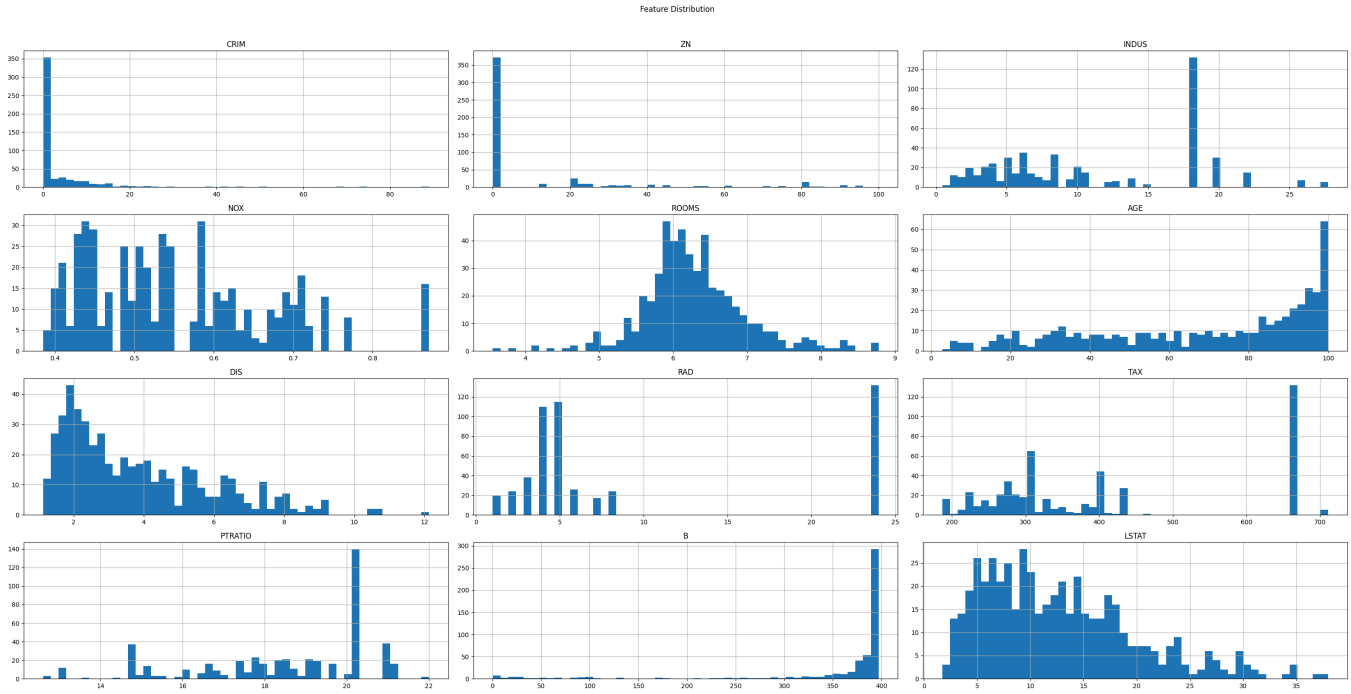
	CRIM	ZN	INDUS	CHAS	NOX	RM	AG
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.57490
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.14886
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.90000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.02500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.50000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.07500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.00000

```
minimum_price = np.min(data["MEDV"])

maximum_price = np.max(data["MEDV"])

mean_price = np.mean(data["MEDV"])

median_price = np.median(data["MEDV"])

std_price = np.std(data["MEDV"])

#Show the calculated statistics

print("Statistics for Boston housing dataset:\n")
print("Minimum price: ${:,.2f}".format(minimum_price))
print("Maximum price: ${:,.2f}".format(maximum_price))
print("Mean price: ${:,.2f}".format(mean_price))
print("Median price ${:,.2f}".format(median_price))
print("Standard deviation of prices: ${:,.2f}".format(std_price))
```

```
Statistics for Boston housing dataset:

Minimum price: $5.00
Maximum price: $50.00
Mean price: $22.53
Median price $21.20
Standard deviation of prices: $9.19
```

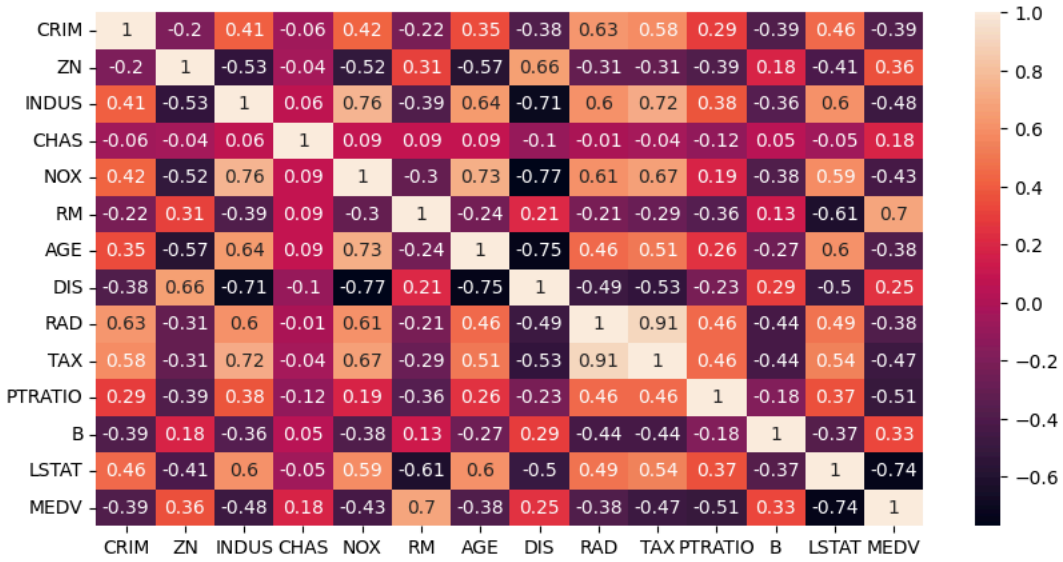
DATA VISUALIZATION

```
data.hist(bins=50, figsize=(30,15))
plt.suptitle('Feature Distribution', x=0.5, y=1.02, ha='center', fontsize='large')
plt.tight_layout()
plt.show()
```




```
# Correlation

plt.figure(figsize=(10,5))
correlation_matrix = data.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<Axes: >
```




```
data["MEDV"]=pd.cut(data["MEDV"],bins=[0,15,30,50],labels=["LOW","MEDIUM","HIGH"])
data.sample(5)
```

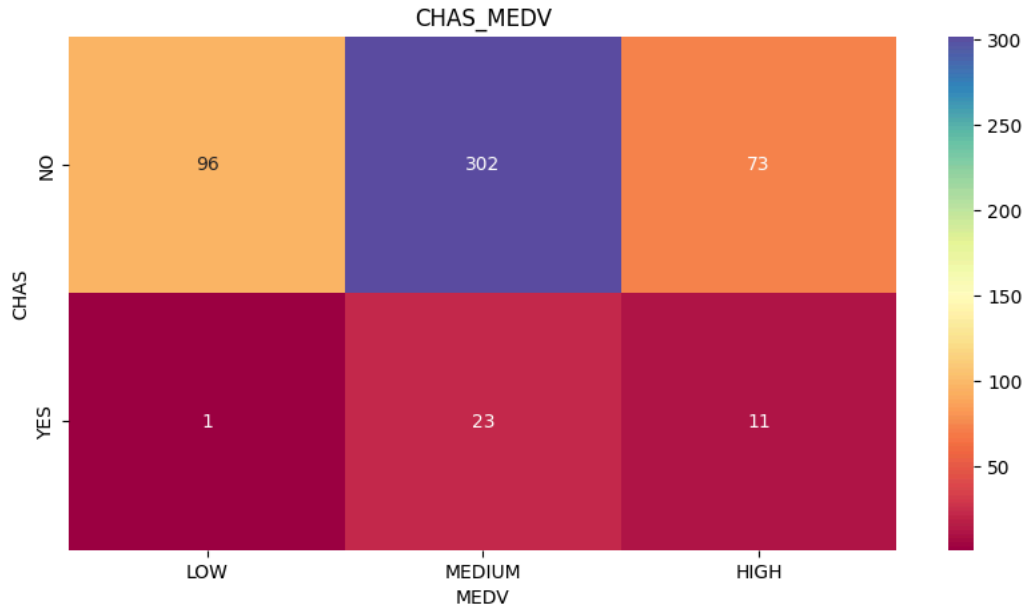
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
287	0.03871	52.5	5.32	0	0.405	6.209	31.3	7.3172	6	293.0	16.6	396.90	7.14	MEDIUM
218	0.11069	0.0	13.89	1	0.550	5.951	93.8	2.8893	5	276.0	16.4	396.90	17.92	MEDIUM
143	4.09740	0.0	19.58	0	0.871	5.468	100.0	1.4118	5	403.0	14.7	396.90	26.42	MEDIUM
72	0.09164	0.0	10.81	0	0.413	6.065	7.8	5.2873	4	305.0	19.2	390.91	5.52	MEDIUM
170	1.20742	0.0	19.58	0	0.605	5.875	94.6	2.4259	5	403.0	14.7	292.29	14.43	MEDIUM

```
data["CHAS"]=data["CHAS"].replace({0:"NO",1:"YES"})
data.sample(5)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
399	9.91655	0.0	18.10	NO	0.693	5.852	77.8	1.5004	24	666.0	20.2	338.16	29.97	LOW
503	0.06076	0.0	11.93	NO	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	MEDIUM
462	6.65492	0.0	18.10	NO	0.713	6.317	83.0	2.7344	24	666.0	20.2	396.90	13.99	MEDIUM
291	0.07886	80.0	4.95	NO	0.411	7.148	27.7	5.1167	4	245.0	19.2	396.90	3.56	HIGH
373	11.10810	0.0	18.10	NO	0.668	4.906	100.0	1.1742	24	666.0	20.2	396.90	34.77	LOW

```
data1=pd.crosstab(data["CHAS"],data["MEDV"])

plt.figure(figsize=(10,5))
sns.heatmap(data1,annot=True,fmt="d",cmap="Spectral")
plt.title("CHAS_MEDV")
```

```
Text(0.5, 1.0, 'CHAS_MEDV')
```




```
data.rename(columns={"RM":"ROOMS"},inplace=True)
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	ROOMS	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	NO	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	MEDIUM
1	0.02731	0.0	7.07	NO	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	MEDIUM
2	0.02729	0.0	7.07	NO	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	HIGH
3	0.03237	0.0	2.18	NO	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	HIGH
4	0.06905	0.0	2.18	NO	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	HIGH

Next steps: Generate code with `data` View recommended plots

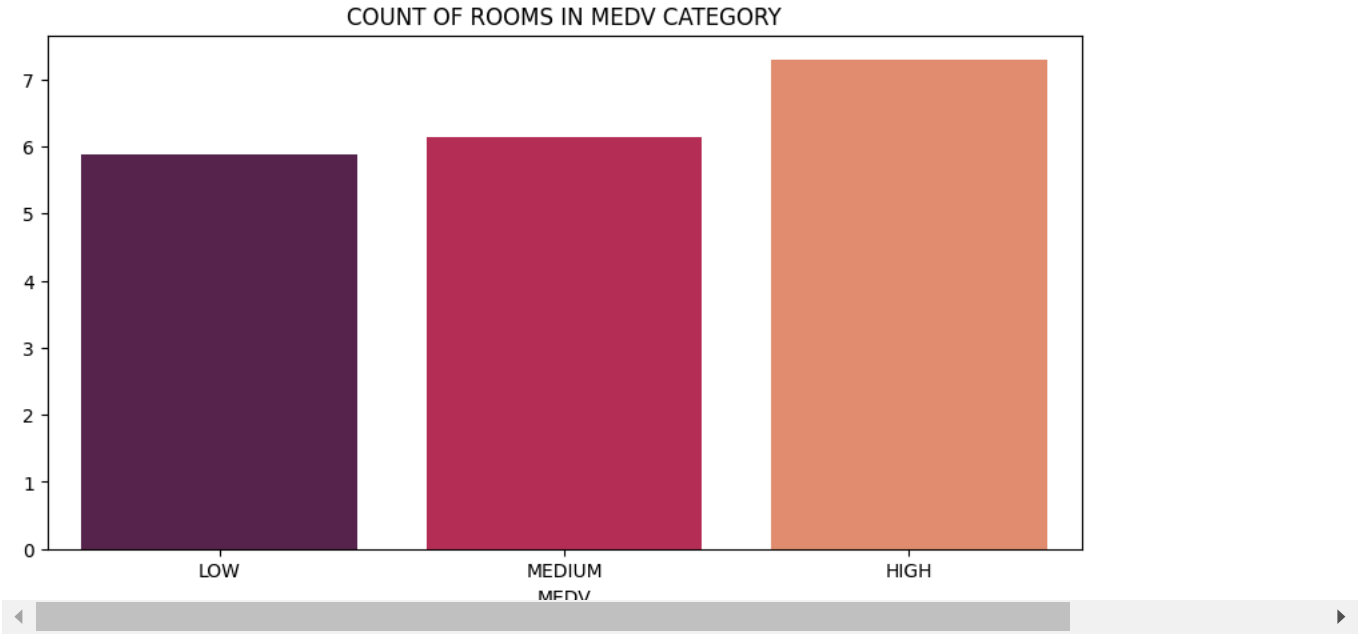
```
room_medv=data.groupby("MEDV")["ROOMS"].mean()
```

```
room_medv
```

```
MEDV
LOW       5.878371
MEDIUM    6.145129
HIGH      7.293524
Name: ROOMS, dtype: float64
```

```
plt.figure(figsize=(10,5))
sns.barplot(x=room_medv.index,y=room_medv.values,palette="rocket")
plt.title("COUNT OF ROOMS IN MEDV CATEGORY")
```

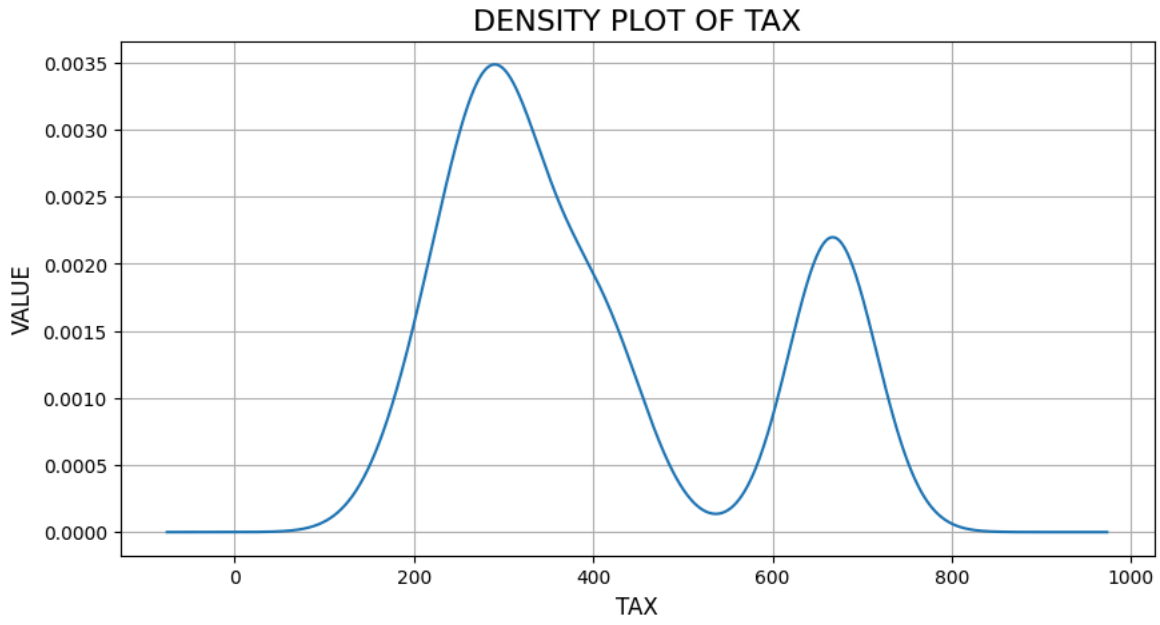
```
<ipython-input-20-9ffad6da7c5a>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=room_medv.index,y=room_medv.values,palette="rocket")
Text(0.5, 1.0, 'COUNT OF ROOMS IN MEDV CATEGORY')
```




```
# Calculating percentage of houses along the Charles river

p=data["CHAS"].value_counts().values[1]
s=data["CHAS"].value_counts().values[0]
pp=(p/(p+s))*100
print(f"The answer is : {pp:.2f}%")
```

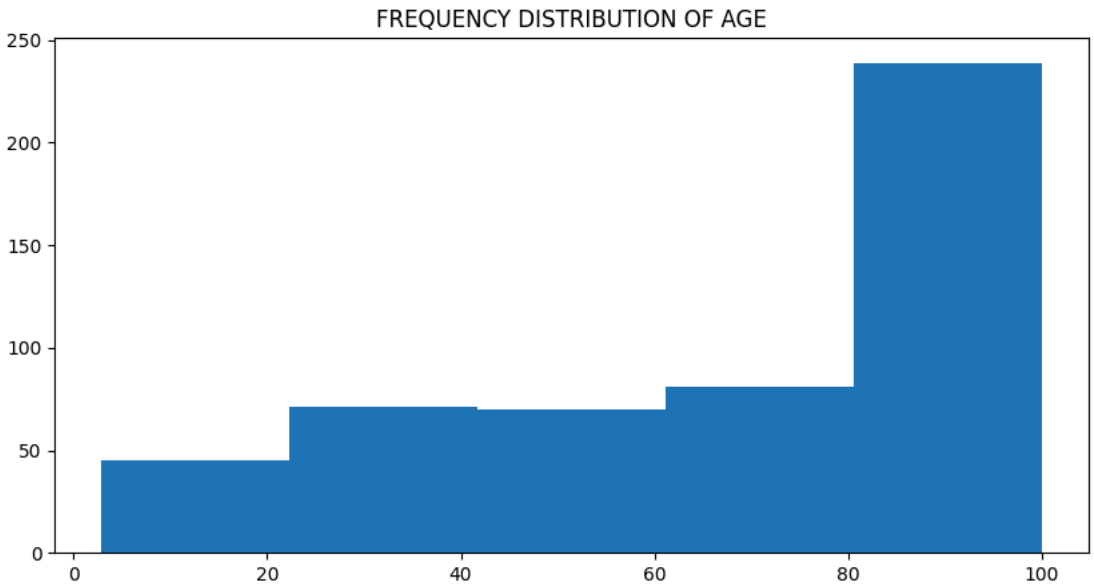
```
The answer is : 6.92%
```

```
# Calculating Tax distribution

plt.figure(figsize=(10,5))
data["TAX"].plot.density()
plt.title("DENSITY PLOT OF TAX",fontsize=16)
plt.xlabel("TAX",fontsize=12)
plt.ylabel("VALUE",fontsize=12)
plt.grid(True)
```

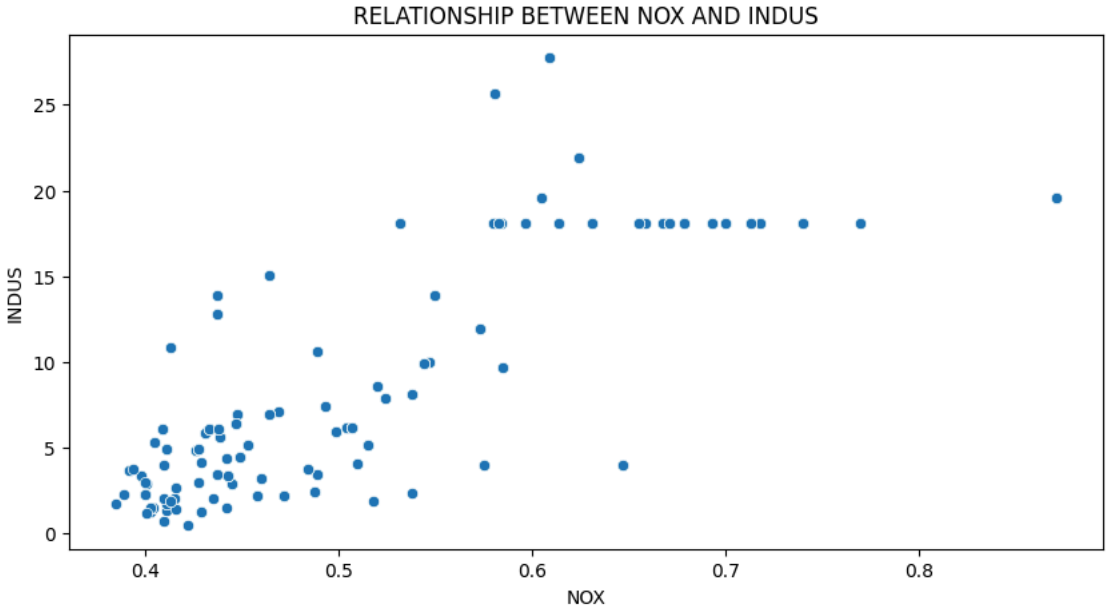


```
# Calculating frequency distribution of age

plt.figure(figsize=(10,5))
plt.hist(data["AGE"],bins=5)
plt.title("FREQUENCY DISTRIBUTION OF AGE")
```

```
Text(0.5, 1.0, 'FREQUENCY DISTRIBUTION OF AGE')
```




```
# Relationship between Nox and Indus

plt.figure(figsize=(10,5))
sns.scatterplot(x=data["NOX"], y=data["INDUS"], data=data)
plt.title("RELATIONSHIP BETWEEN NOX AND INDUS")
plt.show()
```




```
# Pupil to Teacher ratio

plt.figure(figsize=(10,5))
sns.distplot(data["PTRATIO"],bins=10)
plt.title("HISTOGRAM FOR THE PUPIL TO TEACHER RATIO VARIABLE")
```

```
<ipython-input-29-9c83e0b0b689>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data["PTRATIO"],bins=10)
Text(0.5, 1.0, 'HISTOGRAM FOR THE PUPIL TO TEACHER RATIO VARIABLE')
```