Experiment No. 2

Name of the Experiment: Data Visualization / Exploratory Data Analysis for the selected dataset using Matplotlib and Seaborn.

Aim: a. To create a bar graph using any two variables.
b. To create a normalized histogram.
c. To describe what the graphs and tables indicate.

Requirements (one-line format): Python 3.x with Matplotlib, Seaborn, NumPy and Pandas libraries installed using Anaconda Distribution or Google Colab environment.

Pre-requisite: Basic knowledge of Python programming, Pandas DataFrame operations, and understanding of basic statistical concepts.

Theory: Data visualization is an essential part of exploratory data analysis as it helps in understanding patterns, trends, and relationships within the dataset. Visual representation of data makes it easier to interpret large volumes of information and supports effective decision making.

Matplotlib is a widely used Python library for creating static, animated, and interactive visualizations. It is built on top of NumPy and is used for plotting 2-dimensional graphs such as line graphs, bar charts, scatter plots, and histograms. Matplotlib provides flexibility in customizing plots and is useful for basic data visualization tasks.

Bar graphs are used to represent categorical data and show comparisons between different groups. Histograms are used to represent the distribution of numerical data. A normalized histogram displays probability distribution instead of frequency, making it easier to compare datasets of different sizes.

Seaborn is a high-level statistical visualization library built on top of Matplotlib. It works seamlessly with Pandas DataFrames and provides attractive and informative statistical graphics. Seaborn simplifies complex visualizations and automatically handles statistical aggregation and labeling.

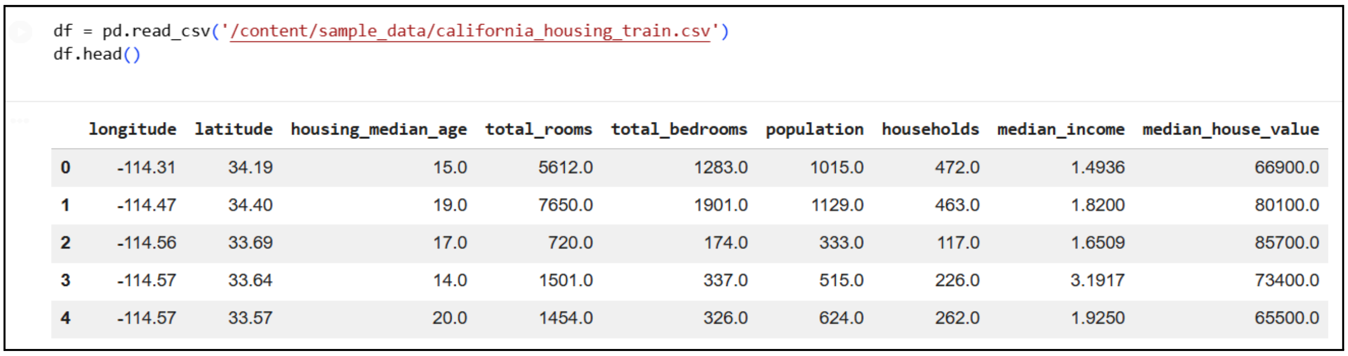
By using Matplotlib and Seaborn together, exploratory data analysis becomes more effective, allowing better understanding of data distribution, relationships between variables, and overall dataset behavior.

Advantages of Matplotlib and Seaborn:

Matplotlib is a powerful and flexible visualization library that allows detailed customization of graphs such as titles, labels, legends, colors, and scales. It supports a wide range of plots including line charts, bar graphs, scatter plots, and histograms, making it suitable for basic as well as advanced data visualization tasks. Matplotlib provides precise control over plot elements, which is helpful for presenting accurate and publication-quality visualizations.

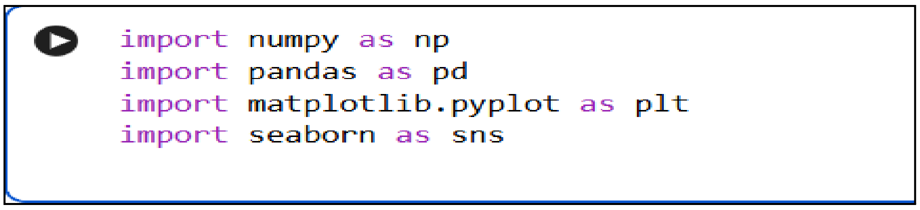
Seaborn is built on top of Matplotlib and provides a high-level interface for creating statistically informative and visually attractive plots. It works seamlessly with Pandas DataFrames and automatically handles grouping, aggregation, and statistical calculations. Seaborn reduces the amount of code required to generate complex plots such as box plots, pair plots, and distribution plots. It also provides better default styles and color palettes, making visual analysis easier and more effective.

1. Load the Dataset/ Basic Dataset Information-



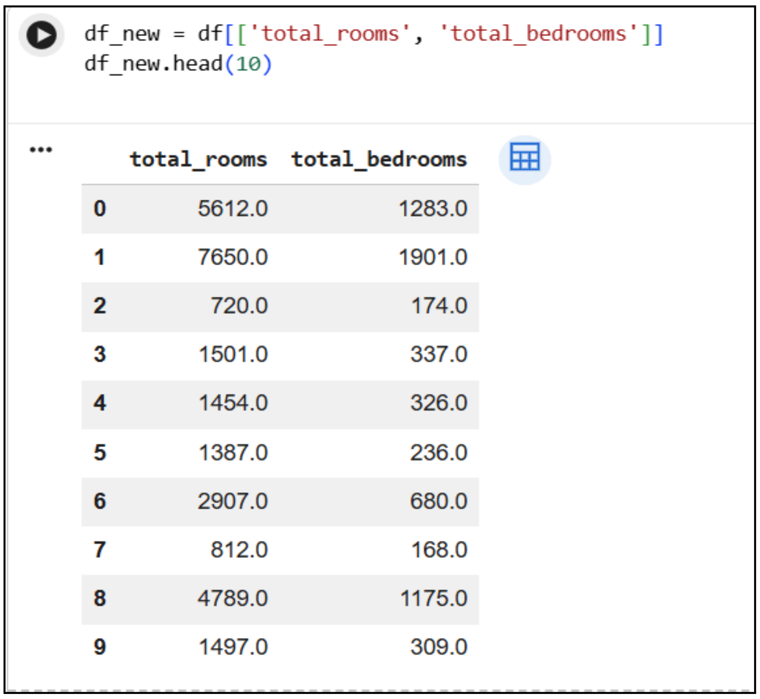
```
df = pd.read_csv('/content/sample_data/california_housing_train.csv')
df.head()
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  median_house_value
0    -114.31     34.19                15.0       5612.0          1283.0      1015.0       472.0         1.4936             66900.0
1    -114.47     34.40                19.0       7650.0          1901.0      1129.0       463.0         1.8200             80100.0
2    -114.56     33.69                17.0        720.0           174.0       333.0       117.0         1.6509             85700.0
3    -114.57     33.64                14.0       1501.0           337.0       515.0       226.0         3.1917             73400.0
4    -114.57     33.57                20.0       1454.0           326.0       624.0       262.0         1.9250             65500.0
```



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

2. Select Required Variables-

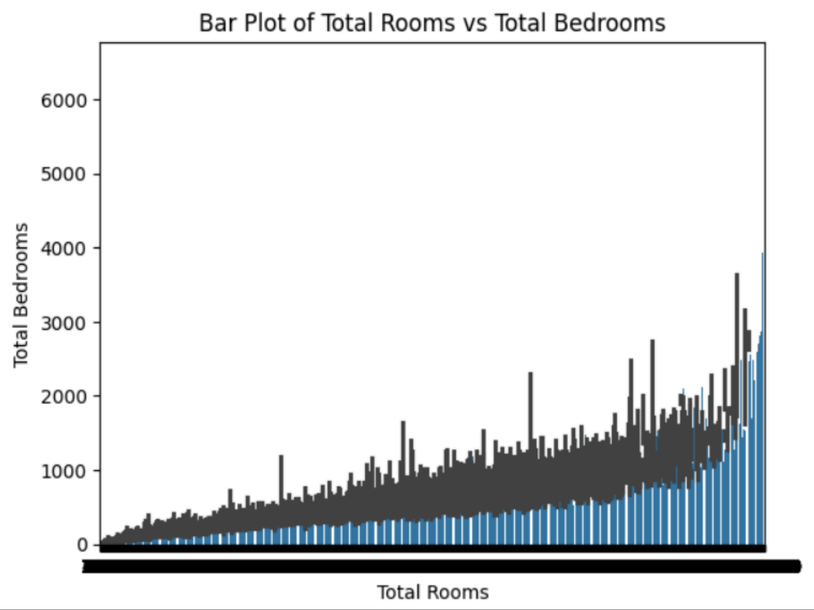


```
df_new = df[['total_rooms', 'total_bedrooms']]
df_new.head(10)
```

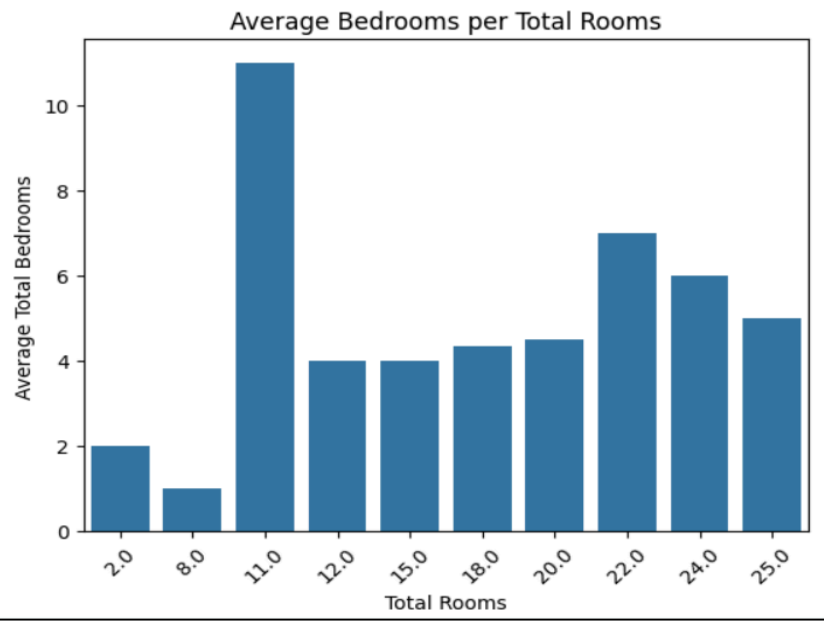
```
   total_rooms  total_bedrooms
0       5612.0          1283.0
1       7650.0          1901.0
2        720.0           174.0
3       1501.0           337.0
4       1454.0           326.0
5       1387.0           236.0
6       2907.0           680.0
7        812.0           168.0
8       4789.0          1175.0
9       1497.0           309.0
```

3. Create Bar Graph-

```python
sns.barplot(x=df_new['total_rooms'], y=df_new['total_bedrooms'])
plt.title("Bar Plot of Total Rooms vs Total Bedrooms")
plt.xlabel("Total Rooms")
plt.ylabel("Total Bedrooms")
plt.show()
```



```python
sns.barplot(x='total_rooms', y='total_bedrooms', data=df_grouped)
plt.title("Average Bedrooms per Total Rooms")
plt.xlabel("Total Rooms")
plt.ylabel("Average Total Bedrooms")
plt.xticks(rotation=45)
plt.show()
```



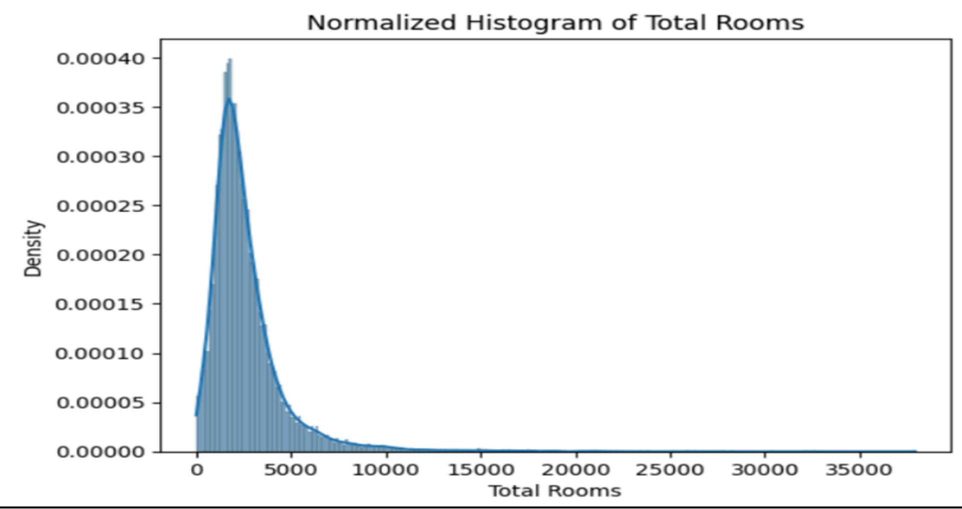
4. Create Contingency Table-

```
contingency_table = pd.crosstab(df_new['total_rooms'], df_new['total_bedrooms'])
contingency_table.head()
```

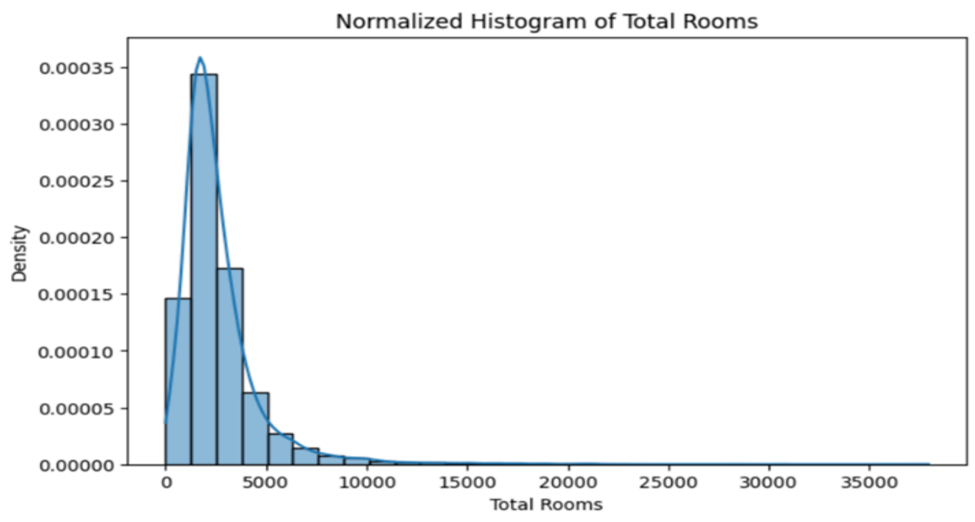
```
total_bedrooms  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0  ...  4407.0  4457.0  4492.0  4798.0  4819.0  4952.0  4957.0  5290.0  5471.0  6445.0
   total_rooms
      2.0         0    1    0    0    0    0    0    0    0     0  ...       0       0       0       0       0       0       0       0       0       0
      8.0         1    0    0    0    0    0    0    0    0     0  ...       0       0       0       0       0       0       0       0       0       0
     11.0         0    0    0    0    0    0    0    0    0     0  ...       0       0       0       0       0       0       0       0       0       0
     12.0         0    0    0    1    0    0    0    0    0     0  ...       0       0       0       0       0       0       0       0       0       0
     15.0         0    0    1    0    1    0    0    0    0     0  ...       0       0       0       0       0       0       0       0       0       0
```

5. Create Normalized Histogram-

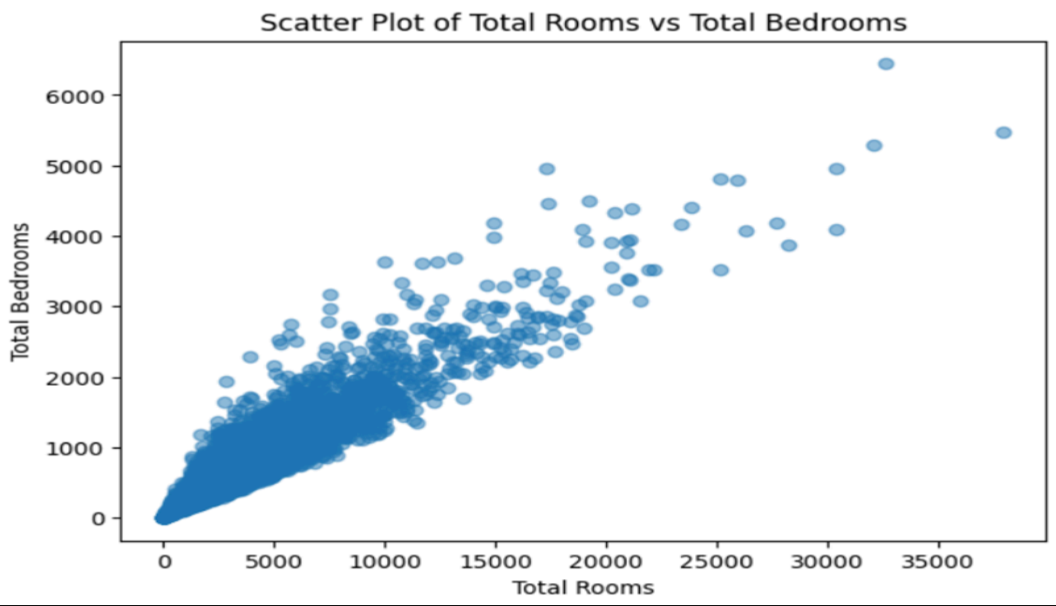
```
sns.histplot(df['total_rooms'], stat='density', kde=True)
plt.title("Normalized Histogram of Total Rooms")
plt.xlabel("Total Rooms")
plt.ylabel("Density")
plt.show()
```



```
plt.title("Normalized Histogram of Total Rooms")
plt.xlabel("Total Rooms")
plt.ylabel("Density")
plt.show()
```

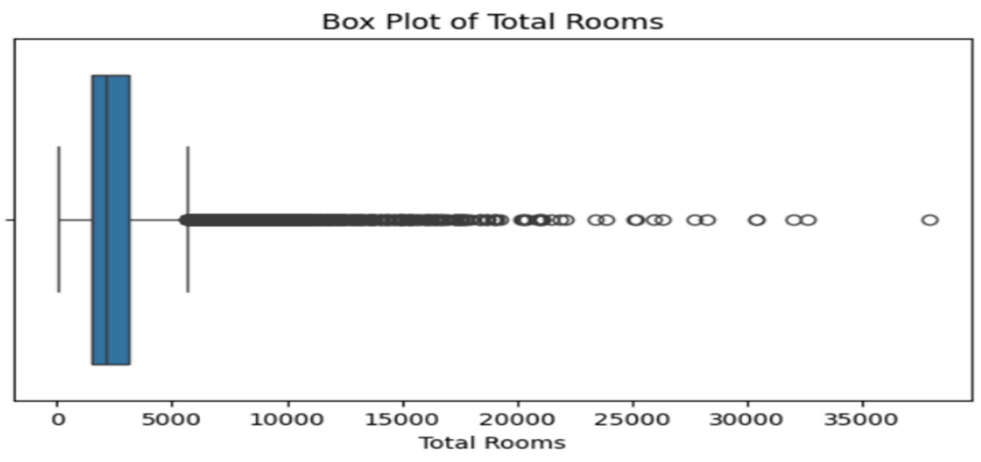


```
plt.title("Scatter Plot of Total Rooms vs Total Bedrooms")
plt.xlabel("Total Rooms")
plt.ylabel("Total Bedrooms")
plt.show()
```



```
sns.boxplot(x=df['total_rooms'])

plt.title("Box Plot of Total Rooms")
plt.xlabel("Total Rooms")
plt.show()
```



6. Describe the Graphs and Tables indicate-

The **bar graph** visually represents the relationship between two selected variables and helps in comparing their values across different categories or ranges. The **contingency table** provides a structured frequency distribution between the two variables, making it easier to understand how the values of one attribute are associated with the other.

The **normalized histogram** shows the probability distribution of a numerical variable. It helps in understanding the spread and concentration of data values and indicates the shape of the distribution, such as skewness or presence of extreme values. Normalization allows meaningful comparison by representing data in terms of density rather than frequency.

The **scatter plot and box plot** further support exploratory data analysis by showing relationships, data spread, and outliers. While the scatter plot helps in identifying trends and correlations between variables, the box plot summarizes the data distribution and clearly highlights outliers. Together, these visualizations provide better insight into the characteristics of the dataset.

Conclusion:
For the above AIDS experiment the dataset used is *california_housing_test.csv*. Visualization library like Seaborn helps in API abstraction across visualizations, statistical estimation and error bars, informative distributional summaries, specialized plots for categorical data, composite views onto multivariate datasets, and classes and functions for making complex graphics. Seaborn’s integration with Matplotlib allows you to use it across the many environments that Matplotlib supports, including exploratory analysis in notebooks, real-time interaction in GUI applications, and archival output in a number of raster and vector formats.