

## DA ASSIGNMENT – 3

**NAME** : Manikandan

**REGISTER NO** :911720104309

Load the dataset :

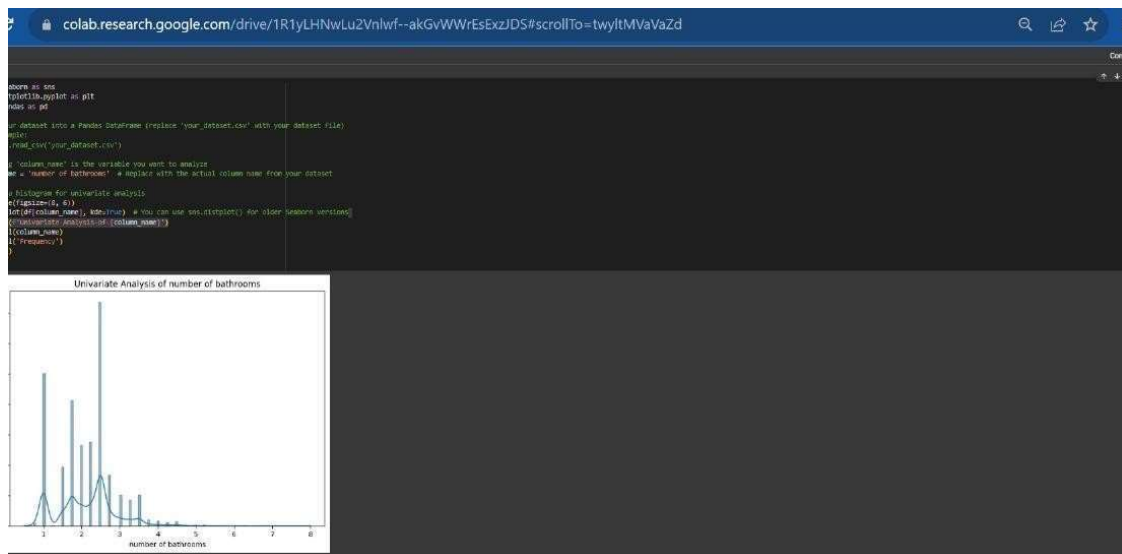
```
colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS

Untitled0.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 5
+ Code + Text
Connect
import pandas as pd
df=pd.read_csv("/content/drive/MyDrive/House Price India.csv")
df.head()
```

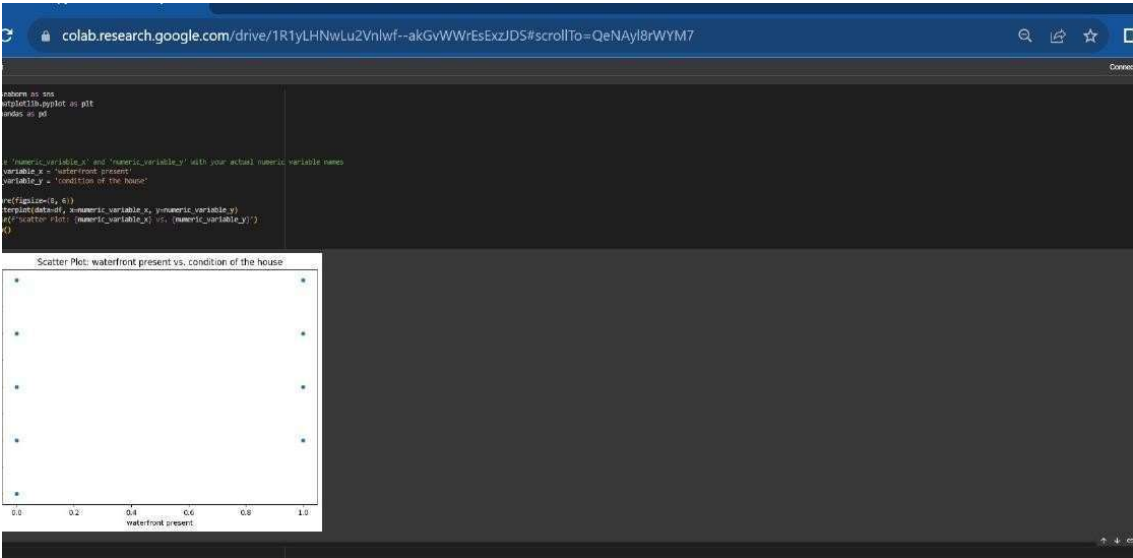
	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Postal Code	Latitude	Longitude
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	52.8645	-114.557
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	52.8878	-114.470
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004	52.8852	-114.468
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005	52.9532	-114.321
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006	52.9047	-114.485

5 rows x 23 columns

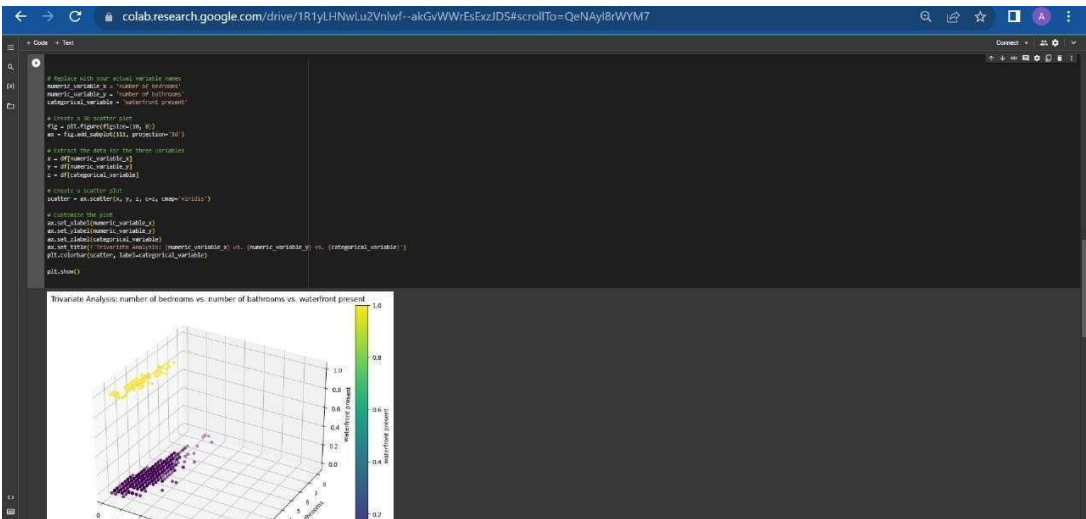
Univariate Analysis :



Bi - Variate Analysis :



Multi-Variate Analysis :



### Descriptive statistics on the dataset :

Unlabeled Jupyter - Collaboratory

colab.research.google.com/drive/1R1yLHNWLu2Vnlwf--akGvWWrEsExZJD5?scrollTo=QvXt9Ez-Xizf

Search

Share

Star

Fullscreen

Print

Close

Code

Text

Run

Reset

Undo

Redo

Copy

Paste

Clear

Close

Connect

Help

```
# Recursive descriptive statistics for numerical columns
import numpy as np
import pandas as pd
import sys

def recursive_stats(df, column):
    """Recursive function to calculate descriptive statistics for a column"""
    if column in df.columns:
        print(f'Column: {column}')
        # Calculate basic statistics
        count = df[column].count()
        min = df[column].min()
        max = df[column].max()
        mean = df[column].mean()
        std = df[column].std()
        median = df[column].median()
        mode = df[column].mode().tolist()
        skewness = df[column].skew()
        kurtosis = df[column].kurt()
        iqr = df[column].iqr()
        range = df[column].max() - df[column].min()
        unique = df[column].nunique()
        top_values = df[column].value_counts().nlargest(5)

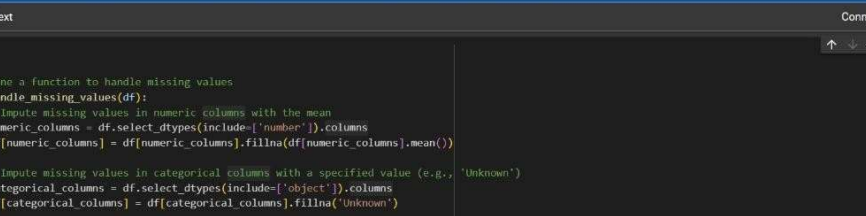
        # Print statistics
        print(f'Count: {count}')
        print(f'Min: {min}')
        print(f'Max: {max}')
        print(f'Mean: {mean}')
        print(f'Std: {std}')
        print(f'Median: {median}')
        print(f'Mode: {mode}')
        print(f'Skewness: {skewness}')
        print(f'Kurtosis: {kurtosis}')
        print(f'IQR: {iqr}')
        print(f'Range: {range}')
        print(f'Unique: {unique}')
        print(f'Top Values: {top_values}')

        # Recursive call for nested data
        if df[column].nunique() > 1:
            for value in df[column].unique():
                if value != '':
                    new_df = df[df[column] == value]
                    recursive_stats(new_df, column)

    else:
        print(f'Column {column} not found in DataFrame')

# Example usage
df = pd.read_csv('data.csv')
recursive_stats(df, 'column_name')
```

### Handle the Missing values :



The screenshot displays a Jupyter Notebook environment. The top toolbar includes icons for search, share, star, and window management. Below the toolbar, there are tabs for '+ Code' and '+ Text'. The main area shows a Python script with the following content:

```
# Define a function to handle missing values
def handle_missing_values(df):
    # Impute missing values in numeric columns with the mean
    numeric_columns = df.select_dtypes(include=['number']).columns
    df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

    # Impute missing values in categorical columns with a specified value (e.g., 'Unknown')
    categorical_columns = df.select_dtypes(include=['object']).columns
    df[categorical_columns] = df[categorical_columns].fillna("Unknown")

    return df

# Apply the function to handle missing values
df = handle_missing_values(df)

# Check if there are any remaining missing values
missing_values_count = df.isnull().sum().sum()
if missing_values_count == 0:
    print("All missing values have been handled.")
else:
    print(f"There are still {missing_values_count} missing values in the dataset.")

# Save the cleaned dataset to a new file (optional)
df.to_csv('cleaned_dataset.csv', index=False) # Replace with your desired file name
```

At the bottom of the notebook, a status bar indicates: "All missing values have been handled."