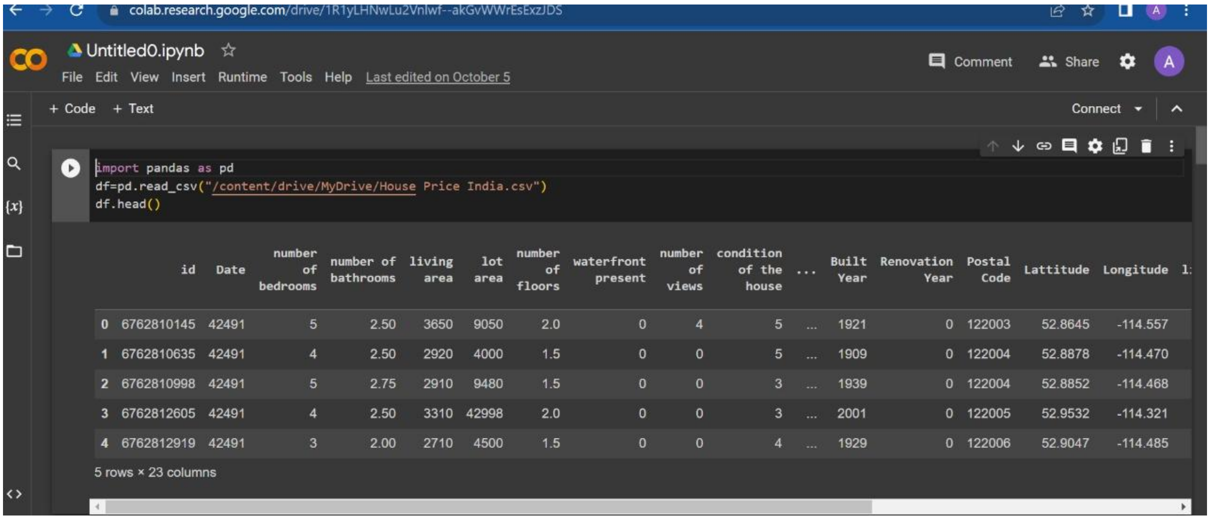
DA ASSIGNMENT – 3

NAME : G.POOJITH

REGISTER NO : 723920243013

Load the dataset :

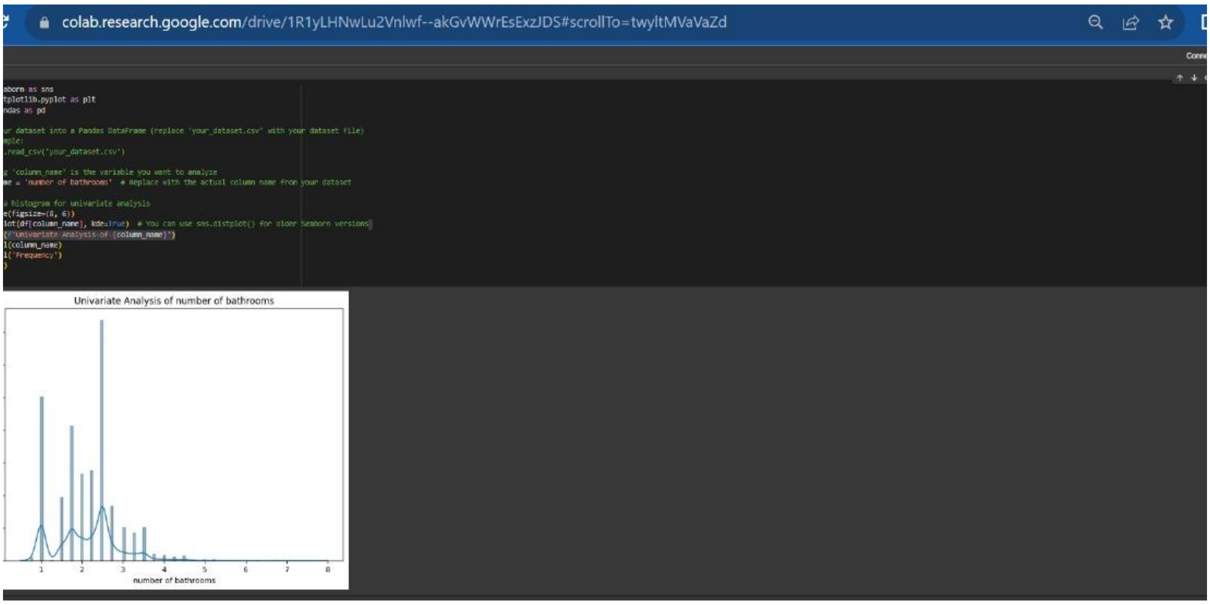

colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS
Untitled0.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 5
Comment Share
+ Code + Text
Connect

```
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	Lattitude	Longitude	1:
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	

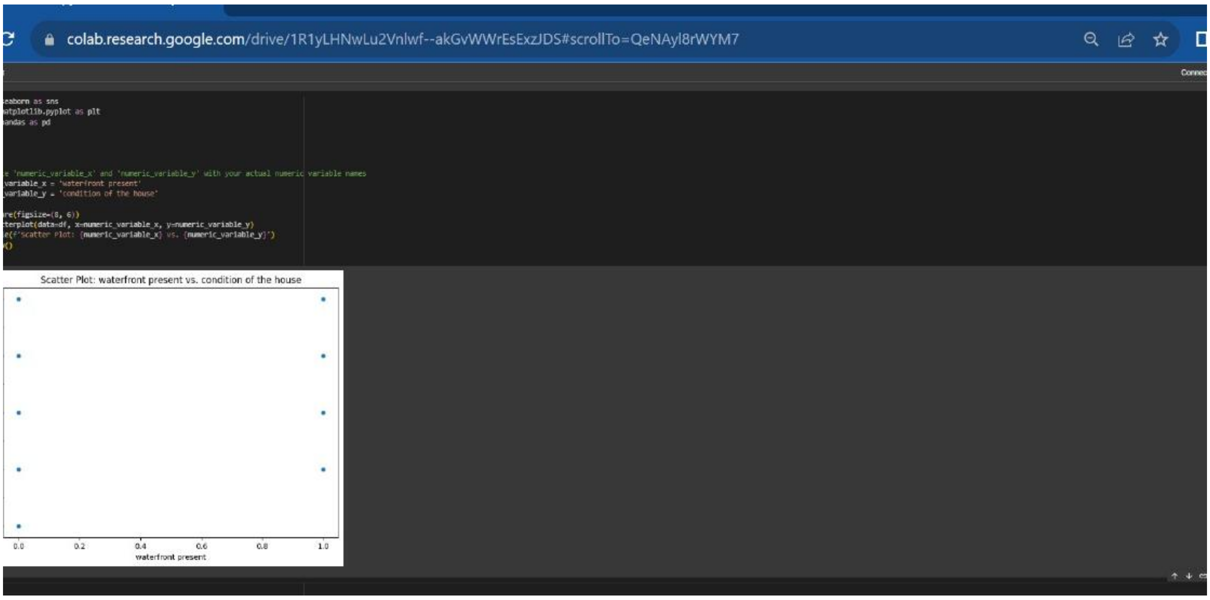
5 rows × 23 columns

Univariate Analysis :


colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS#scrollTo=twyltMVaVaZd
Univariate Analysis of number of bathrooms
1
2
3
4
5
6
7
8
number of bathrooms

```
aborn as sns
tplotlib.pyplot as plt
ndas as pd

ur dataset into a Pandas DataFrame (replace 'your_dataset.csv' with your dataset file)
mple:
.read_csv('your_dataset.csv')

g 'column_name' is the variable you want to analyze
me = 'number of bathrooms'  # Replace with the actual column name from your dataset

a histogram for univariate analysis
e(figsize=(8, 6))
lot(df[column_name], kde=True)  # You can use sns.distplot() for older Seaborn versions
(f'Univariate Analysis of {column_name}')
l(column_name)
l('Frequency')
)
```

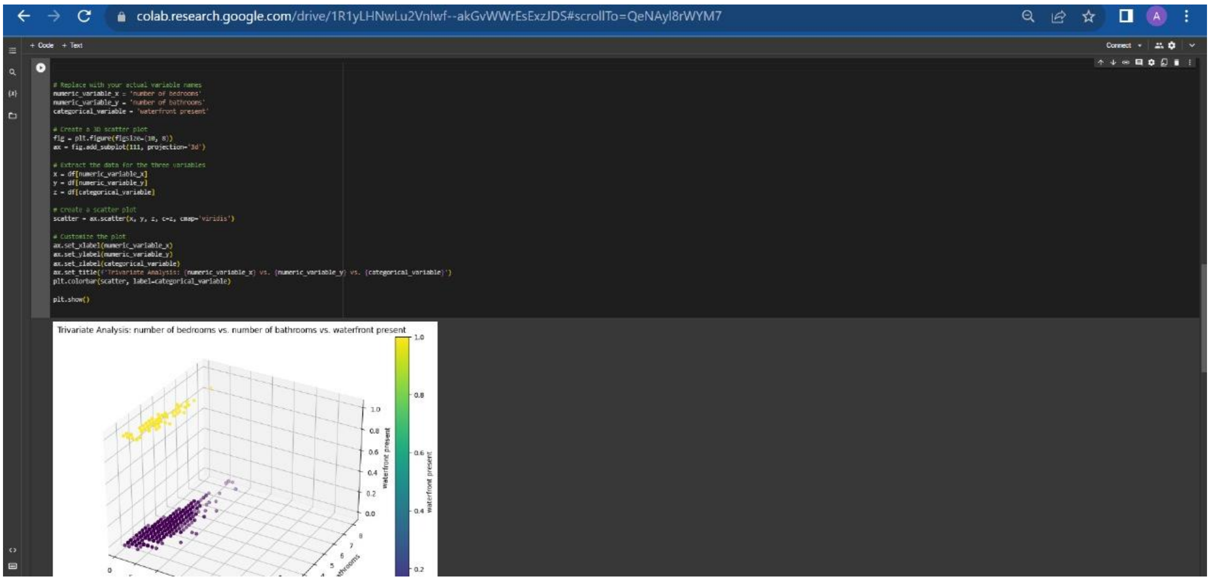
Bi - Variate Analysis :



```
eaborn as sns
atplotlib.pyplot as plt
andas as pd


e 'numeric_variable_x' and 'numeric_variable_y' with your actual numeric variable names
variable_x = 'waterfront present'
variable_y = 'condition of the house'

re(figsize=(8, 6))
terplot(data=df, x=numeric_variable_x, y=numeric_variable_y)
e(f'Scatter Plot: {numeric_variable_x} vs. {numeric_variable_y}')
()
```

colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS#scrollTo=QeNAyl8rWYM7
Connec
Scatter Plot: waterfront present vs. condition of the house
0.0
0.2
0.4
0.6
0.8
1.0
waterfront present

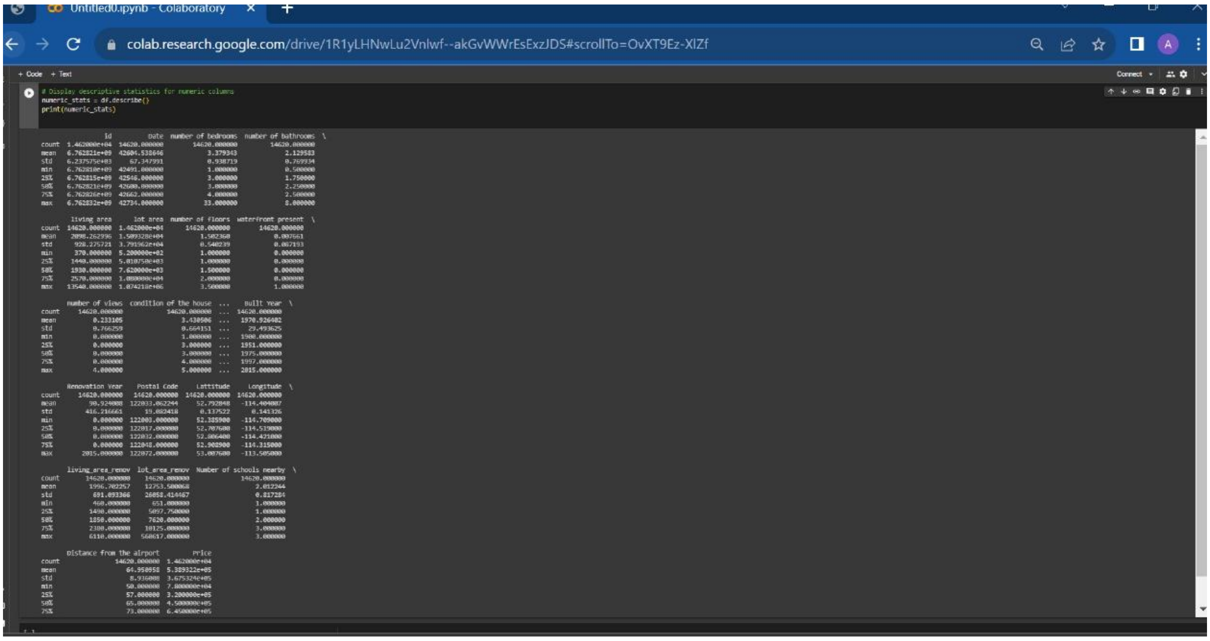
Multi-Variate Analysis :



```
# Replace with your actual variable names
numeric_variable_x = 'number of bedrooms'
numeric_variable_y = 'number of bathrooms'
categorical_variable = 'waterfront present'

# Create a 3D scatter plot
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Extract the data for the three variables
x = df[numeric_variable_x]
y = df[numeric_variable_y]
z = df[categorical_variable]

# create a scatter plot
scatter = ax.scatter(x, y, z, c=z, cmap='viridis')

# Customize the plot
ax.set_xlabel(numeric_variable_x)
ax.set_ylabel(numeric_variable_y)
ax.set_zlabel(categorical_variable)
ax.set_title(f'Trivariate Analysis: {numeric_variable_x} vs. {numeric_variable_y} vs. {categorical_variable}')
plt.colorbar(scatter, label=categorical_variable)

plt.show()
```

colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS#scrollTo=QeNAyl8rWYM7
+ Code
+ Text
Connect
Trivariate Analysis: number of bedrooms vs. number of bathrooms vs. waterfront present
1.0
0.8
0.6
0.4
0.2
0.0
waterfront present
0
5
6
7
8
bathrooms
1.0
0.8
0.6
0.4
0.2
waterfront present

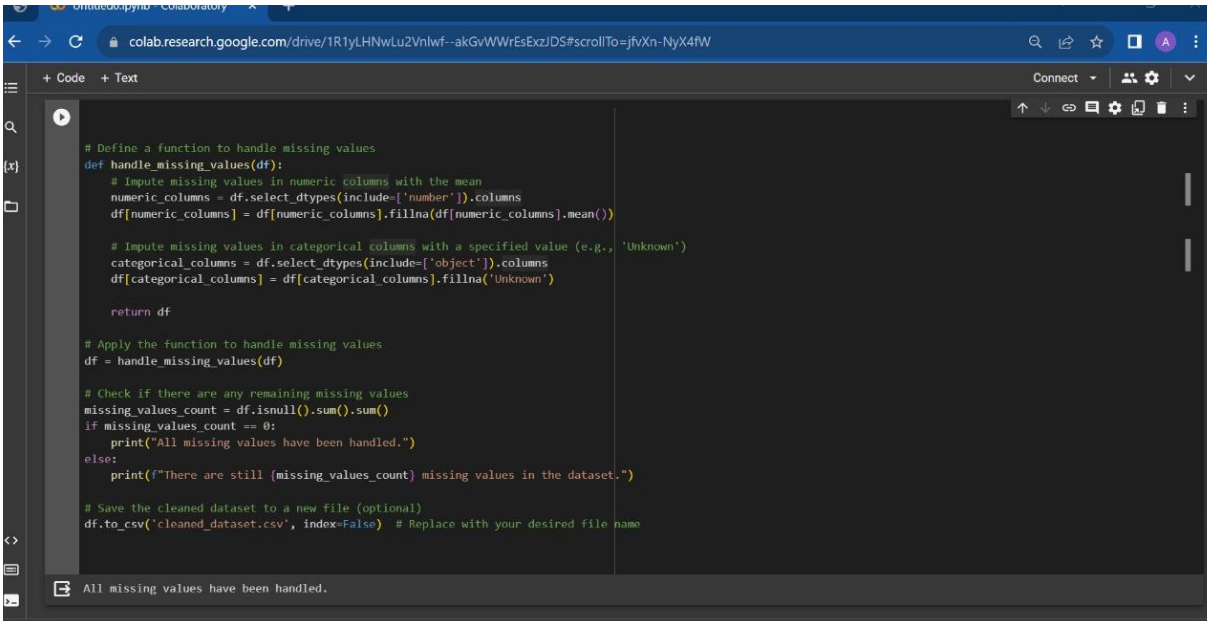
Descriptive statistics on the dataset :


Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS#scrollTo=OvXT9Ez-XlZf
+ Code
+ Text
Connect

```
# Display descriptive statistics for numeric columns
numeric_stats = df.describe()
print(numeric_stats)
```

```
                 Id          Date  number of bedrooms  number of bathrooms  \
count  1.462000e+04  14620.000000        14620.000000         14620.000000
mean   6.762821e+09  42604.538646            3.379343             2.129583
std    6.237575e+03     67.347991            0.938719             0.769934
min    6.762810e+09  42491.000000            1.000000             0.500000
25%    6.762815e+09  42546.000000            3.000000             1.750000
50%    6.762821e+09  42600.000000            3.000000             2.250000
75%    6.762826e+09  42662.000000            4.000000             2.500000
max    6.762832e+09  42734.000000           33.000000             8.000000

        living area      lot area  number of floors  waterfront present  \
count  14620.000000  1.462000e+04      14620.000000        14620.000000
mean    2098.262996  1.509328e+04          1.502360            0.007661
std      928.275721  3.791962e+04          0.540239            0.087193
min      370.000000  5.200000e+02          1.000000            0.000000
25%     1440.000000  5.010750e+03          1.000000            0.000000
50%     1930.000000  7.620000e+03          1.500000            0.000000
75%     2570.000000  1.080000e+04          2.000000            0.000000
max    13540.000000  1.074218e+06          3.500000            1.000000

       number of views  condition of the house  ...    Built Year  \
count     14620.000000            14620.000000  ...  14620.000000
mean          0.233105                3.430506  ...   1970.926402
std           0.766259                0.664151  ...     29.493625
min           0.000000                1.000000  ...   1900.000000
25%           0.000000                3.000000  ...   1951.000000
50%           0.000000                3.000000  ...   1975.000000
75%           0.000000                4.000000  ...   1997.000000
max           4.000000                5.000000  ...   2015.000000

       Renovation Year    Postal Code     Lattitude     Longitude  \
count     14620.000000   14620.000000  14620.000000  14620.000000
mean         90.924008  122033.062244     52.792848   -114.404007
std         416.216661      19.082418      0.137522      0.141326
min           0.000000  122003.000000     52.385900   -114.709000
25%           0.000000  122017.000000     52.707600   -114.519000
50%           0.000000  122032.000000     52.806400   -114.421000
75%           0.000000  122048.000000     52.908900   -114.315000
max        2015.000000  122072.000000     53.007600   -113.505000

       living_area_renov  lot_area_renov  Number of schools nearby  \
count       14620.000000    14620.000000              14620.000000
mean         1996.702257    12753.500068                  2.012244
std           691.093366    26058.414467                  0.817284
min           460.000000      651.000000                  1.000000
25%          1490.000000     5097.750000                  1.000000
50%          1850.000000     7620.000000                  2.000000
75%          2300.000000    10125.000000                  3.000000
max          6110.000000   560617.000000                  3.000000

       Distance from the airport         Price
count               14620.000000  1.462000e+04
mean                   64.950958  5.389322e+05
std                     8.936008  3.675324e+05
min                    50.000000  7.800000e+04
25%                    57.000000  3.200000e+05
50%                    65.000000  4.500000e+05
75%                    73.000000  6.450000e+05
```

Handle the Missing values :


Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1R1yLHNwLu2Vnlwf--akGvWWrEsExzJDS#scrollTo=jfvXn-NyX4fW
+ Code
+ Text
Connect

```
# 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
```

```
All missing values have been handled.
```