MOUNT GOOGLE DRIVE

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

IMPORT REQUIRED PACKAGES

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

LOAD THE DATASET

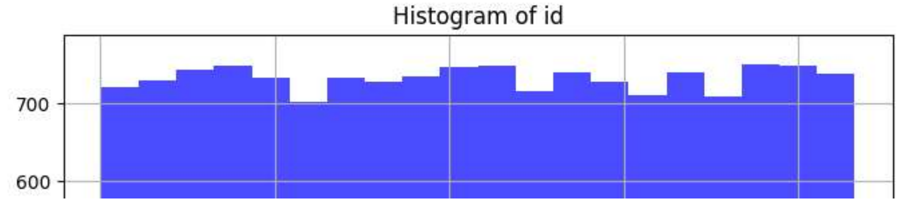
```
df=pd.read_csv("/content/drive/MyDrive/House Price India.csv")
df.head()
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0

5 rows × 23 columns

UNIVARIATE ANALYSIS

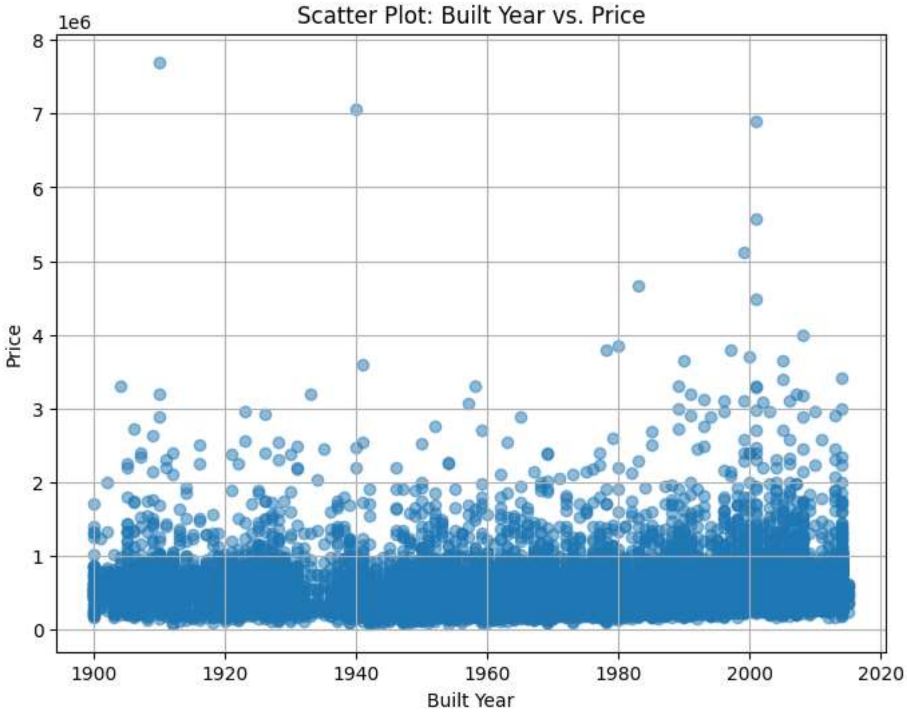
```
plt.figure(figsize=(8, 6))
column_name="id"
plt.hist(df[column_name], bins=20, color='blue', alpha=0.7)
plt.title("Histogram of " + column_name)
plt.xlabel(column_name)
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```



BIVARIATE ANALYSIS

```
x_variable = 'Built Year'
y_variable = 'Price'

# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df[x_variable], df[y_variable], alpha=0.5)
plt.title("Scatter Plot: " + x_variable + " vs. " + y_variable)
plt.xlabel(x_variable)
plt.ylabel(y_variable)
plt.grid(True)
plt.show()
```



MULTIVARIATE ANALYSIS

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

# Read in the DataFrame
df = pd.read_csv('/content/drive/MyDrive/House Price India.csv')

# plotting histogram
plt.hist(df['Price'],bins = 35,
         alpha = 0.45, color = 'red')
plt.show()
```

6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8

DESCRIPTIVE STATISTICS

```
descriptive_stats = df.describe()

# Display the descriptive statistics
print("Descriptive Statistics:")
print(descriptive_stats)
```

```
Descriptive Statistics:
                 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%          2380.000000    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  
```

```
mean_value = df['Price'].mean()
print("Mean of 'your_column':", mean_value)
```

```
Mean of 'your_column': 538932.2183310534
```

Handle the Missing values.

```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)

# Option 1: Remove rows with missing values
df_cleaned = df.dropna()

# Option 2: Fill missing values with a specific value (e.g., mean or median)
# Replace 'your_column' with the actual column name
mean_value = df['Price'].mean()
data_filled = df.fillna(mean_value)

# Option 3: Forward fill or backward fill missing values
data_ffill = df.ffill()  # Forward fill missing values
data_bfill = df.bfill()  # Backward fill missing values

# Option 4: Interpolate missing values
data_interpolated = df.interpolate()
```

```
Missing Values:
id                                       0
Date                                     0
number of bedrooms                       0
number of bathrooms                      0
living area                              0
lot area                                 0
number of floors                         0
waterfront present                       0
number of views                          0
condition of the house                   0
grade of the house                       0
Area of the house(excluding basement)    0
Area of the basement                     0
Built Year                               0
Renovation Year                          0
Postal Code                              0
Lattitude                                0
Longitude                                0
living_area_renov                        0
lot_area_renov                           0
Number of schools nearby                 0
Distance from the airport                0
Price                                    0
dtype: int64
```