Name-Tanaji Kolekar
Div-CC
Batch-CC2
Data set -House Price
Roll No-26
PRN-202401050022

```
import pandas as pd
import numpy as np
```

Double-click (or enter) to edit

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files house_price.csv
• house_price.csv(text/csv) - 29981 bytes, last modified: 4/27/2025 - 100% done
Saving house_price.csv to house_price (1).csv
```

```
df = pd.read_csv('house_price.csv')
```

1. Total number of houses and features

```
print(f"Rows: {df.shape[0]}, Columns: {df.shape[1]}")
```

```
Rows: 545, Columns: 13
```

2. Display first 5 houses using .sample()

```
print(df.sample(5))
```

```
       price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
50   7420000  7440         3          2        4      yes        no       no
441  3220000  4370         3          1        2      yes        no       no
530  2240000  1950         3          1        1       no        no       no
195  4970000  4410         4          3        2      yes        no      yes
76   6650000  6420         3          2        3      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
50               no              no        1      yes      unfurnished
441              no              no        0       no      unfurnished
530             yes              no        0       no      unfurnished
195              no              no        2       no   semi-furnished
76               no             yes        0      yes        furnished
```

3. Display last 5 houses with selected columns

```
print(df.tail(5)[['area','price','basement']])
```

```
     area    price basement
540  3000  1820000      yes
541  2400  1767150       no
542  3620  1750000       no
543  2910  1750000       no
544  3850  1750000       no
```

4. List all feature names sorted alphabetically

```
print(sorted(df.columns))
```

```
['airconditioning', 'area', 'basement', 'bathrooms', 'bedrooms', 'furnishingstatus', 'guestroom', 'hotwaterheating', 'mainroad', 'parking', 'prefarea', '
```

5. Check data types and count them

```
print(df.dtypes.value_counts())
```

```
object    7
int64     6
Name: count, dtype: int64
```

6. Find columns having more than 10 missing values

```
missing_cols = df.isnull().sum()
print(missing_cols[missing_cols > 10])
```

```
Series([], dtype: int64)
```

*7. find the correlation between the price and the area *

```
correlation = np.corrcoef(df['price'], df['area'])[0, 1]
print(f"Correlation between price and area: {correlation}")
```

```
Correlation between price and area: 0.5359973457780796
```

** 8.Find the average price of houses for each number of bedrooms.**

```
avg_price_per_bedroom = df.groupby('bedrooms')['price'].mean()
print(avg_price_per_bedroom)
```

```
bedrooms
1    2.712500e+06
2    3.632022e+06
3    4.954598e+06
4    5.729758e+06
5    5.819800e+06
6    4.791500e+06
Name: price, dtype: float64
```

9.Calculate the average price of houses based on the number of stories

```
avg_price_by_stories = df.groupby('stories')['price'].mean()
print(avg_price_by_stories)
```

```
stories
1    4.170659e+06
2    4.764074e+06
3    5.685436e+06
4    7.208450e+06
Name: price, dtype: float64
```

10. Find the percentage of houses that have access to the main road (assuming mainroad is a binary feature: 1 for access, 0 for no access).

```
mainroad_percentage = np.mean(df['mainroad'] == 1) * 100
print(f"Percentage of houses with main road access: {mainroad_percentage}%")
```

```
Percentage of houses with main road access: 0.0%
```

11. Compare the price of houses with and without a guestroom.

```
price_with_guestroom = df[df['guestroom'] == 1]['price'].mean()
price_without_guestroom = df[df['guestroom'] == 0]['price'].mean()

print(f"Average price of houses with guestroom: {price_with_guestroom}")
print(f"Average price of houses without guestroom: {price_without_guestroom}")
```

```
Average price of houses with guestroom: nan
Average price of houses without guestroom: nan
```

12. Calculate average SalePrice rounded to 2 decimals

```
print(round(df['price'].mean(), 2))
```

```
4766729.25
```

13. Find houses with SalePrice > 1.5 times mean using np.where

```
high_price = np.where(df['price'] > 1.5 * df['price'].mean(), True, False)
print(df[high_price])
```

18	no	yes	2	no	furnished
19	no	yes	1	yes	semi-furnished
20	yes	no	2	no	semi-furnished
21	no	yes	2	no	unfurnished
22	no	yes	1	no	furnished
23	no	yes	1	no	furnished
24	no	yes	2	no	furnished
25	no	yes	2	yes	furnished
26	no	yes	0	yes	semi-furnished
27	no	no	1	no	semi-furnished
28	yes	no	2	no	unfurnished
29	no	yes	1	yes	semi-furnished
30	no	yes	2	no	unfurnished
31	no	yes	2	no	semi-furnished
32	no	yes	1	yes	furnished
33	no	no	1	no	unfurnished
34	no	yes	1	no	furnished
35	no	yes	2	no	furnished
36	yes	no	1	yes	furnished
37	no	yes	2	no	furnished
38	no	yes	2	no	unfurnished
39	no	yes	1	no	semi-furnished
40	no	yes	0	yes	furnished
41	no	yes	0	yes	furnished
42	no	yes	2	no	unfurnished
43	no	no	2	no	semi-furnished
44	no	yes	1	no	furnished
45	no	yes	0	no	semi-furnished
46	no	yes	1	no	furnished
47	no	yes	3	yes	furnished
48	no	no	1	no	unfurnished
49	no	yes	0	yes	semi-furnished
50	no	no	1	yes	unfurnished
51	no	yes	1	no	unfurnished
52	no	yes	1	no	furnished
53	no	yes	2	no	semi-furnished
54	no	yes	1	no	semi-furnished
55	no	yes	1	no	unfurnished
56	no	no	1	yes	semi-furnished
57	no	yes	1	yes	furnished
58	no	yes	1	no	semi-furnished
59	no	yes	1	no	furnished

14. Average SalePrice per Area (descending)

```
print(df.groupby('area')['price'].mean().sort_values(ascending=False))
```

```
area
7420     12355000.0
8960     12250000.0
```

```
9960     12250000.0
7500     11532500.0
16200    10150000.0
            ...
2400      1933575.0
1700      1890000.0
3649      1890000.0
2990      1855000.0
3620      1750000.0
Name: price, Length: 284, dtype: float64
```

15. Create a new column 'TotalArea' with apply() row-wise

```
df['TotalArea'] = df.apply(lambda row: row['stories'] + row['bathrooms'], axis=1)
print(df[['furnishingstatus', 'TotalArea']])
```

```
    furnishingstatus  TotalArea
0          furnished          5
1          furnished          8
2     semi-furnished          4
3          furnished          4
4          furnished          3
..               ...        ...
540      unfurnished          2
541   semi-furnished          2
542      unfurnished          2
543        furnished          2
544      unfurnished          3

[545 rows x 2 columns]
```

16. Find correlation between important features

```
print(df[['area', 'stories', 'price']].corr())
```

```
             area   stories     price
area     1.000000  0.083996  0.535997
stories  0.083996  1.000000  0.420712
price    0.535997  0.420712  1.000000
```

17. Calculate standard deviation of SalePrice manually

```
saleprice_std = np.sqrt(np.mean((df['price'] - df['price'].mean())**2))
print(f"Standard Deviation of price: {saleprice_std:.2f}")
```

```
Standard Deviation of price: 1868722.83
```

18. Descriptive stats for numerical columns using .agg()

```
print(df.select_dtypes(include=[np.number]).agg(['mean', 'std', 'min', 'max']))
```

```
              price          area  bedrooms  bathrooms   stories   parking  \
mean  4.766729e+06   5150.541284  2.965138   1.286239  1.805505  0.693578
std   1.870440e+06   2170.141023  0.738064   0.502470  0.867492  0.861586
min   1.750000e+06   1650.000000  1.000000   1.000000  1.000000  0.000000
max   1.330000e+07  16200.000000  6.000000   4.000000  4.000000  3.000000

      TotalArea
mean   3.091743
std    1.135501
min    2.000000
max    8.000000
```

19. Calculate the average price for houses with and without air conditioning.

```
price_with_ac = df[df['airconditioning'] == 1]['price'].mean()
price_without_ac = df[df['airconditioning'] == 0]['price'].mean()

print(f"Average price of houses with air conditioning: {price_with_ac}")
print(f"Average price of houses without air conditioning: {price_without_ac}")
```

```
Average price of houses with air conditioning: nan
Average price of houses without air conditioning: nan
```

20. Find the number of houses for each parking space category.

```
parking_counts = df['parking'].value_counts()
print(parking_counts)
```

```
parking
0    299
1    126
```