DAP --> DATA ANALYSIS PROCESS

1. Problem Finding
2. Data wrangling
    - 2.1 Data Gathering (csv ,xlsx ,db ,api)
    - 2.2 Data Assessing
    - 2.3 Data Preprocessing
3. EDA --> Exploratory Data Analysis ---> Visualise Data, Statistical Methods
4. Reporting , Conclusions, Summary, Advisories
5. Dashboarding --> Power Bi and Tableau

In [1]:
```
# importing major libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# additional libraries
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#importing dataset
df = pd.read_csv('Magicbricks.csv')
```

Data Assessing

About Company

- Magicbricks is India’s leading online real estate platform that connects property buyers, sellers, and renters. It offers verified listings, market insights, price trends, and smart tools to help users make informed property decisions. Magicbricks simplifies property search by providing trusted information, advanced filters, and expert guidance across residential and commercial real estate markets.

In [3]:
```
# overview data
df.head()
```

Out[3]:

	Area	BHK	Bathroom	Furnishing	Location	District	Locality	Parking	
0	950.0	2	2.0	Furnished	Karol Bagh	Central Delhi	DDA MIG Flats Prasad Nagar Phase 2, Prasad Nag...	1.0	Rea
1	535.0	2	1.0	Furnished	Karol Bagh	Central Delhi	Dev Nagar, Karol Bagh	1.0	Rea
2	1280.0	3	3.0	Furnished	Karol Bagh	Central Delhi	Karol Bagh	2.0	Rea
3	1135.0	3	3.0	Furnished	Karol Bagh	Central Delhi	The Amaryllis, Karol Bagh	2.0	A
4	1135.0	3	3.0	Furnished	Karol Bagh	Central Delhi	The Amaryllis, Karol Bagh	2.0	A

In [4]:
```
#shape
df.shape
```

Out[4]:
```
(1214, 13)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Area', 'BHK', 'Bathroom', 'Furnishing', 'Location', 'District',
       'Locality', 'Parking', 'Status', 'Transaction', 'Type', 'Per_Sqft',
       'Price'],
      dtype='object')
```

📌 Data Card: MagicBricks Property Dataset (Sample)

Dataset Name

MagicBricks Residential Property Listings – Sample Data

Dataset Description

This dataset is a **sample of residential property listings** sourced from MagicBricks. It contains structured information about apartments and builder floors

in **Karol Bagh, Central Delhi**, including property size, configuration, furnishing status, price, and transaction details.

Number of Records

- **5 property listings**

Number of Features

- **14 attributes**

Feature Description

Column Name	Description
Area	Built-up area of the property (in sq. ft.)
BHK	Number of bedrooms
Bathroom	Number of bathrooms
Furnishing	Furnishing status (Furnished)
Location	Area name
District	Administrative district
Locality	Specific locality or project name
Parking	Number of parking spaces
Status	Property possession status
Transaction	Transaction type (Resale / New Property)
Type	Property type (Apartment / Builder Floor)
Per_Sqft	Price per square foot
Price	Total property price
Index	Row identifier

📌 Types of Data Errors

1. Completeness Errors

- Occur when **required data is missing**

- Values may be `NULL`, blank, or not recorded

Examples:

- Missing age of a customer
- Empty salary field
- Address not provided

---

2. Validity Errors

- Occur when data **does not follow defined rules, constraints, or data types**

Examples:

- Number of children stored as `float` instead of `integer`
- Duplicate values in a unique column (e.g., `patient_id`)
- Salary recorded as `-10000`
- Age recorded as a negative value

---

3. Accuracy Errors

- Data is **technically valid** but **factually incorrect or unrealistic**

Examples:

- Body weight of an adult recorded as `13 kg`
- Age recorded as `200 years`
- Height recorded as `20 cm` for an adult

---

4. Inconsistency Errors

- Same information stored in **different formats or representations**

Examples:

- `New York City`, `NYC`, `NewYork`
- Name variations: `Gaurav` → `Gauri`
- Gender stored as `M`, `Male`, `male`

✅ Summary Table

Error Type	Description	Example
Completeness	Missing data	Age = NULL
Validity	Rule or datatype violation	Salary = -10000
Accuracy	Unrealistic values	Age = 200
Inconsistency	Multiple formats	NYC vs New York City

📌 Types of Data

1. Dirty Data

Data with **quality issues**.

- **Completeness** – missing values
- **Validity** – wrong data type, negative age/salary
- **Accuracy** – unrealistic values (age = 200, adult weight = 13 kg)
- **Inconsistency** – same data in different formats (NYC / New York City)

---

2. Messy Data

Data with **structural issues**.

- **Pivoted structure** (months as columns)
- **Sparsity issue** (too many NULLs)
- **Wide tables**

🔑 Key Point

- **Dirty Data → value problems**
- **Messy Data → structure problems**

In [6]:
```
# Seaking Information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1214 entries, 0 to 1213
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Area         1214 non-null   float64
 1   BHK          1214 non-null   int64  
 2   Bathroom     1212 non-null   float64
 3   Furnishing   1214 non-null   object 
 4   Location     1214 non-null   object 
 5   District     1214 non-null   object 
 6   Locality     1214 non-null   object 
 7   Parking      1182 non-null   float64
 8   Status       1214 non-null   object 
 9   Transaction  1214 non-null   object 
 10  Type         1209 non-null   object 
 11  Per_Sqft     973 non-null    float64
 12  Price        1214 non-null   int64  
dtypes: float64(4), int64(2), object(7)
memory usage: 123.4+ KB
```

📌 Data Quality Observations

- The dataset contains **null (missing) values**.
- **Bathroom** is stored as a `float`, which is a **validity issue** since it should be an integer.
- **Parking** is also stored as a `float`, indicating a **data type validity issue**.
- The dataset consists of **6 numerical features** and **7 categorical (object) features**.

In [7]:
```
# Seeking description
df.describe()
```

Out[7]:

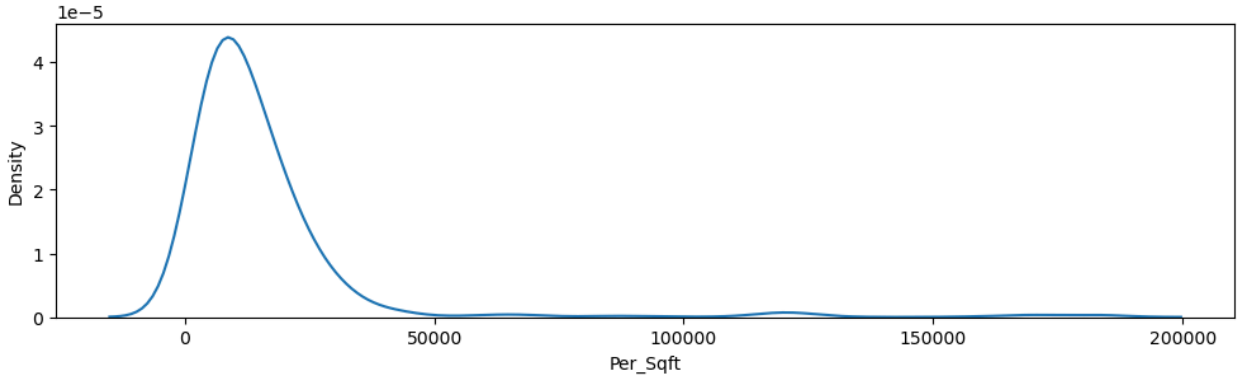
	Area	BHK	Bathroom	Parking	Per_Sqft	
count	1214.000000	1214.000000	1212.000000	1182.000000	973.000000	1.21
mean	1451.850751	2.778418	2.523927	1.708122	15574.885920	2.07
std	1586.472855	0.946811	1.017723	5.717177	21574.389007	2.56
min	28.000000	1.000000	1.000000	1.000000	1259.000000	1.00
25%	800.000000	2.000000	2.000000	1.000000	6154.000000	5.60
50%	1150.000000	3.000000	2.000000	1.000000	10838.000000	1.40
75%	1620.000000	3.000000	3.000000	2.000000	17647.000000	2.50
max	24300.000000	10.000000	7.000000	114.000000	183333.000000	2.40

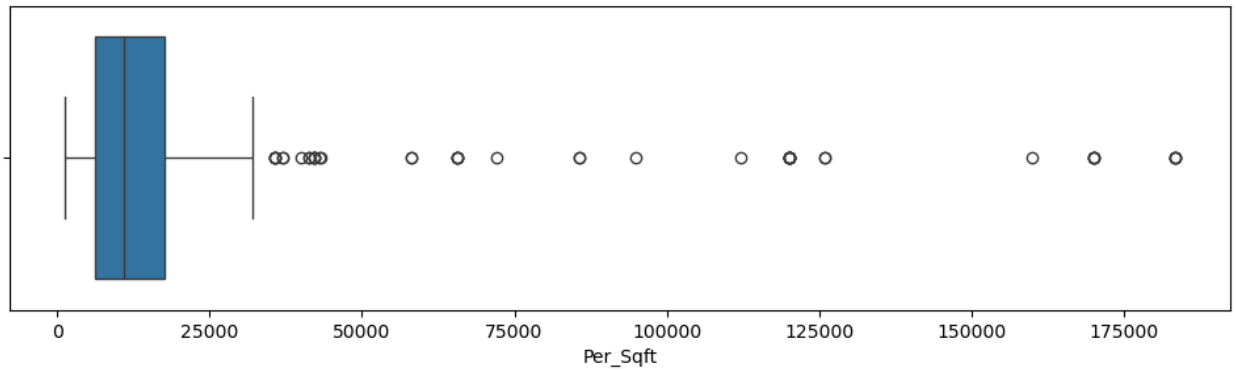
- **Accuracy issues identified in the dataset**:
    - The **Area** column shows extreme values, with a minimum of **28 sq ft** and a maximum of **24,300 sq ft**, which requires validation.
    - Properties with **10 BHK** need to be reviewed, as such configurations are rare and may be data entry errors.
    - A record showing **114 parking spaces** is highly unrealistic and indicates a possible accuracy issue.
- House prices range from **₹10 lakh to ₹24 crore**

In [8]:
```
# Completness
df.isnull().sum().sum()
# Percentage
df.isnull().mean()*100
```

Out[8]:
```
Area            0.000000
BHK             0.000000
Bathroom        0.164745
Furnishing      0.000000
Location        0.000000
District        0.000000
Locality        0.000000
Parking         2.635914
Status          0.000000
Transaction     0.000000
Type            0.411862
Per_Sqft       19.851730
Price           0.000000
dtype: float64
```

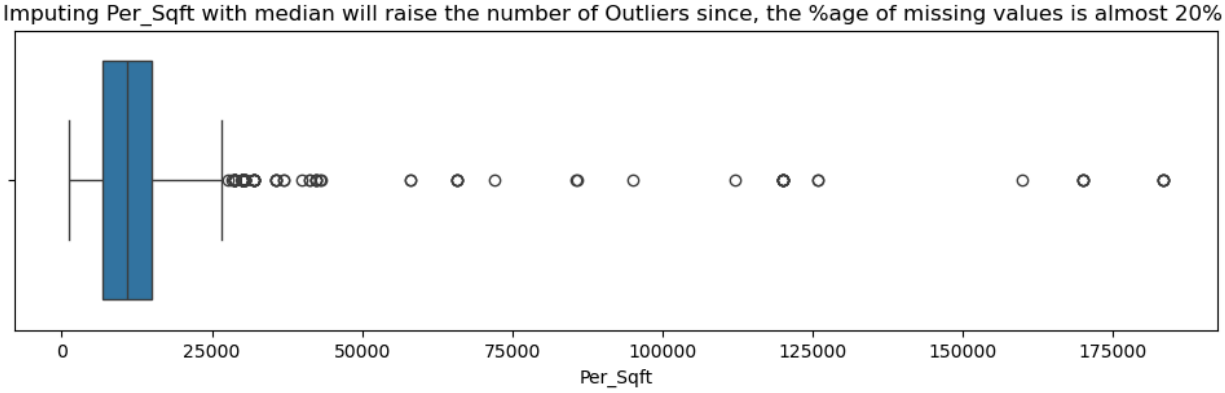
In [9]:
```
df.Per_Sqft.describe()
# Visualisation
plt.figure(figsize=(12,3))
sns.kdeplot(data=df,x='Per_Sqft')
plt.show()
plt.figure(figsize=(12,3))
sns.boxplot(data=df,x='Per_Sqft')
plt.show()

print('skewness',df.Per_Sqft.skew())
```






```
skewness 5.264871991245157
```

In [10]:
```
plt.figure(figsize=(12,3))
plt.title('Imputing Per_Sqft with median will raise the number of Outliers sir
sns.boxplot(x=df.Per_Sqft.fillna(df.Per_Sqft.median()))
plt.show()
```



In [11]:
```
df.columns
```

Out[11]:
```
Index(['Area', 'BHK', 'Bathroom', 'Furnishing', 'Location', 'District',
       'Locality', 'Parking', 'Status', 'Transaction', 'Type', 'Per_Sqft',
       'Price'],
      dtype='object')
```

In [12]:
```
# 'Area' --> 'Price'
# 'Per_Sqft' --> Price/Area

df.Per_Sqft = df.Per_Sqft.fillna(df.Price/df.Area)
```

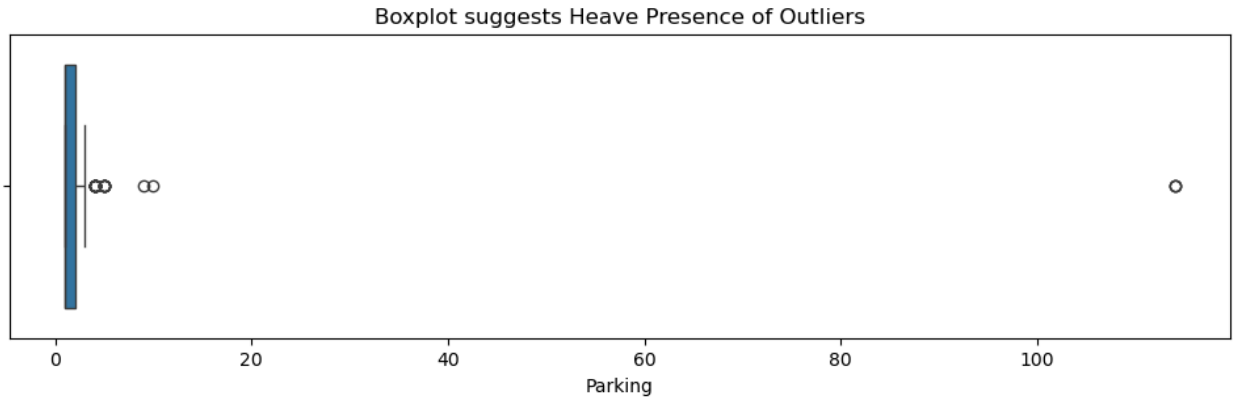
In [13]:
```
df.Per_Sqft.isnull().sum()
# df.Per_Sqft is free from issue of completeness
```

Out[13]:
```
np.int64(0)
```

In [14]:
```
df.isnull().sum()
df.isnull().mean()*100
```

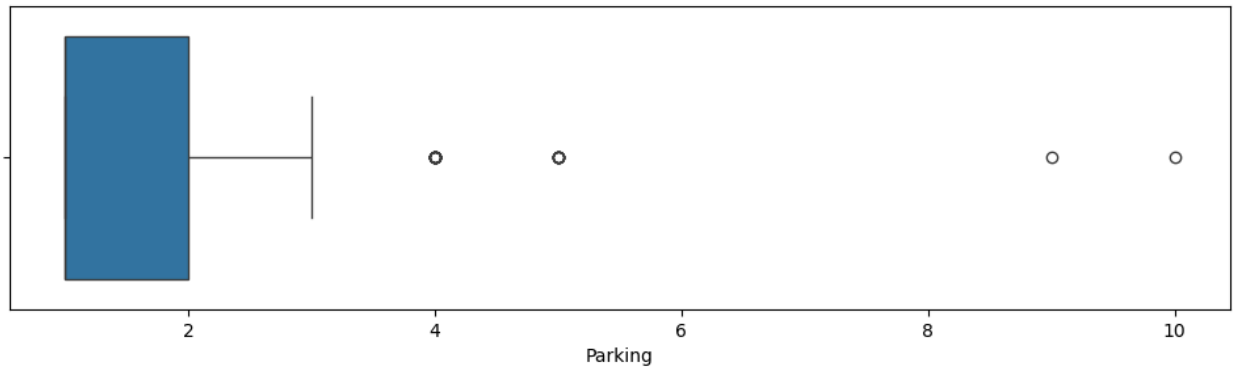
Out[14]:
```
Area           0.000000
BHK            0.000000
Bathroom       0.164745
Furnishing     0.000000
Location       0.000000
District       0.000000
Locality       0.000000
Parking        2.635914
Status         0.000000
Transaction    0.000000
Type           0.411862
Per_Sqft       0.000000
Price          0.000000
dtype: float64
```

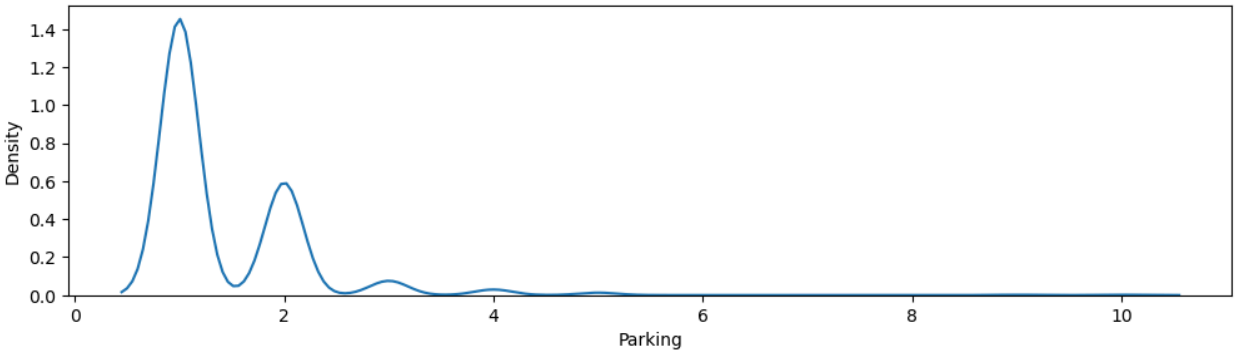
In [15]:
```
# outlier
plt.figure(figsize=(12,3))
plt.title('Boxplot suggests Heave Presence of Outliers')
sns.boxplot(x=df.Parking)
plt.show()
```



In [16]:
```
df[df.Parking>100]
df.Parking = np.where(df.Parking>100,1,df.Parking)
```

In [17]:
```
plt.figure(figsize=(12,3))
sns.boxplot(x=df.Parking)
plt.show()
plt.figure(figsize=(12,3))
sns.kdeplot(x=df.Parking)
plt.show()
df.Parking.skew()
```



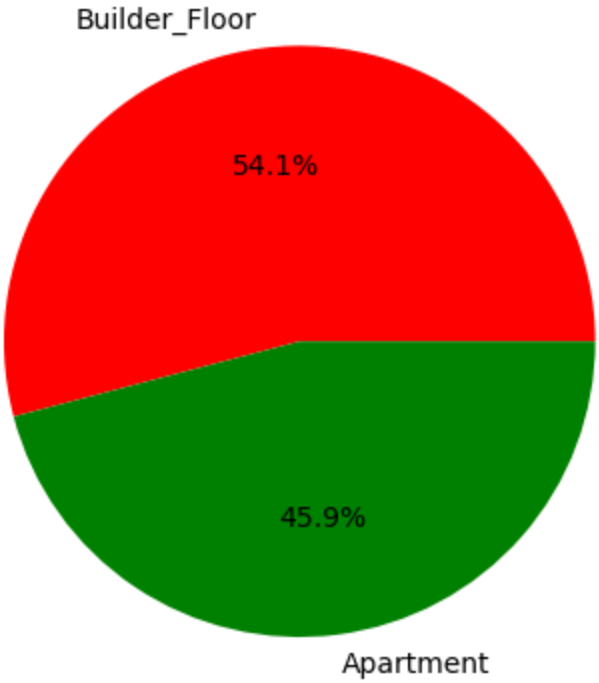


Out[17]:
```
np.float64(3.5598955904673226)
```

In [18]:
```
df.Parking = df.Parking.fillna(1)
df.Parking = df.Parking.astype(int)
df.Parking.sample(5)
```

Out[18]:
```
1085    1
876     1
230     1
219     1
1054    1
Name: Parking, dtype: int64
```

In [19]:
```
#Type
df.isnull().sum()
colors = ["r","g"]
temp=df.Type.value_counts().reset_index()
plt.pie(temp['count'],labels=temp.Type,autopct='%1.1f%%',colors=colors)
plt.show()
temp
```



Out[19]:

	Type	count
0	Builder_Floor	654
1	Apartment	555

In [20]:
```
df.Type.fillna('Builder_Floor',inplace=True)
df.Type.mode().values
```

Out[20]:
```
array(['Builder_Floor'], dtype=object)
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
Area           0
BHK            0
Bathroom       2
Furnishing     0
Location       0
District       0
Locality       0
Parking        0
Status         0
Transaction    0
Type           0
Per_Sqft       0
Price          0
dtype: int64
```

In [22]:
```
df.dropna(inplace=True)
```

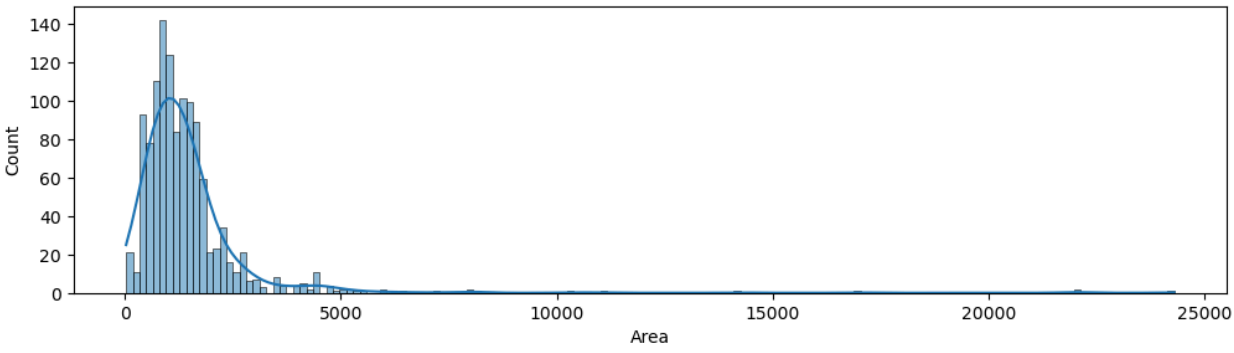
In [23]:
```
df.shape
```

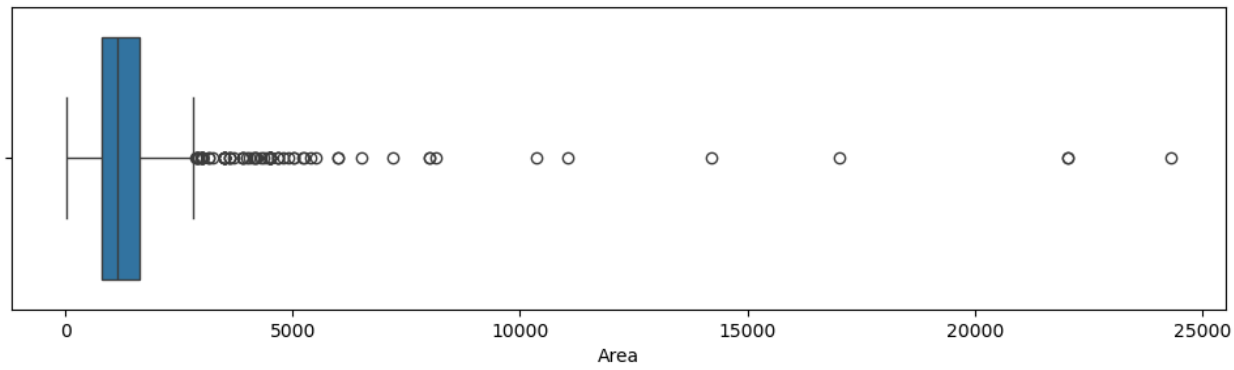
Out[23]:
```
(1212, 13)
```

In [24]:
```
# numerical columns
# categorical columns
df.columns
num=list(df.describe().columns)
cat=[]
for i in df.columns:
    if i not in num:
        cat.append(i)

print('numerical\t',num)
print('categorical\t',cat)
```

```
numerical	 ['Area', 'BHK', 'Bathroom', 'Parking', 'Per_Sqft', 'Price']
categorical	 ['Furnishing', 'Location', 'District', 'Locality', 'Statu
s', 'Transaction', 'Type']
```

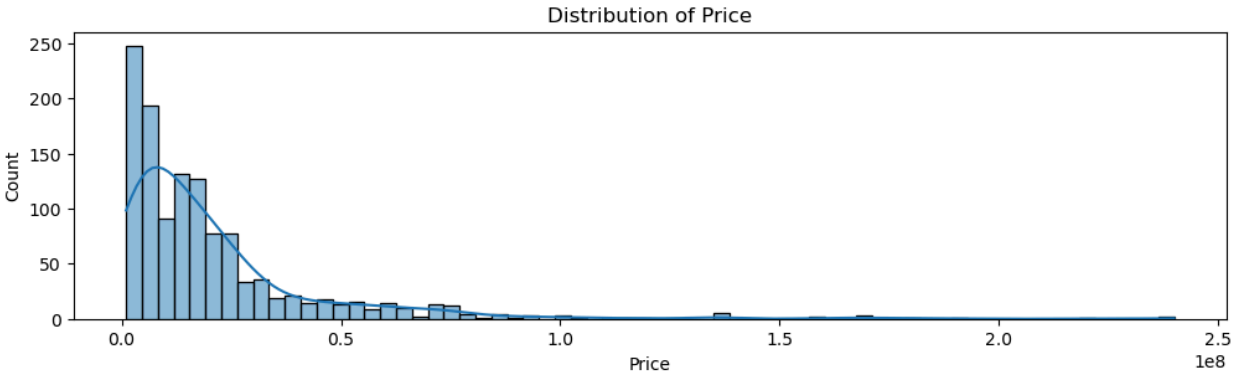
In [25]:
```
# Univariate Analysis
# Numerical columns
# Distribution
plt.figure(figsize=(12,3))
sns.histplot(df.Area,kde=True)
plt.show()

plt.figure(figsize=(12,3))
sns.boxplot(x=df.Area)
plt.show()
df.Area.skew()
```

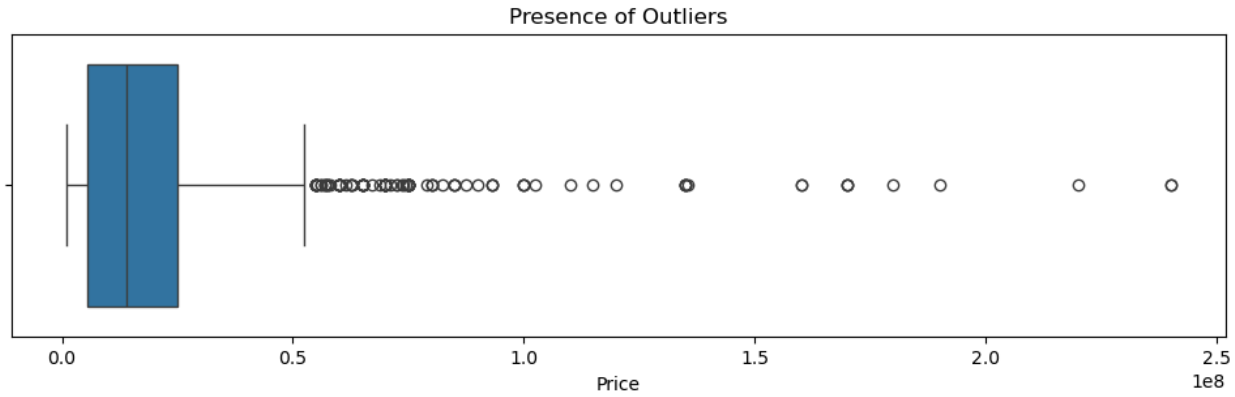




Out[25]: np.float64(8.099780784747038)

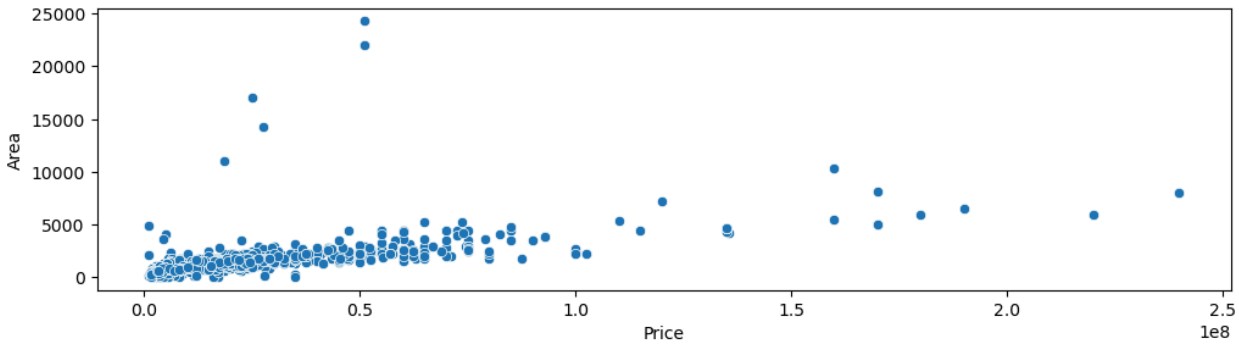
In [26]:
```
# Price
plt.figure(figsize=(12,3))
plt.title('Distribution of Price')
sns.histplot(df.Price,kde=True)
plt.show()

plt.figure(figsize=(12,3))
plt.title('Presence of Outliers')
sns.boxplot(x=df.Price)
plt.show()
df.Area.skew()
```





Out[26]: np.float64(8.099780784747038)

In [27]:
```
#Bivariate analysis
plt.figure(figsize=(12,3))
sns.scatterplot(data=df,x='Price',y='Area')
plt.show()
```



Potential Issue:

Areas above 10,000 sq. ft. are shown as very affordable, which is unrealistic for a Delhi-based location.

In [28]:
```
df[df.Area>10000]
```

Out[28]:

	Area	BHK	Bathroom	Furnishing	Location	District	Locality	Parl
429	22050.0	4	4.0	Semi-Furnished	Greater Kailash	South Delhi	Greater Kailash 1	
431	22050.0	4	4.0	Semi-Furnished	Greater Kailash	South Delhi	Greater Kailash 1	
515	10350.0	4	7.0	Semi-Furnished	Friends Colony	South Delhi	Maharani Bagh, New Friends Colony	
603	24300.0	4	5.0	Semi-Furnished	Saket	South Delhi	Saket	
806	14220.0	3	3.0	Semi-Furnished	Paschim Vihar	West Delhi	Paschim Vihar Block B4	
835	17010.0	3	3.0	Semi-Furnished	Punjabi Bagh	West Delhi	Punjabi Bagh West	
978	11050.0	3	3.0	Unfurnished	Chittaranjan Park	South Delhi	Chittaranjan Park	

In [29]:
```
22050.0*30556.0
```

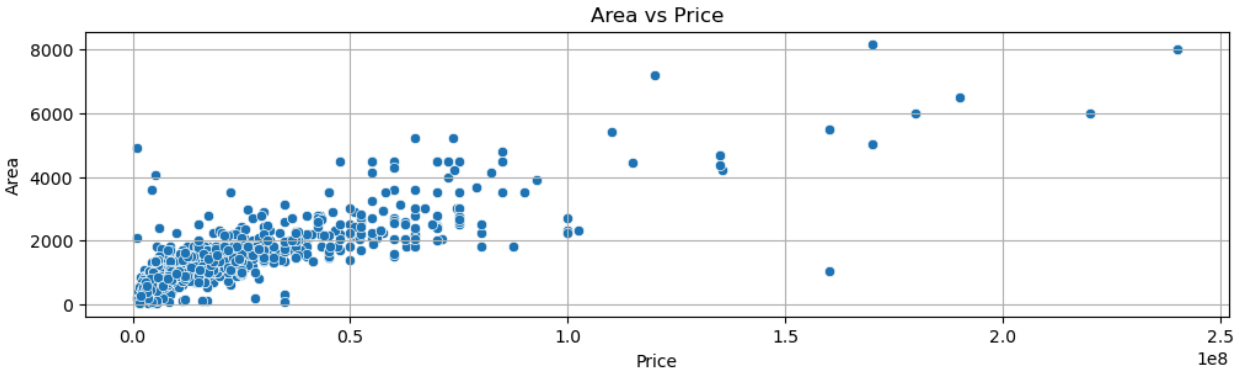
Out[29]:
```
673759800.0
```

In [30]:
```
(673759800.0/51000000)/10
```

Out[30]:
```
1.3210976470588236
```

In [31]:
```
df.Area = np.where(df.Area>10000,df.Area/10,df.Area)
```

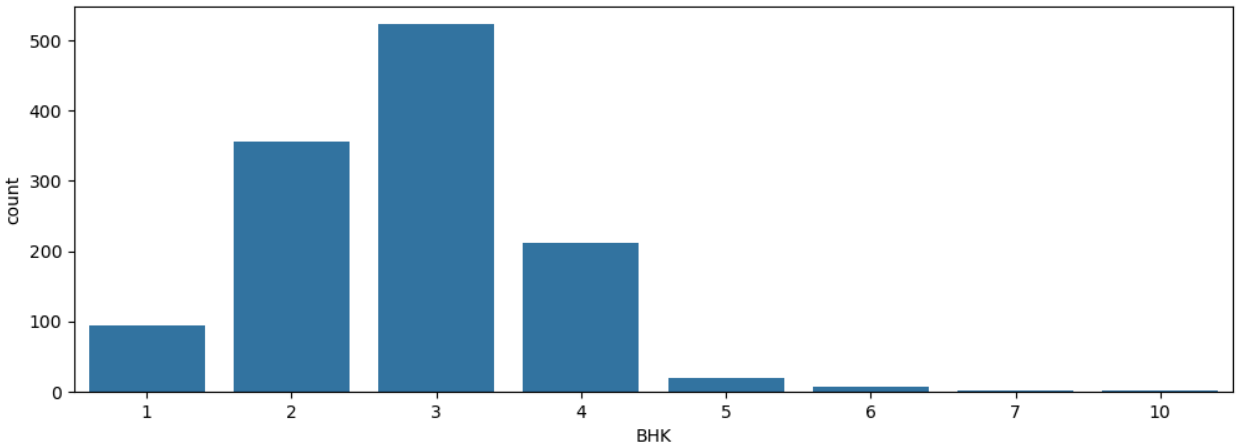
In [32]:
```
#Bivariate analysis
plt.figure(figsize=(12,3))
plt.title('Area vs Price')
sns.scatterplot(data=df,x='Price',y='Area')
plt.grid()
plt.show()
```



In [33]:
```
#plotly
px.scatter(df,x='Price',y='Area',hover_data=['Area','Location','Price'],height
```

Exploratory Analysis

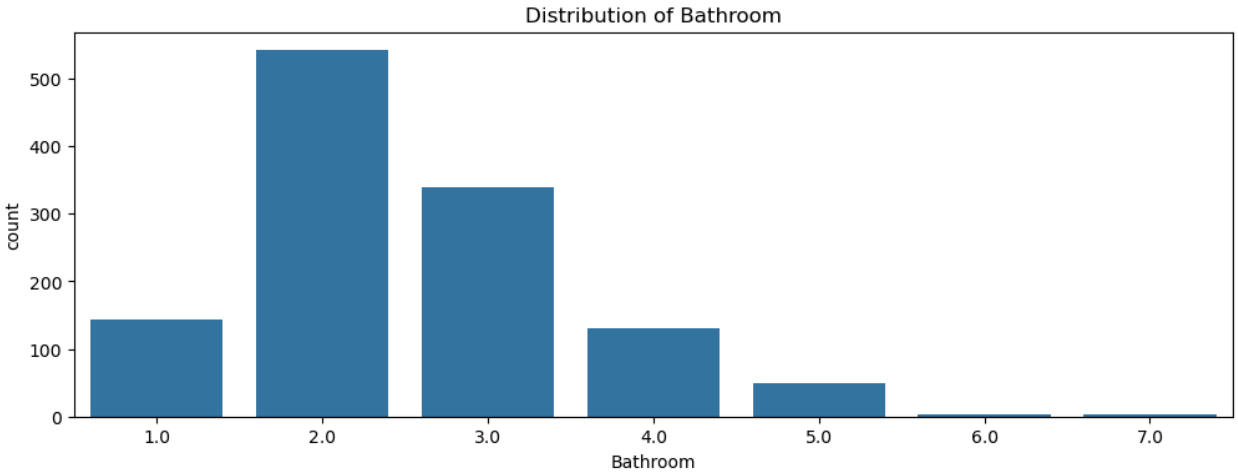
In [34]:
```
# Univariate analysis

num

#BHK
plt.figure(figsize=(12,4))
sns.countplot(data=df,x='BHK')
temp=df.BHK.value_counts().reset_index()
temp

px.pie(temp,names='BHK',values='count',
       color_discrete_sequence=px.colors.sequential.Blues,
       height=400,title='Distribution of BHK')
# feature egg ---> 5,6,7,10 --> 4+ BHK Entries
```



Key Insights

- 3 BHK apartments have the highest number of listings on MagicBricks, indicating strong demand and builder preference.
- 2 BHK units follow closely, driven by affordability and suitability for smaller or nuclear families.
- 4 BHK and 1 BHK configurations show moderate to low availability, reflecting niche demand.
- Very few listings exist for 5, 6, 7, and 10 BHK units, indicating low demand and a more luxury-focused market segment

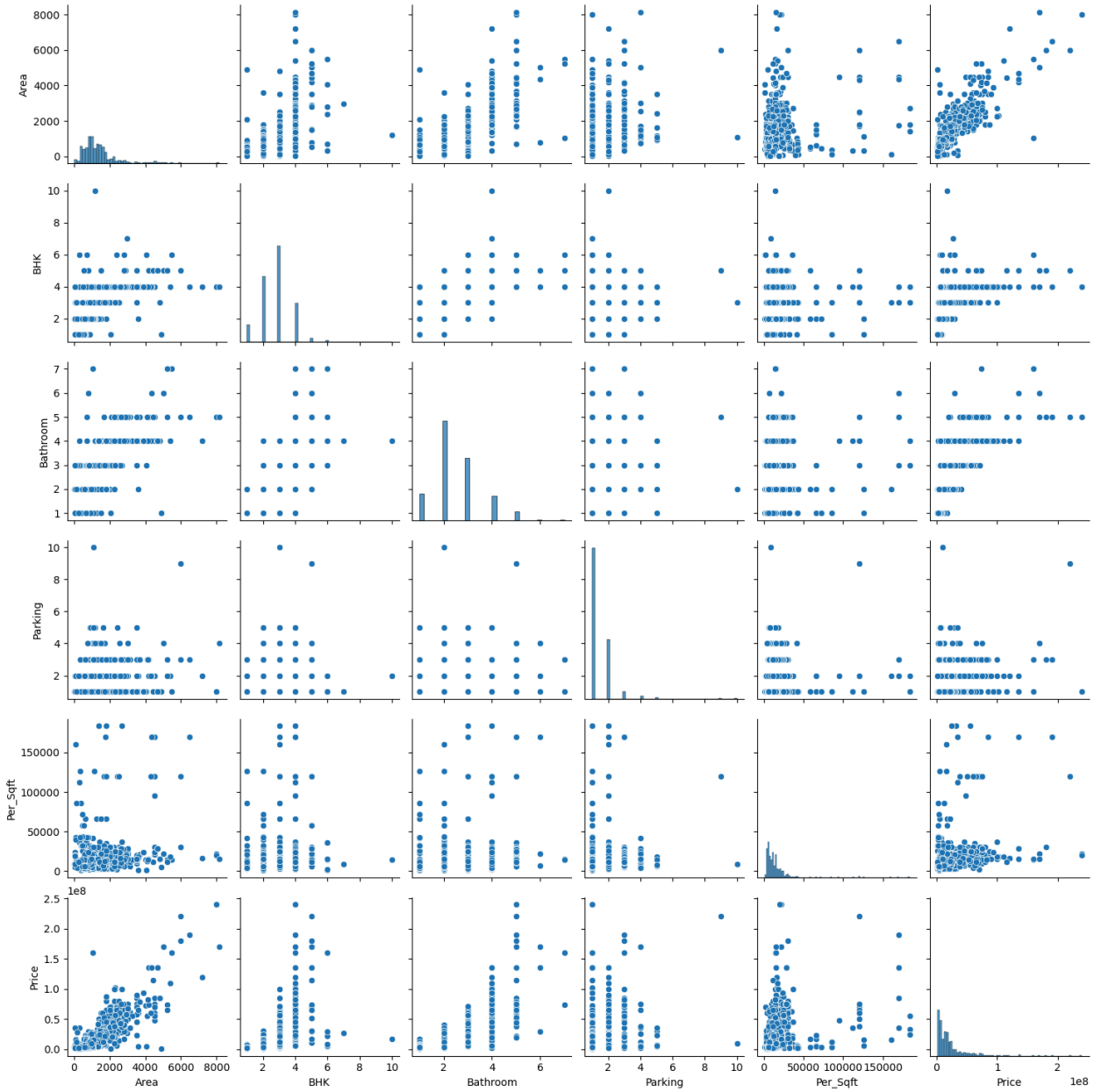
In [35]:
```
df.Bathroom.value_counts()
plt.figure(figsize=(12,4))
sns.countplot(data=df,x='Bathroom')
plt.title('Distribution of Bathroom')
plt.show()
```



Key Observations

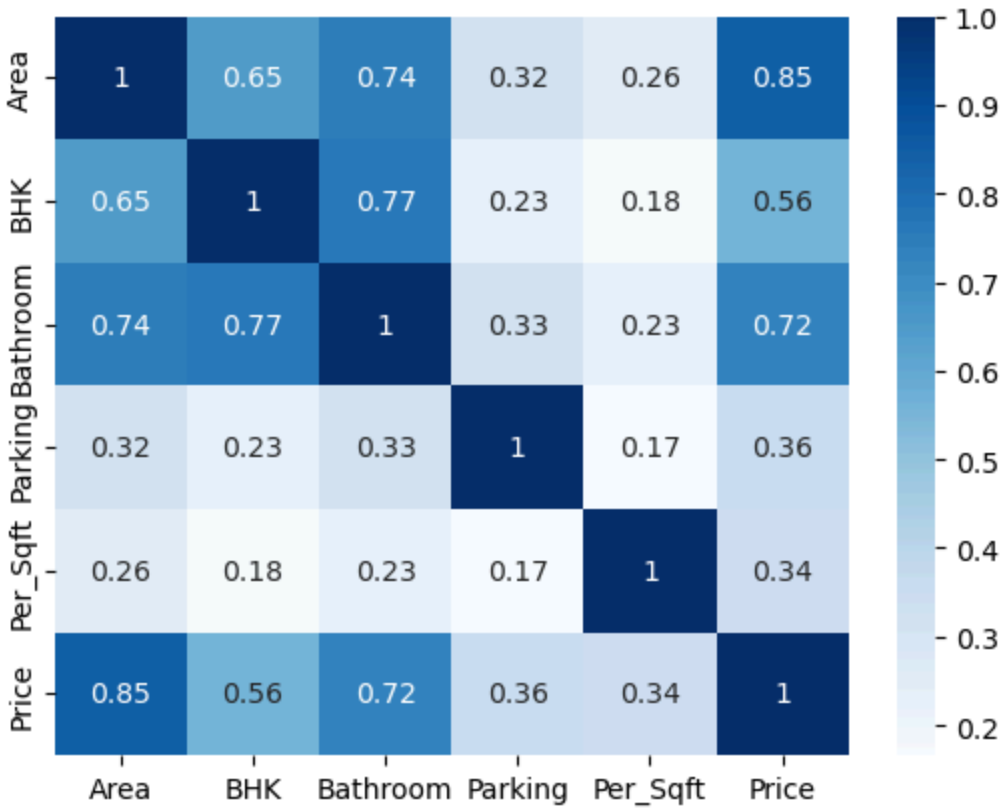
- Properties with 2 bathrooms have the highest number of listings, followed by 3-bathroom configurations, indicating common residential preferences.
- 1-bathroom and 4-bathroom units show almost similar listing counts, suggesting moderate demand.
- 5, 6, and 7 bathroom configurations are extremely rare, reflecting limited availability and niche demand.
- The overall distribution suggests that most listings are apartment-type properties rather than large independent houses.

In [36]:
```
# multivariate analysis

sns.pairplot(df)
plt.show()
```



In [38]:

```
# corr
# heat map
sns.heatmap(df.corr(numeric_only=True),cmap="Blues",annot=True)
plt.show()
```



Correlation Insights

- **Price and Area** show a very strong positive correlation (**0.84**), indicating that larger properties are priced significantly higher.
- **BHK and Bathroom** have strong correlations with both **Area** and **Price**, reflecting that bigger homes generally include more rooms and bathrooms, which increases property value.
- **Per Sqft Price** and **Parking** exhibit weak or negligible correlations with most other variables, suggesting limited influence on overall pricing and property size patterns.

In [39]:
```
df.District.value_counts().reset_index()
```

Out[39]:

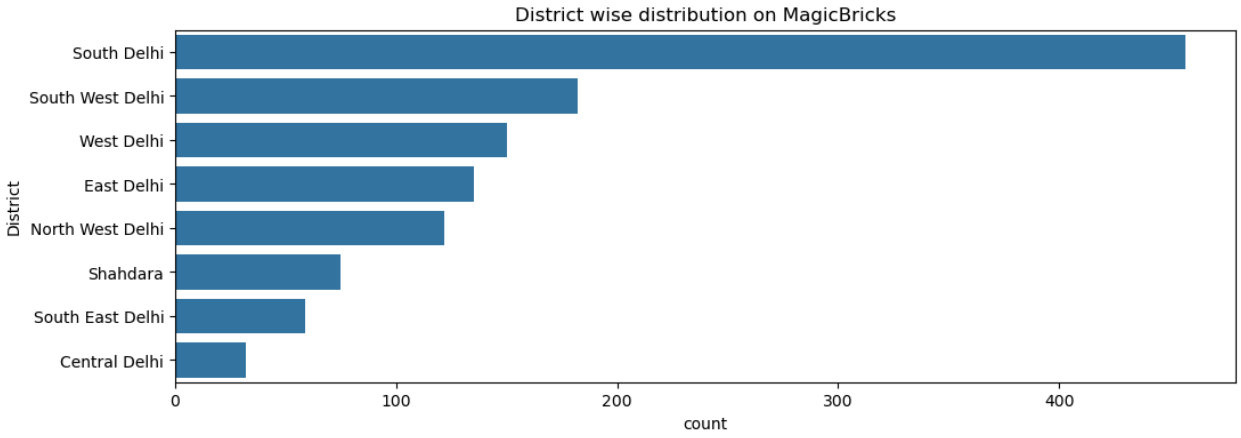
	District	count
0	South Delhi	457
1	West Delhi	150
2	East Delhi	135
3	North West Delhi	122
4	South West Delhi	95
5	South West Delhi	87
6	Shahdara	75
7	South East Delhi	59
8	Central Delhi	32

In [40]:
```
df.District.unique()
```

Out[40]:
```
array(['Central Delhi', 'East Delhi', 'North West Delhi', 'Shahdara',
       'South Delhi', 'South East Delhi', 'South West Delhi',
       'South West Delhi ', 'West Delhi'], dtype=object)
```

In [41]:
```
df.District=df.District.str.strip()
```

In [44]:
```
temp=df.District.value_counts().reset_index()
plt.figure(figsize=(12,4))
sns.countplot(data=df,y='District',order=temp.District)
plt.title('District wise distribution on MagicBricks')
plt.show()

px.pie(temp,values='count',names='District',height=400,
       color_discrete_sequence=px.colors.sequential.RdBu,hole=0.5,
       title = 'Distribution of District Listing on MagicBricks').show()
temp
```



Out[44]:

	District	count
0	South Delhi	457
1	South West Delhi	182
2	West Delhi	150
3	East Delhi	135
4	North West Delhi	122
5	Shahdara	75
6	South East Delhi	59
7	Central Delhi	32

Location-wise Property Listing Analysis

- **South Delhi** records an exceptionally high number of property listings, clearly dominating the market in terms of supply.
- **South West, East, West, and North West Delhi** show comparable listing volumes, each ranging roughly between **100–125 listings**, indicating balanced availability across these regions.
- **Central Delhi** has the lowest number of listings, reflecting limited

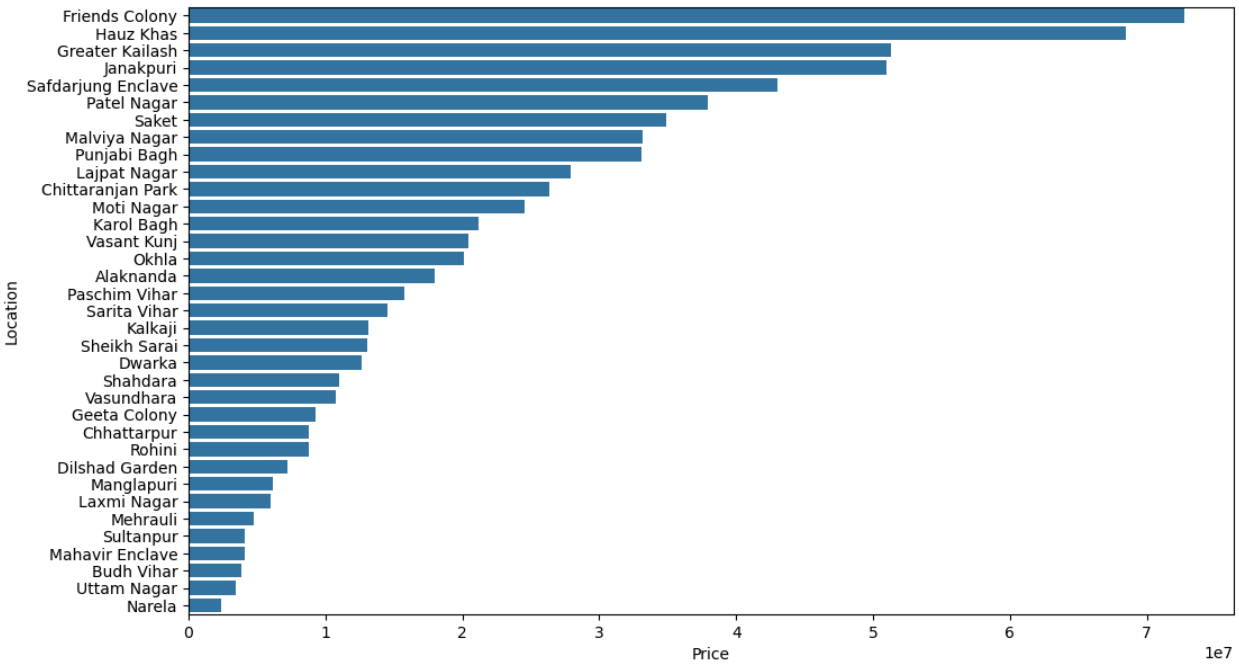
housing supply in this area.

In [46]:
```
# Location
# price
# bivariate analysis
df.loc[:,['Location','Price']].sample(5)
```

Out[46]:

	Location	Price
358	Alaknanda	14000000
1077	Alaknanda	18500000
1024	Lajpat Nagar	30000000
62	Shahdara	1900000
1157	Paschim Vihar	8600000

In [47]:
```
temp=df.groupby('Location')['Price'].mean().sort_values(ascending=False).reset
plt.figure(figsize=(12,7))
sns.barplot(data=df,y='Location',x='Price',
            ci=False,order=temp.Location)
plt.show()
```



Location-wise Property Listing Analysis

- **Friends Colony** and **Hauz Khas** have the highest number of property listings, indicating strong housing supply and high market activity in these prime locations.

- **Greater Kailash, Janakpuri, and Safdarjung Enclave** also show relatively high listing counts, reflecting consistent demand in well-established residential areas.
- **Mid-range locations** such as Lajpat Nagar, Malviya Nagar, and Karol Bagh display moderate availability, suggesting balanced market conditions.
- **Narela, Uttam Nagar, and Budh Vihar** record the lowest number of listings, indicating limited property supply or lower market participation in these areas.

In [48]:
```
temp=df.Furnishing.value_counts().reset_index()
px.pie(temp,names='Furnishing'
       ,values='count',title='Distribution of Furnishing',hole=0.4,
       color_discrete_sequence=px.colors.sequential.RdBu).show()

temp2= df.groupby('Furnishing')['Price'].median().reset_index()

px.bar(temp2,x='Price',y='Furnishing',title='Price wise comparision of  Furnis
       color_discrete_sequence=px.colors.sequential.RdBu).show()
```

Furnishing-wise Property Analysis

- **Semi-Furnished properties** dominate the market, accounting for the largest share of listings (≈55.9%), indicating strong buyer preference for partially equipped homes.
- **Unfurnished properties** form a significant portion (≈29.6%), appealing to buyers who prefer customization or lower upfront costs.
- **Fully Furnished properties** are the least common (≈14.5%), suggesting limited supply or higher pricing constraints.
- In terms of price, **Semi-Furnished homes command the highest average prices**, followed by **Furnished**, while **Unfurnished properties** are comparatively more affordable.

In [50]:
```
df.Transaction.value_counts()
```

```
colors = ['#1F3A5F','#2EC4B6'] #deep red, muted blue
sns.countplot(data=df , x='Transaction',hue='Type',palette=colors)
plt.show()
```

Transaction-wise Property Type Analysis

- **Resale properties** are dominated by **Apartments**, indicating higher availability and activity in the secondary housing market.
- **New Property transactions** show a stronger presence of **Builder Floors**, suggesting preference for newly constructed independent-style units.
- Overall, **Apartments are more common in Resale**, while **Builder Floors are more prevalent in New Property listings**, highlighting a clear market segmentation by property type.

Conclusion

This exploratory data analysis reveals clear patterns in Delhi’s real estate market. Property prices are strongly influenced by **area, BHK, and number of**

bathrooms, confirming that larger and more spacious homes command higher values. **Location plays a crucial role**, with premium areas and South Delhi dominating property listings, while central and peripheral regions show limited supply.

The market shows a strong preference for **semi-furnished properties**, both in availability and pricing, indicating buyer demand for ready-to-move yet customizable homes. Additionally, **transaction type and property type** are closely linked—apartments dominate resale transactions, whereas builder floors are more prevalent in new property listings.

Overall, these insights highlight how **size, location, furnishing, and transaction type** collectively shape pricing and availability, providing a solid foundation for further predictive modeling and decision-making in the real estate domain.

In [ ]: