In [82]:
```
print ('بسم الله الرحمن الرحيم')
print ('اختبار رقم 1 لمادة')
print ('Practical Machine Learning')
print ('تقديم الطالب / حسن محمد احمد الأمير خليفة')
print ('253002941 ,الرقم الجامعي')
print ('امداد الدكتورة / ماجر صالح ')
print ('الداتا ست التي تم استخدمها https://www.kaggle.com/datasets/monishkumarm/realestate')
```

```
بسم الله الرحمن الرحيم
اختبار رقم 1 لمادة
Practical Machine Learning
تقديم الطالب / حسن محمد احمد الأمير خليفة
253002941 الرقم الجامعي
امداد الدكتورة / ماجر صالح
الداتا ست التي تم استخدمها https://www.kaggle.com/datasets/monishkumarm/realestate
```

In [29]:
```
import pandas as pd
```

In [30]:
```
print ('لعرض أول خمسة صفوف من الداتا ست')
df = pd.read_csv('RealEstateRegression.csv')
df.head()
```

```
لعرض أول خمسة صفوف من الداتا ست
```

Out[30]:

	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	X7 Random column	Y house price of unit area
0	2012.917	32.0	84.87882	10.0	24.98298	121.54024	84.87882	37.9
1	2012.917	19.5	306.59470	9.0	24.98034	121.53951	306.59470	42.2
2	2013.583	13.3	561.98450	5.0	24.98746	121.54391	561.98450	47.3
3	2013.500	13.3	561.98450	5.0	24.98746	121.54391	NaN	54.8
4	2012.833	5.0	390.56840	5.0	24.97937	121.54245	NaN	43.1

In [31]:
```
len(df) #لعرض عدد الصفوف الموجودة في الداتا ست#
```

Out[31]:
```
418
```

In [32]:
```
#لعرض أول خمسة صفوف في الداتا ست#
df = pd.read_csv('RealEstateRegression.csv')
df.head()
```

Out[32]:

	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	X7 Random column	Y house price of unit area
0	2012.917	32.0	84.87882	10.0	24.98298	121.54024	84.87882	37.9
1	2012.917	19.5	306.59470	9.0	24.98034	121.53951	306.59470	42.2
2	2013.583	13.3	561.98450	5.0	24.98746	121.54391	561.98450	47.3
3	2013.500	13.3	561.98450	5.0	24.98746	121.54391	NaN	54.8
4	2012.833	5.0	390.56840	5.0	24.97937	121.54245	NaN	43.1

In [33]:
```
df.describe() #لعرض بعض العمليات الحسابية على الداتا ست#
```

Out[33]:

	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	X7 Random column	Y house price of unit area
count	418.000000	410.000000	418.000000	415.000000	418.000000	418.000000	161.000000	418.000000
mean	2013.147945	17.738293	1083.206582	4.108434	24.969107	121.533329	1150.847871	37.975598
std	0.282372	11.443296	1258.198966	2.951345	0.012417	0.015310	1411.023669	13.547942
min	2012.667000	0.000000	23.382840	0.000000	24.932070	121.473530	23.382840	7.600000
25%	2012.917000	8.925000	289.324800	1.000000	24.963035	121.528085	289.324800	27.700000
50%	2013.167000	16.150000	492.231300	4.000000	24.971100	121.538630	492.231300	38.450000
75%	2013.417000	28.350000	1454.279000	6.000000	24.977828	121.543267	1643.499000	46.550000
max	2013.583000	43.800000	6488.021000	10.000000	25.014590	121.566270	6488.021000	117.500000

In [34]:
```
#لعرض عدد البيانات المفقودة في كل عمود في الداتا ست#
df.isnull() .sum()
```

Out[34]:
```
X1 transaction date                       0
X2 house age                              8
X3 distance to the nearest MRT station    0
X4 number of convenience stores           3
X5 latitude                               0
X6 longitude                              0
X7 Random column                        257
Y house price of unit area                0
dtype: int64
```

In [11]:
```
missing_percentage  = df.isnull().mean() * 100
columns_with_missing_30 = missing_percentage[missing_percentage > 30 ]
print ("عدد الأعمدة  اكثر من 30% متوفر فيها قيمة مفقودة")
print (columns_with_missing_30)
```

```
عدد الأعمدة  اكثر من 30% متوفر فيها قيمة مفقودة
X7 Random column    61.483254
dtype: float64
```

In [35]:
```
missing_percentage = df.isnull().mean() * 100
threshold = 30
columns_to_drop = missing_percentage[missing_percentage > threshold].index
# لحذف بيانات المفقودة في العمود
df_cleaned = df.drop(columns=columns_to_drop)
```

In [36]:
```
#لعرض البيانات المحذوفة من الداتا ست من جميع الأعمدة#
print(f"العمود الذي تم حذفة هو : {list(columns_to_drop)}")
print(f"الأعمدة المتبقية من الداتا ست هي: {df_cleaned.columns}")
```

```
العمود الذي تم حذفة هو : ['X7 Random column']
الأعمدة المتبقية من الداتا ست هي: Index(['X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

In [37]:
```
import numpy as np
import pandas as pd
```

In [38]:
```
data=df.copy()

def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
outliers_only
```

Out[38]:

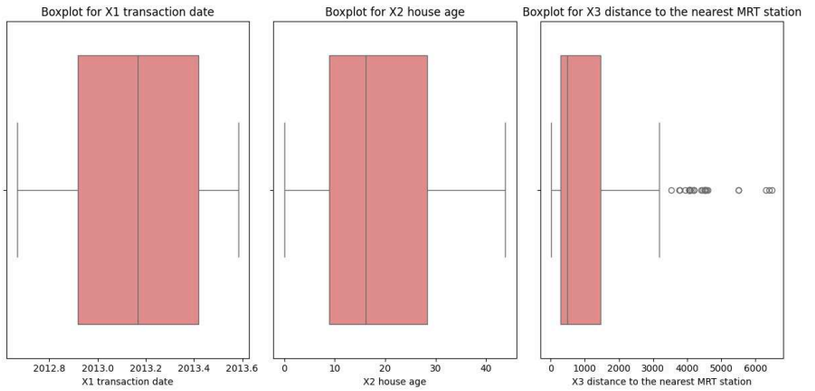
	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	X7 Random column	Y house price of unit area	X1 transaction date_Outlier	X2 house age_Outlier	X3 distance to the nearest MRT station_Outlier	X4 number of convenience stores_Outlier	X5 latitude_Outlier	X6 longitude_Outlier	X7 Random column_Outlier	Y house price of unit area_Outlier
8	2013.500	31.7	5512.0380	1.0	24.95095	121.48458	5512.038	18.8	False	False	True	False	False	True	True	False
30	2013.500	25.9	4519.6900	0.0	24.94826	121.49587	NaN	22.1	False	False	True	False	False	True	False	False
35	2013.500	NaN	4079.4180	0.0	25.01459	121.51816	NaN	27.3	False	False	True	False	True	False	False	False
40	2013.000	13.6	4082.0150	0.0	24.94155	121.50381	NaN	15.9	False	False	True	False	False	True	False	False
41	2013.500	16.8	4066.5870	0.0	24.94297	121.50342	4066.587	18.2	False	False	True	False	False	True	True	False
48	2013.417	24.2	4605.7490	0.0	24.94684	121.49578	4605.749	13.4	False	False	True	False	False	True	True	False
49	2012.667	29.4	4510.3590	1.0	24.94925	121.49542	4510.359	13.2	False	False	True	False	False	True	True	False
58	2013.500	30.3	4510.3590	1.0	24.94925	121.49542	NaN	22.6	False	False	True	False	False	True	False	False
73	2013.167	13.8	4082.0150	0.0	24.94155	121.50381	4082.015	20.0	False	False	True	False	False	True	True	False
87	2013.583	16.9	4066.5870	0.0	24.94297	121.50342	NaN	18.3	False	False	True	False	False	True	False	False
89	2013.500	23.0	3947.9450	0.0	24.94783	121.50243	3947.945	25.3	False	False	True	False	False	True	True	False
116	2013.000	30.9	6396.2830	1.0	24.94375	121.47883	6396.283	12.2	False	False	True	False	False	True	True	False
117	2013.000	13.6	4197.3490	0.0	24.93885	121.50383	NaN	13.0	False	False	True	False	True	True	False	False
148	2013.500	16.4	3780.5900	0.0	24.93293	121.51203	NaN	45.1	False	False	True	False	True	False	False	False
154	2013.500	16.9	4066.5870	0.0	24.94297	121.50342	NaN	20.7	False	False	True	False	False	True	False	False
155	2013.167	13.8	4082.0150	0.0	24.94155	121.50381	NaN	15.6	False	False	True	False	False	True	False	False
162	2012.750	16.0	4066.5870	0.0	24.94297	121.50342	4066.587	11.6	False	False	True	False	False	True	True	False
170	2013.333	24.0	4527.6870	0.0	24.94741	121.49628	NaN	14.4	False	False	True	False	False	True	False	False
176	2012.833	13.9	4573.7790	0.0	24.94867	121.49507	4573.779	19.2	False	False	True	False	False	True	True	False
180	2012.667	26.9	4449.2700	0.0	24.94898	121.49621	NaN	15.5	False	False	True	False	False	True	False	False
183	2013.500	17.0	4082.0150	0.0	24.94155	121.50381	NaN	18.8	False	False	True	False	False	True	False	False
187	2013.000	8.9	3078.1760	0.0	24.95464	121.56627	3078.176	22.0	False	False	False	False	False	True	False	False
189	2012.917	16.3	4066.5870	0.0	24.94297	121.50342	4066.587	20.5	False	False	True	False	False	True	True	False
194	2013.500	15.2	3771.8950	0.0	24.93363	121.51158	NaN	29.3	False	False	True	False	True	False	False	False
220	2013.333	37.2	186.5101	9.0	24.97703	121.54265	NaN	78.3	False	False	False	False	False	False	False	True
226	2013.000	16.5	4082.0150	0.0	24.94155	121.50381	NaN	12.8	False	False	True	False	False	True	False	False
228	2013.417	11.9	3171.3290	0.0	25.00115	121.51776	NaN	46.6	False	False	False	False	True	False	False	False
231	2012.833	16.2	4074.7360	0.0	24.94235	121.50357	4074.736	14.7	False	False	True	False	False	True	True	False
232	2012.917	27.1	4412.7650	1.0	24.95032	121.49587	NaN	17.4	False	False	True	False	False	True	False	False
249	2012.833	18.0	6306.1530	1.0	24.95743	121.47516	6306.153	15.0	False	False	True	False	False	True	True	False
255	2013.417	31.5	5512.0380	1.0	24.95095	121.48458	5512.038	17.4	False	False	True	False	False	True	True	False
270	2013.333	10.8	252.5822	1.0	24.97460	121.53046	NaN	117.5	False	False	False	False	False	False	False	True
298	2013.333	16.7	4082.0150	0.0	24.94155	121.50381	NaN	16.7	False	False	True	False	False	True	False	False
307	2012.833	10.3	3079.8900	0.0	24.95460	121.56627	NaN	24.7	False	False	False	False	False	True	False	False

312	2013.583	35.4	318.5292	9.0	24.97071	121.54069	NaN	78.0	False	False	False	False	False	False	False	True
320	2012.750	13.5	4197.3490	0.0	24.93885	121.50383	NaN	18.6	False	False	True	False	True	True	False	False
329	2013.000	13.6	4197.3490	0.0	24.93885	121.50383	NaN	19.2	False	False	True	False	True	True	False	False
331	2013.333	25.6	4519.6900	0.0	24.94826	121.49587	4519.690	15.6	False	False	True	False	False	True	True	False
347	2013.583	17.4	6488.0210	1.0	24.95719	121.47353	6488.021	11.2	False	False	True	False	False	True	True	False
382	2013.000	16.3	3529.5640	0.0	24.93207	121.51597	3529.564	29.3	False	False	True	False	True	False	False	False
384	2012.750	16.1	4066.5870	0.0	24.94297	121.50342	NaN	12.9	False	False	True	False	False	True	False	False
394	2013.500	32.6	4136.2710	1.0	24.95544	121.49630	NaN	24.7	False	False	True	False	False	True	False	False
409	2013.000	13.7	4082.0150	0.0	24.94155	121.50381	NaN	15.4	False	False	True	False	False	True	False	False

In [39]:
```
print ('استخدام التصور / Using Visualization:')
```

```
استخدام التصور / Using Visualization:
```

In [40]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))

for i, col in enumerate(df.columns[:3]):
    plt.subplot(1, 3, i + 1)
    sns.boxplot(x=df[col], color='lightcoral')
    plt.title(f"Boxplot for {col}")

plt.tight_layout()
plt.show()
```



In [41]:
```
print ('طرق الكشف من القيم المتطرفة / Methods to Detect Outliers')
```

```
طرق الكشف من القيم المتطرفة / Methods to Detect Outliers
```

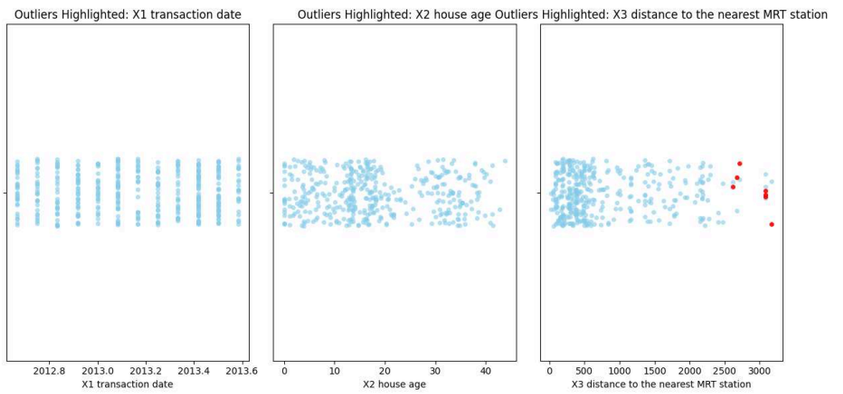
In [42]:
```
import numpy as np
data = df.copy()

def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)

# Replace outliers with NaN for each feature
for col in data.columns:
    data[col] = replace_outliers_with_nan(data[col])

# Check how many NaN values were introduced
null_counts = data.isnull().sum()

print("مجموع القيم الصفرية في كل عمود:")
print(null_counts)
```

```
مجموع القيم الصفرية في كل عمود:
X1 transaction date                       0
X2 house age                              8
X3 distance to the nearest MRT station   37
X4 number of convenience stores           3
X5 latitude                               8
X6 longitude                             35
X7 Random column                        272
Y house price of unit area                3
dtype: int64
```

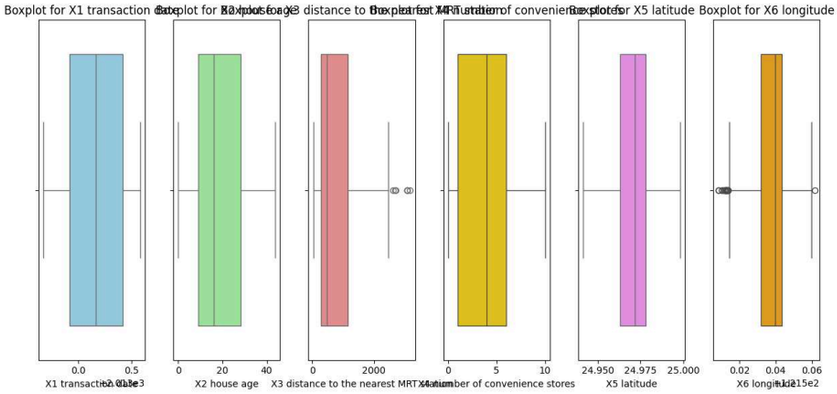
In [43]:
```
print('طرق اخرى ايجاد القيمة الصفرية في كل عمود')
```

```
طرق اخرى ايجاد القيمة الصفرية في كل عمود
```

In [44]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

plt.figure(figsize=(12, 6))

for i, col in enumerate(data.columns[:3]):  # adjust number of columns as needed
    plt.subplot(1, 3, i + 1)

    # Detect outliers using IQR
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    # Plot all points
    sns.stripplot(x=data[col], color='skyblue', alpha=0.6)

    # Highlight outliers in red
    outliers = data[(data[col] < lower) | (data[col] > upper)]
    sns.stripplot(x=outliers[col], color='red', alpha=0.9)

    plt.title(f"Outliers Highlighted: {col}")

plt.tight_layout()
plt.show()
```



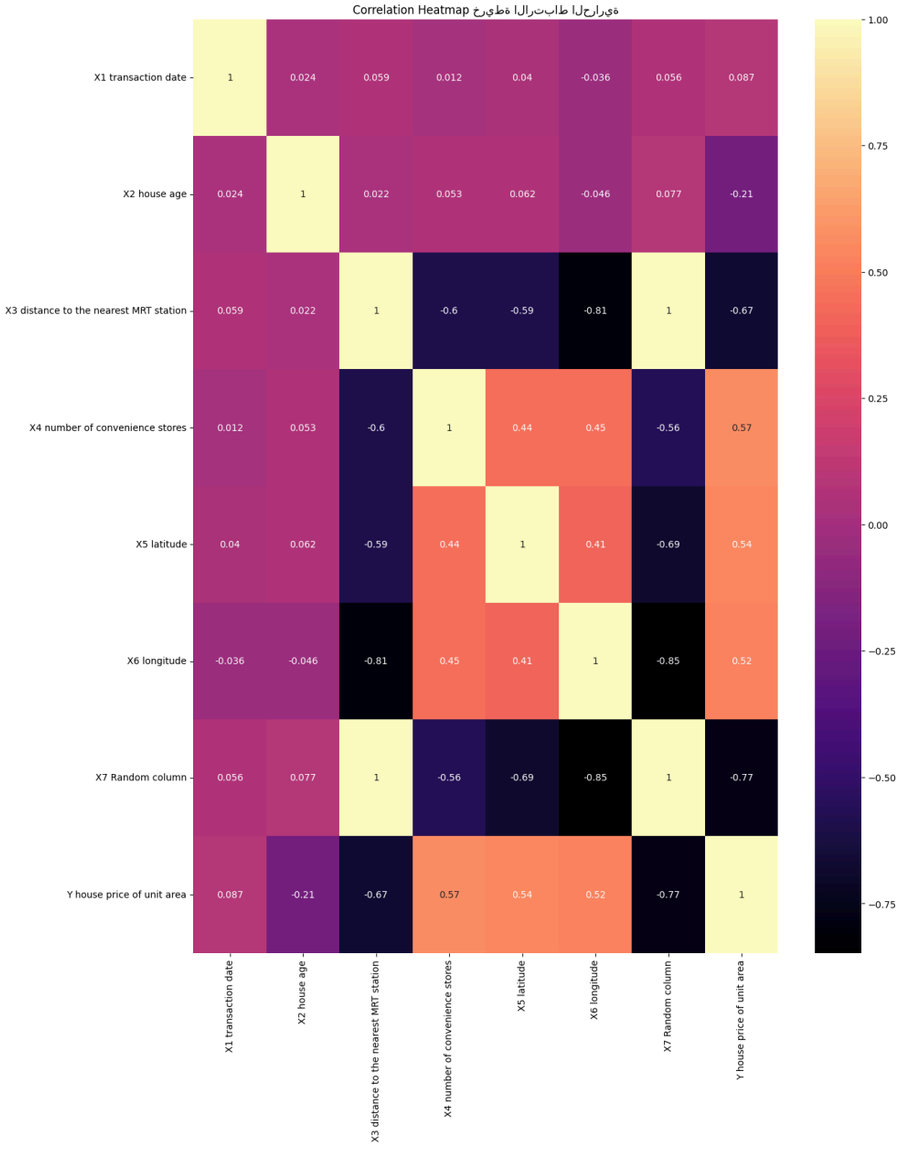
In [27]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))

# Custom color palette
colors = ['skyblue', 'lightgreen', 'lightcoral', 'gold', 'violet', 'orange']

for i, col in enumerate(data.columns[:len(colors)]):
    plt.subplot(1, len(data.columns[:len(colors)]), i + 1)
    sns.boxplot(x=data[col], color=colors[i])
    plt.title(f"Boxplot for {col}")

plt.tight_layout()
plt.show()
```



In [26]:
```
print('لعرض خريطة الارتباط الحرارية / Correlation heatmap')
```

لعرض خريطة الارتباط الحرارية / Correlation heatmap

In [45]:
```
plt.figure(figsize=(14, 18))
sns.heatmap(df.corr(), annot=True, cmap='magma')
plt.title('Correlation Heatmap خريطة الارتباط الحرارية')
plt.show()
```



In [46]:
```
print ('لعرض اسماء الاعمدة في الداتا سيت')
```

لعرض اسماء الاعمدة في الداتا سيت

In [29]:
```
df.columns
```

Out[29]:
```
Index(['X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'X7 Random column', 'Y house price of unit area'],
      dtype='object')
```

In [55]:
```
print ('لعرض نوع البيانات في كل عمود')
```

لعرض نوع البيانات في كل عمود

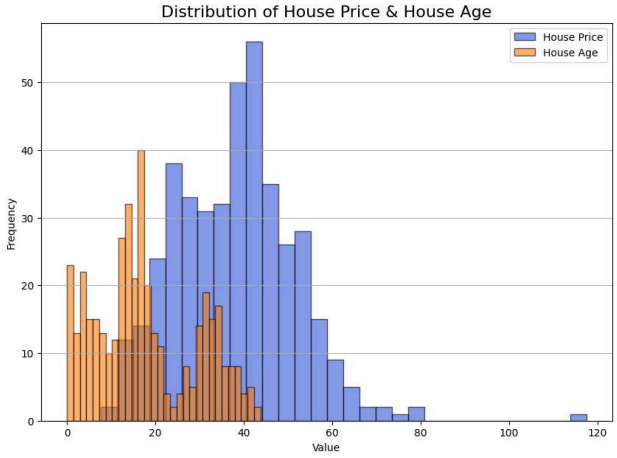
In [47]:
```
df.dtypes
```

Out[47]:
```
X1 transaction date                       float64
X2 house age                              float64
X3 distance to the nearest MRT station    float64
X4 number of convenience stores           float64
X5 latitude                               float64
X6 longitude                              float64
X7 Random column                          float64
Y house price of unit area                float64
dtype: object
```

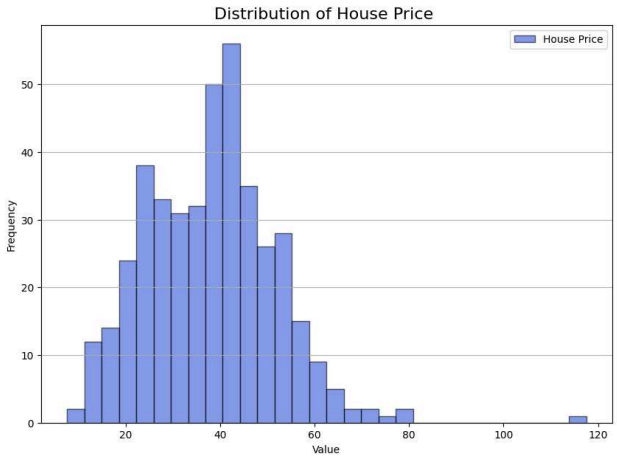
In [48]:
```
print ('جدول او رسم بياني يوضح اسعار المنازل وعمر المنازل ')
```

جدول او رسم بياني يوضح اسعار المنازل وعمر المنازل

In [49]:
```
plt.figure(figsize=(10, 7))

plt.hist(df['Y house price of unit area'], bins=30, alpha=0.6, label='House Price', color='#3458db', edgecolor='black')
plt.hist(df['X2 house age'], bins=30, alpha=0.6, label='House Age', color='#ff7f0e', edgecolor='black')

plt.title('Distribution of House Price & House Age', fontsize=16)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='-', alpha=0.9)
plt.legend()
plt.show()
```



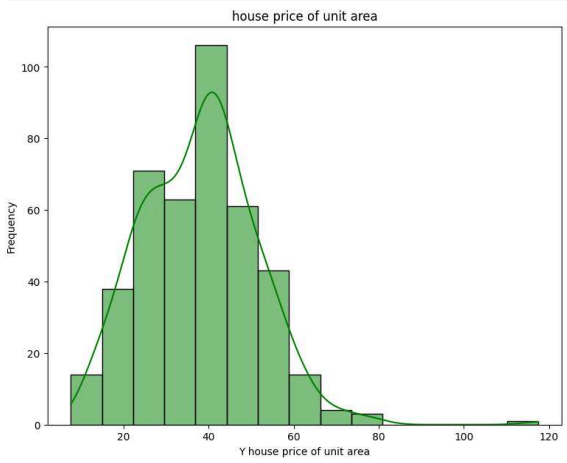
In [50]:
```
plt.figure(figsize=(10, 7))

plt.hist(df['Y house price of unit area'], bins=30, alpha=0.6, label='House Price', color='#3458db', edgecolor='black')
plt.title('Distribution of House Price ', fontsize=16)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='-', alpha=0.9)
plt.legend()
plt.show()
```



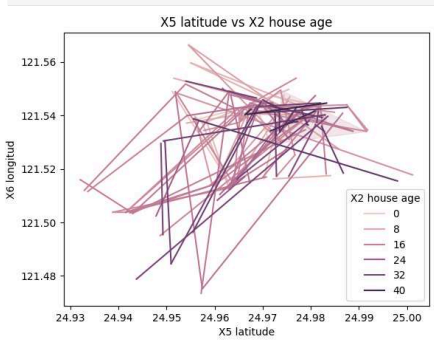
In [67]:
```
df.dtypes
```

Out[67]:
```
X1 transaction date                     float64
X2 house age                            float64
X3 distance to the nearest MRT station  float64
X4 number of convenience stores         float64
X5 latitude                             float64
X6 longitude                            float64
X7 Random column                        float64
Y house price of unit area              float64
dtype: object
```

In [51]:
```
cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols: print(f'{col} has {df[col].unique()} values')
plt.figure(figsize=(9, 7))
sns.histplot(df['Y house price of unit area'], bins=15, kde=True, color='green')
plt.title('house price of unit area')
plt.xlabel('Y house price of unit area')
plt.ylabel('Frequency')
plt.show()
```



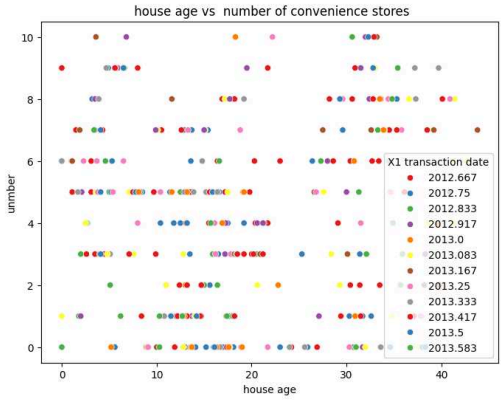
In [75]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.lineplot(x='X5 latitude', y='X6 longitude', hue='X2 house age', data=df)
plt.title("X5 latitude vs X2 house age")
plt.xlabel("X5 latitude")
plt.ylabel("X6 longitud")
plt.show()
```



In [64]:
```
print(df.columns)
```

```
Index(['X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'X7 Random column', 'Y house price of unit area'],
      dtype='object')
```

In [80]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='X2 house age', y='X4 number of convenience stores', data=df, hue='X1 transaction date', palette='Set1')
plt.title(' house age vs  number of convenience stores')
plt.xlabel('house age')
plt.ylabel('unmber')
plt.show()
```



In [83]:
```
# تحويل العمود إلى سنة وشهر
df['year'] = df['X1 transaction date'].astype(int)
df['month'] = ((df['X1 transaction date'] % 1) * 12 + 1).astype(int)

# إنشاء عمود تاريخ
df['date'] = pd.to_datetime(df[['year', 'month']].assign(day=1))

df[['X1 transaction date', 'date']].head()
```

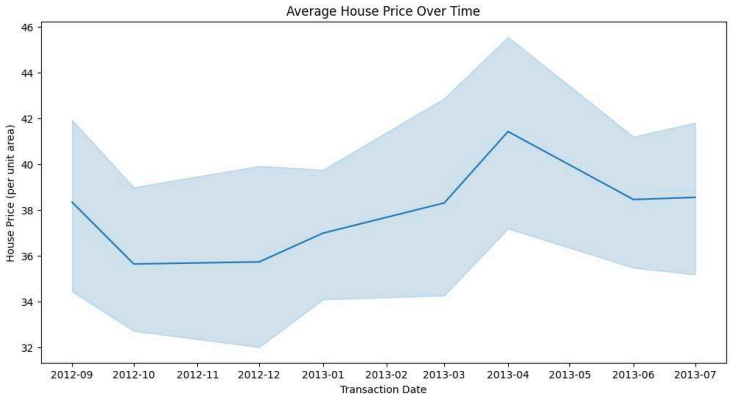
Out[83]:

	X1 transaction date	date
0	2012.917	2012-12-01
1	2012.917	2012-12-01
2	2013.583	2013-07-01
3	2013.500	2013-07-01
4	2012.833	2012-10-01

In [85]:
```
print(' تحليل العلاقة مع سعر الوحدة (Y house price of unit area)')
```

```
تحليل العلاقة مع سعر الوحدة (Y house price of unit area)
```

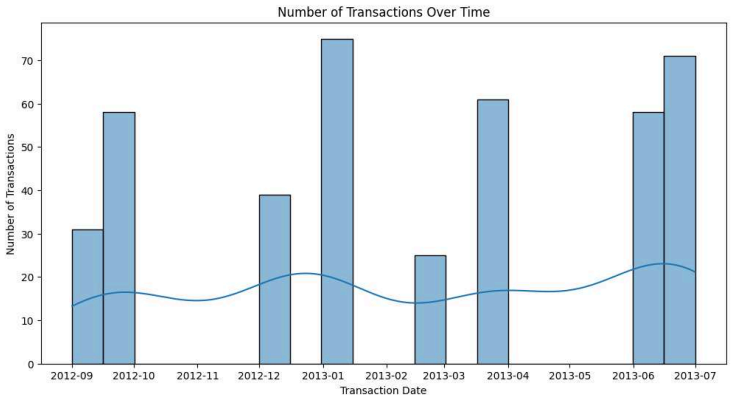
In [84]:
```
plt.figure(figsize=(12,6))
sns.lineplot(x='date', y='Y house price of unit area', data=df)
plt.title('Average House Price Over Time')
plt.xlabel('Transaction Date')
plt.ylabel('House Price (per unit area)')
plt.show()
```



In [87]:
```
print (' توزيع المعاملات حسب التاريخ')
```

```
توزيع المعاملات حسب التاريخ
```

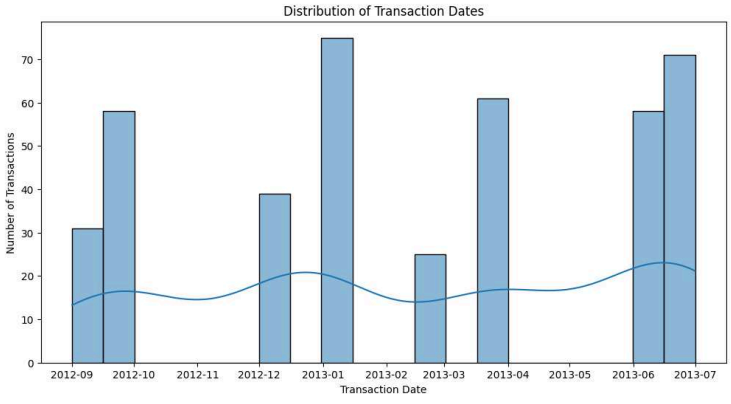
In [86]:
```
plt.figure(figsize=(12,6))
sns.histplot(df['date'], bins=20, kde=True)
plt.title('Number of Transactions Over Time')
plt.xlabel('Transaction Date')
plt.ylabel('Number of Transactions')
plt.show()
```



In [94]:
```
print ('توزيع عدد المعاملات حسب التاريخ')
```

```
توزيع عدد المعاملات حسب التاريخ
```

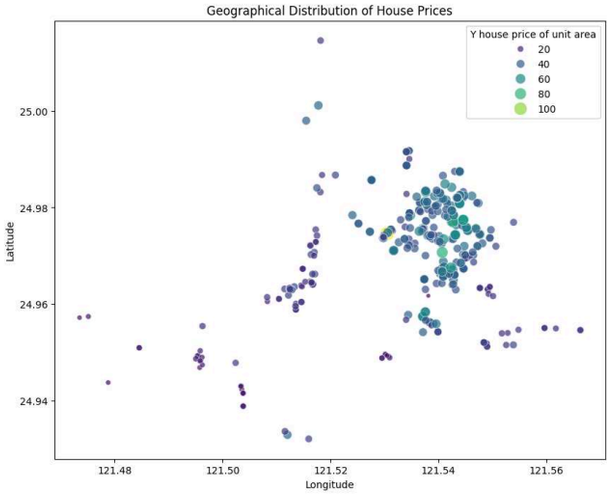
In [93]:
```
plt.figure(figsize=(12,6))
sns.histplot(df['date'], bins=20, kde=True)
plt.title('Distribution of Transaction Dates')
plt.xlabel('Transaction Date')
plt.ylabel('Number of Transactions')
plt.show()
```



In [98]:
```
print (' تحليل المواقع الجغرافية (X5 latitude, X6 longitude)')
```

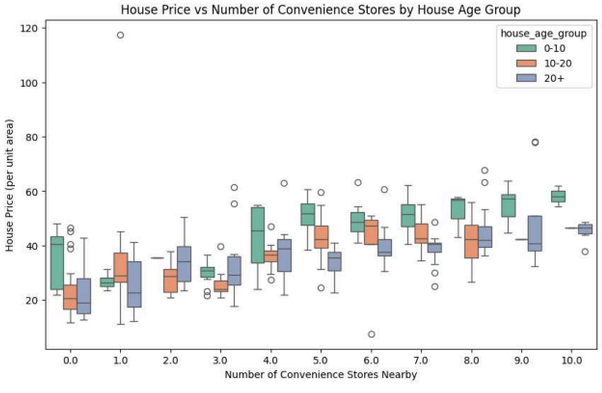
```
تحليل المواقع الجغرافية (X5 latitude, X6 longitude)
```

In [97]:
```
plt.figure(figsize=(10,8))
sns.scatterplot(x='X6 longitude', y='X5 latitude', hue='Y house price of unit area', data=df, palette='viridis', size='Y house price of unit area', sizes=(20,200), alpha=0.7)
plt.title('Geographical Distribution of House Prices')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



In [101…

```
df['house_age_group'] = pd.cut(df['X2 house age'], bins=[0,10,20,50], labels=['0-10','10-20','20+'])

plt.figure(figsize=(10,6))
sns.boxplot(x='X4 number of convenience stores', y='Y house price of unit area',
            hue='house_age_group', data=df, palette='Set2')
plt.title('House Price vs Number of Convenience Stores by House Age Group')
plt.xlabel('Number of Convenience Stores Nearby')
plt.ylabel('House Price (per unit area)')
plt.show()
```



In [106…

```
import pandas as pd

summary = pd.DataFrame({
    "Column": df.columns,
    "Data Type": df.dtypes,
    "Non-Null Count": df.notnull().sum(),
    "Unique Values": df.nunique(),
    "Min": df.min(numeric_only=True),
    "Max": df.max(numeric_only=True),
    "Mean": df.mean(numeric_only=True),
    "Std": df.std(numeric_only=True)
}).reset_index(drop=True)

summary
```

Out[106…

	Column	Data Type	Non-Null Count	Unique Values	Min	Max	Mean	Std
0	X1 transaction date	float64	418	12	2012.66700	2013.58300	2013.147945	0.282372
1	X2 house age	float64	410	234	0.00000	43.80000	17.738293	11.443296
2	X3 distance to the nearest MRT station	float64	418	259	23.38284	6488.02100	1083.206582	1258.198966
3	X4 number of convenience stores	float64	415	11	0.00000	10.00000	4.108434	2.951345
4	X5 latitude	float64	418	234	24.93207	25.01459	24.969107	0.012417
5	X6 longitude	float64	418	232	121.47353	121.56627	121.533329	0.015310
6	X7 Random column	float64	161	122	23.38284	6488.02100	1150.847871	1411.023669
7	Y house price of unit area	float64	418	270	7.60000	117.50000	37.975598	13.547942
8	year	datetime64[ns]	418	8	NaN	NaN	NaN	NaN
9	month	category	393	3	NaN	NaN	NaN	NaN
10	date	int64	418	8	1.00000	12.00000	6.138756	3.468928
11	house_age_group	int64	418	2	2012.00000	2013.00000	2012.693780	0.461475

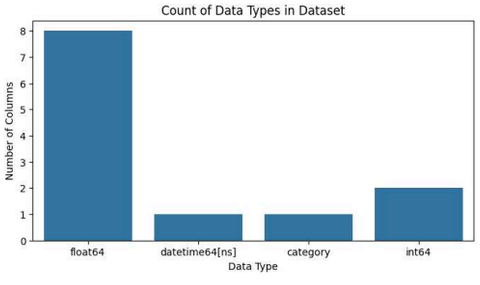
In [109…

```
print ('رسم هذه القيم الفريدة لكل عمود')
```

```
رسم هذه القيم الفريدة لكل عمود
```

In [108…

```
plt.figure(figsize=(8,4))
sns.countplot(x="Data Type", data=summary)
plt.title("Count of Data Types in Dataset")
plt.xlabel("Data Type")
plt.ylabel("Number of Columns")
plt.show()
```



In [110…

```
print ('والحمد لله رب العالمين')
```

```
والحمد لله رب العالمين
```

In [ ]: