In [80]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.linear_model import LinearRegression
```

In [47]:

```
dataset = pd.read_csv('datasethousepred.csv')
```

In [48]:

```
dataset
```

Out[48]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0
...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0

4600 rows × 18 columns

```
In [49]: dataset.head(15)
```

Out[49]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0
5	2014-05-02 00:00:00	490000.0	2.0	1.00	880	6380	1.0	0	0
6	2014-05-02 00:00:00	335000.0	2.0	2.00	1350	2560	1.0	0	0
7	2014-05-02 00:00:00	482000.0	4.0	2.50	2710	35868	2.0	0	0
8	2014-05-02 00:00:00	452500.0	3.0	2.50	2430	88426	1.0	0	0
9	2014-05-02 00:00:00	640000.0	4.0	2.00	1520	6200	1.5	0	0
10	2014-05-02 00:00:00	463000.0	3.0	1.75	1710	7320	1.0	0	0
11	2014-05-02 00:00:00	1400000.0	4.0	2.50	2920	4000	1.5	0	0
12	2014-05-02 00:00:00	588500.0	3.0	1.75	2330	14892	1.0	0	0
13	2014-05-02 00:00:00	365000.0	3.0	1.00	1090	6435	1.0	0	0
14	2014-05-02 00:00:00	1200000.0	5.0	2.75	2910	9480	1.5	0	0

In [50]:

```
dataset.shape
```

Out[50]:

```
(4600, 18)
```

In [51]:

```
fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

```
Float variables: 4
```

In [52]:

```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))
```

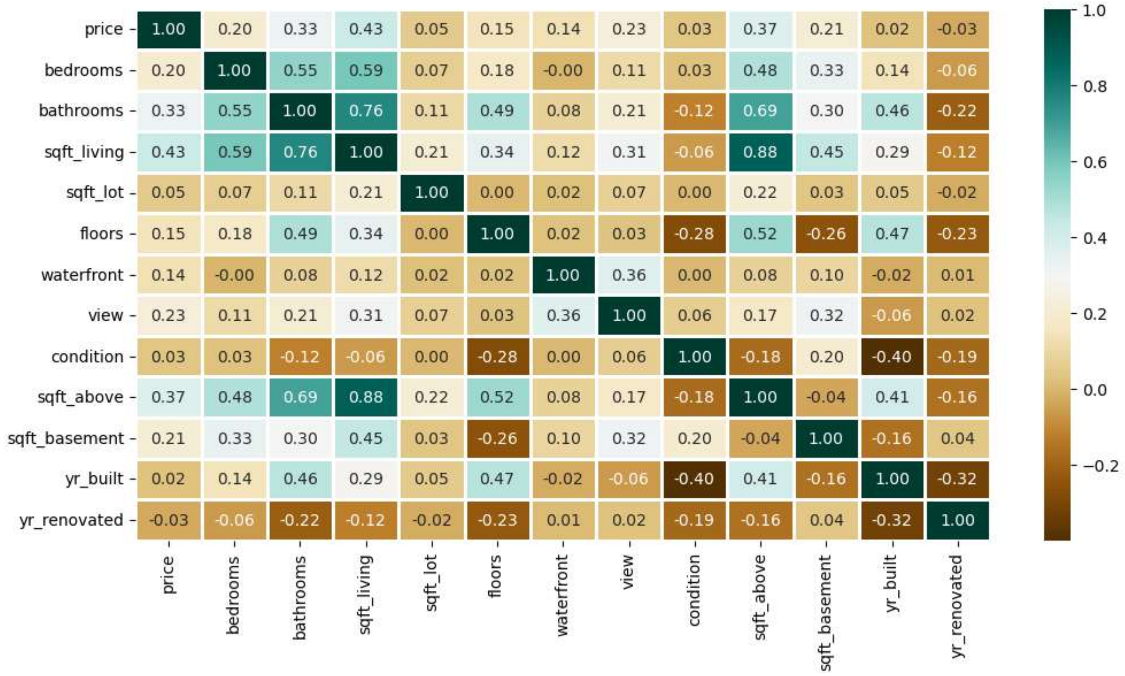
```
Categorical variables: 5
Integer variables: 0
```

In [53]:

```
plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)
```

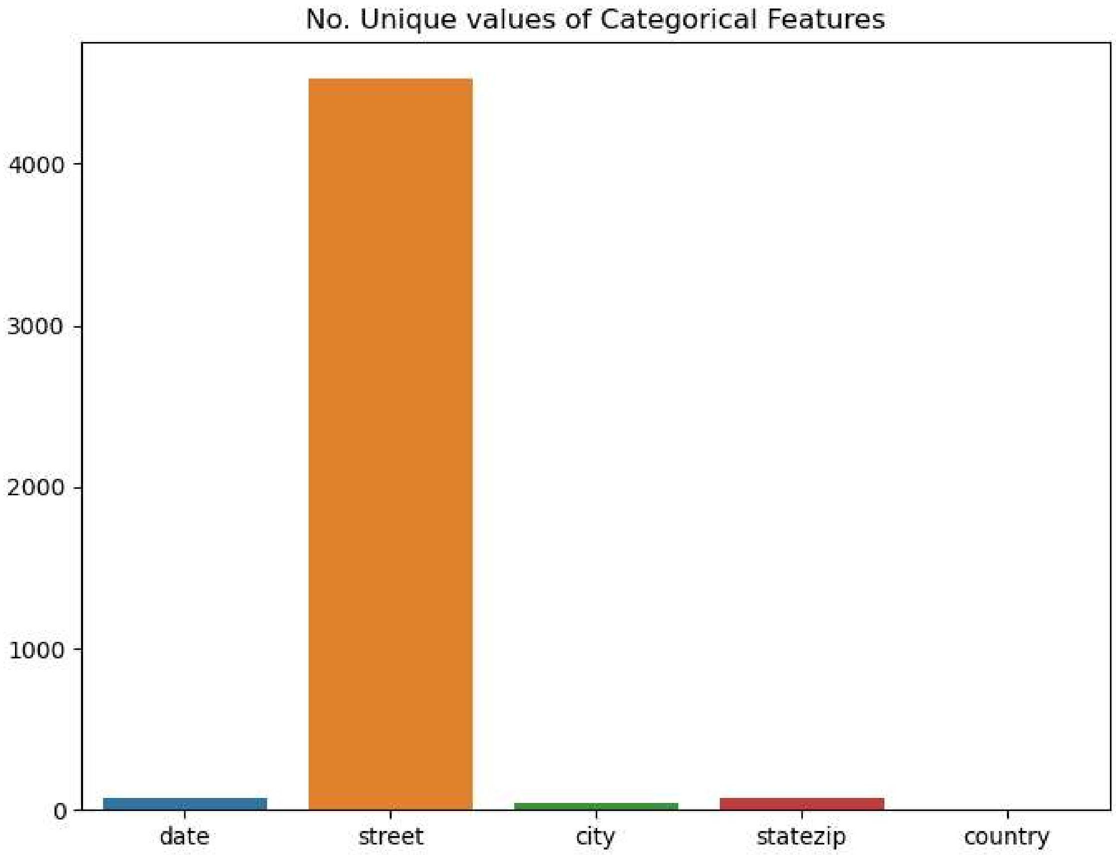
```
C:\Users\Muskan\AppData\Local\Temp\ipykernel_84\3487798585.py:2: FutureWa
rning: The default value of numeric_only in DataFrame.corr is deprecated.
In a future version, it will default to False. Select only valid columns
or specify the value of numeric_only to silence this warning.
  sns.heatmap(dataset.corr(),
```

Out[53]: `<Axes: >`



In [54]:
```
unique_values = []
for col in object_cols:
  unique_values.append(dataset[col].unique().size)
plt.figure(figsize=(8,6))
plt.title('No. Unique values of Categorical Features')
plt.xticks(rotation=0)
sns.barplot(x=object_cols,y=unique_values)
```

Out[54]:
```
<Axes: title={'center': 'No. Unique values of Categorical Features'}>
```



In [55]:

```
dataset.drop(['date'],
             axis=1,
             inplace=True)
dataset['price'] = dataset['price'].fillna(
  dataset['price'].mean())
new_dataset = dataset.dropna()
new_dataset.isnull().sum()
```

Out[55]:

```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [56]:

```
from sklearn.preprocessing import OneHotEncoder

s = (new_dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
print('No. of. categorical features: ',
      len(object_cols))
```

```
Categorical variables:
['street', 'city', 'statezip', 'country']
No. of. categorical features:  4
```

In [57]:

```
OH_encoder = OneHotEncoder(sparse=False)
OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))
OH_cols.index = new_dataset.index
OH_cols.columns = OH_encoder.get_feature_names_out()
df_final = new_dataset.drop(object_cols, axis=1)
df_final = pd.concat([df_final, OH_cols], axis=1)
```

```
C:\anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:972: Fut
ureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and wi
ll be removed in 1.4. `sparse_output` is ignored unless you leave `sparse
` to its default value.
  warnings.warn(
```

In [58]:
```
X = df_final.drop(['price'], axis=1)
Y = df_final['price']
```

In [59]:
```
X
```

Out[59]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_al
0	3.0	1.50	1340	7912	1.5	0	0	3	
1	5.0	2.50	3650	9050	2.0	0	4	5	
2	3.0	2.00	1930	11947	1.0	0	0	4	
3	3.0	2.25	2000	8030	1.0	0	0	4	
4	4.0	2.50	1940	10500	1.0	0	0	4	
...	...	...	...	...	...	...	...	...	
4595	3.0	1.75	1510	6360	1.0	0	0	4	
4596	3.0	2.50	1460	7573	2.0	0	0	3	
4597	3.0	2.50	3010	7014	2.0	0	0	3	
4598	4.0	2.00	2090	6630	1.0	0	0	3	
4599	3.0	2.50	1490	8102	2.0	0	0	4	

4600 rows × 4659 columns

In [60]:
```
Y
```

Out[60]:
```
0       3.130000e+05
1       2.384000e+06
2       3.420000e+05
3       4.200000e+05
4       5.500000e+05
            ...
4595    3.081667e+05
4596    5.343333e+05
4597    4.169042e+05
4598    2.034000e+05
4599    2.206000e+05
Name: price, Length: 4600, dtype: float64
```

In [61]:
```
Features1 = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors']
target = 'price'
X1 = dataset[Features1]
y1 = dataset[target]
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.5,
```

In [62]:

```
X1
```

Out[62]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	3.0	1.50	1340	7912	1.5
1	5.0	2.50	3650	9050	2.0
2	3.0	2.00	1930	11947	1.0
3	3.0	2.25	2000	8030	1.0
4	4.0	2.50	1940	10500	1.0
...	...	...	...	...	...
4595	3.0	1.75	1510	6360	1.0
4596	3.0	2.50	1460	7573	2.0
4597	3.0	2.50	3010	7014	2.0
4598	4.0	2.00	2090	6630	1.0
4599	3.0	2.50	1490	8102	2.0

4600 rows × 5 columns

In [63]:

```
y1
```

Out[63]:

```
0       3.130000e+05
1       2.384000e+06
2       3.420000e+05
3       4.200000e+05
4       5.500000e+05
            ...
4595    3.081667e+05
4596    5.343333e+05
4597    4.169042e+05
4598    2.034000e+05
4599    2.206000e+05
Name: price, Length: 4600, dtype: float64
```

In [64]:

```
model = LinearRegression()
model.fit(X_train, y_train)
LinearRegression()
y_pred = model.predict(X_test)
y_pred
```

Out[64]:

```
array([363715.81566707, 399446.44586546, 834230.23979675, ...,
       748123.84167884, 569569.52453231, 658083.41823346])
```

In [65]:

```
score = model.score(X_test, y_test)
print("Model R^2 Score:", score)
```

```
Model R^2 Score: 0.10933671026237857
```

In [66]:

```
new_house = pd.DataFrame({'bedrooms': [2], 'bathrooms': [2.5], 'sqft_living
predicted_price = model.predict(new_house)
print("Predicted Price:", predicted_price[0])
```

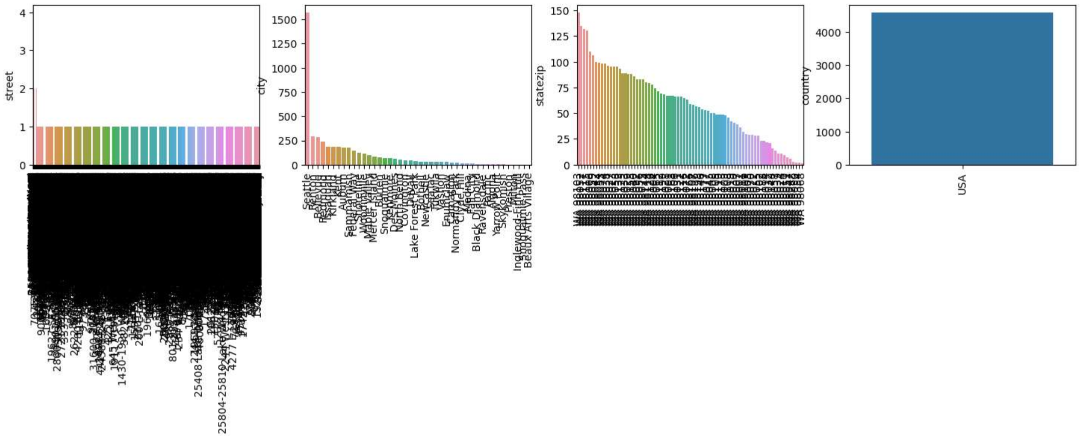
```
Predicted Price: 146977.00302334767
```

In [79]:

```
plt.figure(figsize=(18, 36))
plt.title('Categorical Features: Distribution')
plt.xticks(rotation=90)
index = 1

for col in object_cols:
    y = dataset[col].value_counts()
    plt.subplot(11, 4, index)
    plt.xticks(rotation=90)
    sns.barplot(x=list(y.index), y=y)
    index += 1
```

```
C:\Users\Muskan\AppData\Local\Temp\ipykernel_84\2166598984.py:8: Matpl
otlibDeprecationWarning: Auto-removal of overlapping axes is deprecate
d since 3.6 and will be removed two minor releases later; explicitly c
all ax.remove() as needed.
  plt.subplot(11, 4, index)
```



In [ ]:

In [ ]:

In [ ]: