Visualizing the Data: NYC Rolling Sales

In [57]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
data = pd.read_csv('C:/Users/munazzam/Downloads/nyc-rolling-sales.csv')
```

In [10]:
```
data = data.drop('Unnamed: 0', 1)
data = data.replace(' -  ', np.nan) # empty data points are not set up p
roperly
data = data.replace(' ', np.nan)
data
```

Out[10]:

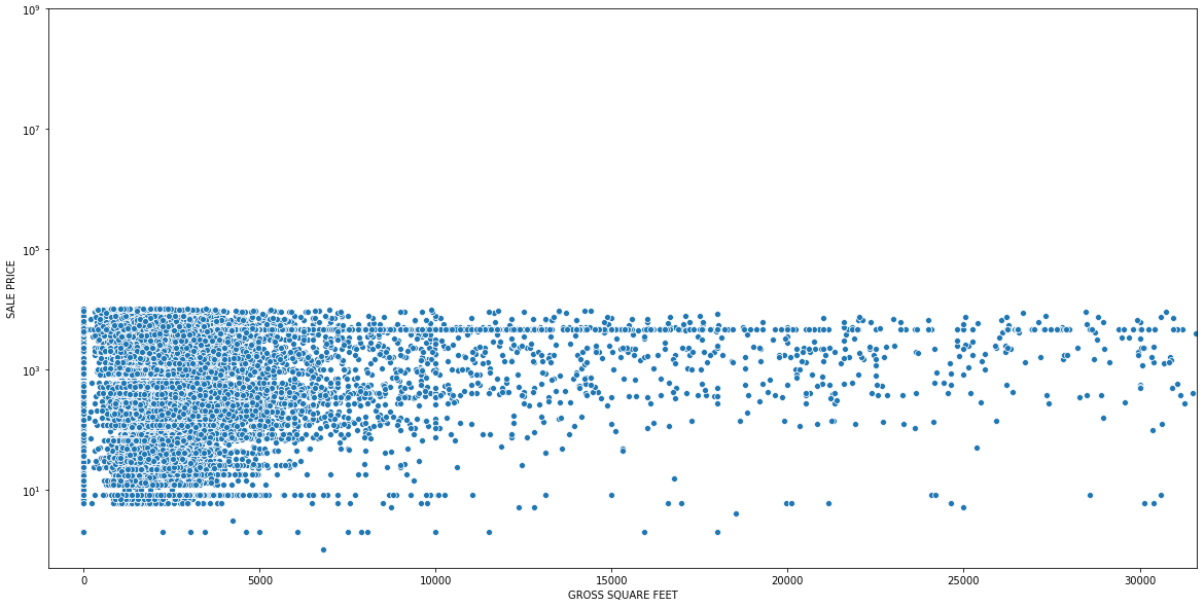
	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT
0	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	392	6	NaN	C2
1	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	26	NaN	C7
2	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	39	NaN	C7
3	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	402	21	NaN	C4
4	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	404	55	NaN	C2
...	...	...	...	...	...	...	...	...
84543	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	34	NaN	B9
84544	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	78	NaN	B9
84545	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7351	60	NaN	B2
84546	5	WOODROW	22 STORE BUILDINGS	4	7100	28	NaN	K6
84547	5	WOODROW	35 INDOOR PUBLIC AND CULTURAL FACILITIES	4	7105	679	NaN	P9

84548 rows × 21 columns

In [11]:
```
data.drop_duplicates(inplace=True)
```

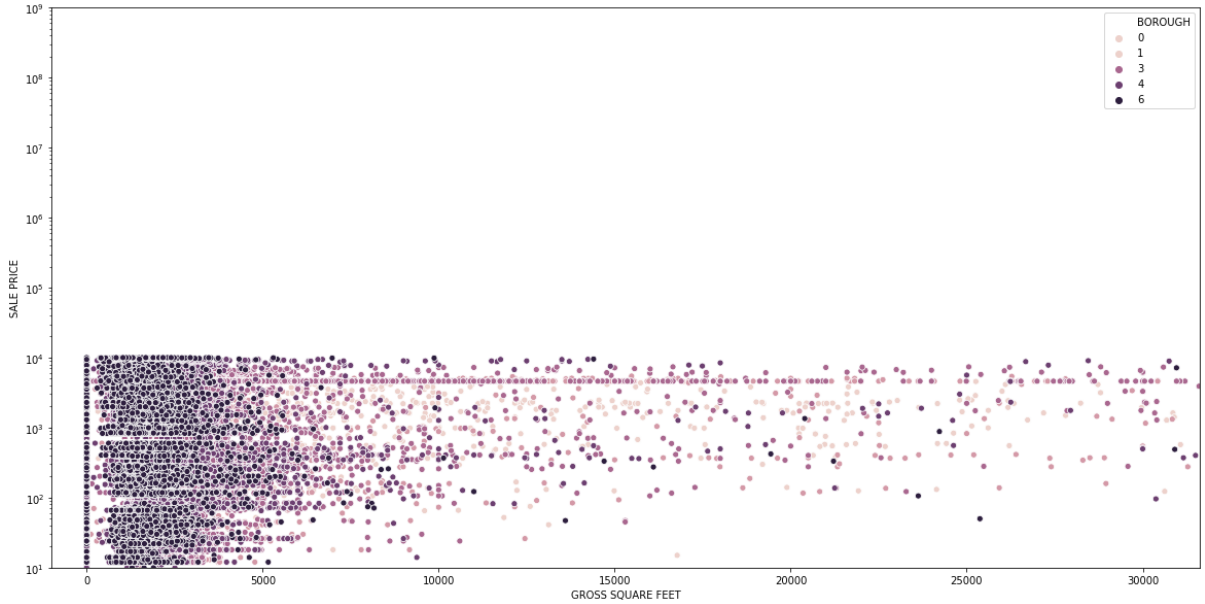
In [17]:
```
#Price against Gross Square Feet

data['GROSS SQUARE FEET'] = data['GROSS SQUARE FEET'].fillna(0).astype(
'int')
plt.rcParams['figure.figsize'] = (20, 10)
fig, ax = plt.subplots()
sns.scatterplot(x='GROSS SQUARE FEET', y='SALE PRICE', data=data)
ax.set_xlim([-10**3, 10**4.5])
ax.set_ylim([.5, 10**9])
ax.set_yscale("log")
plt.show()
```

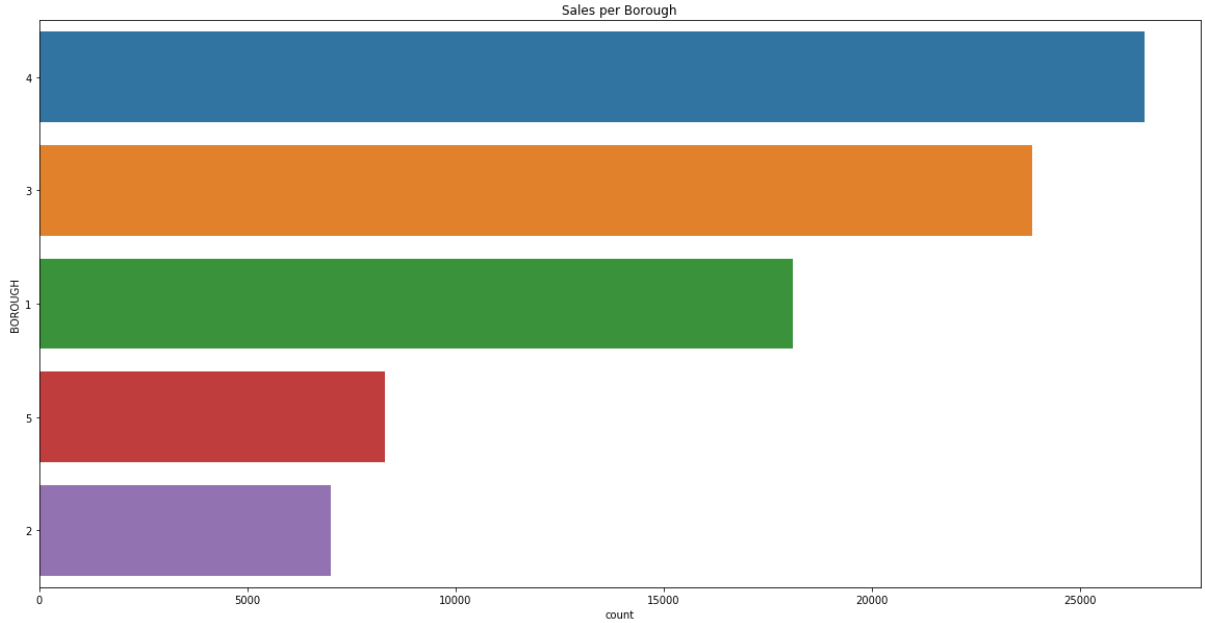


In [21]:
```
#Since borough is a category we can use different color for each

fig, ax = plt.subplots()
sns.scatterplot(x='GROSS SQUARE FEET', y='SALE PRICE', data=data, hue='B
OROUGH')
ax.set_xlim([-10**3, 10**4.5])
ax.set_ylim([10**1, 10**9])
ax.set_yscale("log")
plt.show()
```

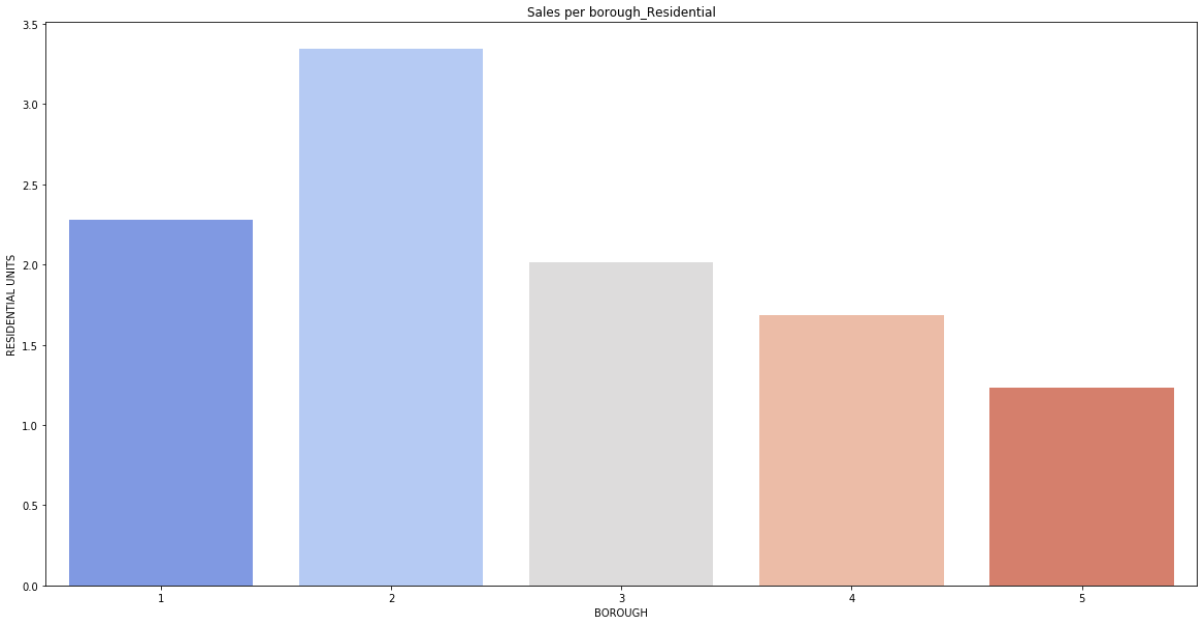


In [41]:
```
#Sales per Borough
sns.countplot(y = 'BOROUGH',
              data = data,
              order = data['BOROUGH'].value_counts().index)
plt.title('Sales per Borough')
plt.show()
```



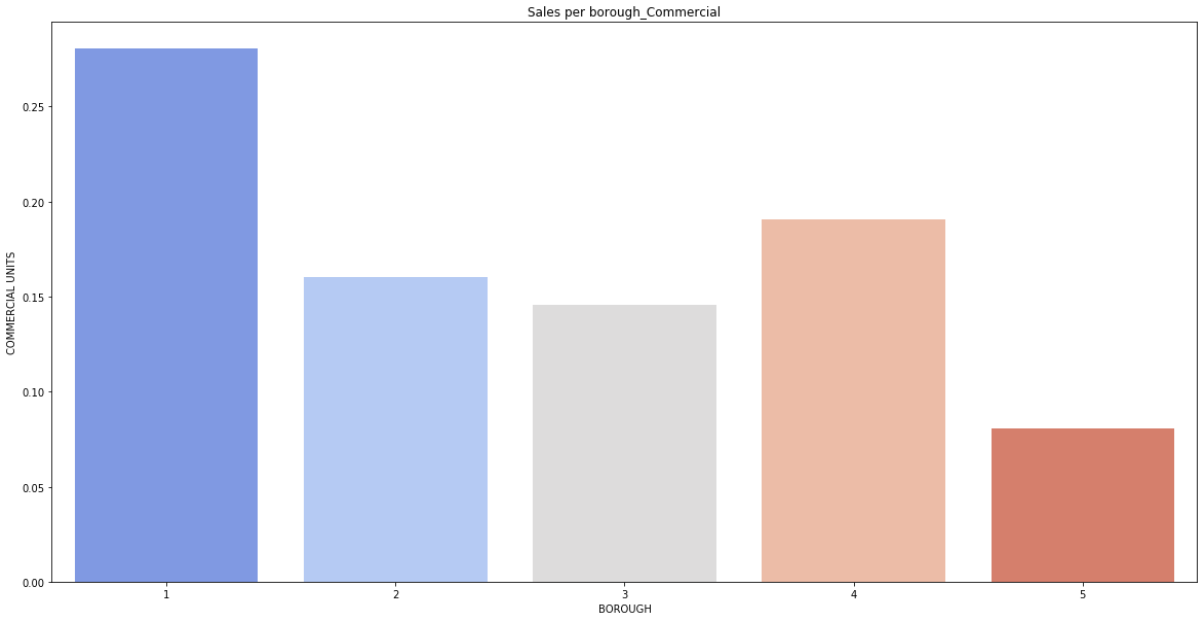
In [42]:
```
#Sales per borough_Residential
sns.barplot(y='RESIDENTIAL UNITS', x='BOROUGH',data=data, palette='coolwarm', ci=None)
plt.title('Sales per borough_Residential')
```

Out[42]: Text(0.5, 1.0, 'Sales per borough_Residential')



In [44]:
```
#Sales per borough_Commercial
sns.barplot(y='COMMERCIAL UNITS', x='BOROUGH',data=data, palette='coolwarm', ci=None)
plt.title('Sales per borough_Commercial')
```

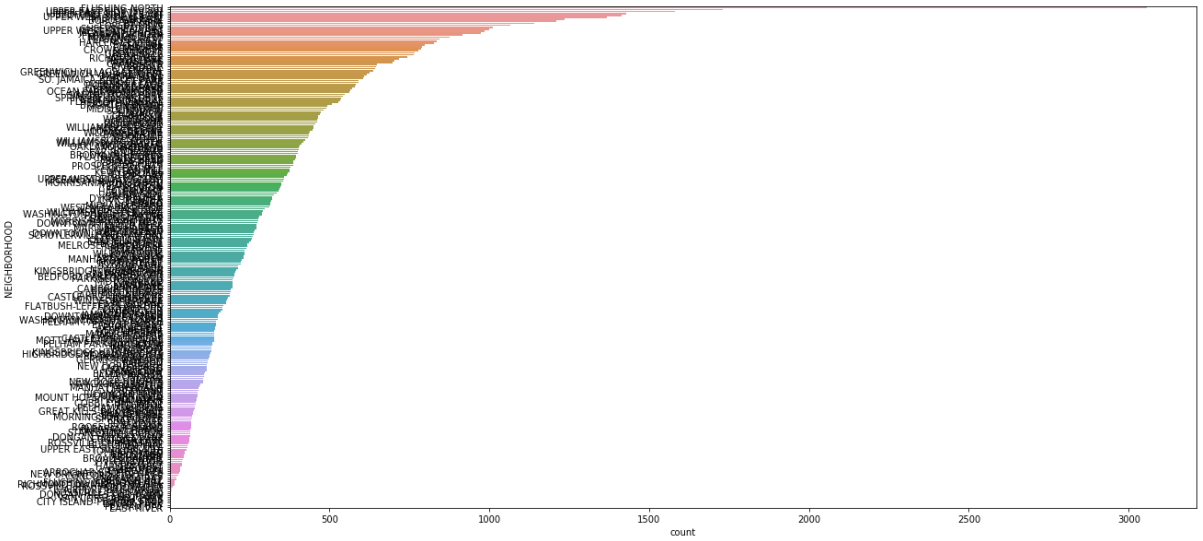
Out[44]: Text(0.5, 1.0, 'Sales per borough_Commercial')



In [23]:
```
data['NEIGHBORHOOD'].value_counts()
```

Out[23]:
```
FLUSHING-NORTH                3058
UPPER EAST SIDE (59-79)       1729
UPPER EAST SIDE (79-96)       1582
BEDFORD STUYVESANT            1429
UPPER WEST SIDE (59-79)       1415
                              ...
FRESH KILLS                      2
CITY ISLAND-PELHAM STRIP         1
BRONX PARK                       1
PELHAM BAY                       1
EAST RIVER                       1
Name: NEIGHBORHOOD, Length: 254, dtype: int64
```

In [24]:
```
sns.countplot(y = 'NEIGHBORHOOD',
              data = data,
              order = data['NEIGHBORHOOD'].value_counts().index)
plt.show()
```



In [25]:
```
data['BUILDING CLASS CATEGORY'].value_counts()
```

Out[25]:

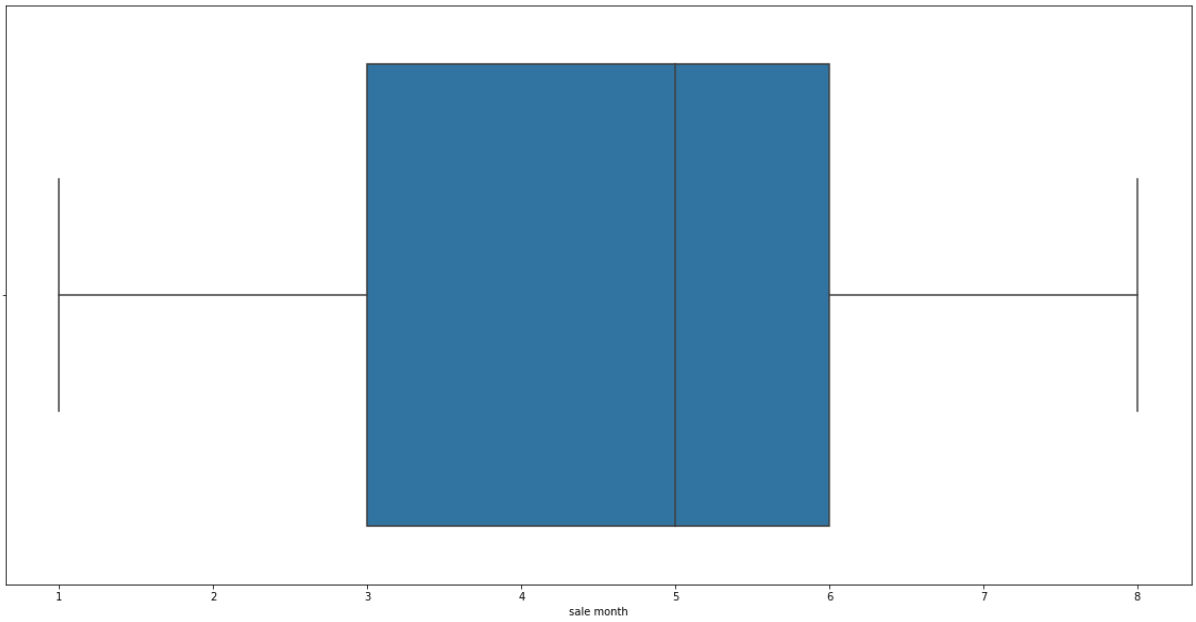
01 ONE FAMILY DWELLINGS	18107
02 TWO FAMILY DWELLINGS	15700
13 CONDOS - ELEVATOR APARTMENTS	12929
10 COOPS - ELEVATOR APARTMENTS	12876
03 THREE FAMILY DWELLINGS	4342
07 RENTALS - WALKUP APARTMENTS	3395
09 COOPS - WALKUP APARTMENTS	2764
04 TAX CLASS 1 CONDOS	1652
44 CONDO PARKING	1437
15 CONDOS - 2-10 UNIT RESIDENTIAL	1274
05 TAX CLASS 1 VACANT LAND	1221
17 CONDO COOPS	1199
22 STORE BUILDINGS	918
12 CONDOS - WALKUP APARTMENTS	871
14 RENTALS - 4-10 UNIT	662
29 COMMERCIAL GARAGES	578
43 CONDO OFFICE BUILDINGS	466
31 COMMERCIAL VACANT LAND	442
47 CONDO NON-BUSINESS STORAGE	373
08 RENTALS - ELEVATOR APARTMENTS	367
21 OFFICE BUILDINGS	343
30 WAREHOUSES	323
27 FACTORIES	198
06 TAX CLASS 1 - OTHER	177
45 CONDO HOTELS	167
41 TAX CLASS 4 - OTHER	154
46 CONDO STORE BUILDINGS	149
37 RELIGIOUS FACILITIES	98
16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT	95
26 OTHER HOTELS	76
33 EDUCATIONAL FACILITIES	69
32 HOSPITAL AND HEALTH FACILITIES	58
48 CONDO TERRACES/GARDENS/CABANAS	47
23 LOFT BUILDINGS	44
11A CONDO-RENTALS	42
49 CONDO WAREHOUSES/FACTORY/INDUS	30
35 INDOOR PUBLIC AND CULTURAL FACILITIES	30
28 COMMERCIAL CONDOS	28
38 ASYLUMS AND HOMES	25
36 OUTDOOR RECREATIONAL FACILITIES	13
42 CONDO CULTURAL/MEDICAL/EDUCATIONAL/ETC	13
25 LUXURY HOTELS	12
34 THEATRES	10
18 TAX CLASS 3 - UNTILITY PROPERTIES	3
11 SPECIAL CONDO BILLING LOTS	2
39 TRANSPORTATION FACILITIES	2
40 SELECTED GOVERNMENTAL FACILITIES	2

Name: BUILDING CLASS CATEGORY, dtype: int64

In [32]:
```
#SALE DATE should be datetime
data['SALE DATE']    = pd.to_datetime(data['SALE DATE'], errors='coerce'
)
data['sale year']    = data['SALE DATE'].dt.year
data['sale month']   = data['SALE DATE'].dt.month
data['sale quarter'] = data['SALE DATE'].dt.quarter
data['sale day']     = data['SALE DATE'].dt.day
data['sale weekday'] = data['SALE DATE'].dt.weekday
```

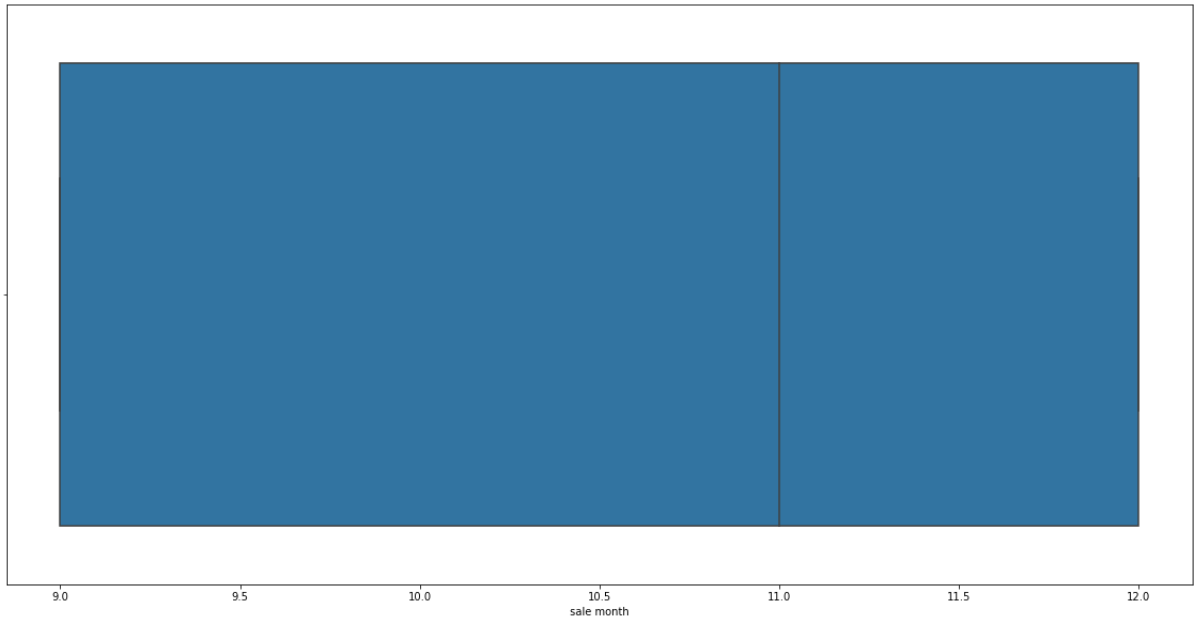
In [58]:
```
#Sale months for 2017
sns.boxplot(x='sale month', data=data[data['sale year']==2017])
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x27262da0948>



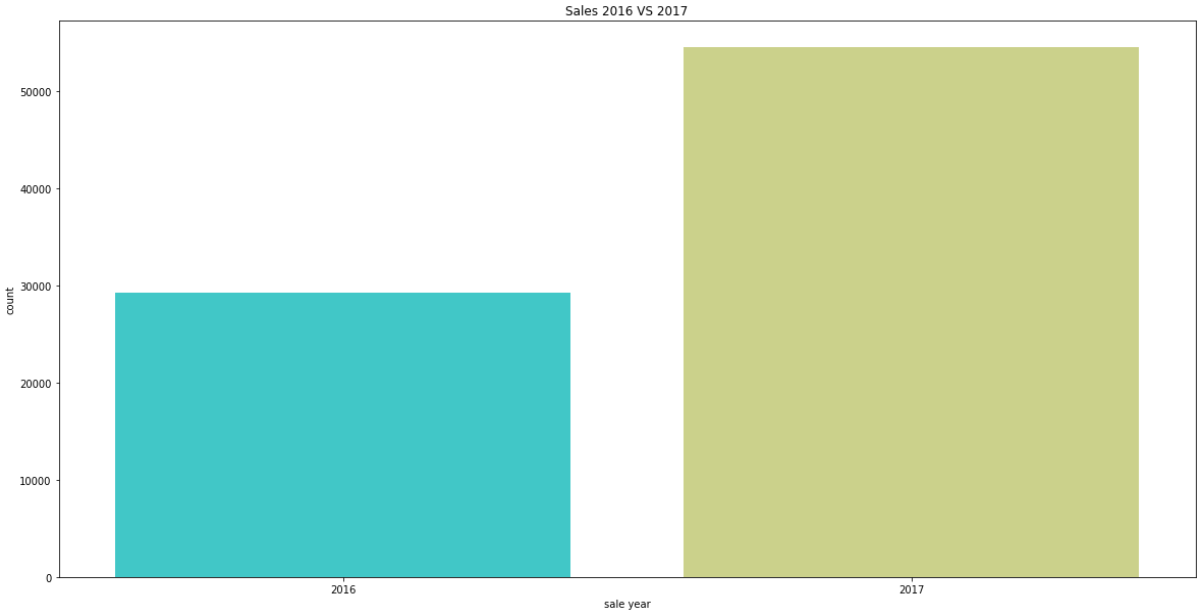
In [38]:
```
#Sale months for 2016
sns.boxplot(x='sale month', data=data[data['sale year']==2016])
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x2726104dcc8>



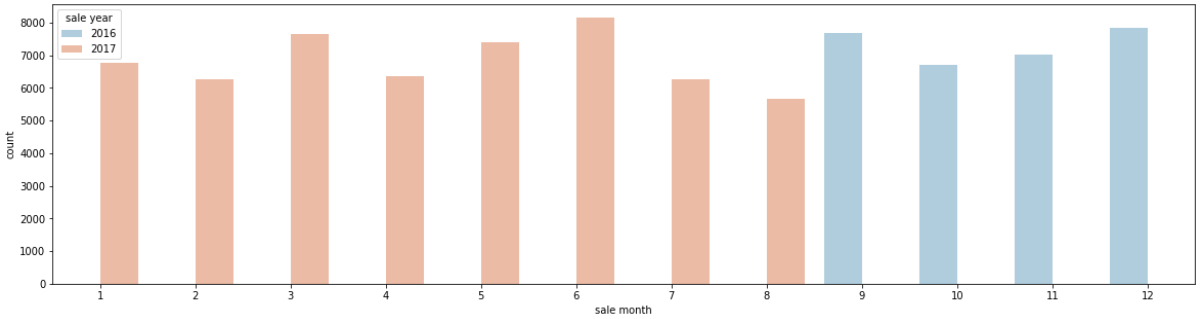
In [48]:
```
#Sales Compararison
sns.countplot(x='sale year', data=data, palette='rainbow')
plt.title('Sales 2016 VS 2017')
```

Out[48]: Text(0.5, 1.0, 'Sales 2016 VS 2017')

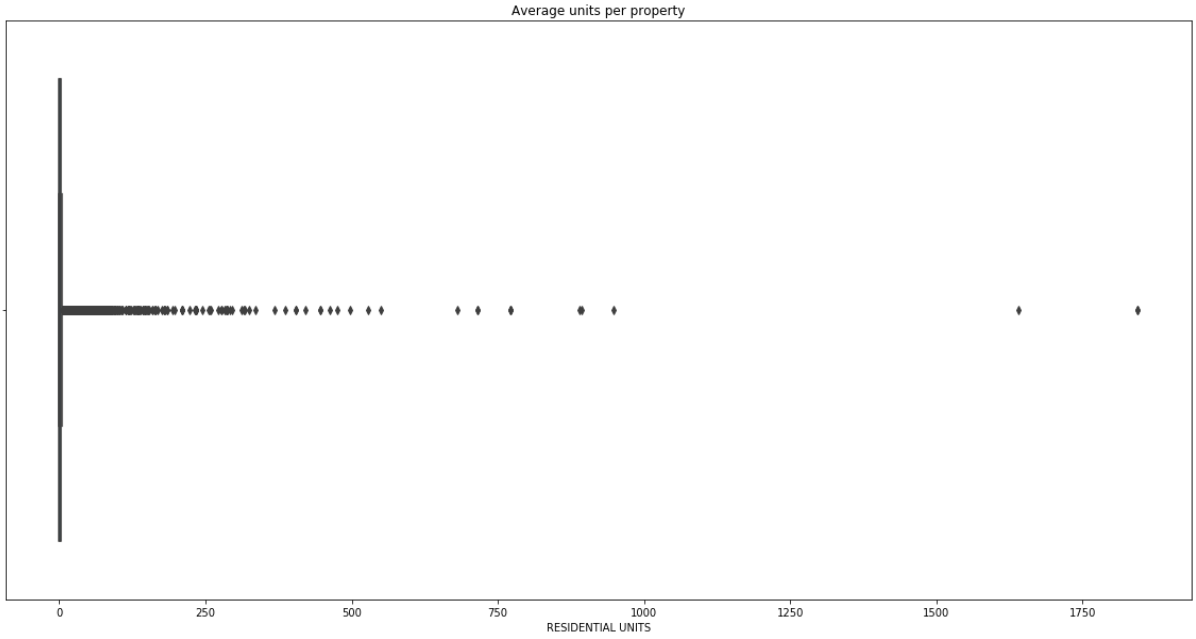


In [55]:
```
#Count of Sales in each month
plt.figure(figsize=(20,5))
sns.countplot('sale month', hue='sale year', data=data, palette='RdBu_r'
)
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x2726126e848>



In [73]:
```
#Average units per property
sns.boxplot(x='RESIDENTIAL UNITS',data=data)
plt.title('Average units per property')
plt.show()
```



In [ ]: