2023 Data Analysis

In [1]:
```
import seaborn as sns
sns.set()
sns.set_palette("Dark2")

# automatically reload utils module when it changes
%load_ext autoreload
%autoreload 2

# import utility functions
import sys
sys.path.append('../')
from src.utils import *

# set plotting defaults
%matplotlib inline
import matplotlib as mpl
mpl.rcParams['figure.dpi']= 300

# requires version 0.12.0 or higher
sns.__version__
```

Out[1]:
```
'0.12.2'
```

In [2]:
```
# load the 2023 data
df = get_data(2023)
```

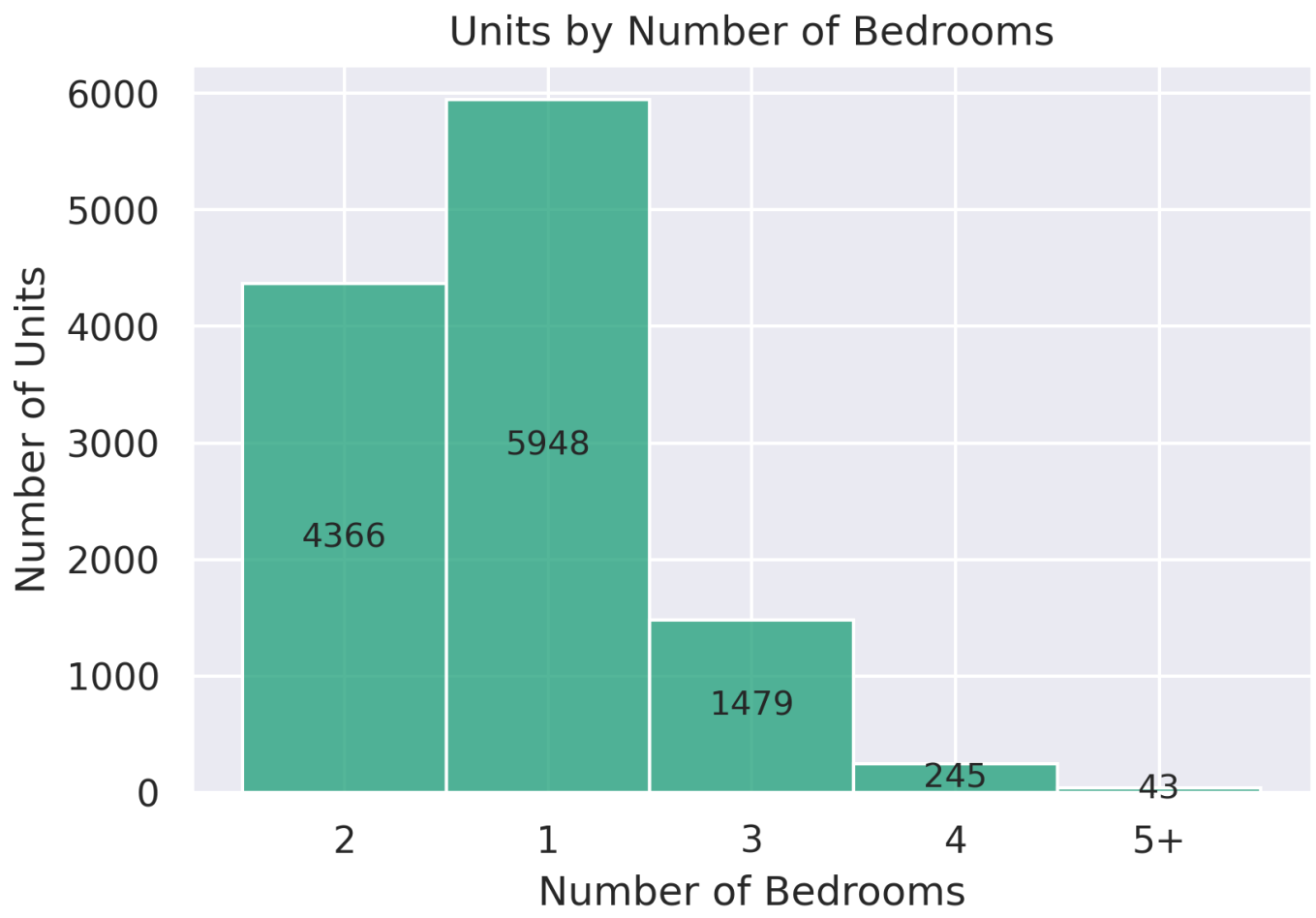
```
Breakdown by Outlier Condition:
        Outlier Rents: 5461 (30%)
        Outlier Increase vs Base: 608 ( 3%)
        Outlier Increase vs Previous: 328 ( 2%)
        Overall: 5905 (33%)

Breakdown by Subset:
        5905 outliers (33%)
        12081 non-outliers (67%)


        8567 rent increase (48%)
        9419 no rent increase (52%)


        7146 exempt (40%)
        10840 not exempt (60%)
```

Population Statistics

In [3]:
```
ax = sns.histplot(
    data=df[~df["outlier"]],
    x='nbrBedRms_grouped'
    )
for bars in ax.containers:
    ax.bar_label(
        bars,
        fmt='%d',
        label_type='center'
        )
ax.set_title("Units by Number of Bedrooms")
```

```
ax.set_xlabel("Number of Bedrooms")
ax.set_ylabel("Number of Units")
```

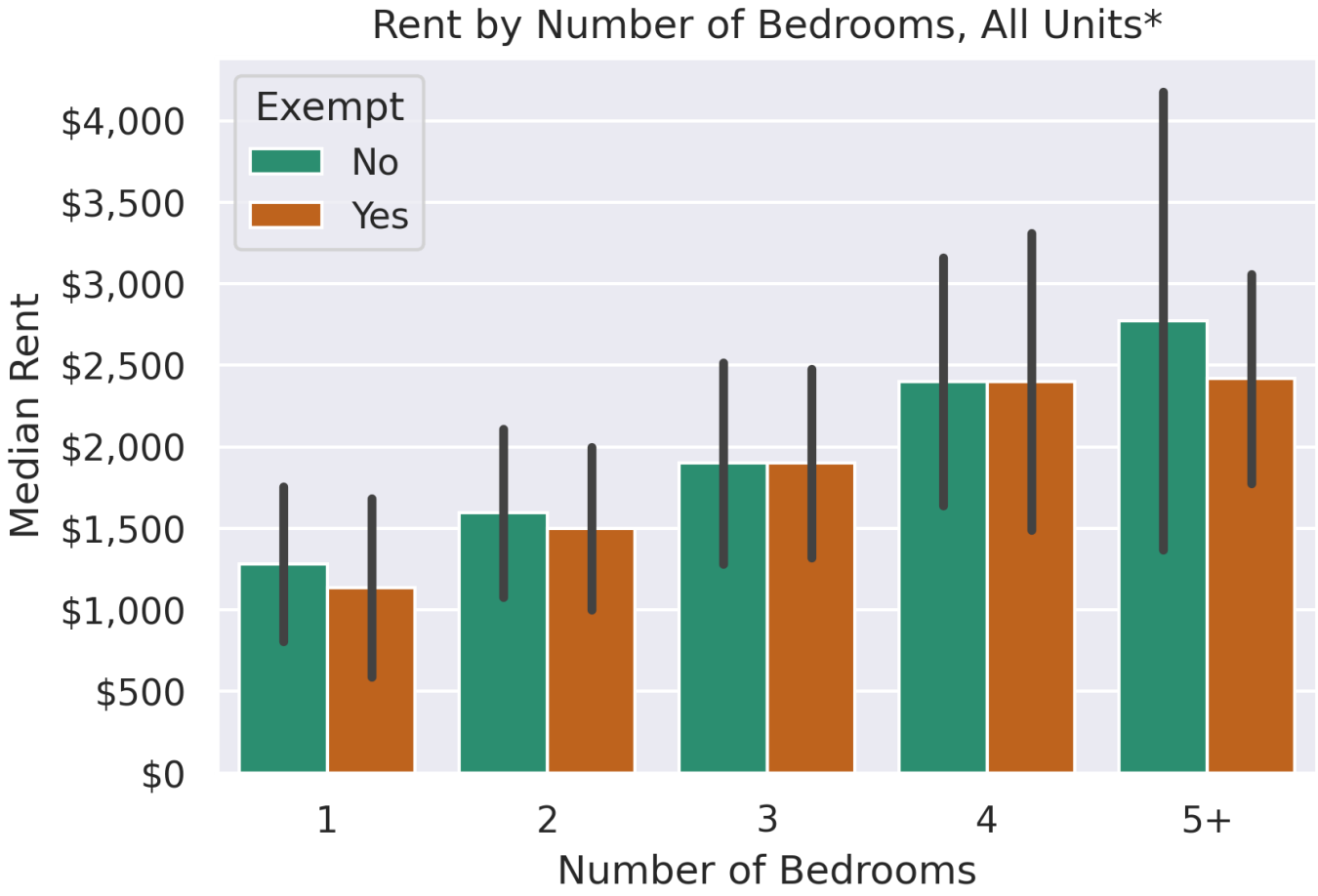
Out[3]:
```
Text(0, 0.5, 'Number of Units')
```



Overall Rent Statistics (Outliers Removed)

In [4]:
```
ax = sns.barplot(
    data=df[~df["outlier"]].sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="CurrentRent1",
    hue="exempt",
    estimator=np.median,
    errorbar='sd'
    )
ax.set_title("Rent by Number of Bedrooms, All Units*")
ax.set_xlabel("Number of Bedrooms")
ax.get_yaxis().set_major_formatter(mpl.ticker.FuncFormatter(lambda x, p: '${:,}'.format(
ax.set_ylabel("Median Rent")
handles, labels = ax.get_legend_handles_labels()
ax.legend(title="Exempt", handles=handles, labels=["No", "Yes"])
```

Out[4]:
```
<matplotlib.legend.Legend at 0x7f653c7f5660>
```

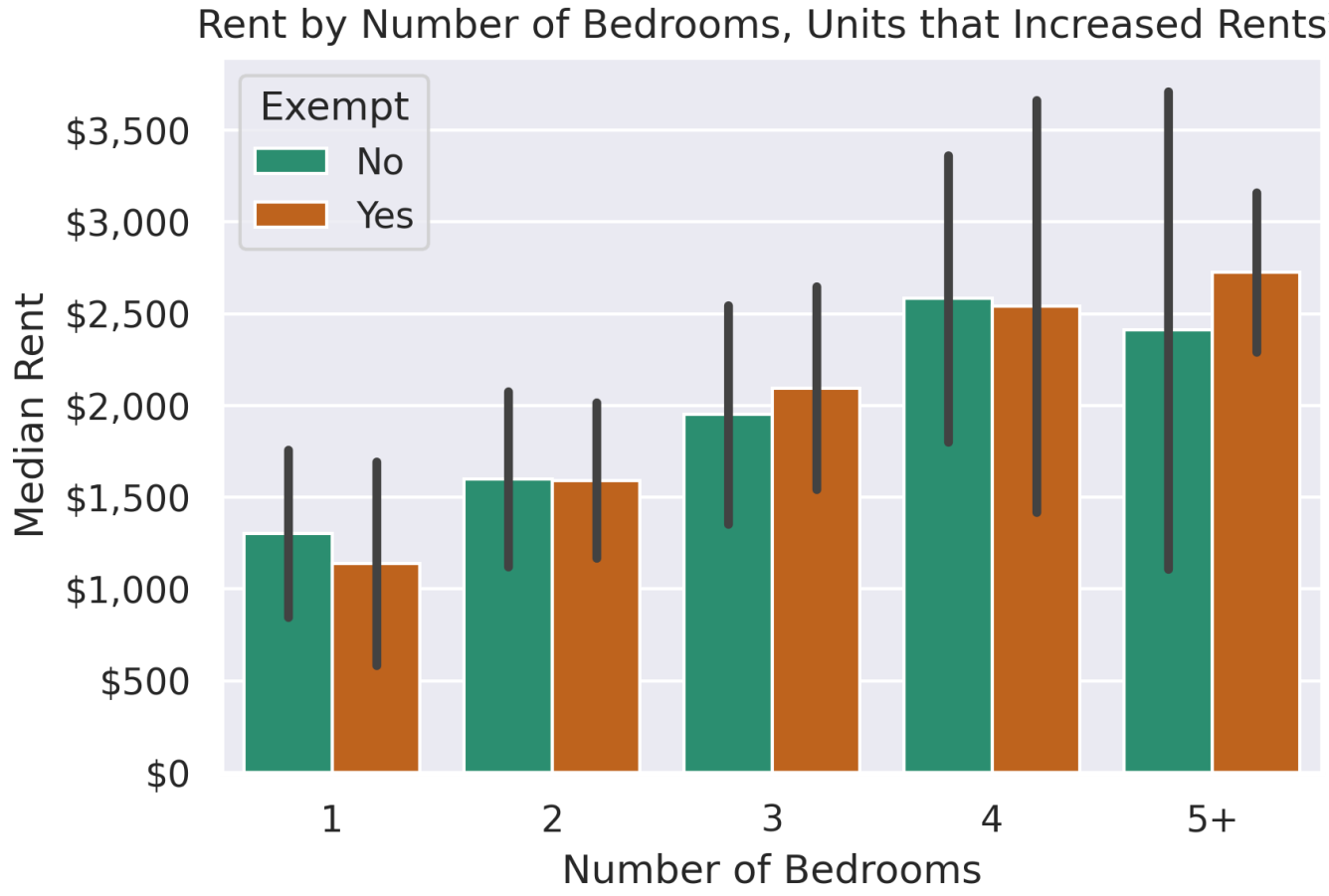


In [5]:
```
ax = sns.barplot(
    data=df[~df["outlier"]].sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="Rent_Inc_percent",
    hue="exempt",
    estimator=np.mean,
    errorbar=None
    )
ax.set_title("Rent Increase by Number of Bedrooms, All Units*")
ax.set_xlabel("Number of Bedrooms")
ax.get_yaxis().set_major_formatter(mpl.ticker.FuncFormatter(lambda x, p: '{:.1f}%'.forma
ax.set_ylabel("Mean Rent Increase")
handles, labels = ax.get_legend_handles_labels()
ax.legend(title="Exempt", handles=handles, labels=["No", "Yes"])
```

Out[5]:
```
<matplotlib.legend.Legend at 0x7f65c7e6f2b0>
```

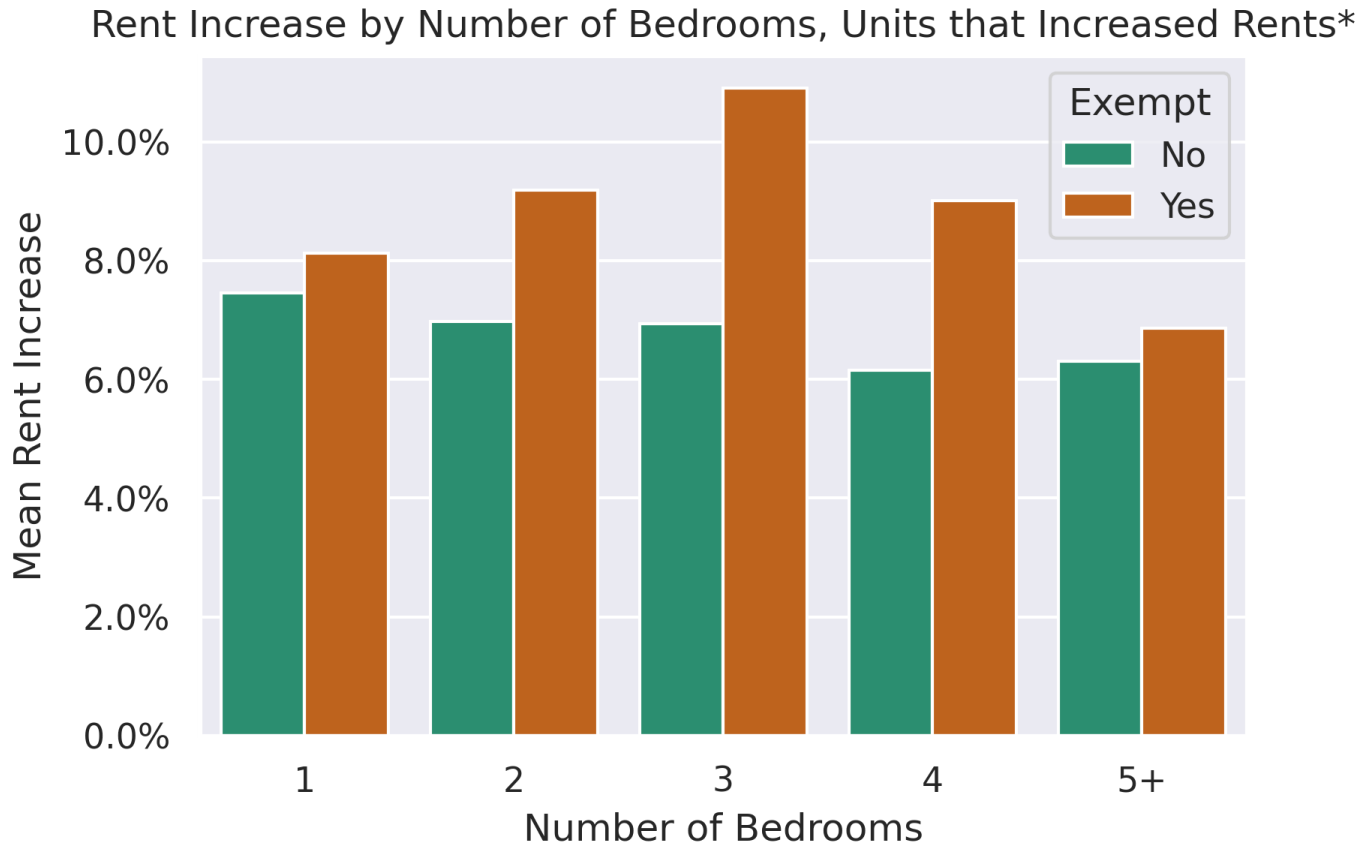
In [6]:
```
ax = sns.barplot(
    data=df[~df["outlier"] & (df["Rent_Inc"] > 0)].sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="CurrentRent1",
    hue="exempt",
    estimator=np.median,
    errorbar='sd'
    )
ax.set_title("Rent by Number of Bedrooms, Units that Increased Rents*")
ax.set_xlabel("Number of Bedrooms")
ax.get_yaxis().set_major_formatter(mpl.ticker.FuncFormatter(lambda x, p: '${:,}'.format(
ax.set_ylabel("Median Rent")
handles, labels = ax.get_legend_handles_labels()
ax.legend(title="Exempt", handles=handles, labels=["No", "Yes"])
```

Out[6]:
```
<matplotlib.legend.Legend at 0x7f653b4281c0>
```



In [7]:
```
ax = sns.barplot(
    data=df[~df["outlier"] & (df["Rent_Inc"] > 0)].sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="Rent_Inc_percent",
    hue="exempt",
    estimator=np.mean,
    errorbar=None
    )
ax.set_title("Rent Increase by Number of Bedrooms, Units that Increased Rents*")
ax.set_xlabel("Number of Bedrooms")
ax.get_yaxis().set_major_formatter(mpl.ticker.FuncFormatter(lambda x, p: '{:.1f}%'.forma
ax.set_ylabel("Mean Rent Increase")
handles, labels = ax.get_legend_handles_labels()
ax.legend(title="Exempt", handles=handles, labels=["No", "Yes"])
```

Out[7]:
```
<matplotlib.legend.Legend at 0x7f653ccd99c0>
```

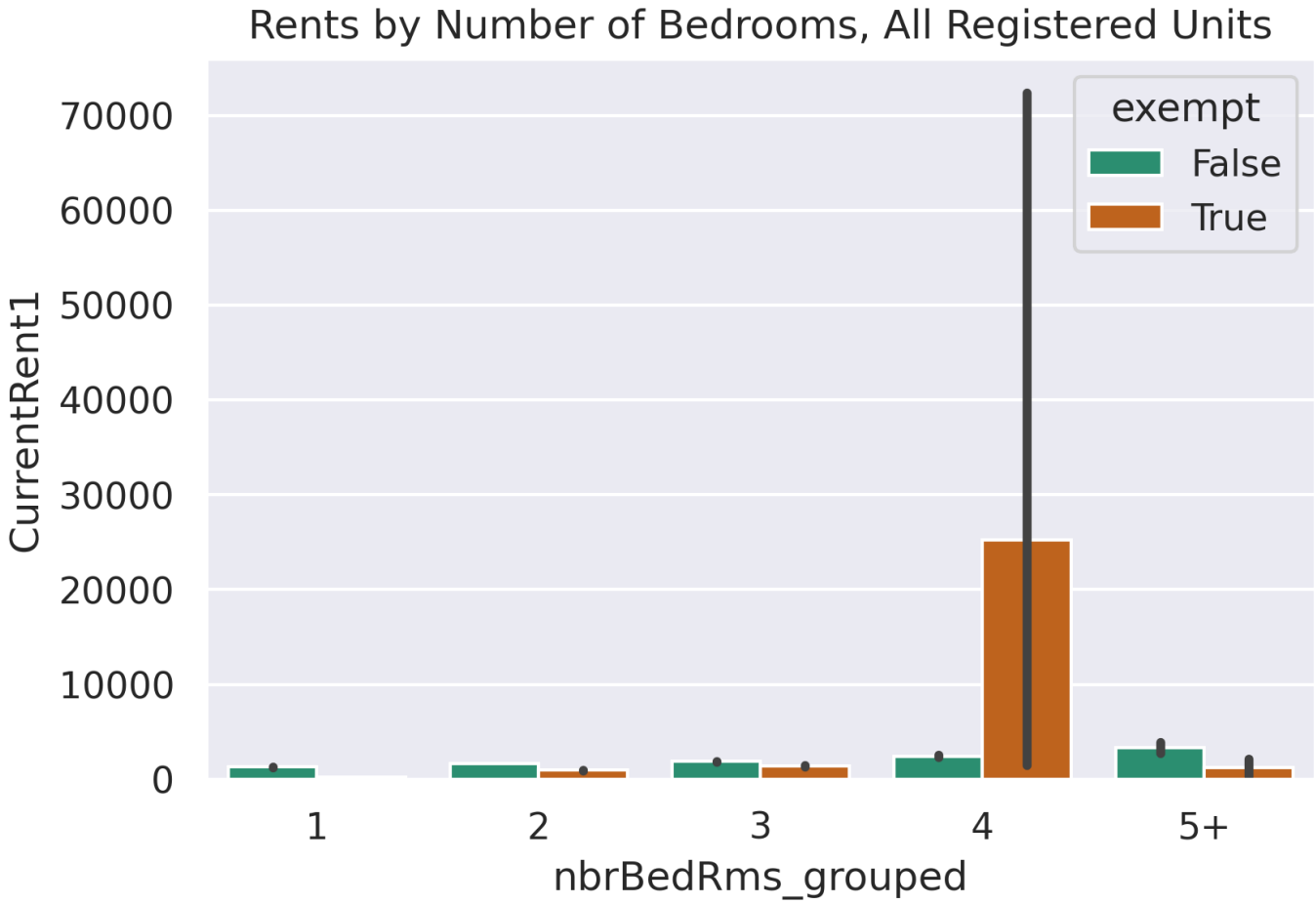


Overall Rent Statistics (Outliers Included)

These charts are included for completeness only and should not be used in subsequent analysis.

In [8]:
```
ax = sns.barplot(
    data=df.sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="CurrentRent1",
    hue="exempt"
    )
ax.set_title("Rents by Number of Bedrooms, All Registered Units")
```

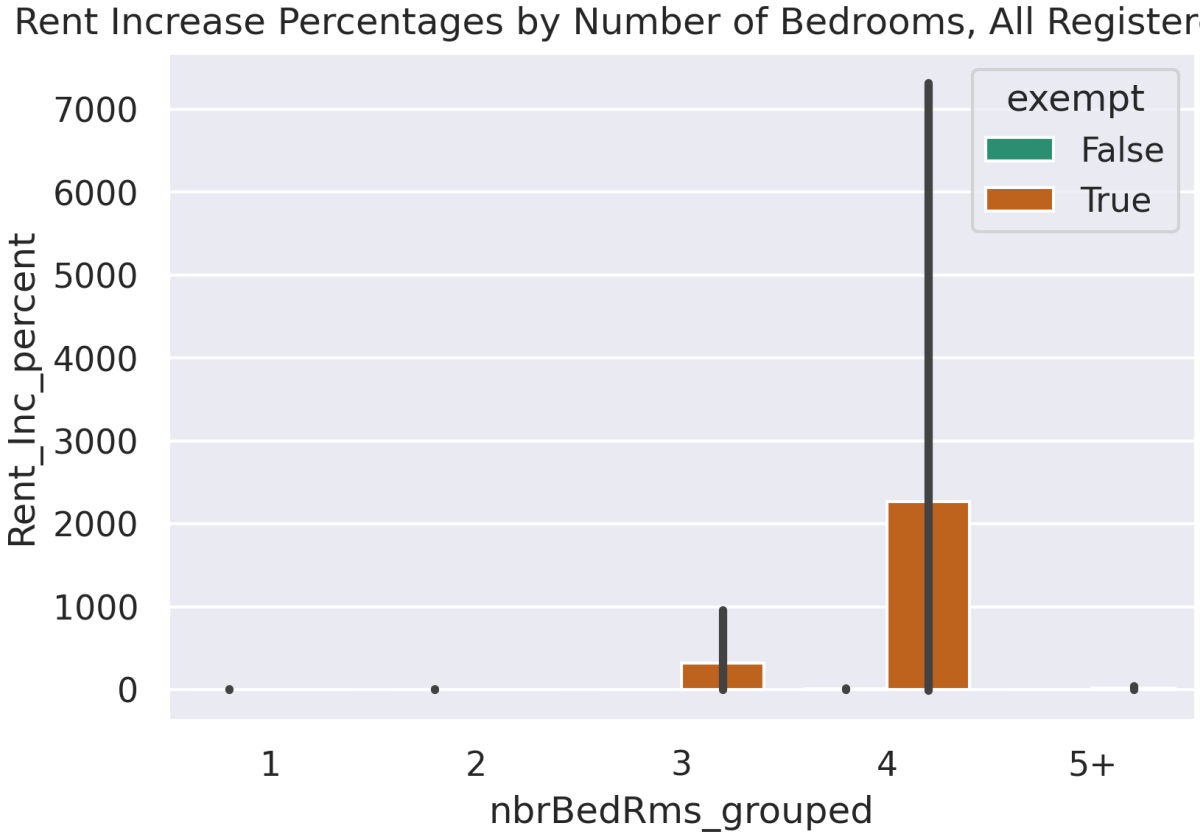
Out[8]:
```
Text(0.5, 1.0, 'Rents by Number of Bedrooms, All Registered Units')
```



In [9]:
```
ax = sns.barplot(
    data=df.sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="Rent_Inc_percent",
    hue="exempt"
    )
ax.set_title("Rent Increase Percentages by Number of Bedrooms, All Registered Units")
```

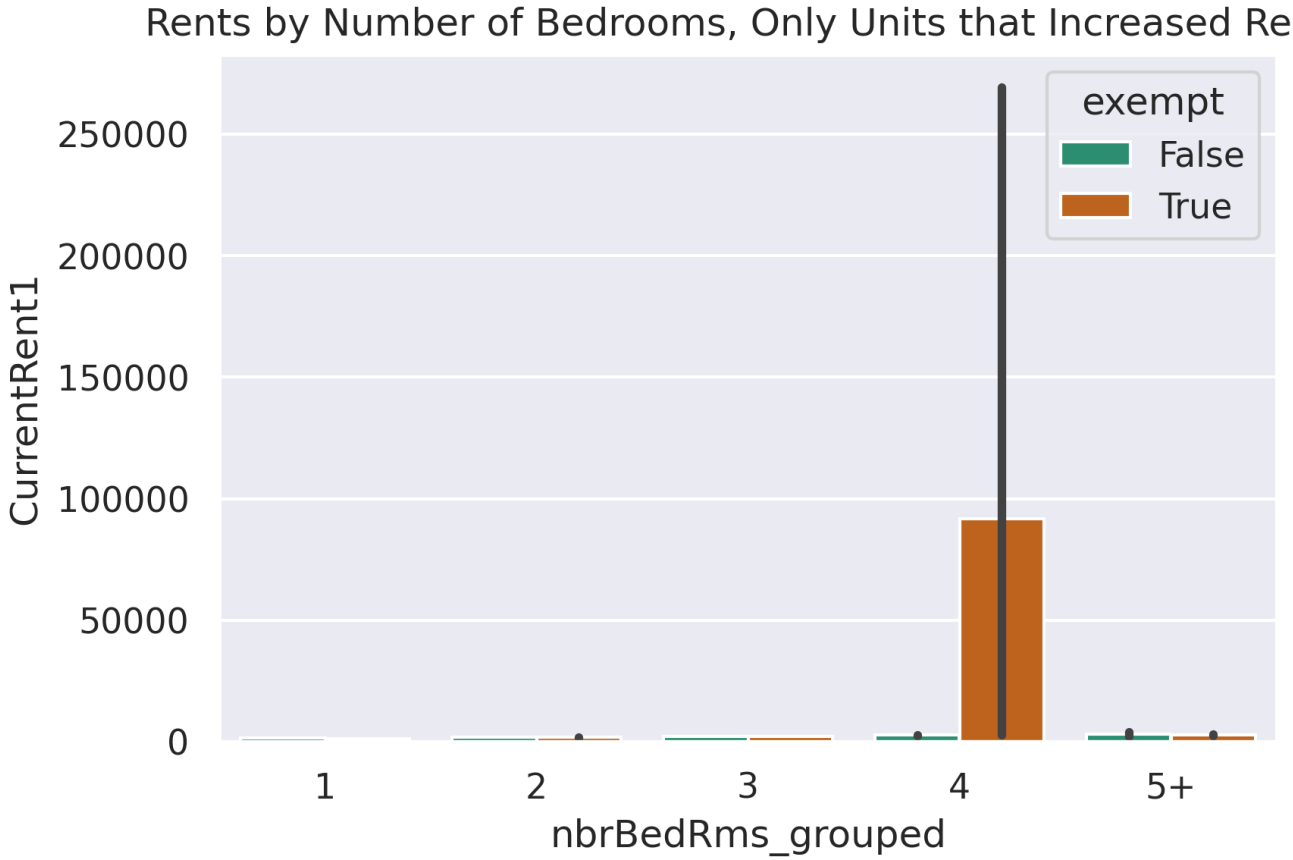
```
/home/philip/miniconda3/envs/ds5110/lib/python3.10/site-packages/seaborn/algorithms.py:98: RuntimeWarning: Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
```

Out[9]:
```
Text(0.5, 1.0, 'Rent Increase Percentages by Number of Bedrooms, All Registered Units')
```



In [10]:
```
ax = sns.barplot(
    data=df[df["Rent_Inc"] > 0].sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="CurrentRent1",
    hue="exempt"
    )
ax.set_title("Rents by Number of Bedrooms, Only Units that Increased Rents")
```

Out[10]:
```
Text(0.5, 1.0, 'Rents by Number of Bedrooms, Only Units that Increased Rents')
```



In [11]:
```
ax = sns.barplot(
    data=df[(df["Rent_Inc"] > 0)].sort_values("nbrBedRms_grouped"),
    x="nbrBedRms_grouped",
    y="Rent_Inc_percent",
    hue="exempt"
    )
ax.set_title("Rents Increase Percentages by Number of Bedrooms, Only Units that Increase
```

Out[11]:
```
Text(0.5, 1.0, 'Rents Increase Percentages by Number of Bedrooms, Only Units that Increa
sed Rents')
```