Housing Rental Analysis for San Francisco

In this challenge, your job is to use your data visualization skills, including aggregation, interactive visualizations, and geospatial analysis, to find properties in the San Francisco market that are viable investment opportunities.

Instructions

Use the `san_francisco_housing.ipynb` notebook to visualize and analyze the real-estate data.

Note that this assignment requires you to create a visualization by using hvPlot and GeoViews. Additionally, you need to read the `sfo_neighborhoods_census_data.csv` file from the `Resources` folder into the notebook and create the DataFrame that you'll use in the analysis.

The main task in this Challenge is to visualize and analyze the real-estate data in your Jupyter notebook. Use the `san_francisco_housing.ipynb` notebook to complete the following tasks:

- Calculate and plot the housing units per year.
- Calculate and plot the average prices per square foot.
- Compare the average prices by neighborhood.
- Build an interactive neighborhood map.
- Compose your data story.

Calculate and Plot the Housing Units per Year

For this part of the assignment, use numerical and visual aggregation to calculate the number of housing units per year, and then visualize the results as a bar chart. To do so, complete the following steps:

1. Use the `groupby` function to group the data by year. Aggregate the results by the `mean` of the groups.
2. Use the `hvplot` function to plot the `housing_units_by_year` DataFrame as a bar chart. Make the x-axis represent the `year` and the y-axis represent the `housing_units`.
3. Style and format the line plot to ensure a professionally styled visualization.
4. Answer the following question:
    - What's the overall trend in housing units over the period that you're analyzing?

Calculate and Plot the Average Sale Prices per Square Foot

For this part of the assignment, use numerical and visual aggregation to calculate the average prices per square foot, and then visualize the results as a bar chart. To do so, complete the following steps:

1. Group the data by year, and then average the results. What's the lowest gross rent that's reported for the years that the DataFrame includes?

2. Create a new DataFrame named `prices_square_foot_by_year` by filtering out the "housing_units" column. The new DataFrame should include the averages per year for only the sale price per square foot and the gross rent.

3. Use hvPlot to plot the `prices_square_foot_by_year` DataFrame as a line plot.

   > **Hint** This single plot will include lines for both `sale_price_sqr_foot` and `gross_rent`.

4. Style and format the line plot to ensure a professionally styled visualization.

5. Use both the `prices_square_foot_by_year` DataFrame and interactive plots to answer the following questions:

   - Did any year experience a drop in the average sale price per square foot compared to the previous year?

   - If so, did the gross rent increase or decrease during that year?

Compare the Average Sale Prices by Neighborhood

For this part of the assignment, use interactive visualizations and widgets to explore the average sale price per square foot by neighborhood. To do so, complete the following steps:

1. Create a new DataFrame that groups the original DataFrame by year and neighborhood. Aggregate the results by the `mean` of the groups.

2. Filter out the "housing_units" column to create a DataFrame that includes only the `sale_price_sqr_foot` and `gross_rent` averages per year.

3. Create an interactive line plot with hvPlot that visualizes both `sale_price_sqr_foot` and `gross_rent`. Set the x-axis parameter to the year (`x="year"`). Use the `groupby` parameter to create an interactive widget for `neighborhood`.

4. Style and format the line plot to ensure a professionally styled visualization.

5. Use the interactive visualization to answer the following question:

   - For the Anza Vista neighborhood, is the average sale price per square foot for 2016 more or less than the price that's listed for 2012?

Build an Interactive Neighborhood Map

For this part of the assignment, explore the geospatial relationships in the data by using interactive visualizations with hvPlot and GeoViews. To build your map, use the `sfo_data_df` DataFrame (created during the initial import), which includes the neighborhood location data with the average prices. To do all this, complete the following steps:

1. Read the `neighborhood_coordinates.csv` file from the `Resources` folder into the notebook, and create a DataFrame named `neighborhood_locations_df`. Be sure to set the `index_col` of the

DataFrame as “Neighborhood”.

2. Using the original `sfo_data_df` Dataframe, create a DataFrame named `all_neighborhood_info_df` that groups the data by neighborhood. Aggregate the results by the `mean` of the group.

3. Review the two code cells that concatenate the `neighborhood_locations_df` DataFrame with the `all_neighborhood_info_df` DataFrame. Note that the first cell uses the Pandas concat function to create a DataFrame named `all_neighborhoods_df`. The second cell cleans the data and sets the “Neighborhood” column. Be sure to run these cells to create the `all_neighborhoods_df` DataFrame, which you’ll need to create the geospatial visualization.

4. Using hvPlot with GeoViews enabled, create a `points` plot for the `all_neighborhoods_df` DataFrame. Be sure to do the following:

   - Set the `geo` parameter to True.
   - Set the `size` parameter to “sale_price_sqr_foot”.
   - Set the `color` parameter to “gross_rent”.
   - Set the `frame_width` parameter to 700.
   - Set the `frame_height` parameter to 500.
   - Include a descriptive title.

5. Use the interactive map to answer the following question:

   - Which neighborhood has the highest gross rent, and which has the highest sale price per square foot?

Compose Your Data Story

Based on the visualizations that you created, answer the following questions:

- How does the trend in rental income growth compare to the trend in sales prices? Does this same trend hold true for all the neighborhoods across San Francisco?

- What insights can you share with your company about the potential one-click, buy-and-rent strategy that they're pursuing? Do neighborhoods exist that you would suggest for investment, and why?

In [1]:

```
# Import the required libraries and dependencies
import pandas as pd
import hvplot.pandas
import panel as pn
from bokeh.sampledata.iris import flowers
from pathlib import Path
```

Import the data

In [2]:

```
# Using the read_csv function and Path module, create a DataFrame
# by importing the sfo_neighborhoods_census_data.csv file from the Resources folder
sfo_data_df = pd.read_csv(Path("Resources/sfo_neighborhoods_census_data.csv"))# YOUR COD

# Review the first and last five rows of the DataFrame
# YOUR CODE HERE
sfo_data_df.head()
# YOUR CODE HERE
sfo_data_df.tail()
```

Out[2]:

	year	neighborhood	sale_price_sqr_foot	housing_units	gross_rent
392	2016	Telegraph Hill	903.049771	384242	4390
393	2016	Twin Peaks	970.085470	384242	4390
394	2016	Van Ness/ Civic Center	552.602567	384242	4390
395	2016	Visitacion Valley	328.319007	384242	4390
396	2016	Westwood Park	631.195426	384242	4390

Step 1: Use the `groupby` function to group the data by year. Aggregate the results by the `mean` of the groups.

In [3]:

```
# Create a numerical aggregation that groups the data by the year and then averages the
housing_units_by_year = sfo_data_df.groupby("year").mean()# YOUR CODE HERE

# Review the DataFrame
# YOUR CODE HERE
housing_units_by_year.head()
```

Out[3]:

year	sale_price_sqr_foot	housing_units	gross_rent
2010	369.344353	372560.0	1239.0
2011	341.903429	374507.0	1530.0
2012	399.389968	376454.0	2324.0
2013	483.600304	378401.0	2971.0
2014	556.277273	380348.0	3528.0

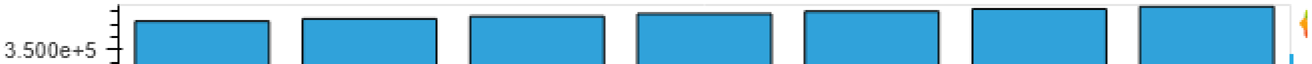
Step 2: Use the `hvplot` function to plot the `housing_units_by_year` DataFrame as a bar chart. Make the x-axis represent the `year` and the y-axis represent the `housing_units`.

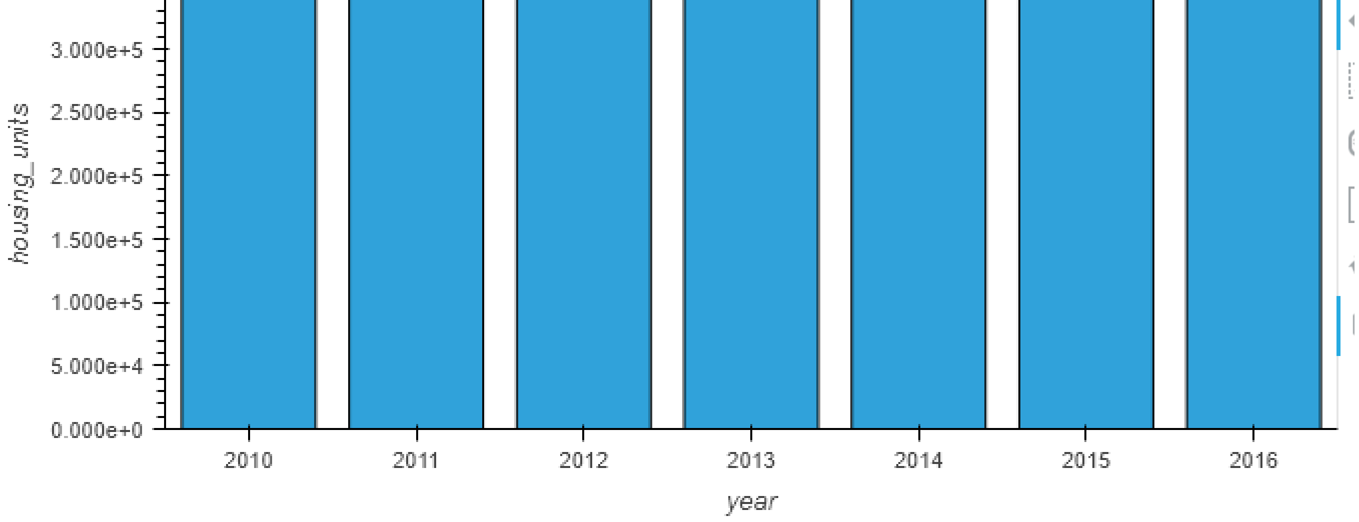
Step 3: Style and format the line plot to ensure a professionally styled visualization.

In [4]:

```
# Create a visual aggregation explore the housing units by year
# YOUR CODE HERE
housing_units_by_year.hvplot.bar(
    x="year",
    y="housing_units"
)
```

Out[4]:





Step 5: Answer the following question:

Question: What is the overall trend in housing_units over the period being analyzed?

Answer: # The number of housing units has remained relatively stable throughout these years.

Step 1: Group the data by year, and then average the results.

In [5]:
```
# Create a numerical aggregation by grouping the data by year and averaging the results
prices_square_foot_by_year = housing_units_by_year.drop("housing_units", axis=1)# YOUR C
#instead of starting from the beginning with the original df, the previous df was alread
# Review the resulting DataFrame
# YOUR CODE HERE
prices_square_foot_by_year
```

Out[5]:

	sale_price_sqr_foot	gross_rent
year		
2010	369.344353	1239.0
2011	341.903429	1530.0
2012	399.389968	2324.0
2013	483.600304	2971.0
2014	556.277273	3528.0
2015	632.540352	3739.0
2016	697.643709	4390.0

Question: What is the lowest gross rent reported for the years included in the DataFrame?

Answer: $1239 for the year 2010

Step 2: Create a new DataFrame named `prices_square_foot_by_year` by filtering out the "housing_units" column. The new DataFrame should include the averages per year for only the sale price per square foot and the gross rent.

In [6]:

```
# Filter out the housing_units column, creating a new DataFrame
# Keep only sale_price_sqr_foot and gross_rent averages per year
prices_square_foot_by_year = # YOUR CODE HERE

#already done in previouss cells.

# Review the DataFrame
# YOUR CODE HERE
```

```
  File "C:\Users\yohan\AppData\Local\Temp\ipykernel_28068\1484772018.py", line 3
    prices_square_foot_by_year = # YOUR CODE HERE
                                                 ^
SyntaxError: invalid syntax
```

Step 3: Use hvPlot to plot the `prices_square_foot_by_year` DataFrame as a line plot.

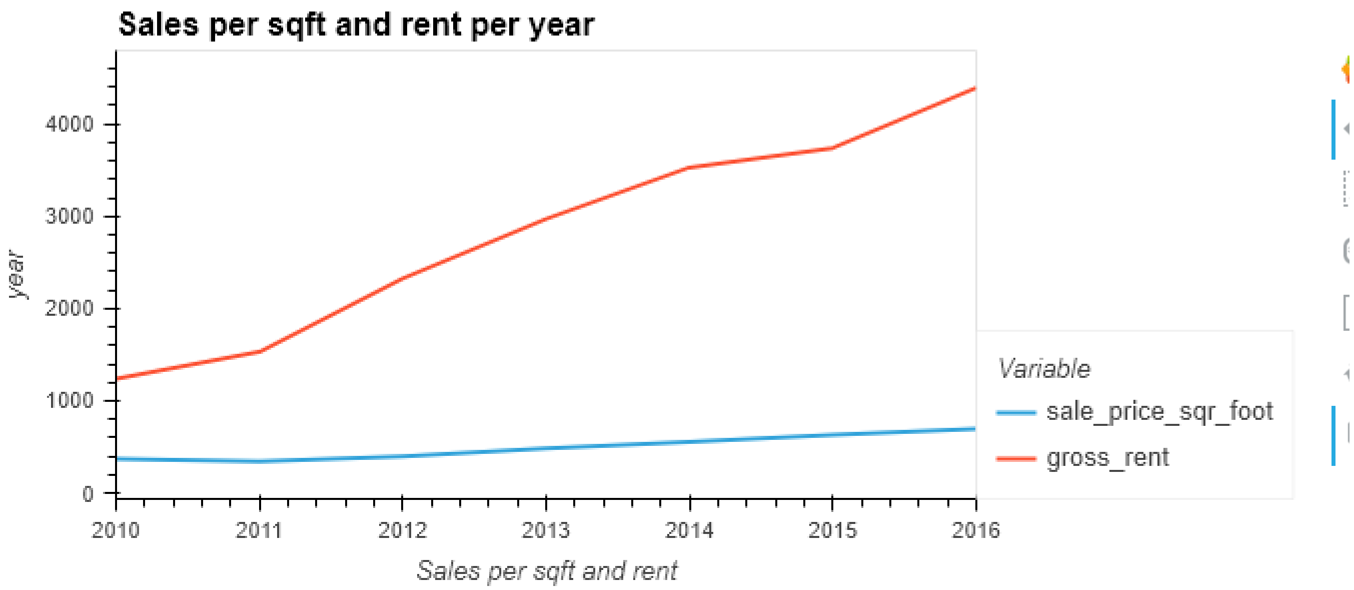
> **Hint** This single plot will include lines for both `sale_price_sqr_foot` and `gross_rent`

Step 4: Style and format the line plot to ensure a professionally styled visualization.

In [7]:

```
# Plot prices_square_foot_by_year.
# Inclued labels for the x- and y-axes, and a title.
# YOUR CODE HERE
prices_square_foot_by_year.hvplot(
    use_index=True,
    y=("sale_price_sqr_foot", "gross_rent"),
    xlabel= "Sales per sqft and rent",
    ylabel = "year"
).opts(
    title="Sales per sqft and rent per year"
)
```

Out[7]:



Step 6: Use both the `prices_square_foot_by_year` DataFrame and interactive plots to answer the following questions:

Question: Did any year experience a drop in the average sale price per square foot compared to the previous year?

Answer: # 2011

Question: If so, did the gross rent increase or decrease during that year?

Answer: # Rent increased that year

Step 1: Create a new DataFrame that groups the original DataFrame by year and neighborhood. Aggregate the results by the `mean` of the groups.

In [8]:

```
# Group by year and neighborhood and then create a new dataframe of the mean values

prices_by_year_by_neighborhood = sfo_data_df.groupby(["year", "neighborhood"]).mean()

# Review the DataFrame
# YOUR CODE HERE
prices_by_year_by_neighborhood
```

Out[8]:

		sale_price_sqr_foot	housing_units	gross_rent
year	**neighborhood**			
2010	**Alamo Square**	291.182945	372560.0	1239.0
	Anza Vista	267.932583	372560.0	1239.0
	Bayview	170.098665	372560.0	1239.0
	Buena Vista Park	347.394919	372560.0	1239.0
	Central Richmond	319.027623	372560.0	1239.0
...	**...**	...	...	...
2016	**Telegraph Hill**	903.049771	384242.0	4390.0
	Twin Peaks	970.085470	384242.0	4390.0
	Van Ness/ Civic Center	552.602567	384242.0	4390.0
	Visitacion Valley	328.319007	384242.0	4390.0
	Westwood Park	631.195426	384242.0	4390.0

397 rows × 3 columns

Step 2: Filter out the “housing_units” column to create a DataFrame that includes only the `sale_price_sqr_foot` and `gross_rent` averages per year.

In [9]:

```
# Filter out the housing_units
prices_by_year_by_neighborhood = prices_by_year_by_neighborhood.drop("housing_units", ax

# Review the first and last five rows of the DataFrame
# YOUR CODE HERE
prices_by_year_by_neighborhood.head()
# YOUR CODE HERE
prices_by_year_by_neighborhood.tail()
```

Out[9]:

		sale_price_sqr_foot	gross_rent
year	neighborhood		
2016	Telegraph Hill	903.049771	4390.0
	Twin Peaks	970.085470	4390.0
	Van Ness/ Civic Center	552.602567	4390.0
	Visitacion Valley	328.319007	4390.0
	Westwood Park	631.195426	4390.0

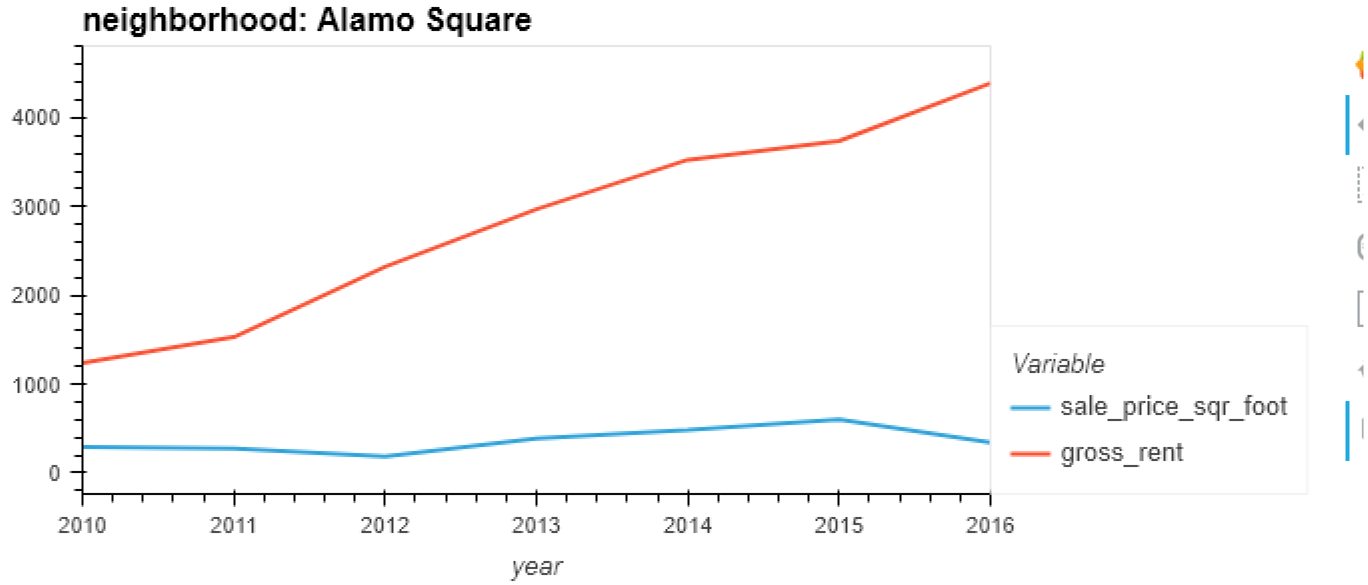
Step 3: Create an interactive line plot with hvPlot that visualizes both `sale_price_sqr_foot` and `gross_rent`. Set the x-axis parameter to the year (`x="year"`). Use the `groupby` parameter to create an interactive widget for `neighborhood`.

Step 4: Style and format the line plot to ensure a professionally styled visualization.

In [10]:

```python
# Use hvplot to create an interactive line plot of the average price per square foot
# The plot should have a dropdown selector for the neighborhood
# YOUR CODE HERE
prices_by_year_by_neighborhood.hvplot(
    x="year",
    y=("sale_price_sqr_foot", "gross_rent"),
    groupby = "neighborhood"
)
```

Out[10]:



Step 6: Use the interactive visualization to answer the following question:

Question: For the Anza Vista neighborhood, is the average sale price per square foot for 2016 more or less than the price that’s listed for 2012?

Answer: # In 2016, avg sale price is less than 2012.

Build an Interactive Neighborhood Map

Step 1: Read the `neighborhood_coordinates.csv` file from the `Resources` folder into the notebook, and create a DataFrame named `neighborhood_locations_df`. Be sure to set the `index_col` of the DataFrame as “Neighborhood”.

In [11]:

```
# Load neighborhoods coordinates data
neighborhood_locations_df = pd.read_csv(Path("Resources/neighborhoods_coordinates.csv"),

# Review the DataFrame
# YOUR CODE HERE
neighborhood_locations_df.head()
```

Out[11]:

	Lat	Lon
Neighborhood		
Alamo Square	37.791012	-122.402100
Anza Vista	37.779598	-122.443451
Bayview	37.734670	-122.401060
Bayview Heights	37.728740	-122.410980
Bernal Heights	37.728630	-122.443050

Step 2: Using the original `sfo_data_df` Dataframe, create a DataFrame named `all_neighborhood_info_df` that groups the data by neighborhood. Aggregate the results by the `mean` of the group.

In [12]:

```
# Calculate the mean values for each neighborhood
all_neighborhood_info_df = sfo_data_df.groupby("neighborhood").mean()  # YOUR CODE HERE

# Review the resulting DataFrame
# YOUR CODE HERE
all_neighborhood_info_df.head(10)
```

Out[12]:

	year	sale_price_sqr_foot	housing_units	gross_rent
neighborhood				
Alamo Square	2013.000000	366.020712	378401.0	2817.285714
Anza Vista	2013.333333	373.382198	379050.0	3031.833333
Bayview	2012.000000	204.588623	376454.0	2318.400000
Bayview Heights	2015.000000	590.792839	382295.0	3739.000000
Bernal Heights	2013.500000	576.746488	379374.5	3080.333333
Buena Vista Park	2012.833333	452.680591	378076.5	2698.833333
Central Richmond	2013.000000	394.422399	378401.0	2817.285714
Central Sunset	2013.000000	423.687928	378401.0	2817.285714
Clarendon Heights	2012.000000	487.244886	376454.0	2250.500000

Corona Heights	2012.400000	587.539067	377232.8	2472.000000

Step 3: Review the two code cells that concatenate the `neighborhood_locations_df` DataFrame with the `all_neighborhood_info_df` DataFrame.

Note that the first cell uses the Pandas concat function to create a DataFrame named `all_neighborhoods_df`.

The second cell cleans the data and sets the “Neighborhood” column.

Be sure to run these cells to create the `all_neighborhoods_df` DataFrame, which you’ll need to create the geospatial visualization.

In [13]:

```
# Using the Pandas `concat` function, join the
# neighborhood_locations_df and the all_neighborhood_info_df DataFrame
# The axis of the concatenation is "columns".
# The concat function will automatially combine columns with
# identical information, while keeping the additional columns.
all_neighborhoods_df = pd.concat(
    [neighborhood_locations_df, all_neighborhood_info_df],
    axis="columns",
    sort=False
)

# Review the resulting DataFrame
display(all_neighborhoods_df.head())
display(all_neighborhoods_df.tail())
```

	Lat	Lon	year	sale_price_sqr_foot	housing_units	gross_rent
Alamo Square	37.791012	-122.402100	2013.000000	366.020712	378401.0	2817.285714
Anza Vista	37.779598	-122.443451	2013.333333	373.382198	379050.0	3031.833333
Bayview	37.734670	-122.401060	2012.000000	204.588623	376454.0	2318.400000
Bayview Heights	37.728740	-122.410980	2015.000000	590.792839	382295.0	3739.000000
Bernal Heights	37.728630	-122.443050	NaN	NaN	NaN	NaN

	Lat	Lon	year	sale_price_sqr_foot	housing_units	gross_rent
Yerba Buena	37.79298	-122.39636	2012.5	576.709848	377427.5	2555.166667
Bernal Heights	NaN	NaN	2013.5	576.746488	379374.5	3080.333333
Downtown	NaN	NaN	2013.0	391.434378	378401.0	2817.285714
Ingleside	NaN	NaN	2012.5	367.895144	377427.5	2509.000000
Outer Richmond	NaN	NaN	2013.0	473.900773	378401.0	2817.285714

In [14]:

```
# Call the dropna function to remove any neighborhoods that do not have data
all_neighborhoods_df = all_neighborhoods_df.reset_index().dropna()

# Rename the "index" column as "Neighborhood" for use in the Visualization
all_neighborhoods_df = all_neighborhoods_df.rename(columns={"index": "Neighborhood"})

# Review the resulting DataFrame
```

```
display(all_neighborhoods_df.head())
display(all_neighborhoods_df.tail())
```

	Neighborhood	Lat	Lon	year	sale_price_sqr_foot	housing_units	gross_rent
0	Alamo Square	37.791012	-122.402100	2013.000000	366.020712	378401.0	2817.285714
1	Anza Vista	37.779598	-122.443451	2013.333333	373.382198	379050.0	3031.833333
2	Bayview	37.734670	-122.401060	2012.000000	204.588623	376454.0	2318.400000
3	Bayview Heights	37.728740	-122.410980	2015.000000	590.792839	382295.0	3739.000000
5	Buena Vista Park	37.768160	-122.439330	2012.833333	452.680591	378076.5	2698.833333

	Neighborhood	Lat	Lon	year	sale_price_sqr_foot	housing_units	gross_rent
68	West Portal	37.74026	-122.463880	2012.25	498.488485	376940.75	2515.500000
69	Western Addition	37.79298	-122.435790	2012.50	307.562201	377427.50	2555.166667
70	Westwood Highlands	37.73470	-122.456854	2012.00	533.703935	376454.00	2250.500000
71	Westwood Park	37.73415	-122.457000	2015.00	687.087575	382295.00	3959.000000
72	Yerba Buena	37.79298	-122.396360	2012.50	576.709848	377427.50	2555.166667

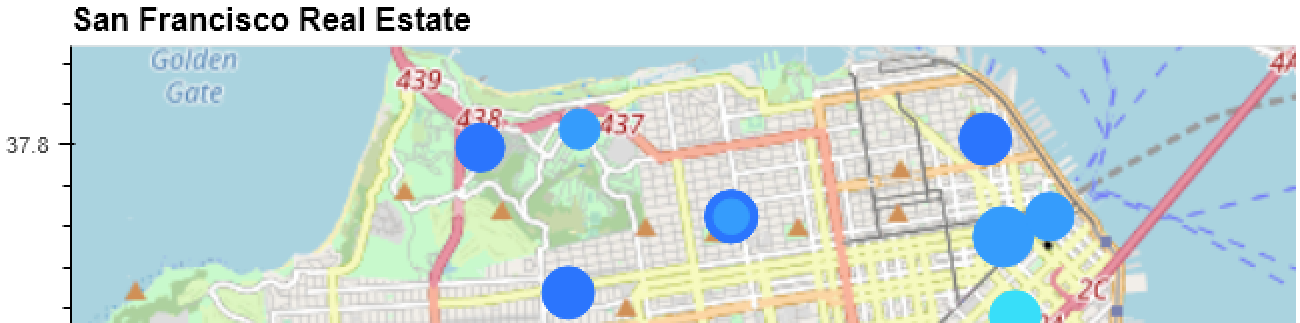
Step 4: Using hvPlot with GeoViews enabled, create a `points` plot for the `all_neighborhoods_df` DataFrame. Be sure to do the following:

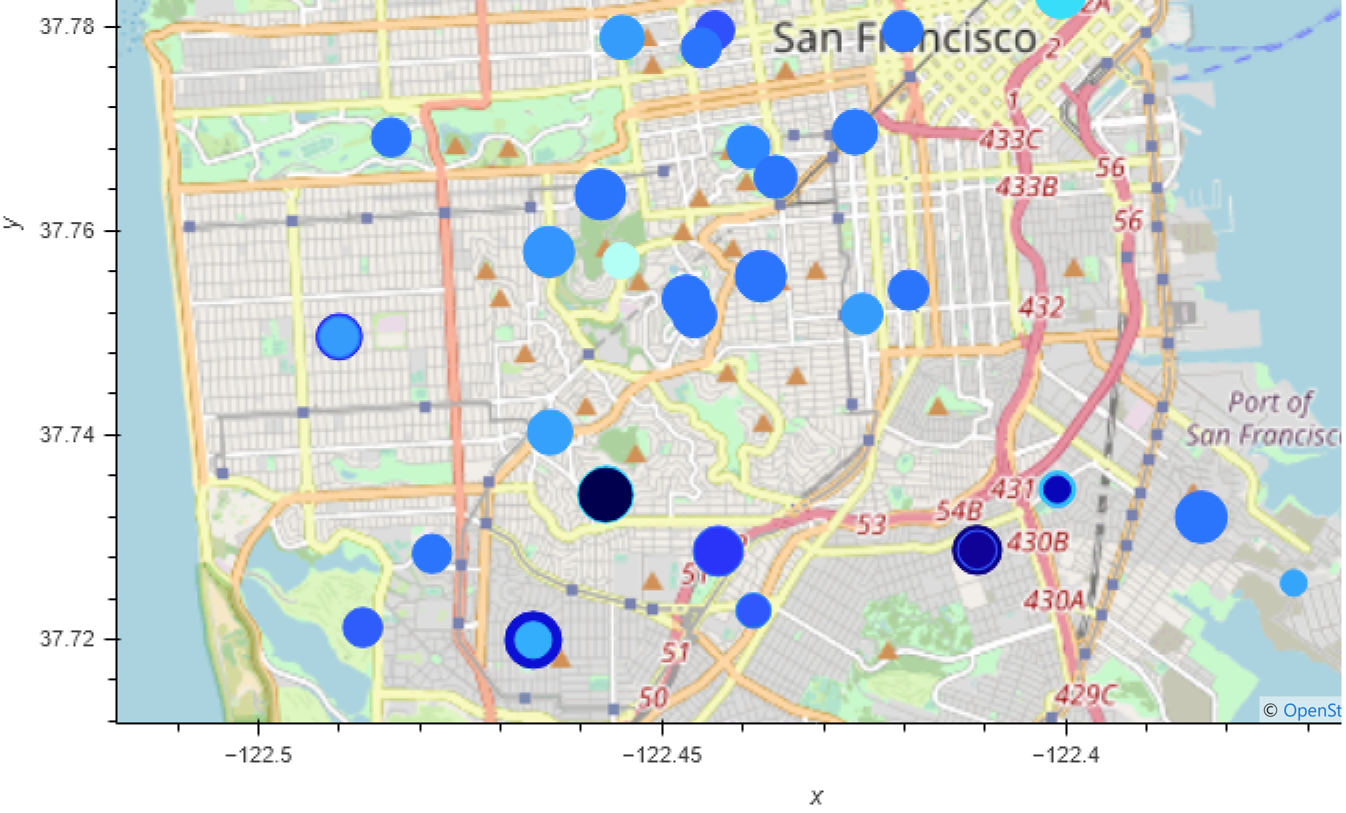
- Set the `geo` parameter to True.
- Set the `size` parameter to “sale_price_sqr_foot”.
- Set the `color` parameter to “gross_rent”.
- Set the `frame_width` parameter to 700.
- Set the `frame_height` parameter to 500.
- Include a descriptive title.

In [15]:

```
# Create a plot to analyze neighborhood info
# YOUR CODE HERE
all_neighborhoods_df.hvplot.points('Lon',
                                   'Lat',
                                   geo=True,
                                   size="sale_price_sqr_foot",
                                   color="gross_rent",
                                   tiles='OSM',
                                   frame_width=700,
                                   frame_height=500
                                   ).opts(
                                    title="San Francisco Real Estate"
                                    )
```

Out[15]:

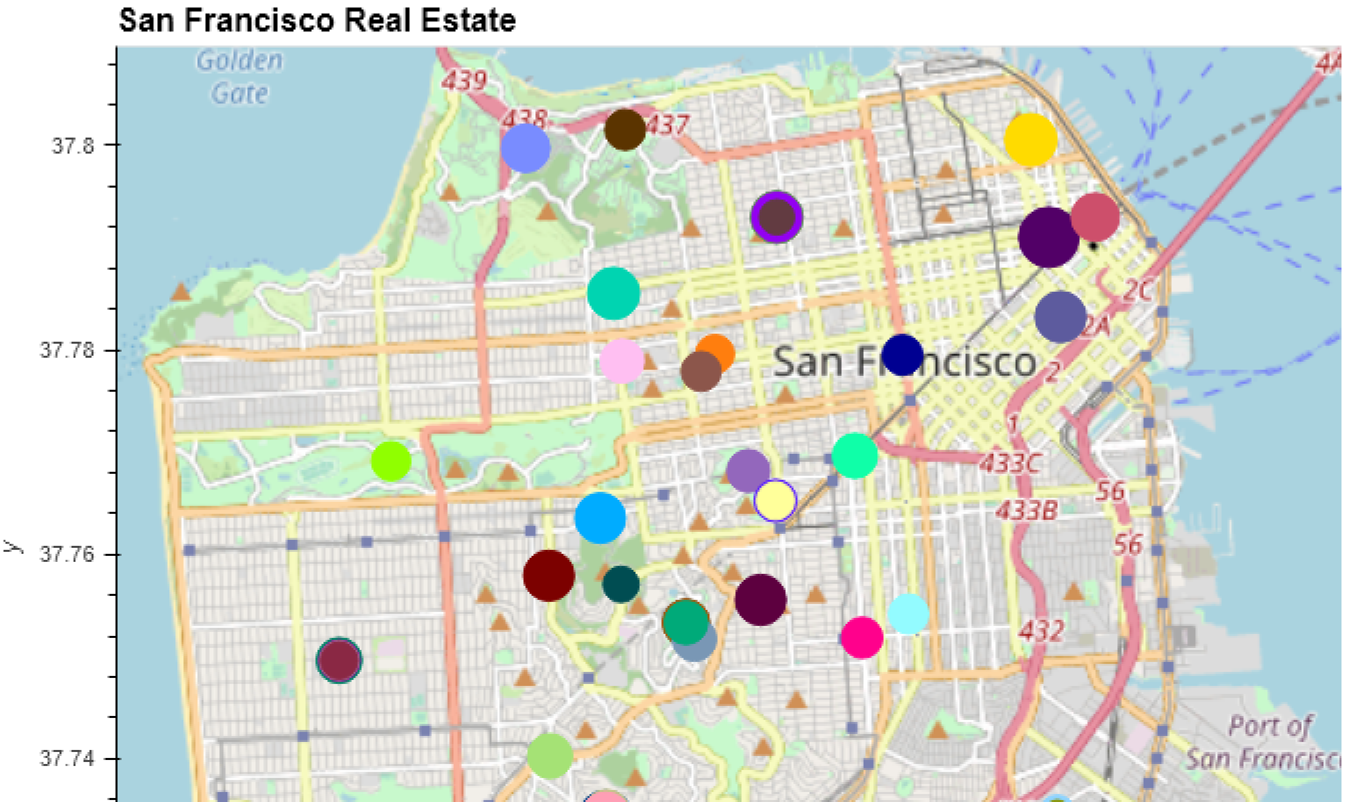


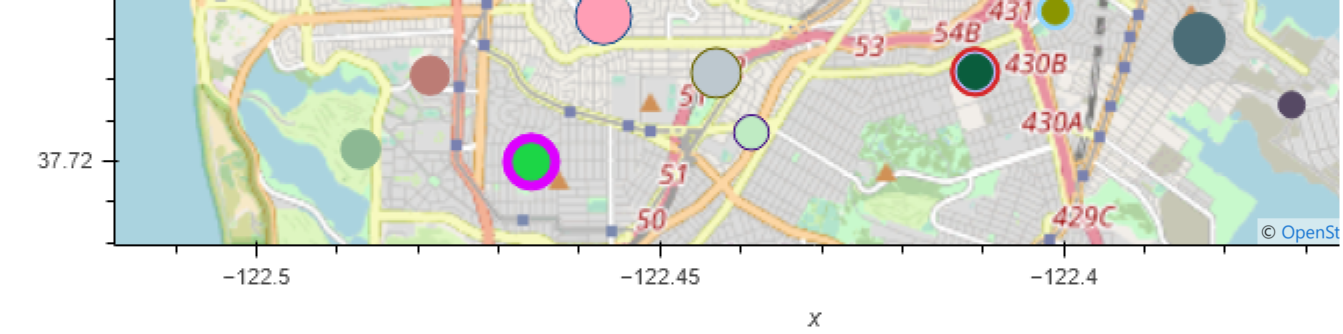


In [16]:

```
#had to plot another one with "Neighborhood" as color because previous plot does not sho
all_neighborhoods_df.hvplot.points('Lon',
                                   'Lat',
                                   geo=True,
                                   size="sale_price_sqr_foot",
                                   color="Neighborhood",
                                   tiles='OSM',
                                   frame_width=700,
                                   frame_height=500
                                   ).opts(
                                    title="San Francisco Real Estate"
                                    )
```

Out[16]:

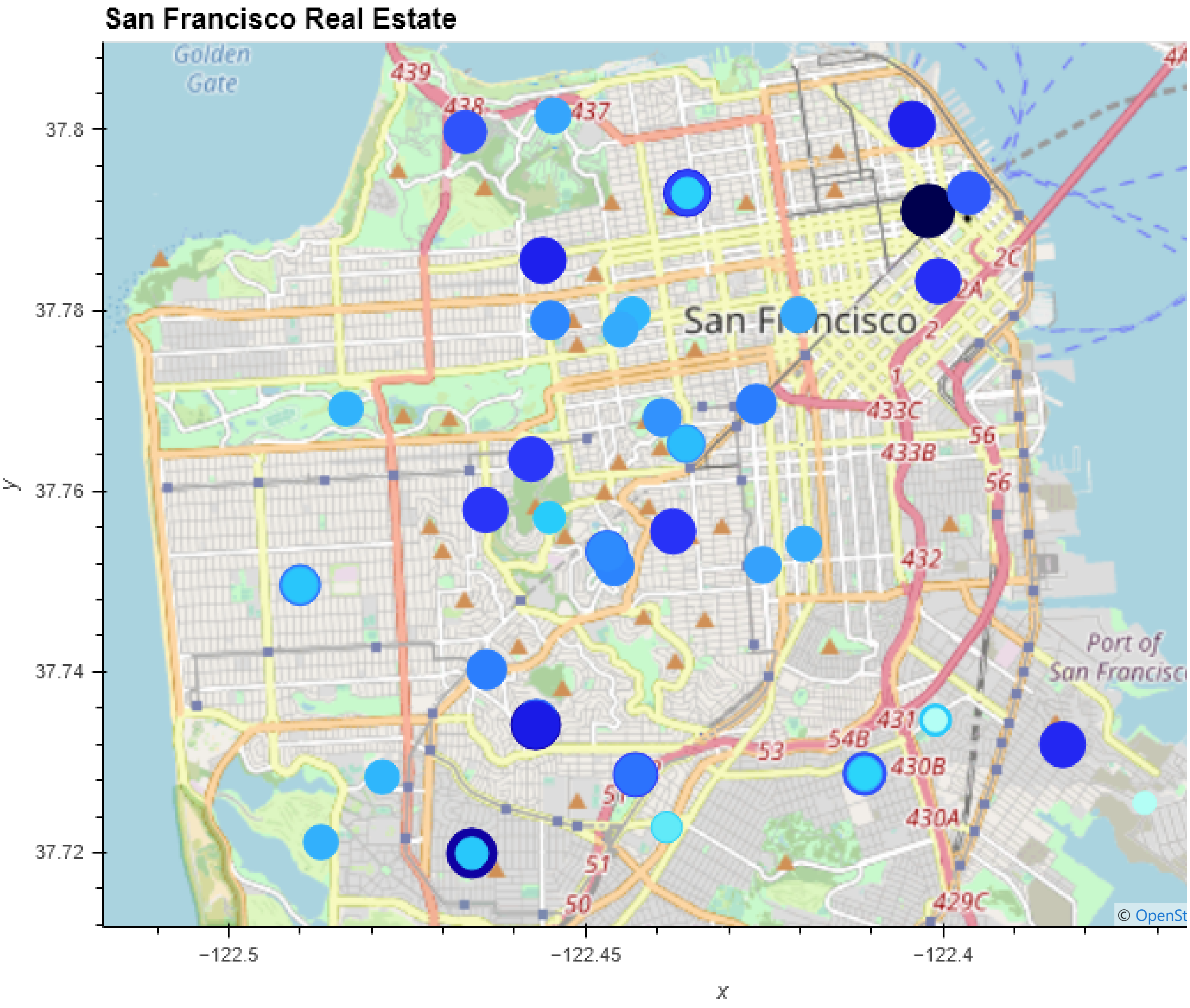




In [17]:

```
#color changed to sqrft sale to easily point out which is darkest.
all_neighborhoods_df.hvplot.points('Lon',
                                   'Lat',
                                   geo=True,
                                   size="sale_price_sqr_foot",
                                   color="sale_price_sqr_foot",
                                   tiles='OSM',
                                   frame_width=700,
                                   frame_height=500
                                   ).opts(
                                    title="San Francisco Real Estate"
                                    )
```

Out[17]:



Step 5: Use the interactive map to answer the following question:

Question: Which neighborhood has the highest gross rent, and which has the highest sale price per square foot?

Answer: # For rents, it is that dot by Bayview Heights, Excelsior and Visitation Valley that have the highest gross rent. For sale price per square foot, the darkest spot is in the Financial District, South of Market, union Square.

Compose Your Data Story

Based on the visualizations that you have created, compose a data story that synthesizes your analysis by answering the following questions:

Question: How does the trend in rental income growth compare to the trend in sales prices? Does this same trend hold true for all the neighborhoods across San Francisco?

Answer: # Based on the linear plot, gross rent has risen much faster than average sale prices. Sale prices on average are pretty stable. In some neighborhoods, it went up, and in some it went down, keeping the average around the same through time.

Question: What insights can you share with your company about the potential one-click, buy-and-rent strategy that they're pursuing? Do neighborhoods exist that you would suggest for investment, and why?

Answer: # YOUR ANSWER HERE

In [ ]:

```
#The data is valuable to see overall trends. If any investor is interested on profits fr
```