# San Francisco Rental Prices Dashboard

In this notebook, you will compile the visualizations from the previous analysis into functions that can be used for a Panel dashboard.

In [41]:
```
# imports
import panel as pn
pn.extension('plotly')
import plotly.express as px
import pandas as pd
import hvplot.pandas
import matplotlib.pyplot as plt
import os
from pathlib import Path
from dotenv import load_dotenv
```

In [42]:
```
# Read the Mapbox API key
load_dotenv()
mapbox_token = os.getenv("MAPBOX_API_KEY")
```

# Import Data

In [43]:
```
# Import the CSVs to Pandas DataFrames
file_path = Path("Data/sfo_neighborhoods_census_data.csv")
sfo_data = pd.read_csv(file_path, index_col="year")

file_path = Path("Data/neighborhoods_coordinates.csv")
df_neighborhood_locations = pd.read_csv(file_path)
```

# Panel Visualizations

In this section, you will copy the code for each plot type from your analysis notebook and place it into separate functions that Panel can use to create panes for the dashboard.

These functions will convert the plot object to a Panel pane.

Be sure to include any DataFrame transformation/manipulation code required along with the plotting code.

Return a Panel pane object from each function that can be used to build the dashboard.

Note: Remove any `.show()` lines from the code. We want to return the plots instead of showing them. The Panel dashboard will then display the plots.

In [44]:
```
# Define Panel Visualization Functions
def housing_units_per_year():
    """Housing Units Per Year."""

    minimum = sfo_data["housing_units"].min()
    maximum = sfo_data["housing_units"].max()

    housing_units_per_year = sfo_data.groupby("year").mean()

    housing_units_per_year_plot = housing_units_per_year.sort_values(["housing_units"], ascending=True).hvplo
t(
        kind="bar",
        x="year",
        y="housing_units",
        ylim=[minimum-5000, maximum+2500],
        xlabel="Year",
        ylabel="Housing Units",
        height=400,
        rot=90,
        title="Housing Units in San Francisco from 2010 to 2016"
    ).opts(yformatter="%.0f")

    return housing_units_per_year_plot


def average_gross_rent():
    """Average Gross Rent in San Francisco Per Year."""

    average_gross_rent = sfo_data.groupby("year").mean()["gross_rent"]

    average_gross_rent_plot = average_gross_rent.hvplot(figsize=(10, 8), title ="Average Gross Rent in San Fr
ancisco")

    return average_gross_rent_plot


def average_sales_price():
    """Average Sales Price Per Year."""

    average_sales_price = sfo_data.groupby("year").mean()["sale_price_sqr_foot"]

    average_sales_price_plot = average_sales_price.hvplot(figsize=(10, 8), title ="Average Sale Price per Squ
```

```
are Foot in San Francisco")

    return average_sales_price_plot


def average_price_by_neighborhood():
    """Average Prices by Neighborhood."""

    average_price_by_neighborhood = sfo_data.groupby(["year","neighborhood"]).mean()
    average_price_by_neighborhood = average_price_by_neighborhood.reset_index(drop=False)
    average_price_by_neighborhood_plot = average_price_by_neighborhood.hvplot(kind="line", x="year", y="sale_
price_sqr_foot", xlabel="Year", ylabel="Avg. Sale Price per Square Foot", groupby="neighborhood")

    return average_price_by_neighborhood_plot


def top_most_expensive_neighborhoods():
    """Top 10 Most Expensive Neighborhoods."""

    top_most_expensive_neighborhoods = sfo_data.groupby(["neighborhood"]).mean()
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods.sort_values(by="sale_price_sqr_foot",
ascending=False)
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods.reset_index()
    top_most_expensive_neighborhoods_plot = top_most_expensive_neighborhoods.head(10).sort_values(["sale_pric
e_sqr_foot"], ascending=False).hvplot(kind="bar", x="neighborhood", y="sale_price_sqr_foot", xlabel="Neighbor
hood", ylabel="Avg. Sale Price per Square Foot", height=400, rot=90, title="Top 10 Expensive Neighborhoods in
SFO")

    return top_most_expensive_neighborhoods_plot


def parallel_coordinates():
    """Parallel Coordinates Plot."""

    top_most_expensive_neighborhoods = sfo_data.groupby(["neighborhood"]).mean()
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods.sort_values(by="sale_price_sqr_foot",
ascending=False)
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods.reset_index()
    parallel_coordinates_plot = px.parallel_coordinates(top_most_expensive_neighborhoods.head(10), width=1000
, color="sale_price_sqr_foot", color_continuous_scale=px.colors.sequential.Inferno)

    return parallel_coordinates_plot
```

```
def parallel_categories():
    """Parallel Categories Plot."""

    top_most_expensive_neighborhoods = sfo_data.groupby(["neighborhood"]).mean()
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods.sort_values(by="sale_price_sqr_foot",
ascending=False)
    top_most_expensive_neighborhoods = top_most_expensive_neighborhoods.reset_index()
    parallel_categories_plot = px.parallel_categories(top_most_expensive_neighborhoods.head(10), width=1000,
color='sale_price_sqr_foot')

    return parallel_categories_plot


def neighborhood_map():
    """Neighborhood Map"""

    average_price_by_neighborhood = sfo_data.groupby(["year","neighborhood"]).mean()
    average_price_by_neighborhood = average_price_by_neighborhood.reset_index(drop=False)

    df_concat_neighborhood = pd.concat([df_neighborhood_locations, average_price_by_neighborhood], axis="colu
mns", join="inner")
    df_concat_neighborhood = df_concat_neighborhood.drop(columns=["neighborhood"])

    px.set_mapbox_access_token(mapbox_token)

    neighborhood_map = px.scatter_mapbox(
        df_concat_neighborhood,
        lat="Lat",
        lon="Lon",
        size="sale_price_sqr_foot",
        color="gross_rent",
        title="Average Sale Price per Square Foot and Gross Rent in San Francisco",
        zoom=10,
        width=1000,
    )

    return neighborhood_map
```

# Panel Dashboard

In this section, you will combine all of the plots into a single dashboard view using Panel. Be creative with your dashboard design!

In [45]:
```
# Combine all plots into single dashboard using Panel
# Create tabs
panel_dashboard = pn.Tabs(
    ("Neighborhood Map", neighborhood_map),
    ("Housing Units Per Year", housing_units_per_year),
    ("Average Gross Rent", average_gross_rent),
    ("Average Sales Price", average_sales_price),
    ("Average Price by Neighborhood", average_price_by_neighborhood),
    ("Top Most Expensive Neighborhoods", top_most_expensive_neighborhoods),
    ("Parallel Coordinates", parallel_coordinates),
    ("Parallel Categories", parallel_categories)
)
panel_dashboard
```

Out[45]:

# Serve the Panel Dashboard

In [46]:
```
panel_dashboard.servable()
```

Out[46]:

In [ ]:
```

```