

NYC Property Sales Analysis (2017–2023)

Author: Abdulla Saleh

Purpose of this project is to analyze trends in property sales prices and volume in NYC before and after COVID-19

✓ Install & Import Libraries

In this project, I used Python libraries that are helpful for working with data. These include:

- `pandas` for reading and cleaning Excel files,
- `seaborn` and `matplotlib` for creating charts,
- `folium` and `plotly` for mapping and interactive visualizations.

These tools helped me explore trends in NYC property sales data from 2017 to 2023.

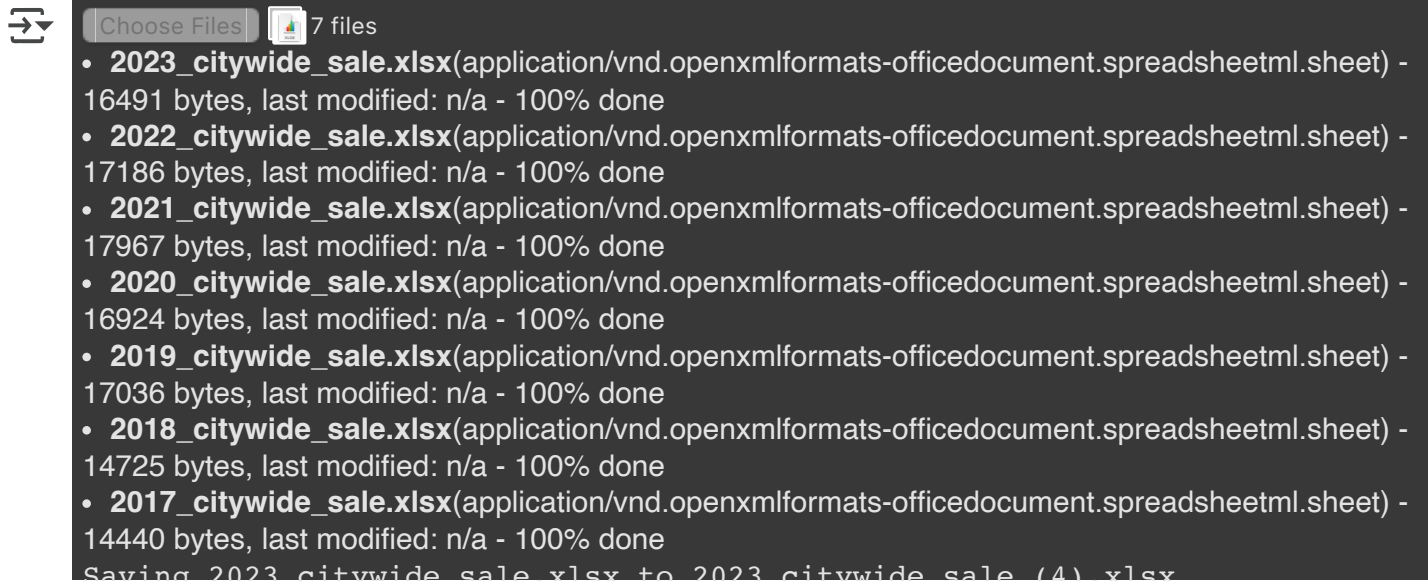
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import folium
from folium.plugins import MarkerCluster
from google.colab import files
import json # for GeoJSON
import copy # for deepcopy
```

✓ Link to Datasets & Upload Excel Files

I downloaded Excel files from NYC Open Data that contain property sales summaries by borough and building type for each year (2017–2023). I uploaded them here so I could clean and analyze the data using Python.

```
# Datasets were downloaded from this link https://www.nyc.gov/site/finance/property
# Upload all 7 Excel files (manually select all in the file picker)
```

```
uploaded = files.upload()
```



The image shows a file upload interface with a 'Choose Files' button and a list of 7 files. Each file is an Excel spreadsheet (xlsx) representing citywide property sales for a specific year from 2017 to 2023. The files are all successfully uploaded (100% done).

- **2023_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 16491 bytes, last modified: n/a - 100% done
- **2022_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 17186 bytes, last modified: n/a - 100% done
- **2021_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 17967 bytes, last modified: n/a - 100% done
- **2020_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 16924 bytes, last modified: n/a - 100% done
- **2019_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 17036 bytes, last modified: n/a - 100% done
- **2018_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 14725 bytes, last modified: n/a - 100% done
- **2017_citywide_sale.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 14440 bytes, last modified: n/a - 100% done

Saving 2023 citywide sale.xlsx to 2023 citywide sale (4).xlsx

Load & Clean Raw Excel Files

```
# Dictionary of year to file name
file_paths = {
    2017: '2017_citywide_sale.xlsx',
    2018: '2018_citywide_sale.xlsx',
    2019: '2019_citywide_sale.xlsx',
    2020: '2020_citywide_sale.xlsx',
    2021: '2021_citywide_sale.xlsx',
    2022: '2022_citywide_sale.xlsx',
    2023: '2023_citywide_sale.xlsx',
}

# Load and combine all files
all_years_data = []

for year, path in file_paths.items():
    sheet_name = f'Boro_Citywide_Yr_{year}' if year >= 2019 else 0
    skiprows = 4

    df = pd.read_excel(path, sheet_name=sheet_name, skiprows=skiprows)

    if year >= 2019:
        df.columns = df.iloc[0]
        df = df[1:]

    df = df[df['BOROUGH'].notnull() & ~df['BOROUGH'].astype(str).str.contains('TO')]
    df['YEAR'] = year

    df = df[[
        'BOROUGH', 'BUILDING CLASS CATEGORY', 'NUMBER OF SALES',
        'MINIMUM SALE PRICE', 'AVERAGE SALE PRICE', 'MEDIAN SALE PRICE',
        'MAXIMUM SALE PRICE', 'YEAR'
    ]]

    for col in ['NUMBER OF SALES', 'MINIMUM SALE PRICE', 'AVERAGE SALE PRICE',
                'MEDIAN SALE PRICE', 'MAXIMUM SALE PRICE']:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    all_years_data.append(df)

df_all_years_clean = pd.concat(all_years_data, ignore_index=True)
```

✓ Summary Data for Visualizations

I created a quick summary of the cleaned data to look at general trends across all years. This includes:

- Average and median sale prices,
- Number of sales per year.

These summaries helped me get an overall idea of how the NYC housing market has changed over time Pre Covid and Post Covid.

```
summary = df_all_years_clean.groupby('YEAR').agg({
    'MEDIAN SALE PRICE': 'mean',
    'AVERAGE SALE PRICE': 'mean',
    'NUMBER OF SALES': 'sum'
}).reset_index()
summary.columns = ['Year', 'Median Sale Price', 'Average Sale Price', 'Total Sale
```

✓ Visualizations (Citywide Trends)

This section shows three charts:

- A line chart for average sale prices per year,
- Another for median sale prices per year.
- Another for total sales volume per year.

These charts give a high-level look at how housing prices have gone up or down across NYC as a whole.

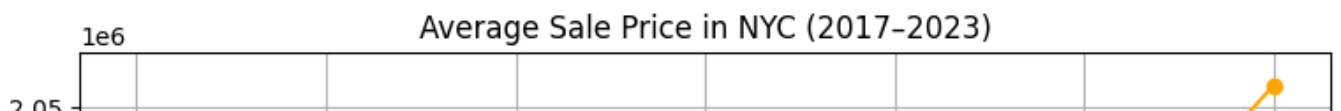
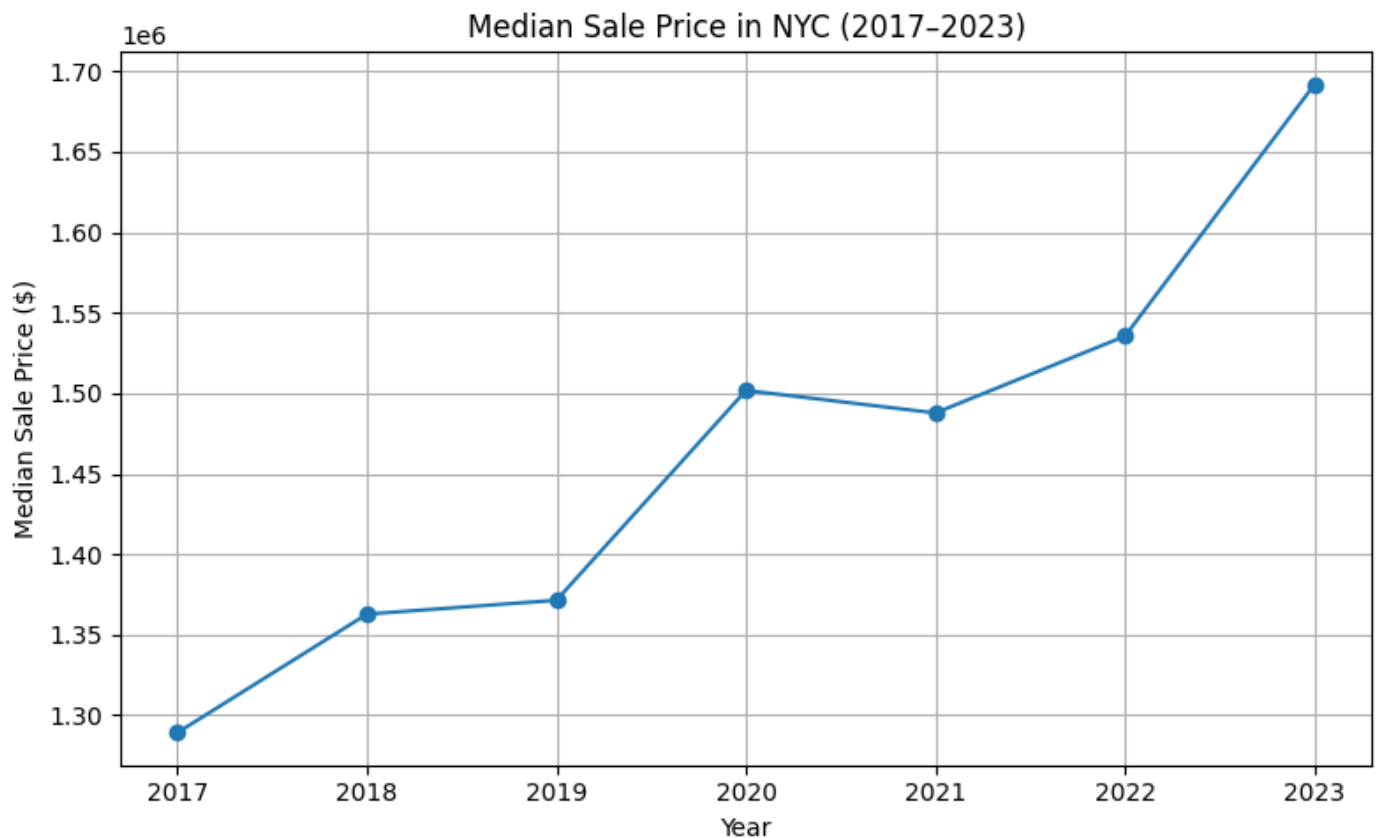
```
# 1. Median Sale Price Over Time
plt.figure(figsize=(8, 5))
plt.plot(summary['Year'], summary['Median Sale Price'], marker='o')
plt.title('Median Sale Price in NYC (2017-2023)')
plt.xlabel('Year')
plt.ylabel('Median Sale Price ($)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

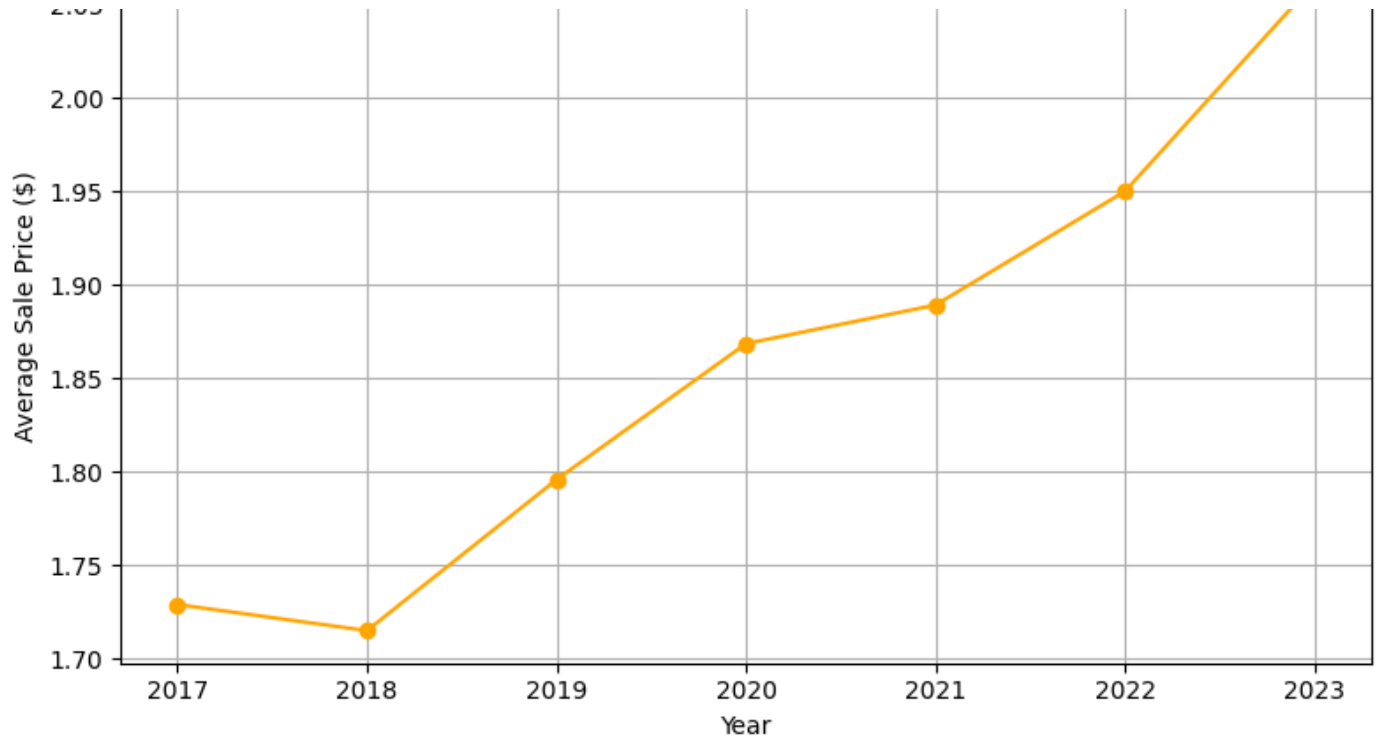
2. Average Sale Price Over Time

```
plt.figure(figsize=(8, 5))
plt.plot(summary['Year'], summary['Average Sale Price'], marker='o', color='orange')
plt.title('Average Sale Price in NYC (2017-2023)')
plt.xlabel('Year')
plt.ylabel('Average Sale Price ($)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

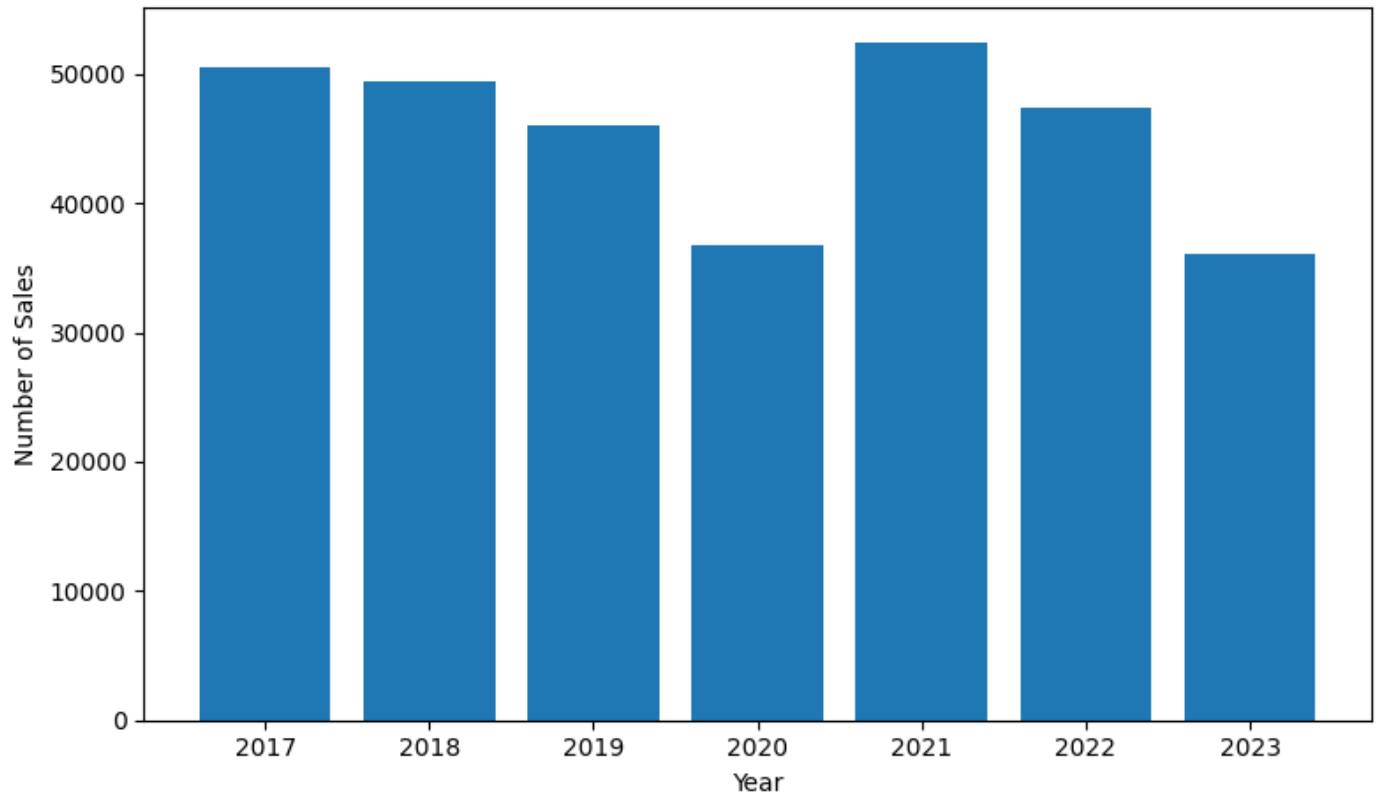
3. Total Number of Sales

```
plt.figure(figsize=(8, 5))
plt.bar(summary['Year'], summary['Total Sales'])
plt.title('Total Sales Volume in NYC (2017-2023)')
plt.xlabel('Year')
plt.ylabel('Number of Sales')
plt.tight_layout()
plt.show()
```





Total Sales Volume in NYC (2017-2023)





✓ Borough-Level Insights (2017–2023)

In this section, I focused on comparing NYC's boroughs to see how property prices and sales activity changed in each one.

The visualizations include:

- Median sale price by borough (line chart)
- Number of sales by borough (line chart)
- Distribution of median sale prices by year (boxplot)

This helped me see patterns like how Manhattan always has the highest prices, how COVID affected sales in 2020, and how some boroughs rebounded faster than others.

```
# Define the relevant price and sales columns again
price_cols = [
    "NUMBER OF SALES",
    "MINIMUM SALE PRICE",
    "AVERAGE SALE PRICE",
    "MEDIAN SALE PRICE",
    "MAXIMUM SALE PRICE"
]

# Remove rows where a header string accidentally appears in the data
df_all_years_clean = df_all_years_clean[
    ~df_all_years_clean["NUMBER OF SALES"].astype(str).str.contains("NUMBER OF SA
]

# Re-apply numeric conversion
df_all_years_clean[price_cols] = df_all_years_clean[price_cols].replace('[\$,]',
df_all_years_clean[price_cols] = df_all_years_clean[price_cols].replace(',', ' ',
```

```

print(df_all_years_clean['BOROUGH'].unique())
print(df_all_years_clean['MEDIAN SALE PRICE'].dtype)

# Clean borough names again (just in case)
df_all_years_clean['BOROUGH'] = df_all_years_clean['BOROUGH'].str.strip().str.title

# 1. Median Sale Price by Borough
borough_median = df_all_years_clean.groupby(['YEAR', 'BOROUGH'])['MEDIAN SALE PRICE'].mean()
plt.figure(figsize=(10, 6))
sns.lineplot(data=borough_median, x='YEAR', y='MEDIAN SALE PRICE', hue='BOROUGH',
plt.title('Median Sale Price by Borough (2017–2023)')
plt.xlabel('Year')
plt.ylabel('Median Sale Price ($)')
plt.legend(title='Borough', bbox_to_anchor=(1.05, 1))
plt.grid(True)
plt.tight_layout()
plt.show()

# 2. Sales Volume by Borough
borough_sales = df_all_years_clean.groupby(['YEAR', 'BOROUGH'])['NUMBER OF SALES'].sum()
plt.figure(figsize=(10, 6))
sns.lineplot(data=borough_sales, x='YEAR', y='NUMBER OF SALES', hue='BOROUGH',
plt.title('Sales Volume by Borough (2017–2023)')
plt.xlabel('Year')
plt.ylabel('Number of Sales')
plt.legend(title='Borough', bbox_to_anchor=(1.05, 1))
plt.grid(True)
plt.tight_layout()
plt.show()

print(df_all_years_clean['MEDIAN SALE PRICE'].dtype)

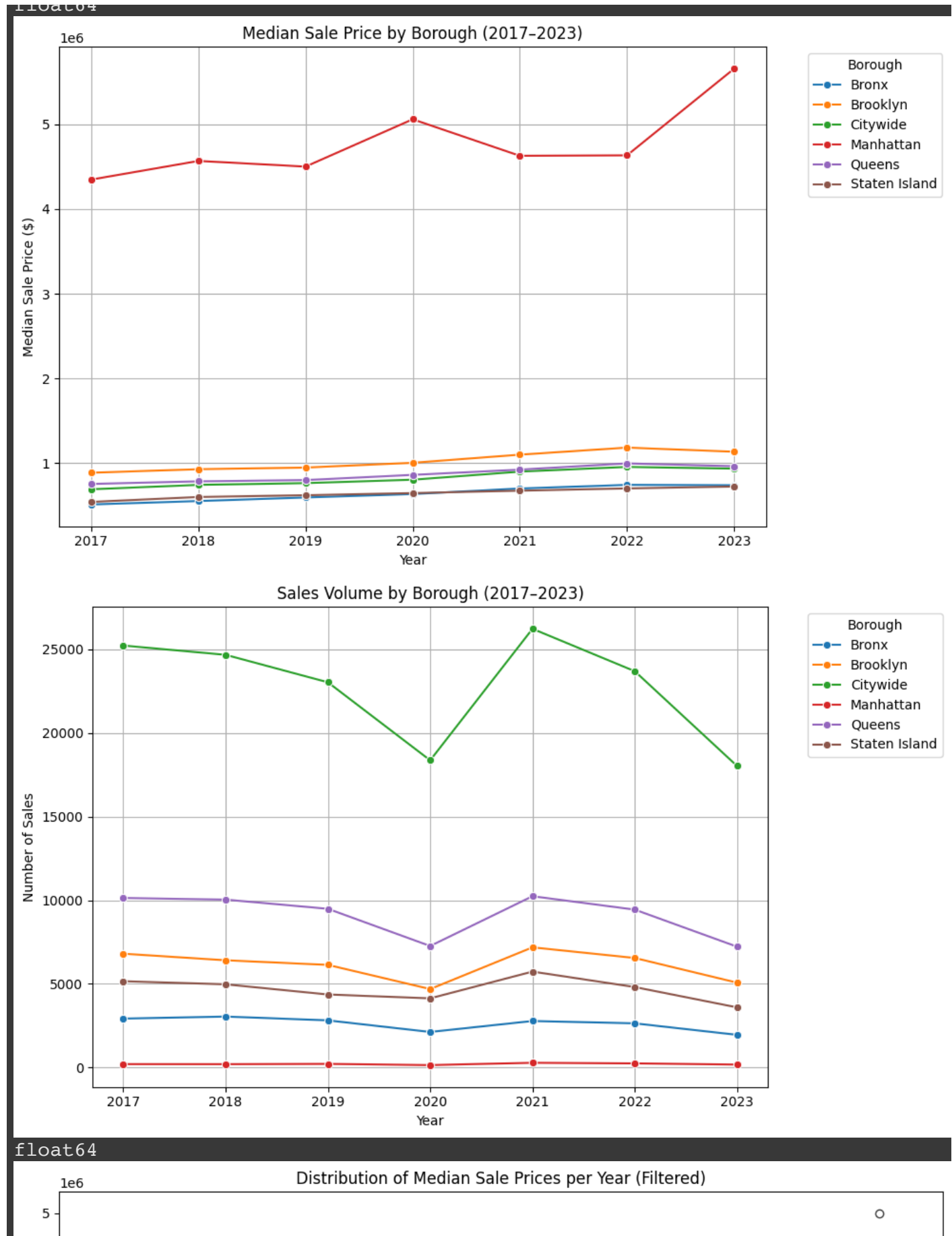
# 3. Boxplot of Median Sale Prices (filtered for clarity)
filtered_df = df_all_years_clean[df_all_years_clean['MEDIAN SALE PRICE'] < 5_000_]
plt.figure(figsize=(10, 6))
sns.boxplot(data=filtered_df, x='YEAR', y='MEDIAN SALE PRICE')
plt.title('Distribution of Median Sale Prices per Year (Filtered)')
plt.xlabel('Year')
plt.ylabel('Median Sale Price ($)')
plt.tight_layout()
plt.show()

```

```

['Manhattan' 'Bronx' 'Brooklyn' 'Queens' 'Staten Island' 'Citywide'
'Manhattan' 'Bronx' 'Brooklyn' 'Queens' 'Staten Island' 'Citywide'
'CITYWIDE' 'MANHATTAN' 'BRONX' 'BROOKLYN' 'QUEENS' 'STATEN ISLAND']
float64

```

float64

Distribution of Median Sale Prices per Year (Filtered)

1e6

5

Distribution

2017

2018

2019

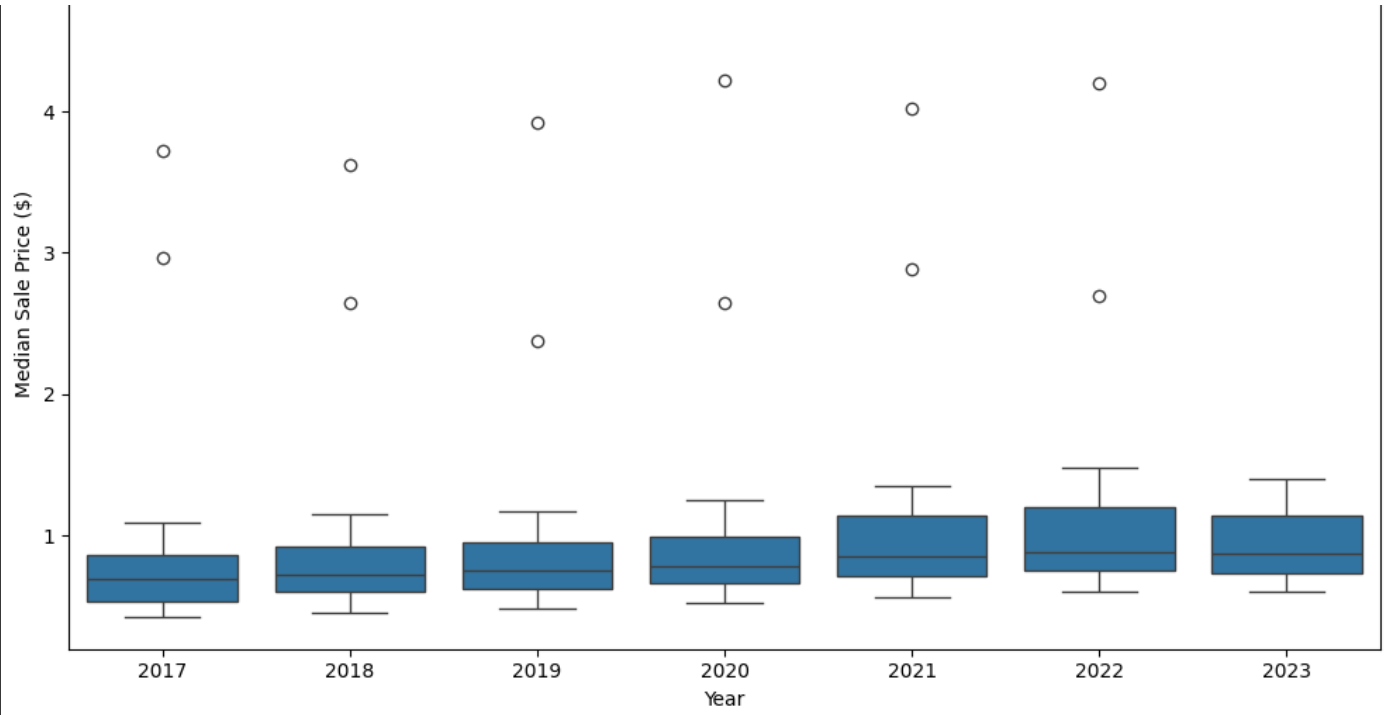
2020

2021

2022

2023

Year



✓ Upload GeoJSON & Preview Map Shapes

To make a geographic map of the data, I uploaded a GeoJSON file with NYC borough boundaries. This file helps me connect each borough's sales data to its shape on the map.

```
# Load the NYC Boroughs GeoJSON from uploaded file
with open('/nyc_boroughs.geojson', 'r') as file:
    nyc_boroughs_geo = json.load(file)

# Preview the GeoJSON properties to confirm the key
nyc_boroughs_geo['features'][0]['properties']
```

```
↔ {'OBJECTID': 1,
   'BoroCode': 5,
   'BoroName': 'Staten Island',
   'Shape__Area': 1623620701.8180466,
   'Shape__Length': 325901.48380982}
```

✓ Folium Map: Interactive Choropleths (2017–2023)

I used the Folium library to make an interactive map that lets you switch between years (2017 to 2023). Each layer shows median sale prices for that year using color shading. This lets us see how prices moved across boroughs over time.

```
# Define color palette by year for visual distinction
color_palette = {
    2017: 'YlGnBu',
    2018: 'BuPu',
    2019: 'PuBuGn',
    2020: 'YlOrBr',
    2021: 'OrRd',
    2022: 'PuRd',
    2023: 'YlOrRd'
}

# Initialize the map centered on NYC
multi_year_map = folium.Map(location=[40.7128, -74.0060], zoom_start=10, tiles='carto')

# Loop through each year and create separate FeatureGroup for toggling
```

```

for year in range(2017, 2024):
    year_data = df_all_years_clean[df_all_years_clean['YEAR'] == year]
    if not year_data.empty:
        median_by_borough = year_data.groupby('BOROUGH')['MEDIAN SALE PRICE'].median()
        median_by_borough.columns = ['Borough', 'Median Sale Price']
        median_by_borough['Borough'] = median_by_borough['Borough'].str.title()

        geojson_copy = copy.deepcopy(nyc_boroughs_geo)
        for feature in geojson_copy['features']:
            boro_name = feature['properties']['BoroName']
            match = median_by_borough[median_by_borough['Borough'].str.lower() == boro_name]
            if not match.empty:
                price = match['Median Sale Price'].values[0]
                feature['properties']['tooltip'] = f"{boro_name} ({year}): ${price:,.2f}"

# Choropleth layer directly on map
folium.Choropleth(
    geo_data=geojson_copy,
    data=median_by_borough,
    columns=['Borough', 'Median Sale Price'],
    key_on='feature.properties.BoroName',
    fill_color=color_palette.get(year, 'YlGnBu'),
    fill_opacity=0.7,
    line_opacity=0.4,
    legend_name=f'Median Sale Price ({year})',
    name=f"{year} Choropleth"
).add_to(multi_year_map)

# Tooltip layer grouped separately
tooltip_group = folium.FeatureGroup(name=f"{year} Tooltips", show=(year == 2024))
folium.GeoJson(
    geojson_copy,
    style_function=lambda x: {
        'fillColor': 'transparent',
        'color': 'black',
        'weight': 0.5,
        'dashArray': '5, 5'
    },
    tooltip=folium.GeoJsonTooltip(
        fields=['tooltip'],
        aliases=[],
        sticky=True,
        direction='top',
        opacity=0.9,
        style=("background-color: white; padding: 6px; border-radius: 4px;")
    )
).add_to(tooltip_group)

```

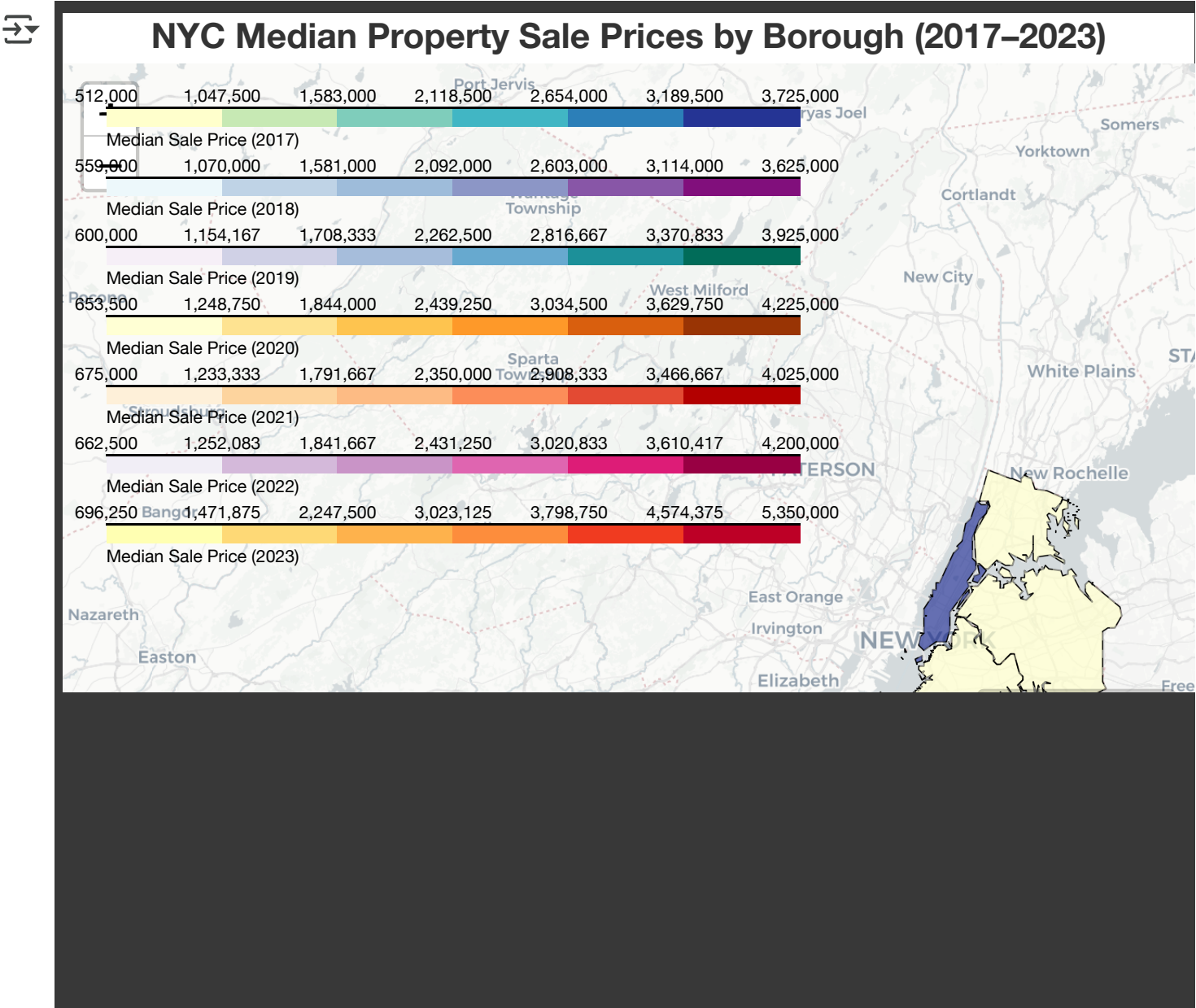
```
    )
    ).add_to(tooltip_group)

    tooltip_group.add_to(multi_year_map)

# Add custom floating title using HTML template
title_html = '''
    <h3 align="center" style="font-size:20px"><b>NYC Median Property Sale Prices by
    ...
multi_year_map.get_root().html.add_child(folium.Element(title_html))

# Add layer control
folium.LayerControl(collapsed=False).add_to(multi_year_map)

# Save map
multi_year_map.save("NYC_Median_Prices_By_Year.html")
multi_year_map
```



✓ Quick Charts & Tables

This section includes extra visuals that helped me quickly compare:

- Median sale price trends by borough,
- Average sale prices by borough and year (in a heatmap style).

These charts supported the main insights I found earlier and gave more detail on borough-by-borough changes.

```
# Normalize borough names BEFORE aggregation to avoid duplicates
```

```

df_clean = df_all_years_clean.copy()
df_clean['BOROUGH'] = df_clean['BOROUGH'].str.strip().str.title()

# Summary table: median sale price by borough and year
summary_table = df_clean.groupby(['YEAR', 'BOROUGH'])['MEDIAN SALE PRICE'].median
pivot_table = summary_table.pivot(index='YEAR', columns='BOROUGH', values='MEDIAN
display(pivot_table.style.format('${:,0f}').set_caption("Median Sale Price by Bo

plt.figure(figsize=(10, 6))
sns.lineplot(data=summary_table, x='YEAR', y='MEDIAN SALE PRICE', hue='BOROUGH', i
plt.title('Median Sale Price Trends by Borough (2017–2023)')
plt.xlabel('Year')
plt.ylabel('Median Sale Price ($)')
plt.grid(True)
plt.tight_layout()
plt.show()

print(" Manhattan had the highest average sale price every year. Brooklyn and Quee

# Summary table: average sale price by borough and year
avg_summary = df_clean.groupby(['YEAR', 'BOROUGH'])['AVERAGE SALE PRICE'].mean().
pivot_avg = avg_summary.pivot(index='YEAR', columns='BOROUGH', values='AVERAGE SA
display(pivot_avg.style.format('${:,0f}').set_caption("Average Sale Price by Bor

plt.figure(figsize=(10, 6))
sns.lineplot(data=avg_summary, x='YEAR', y='AVERAGE SALE PRICE', hue='BOROUGH', m
plt.title('Average Sale Price Trends by Borough (2017–2023)')
plt.xlabel('Year')
plt.ylabel('Average Sale Price ($)')
plt.grid(True)
plt.tight_layout()
plt.show()

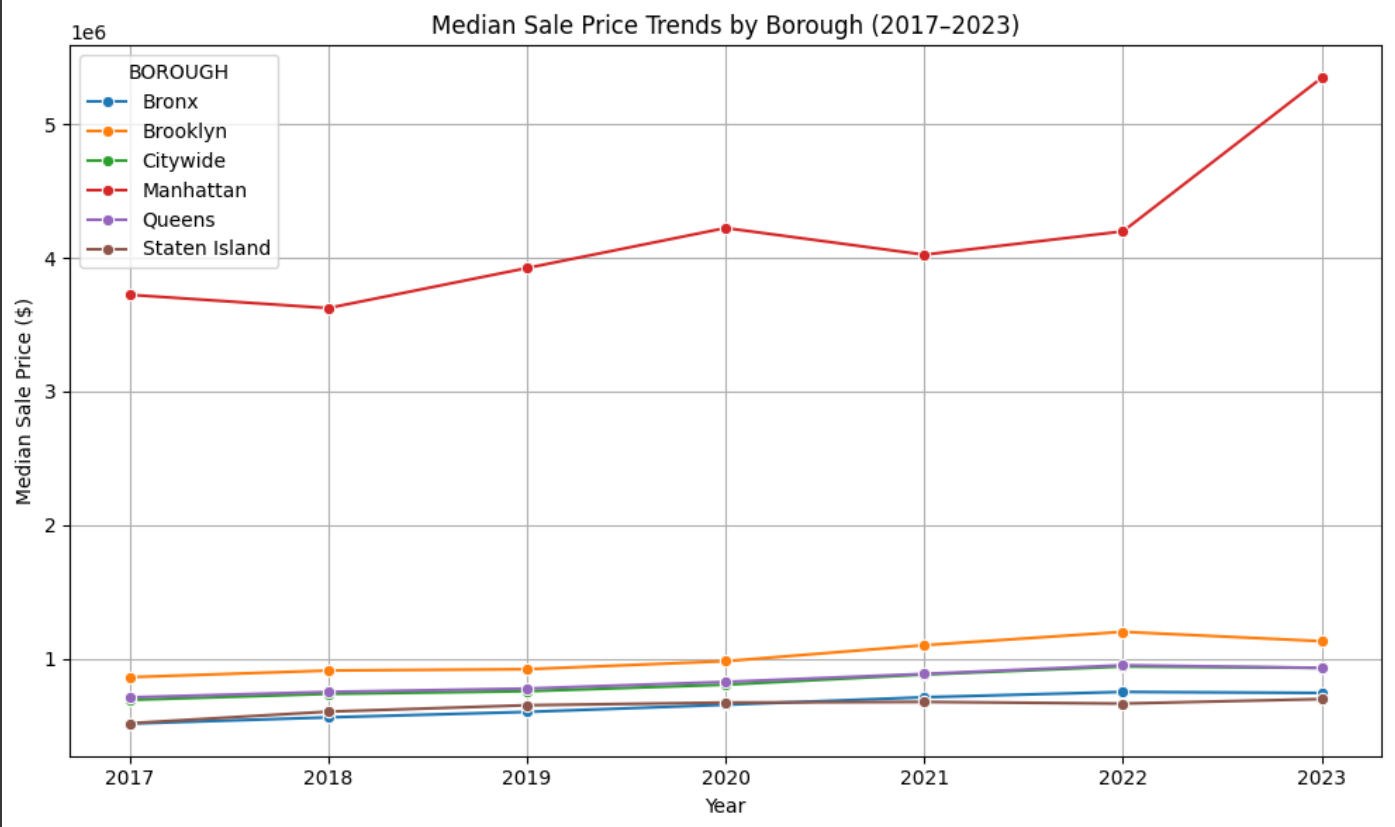
```



Median Sale Price by Borough and Year (Cleaned)

BOROUGH	Bronx	Brooklyn	Citywide	Manhattan	Queens	Staten Island
YEAR						
2017	\$512,000	\$860,000	\$690,000	\$3,725,000	\$709,000	\$515,000
2018	\$559,000	\$910,000	\$735,000	\$3,625,000	\$750,000	\$602,000
2019	\$600,000	\$960,000	\$755,000	\$3,665,000	\$775,000	\$650,000

2019	\$600,000	\$920,000	\$755,000	\$3,925,000	\$775,000	\$650,000
2020	\$653,500	\$980,000	\$804,000	\$4,225,000	\$825,000	\$670,000
2021	\$710,000	\$1,100,000	\$880,000	\$4,025,000	\$885,000	\$675,000
2022	\$750,000	\$1,200,000	\$940,000	\$4,200,000	\$950,000	\$662,500
2023	\$742,250	\$1,130,000	\$930,000	\$5,350,000	\$930,000	\$696,250

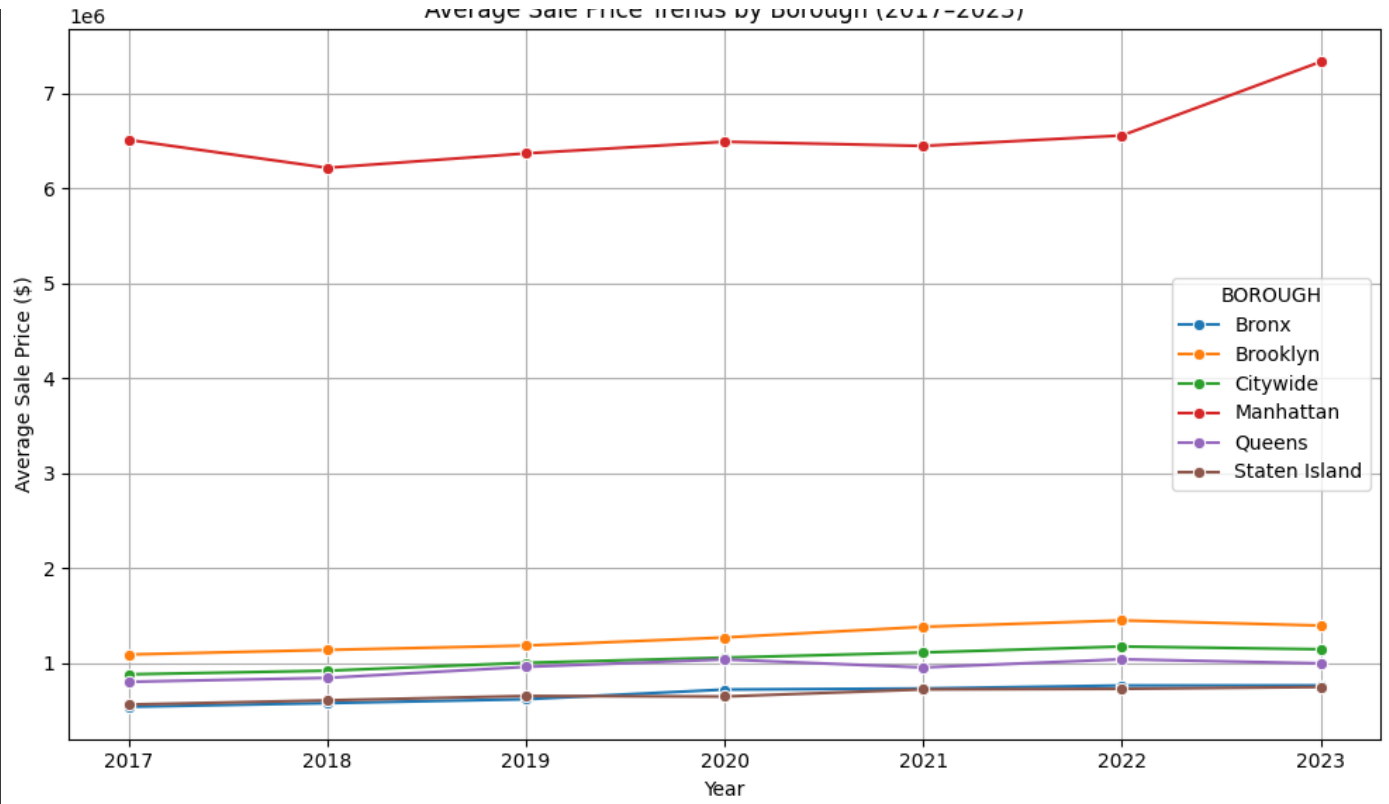


Manhattan had the highest average sale price every year. Brooklyn and Queens

Average Sale Price by Borough and Year

BOROUGH	Bronx	Brooklyn	Citywide	Manhattan	Queens	Staten Island
YEAR						
2017	\$535,981	\$1,087,048	\$878,168	\$6,513,480	\$799,353	\$559,460
2018	\$574,552	\$1,135,492	\$916,476	\$6,218,107	\$840,880	\$604,070
2019	\$615,832	\$1,181,583	\$999,839	\$6,370,465	\$956,810	\$650,079
2020	\$717,726	\$1,266,833	\$1,054,250	\$6,493,439	\$1,034,887	\$644,390
2021	\$728,902	\$1,379,011	\$1,108,489	\$6,449,250	\$950,678	\$719,319
2022	\$760,601	\$1,446,801	\$1,171,685	\$6,559,216	\$1,036,938	\$725,709
2023	\$761,755	\$1,392,523	\$1,142,563	\$7,336,853	\$993,971	\$744,341

Average Sale Price Trends by Borough (2017-2023)



Key Insights

- Manhattan had the highest prices but also the biggest drop in 2020.
- Queens and Brooklyn showed more steady growth, especially after COVID.
- There was a noticeable drop in sales during 2020, followed by a strong recovery in 2021.

Takeaway

This project helped me explore real estate trends using both charts and maps. I learned how to clean messy Excel data, build visualizations, and find patterns in time and location. These skills could be useful for people in real estate, city planning, or anyone interested in housing markets.